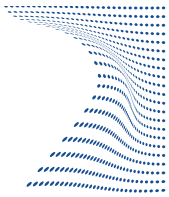
ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR



Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (32nd cycle)

Machine learning techniques to forecast non-linear trends in smart environments

Alessandro Aliberti
* * * * * *

Supervisor
Prof. Macii Enrico

Doctoral Examination Committee:
Prof. Luciano Baresi, Referee, Politecnico di Milano
Prof. Marco Pau, Referee, RWTH Aachen University
Prof. Andrea Calimera, Politecnico di Torino
Prof. Silvia Chiusano, Politecnico di Torino
Prof. Ana-Maria Dumitrescu, Politehnica University of Bucharest

Politecnico di Torino
July 27, 2020

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . . . .

Alessandro Aliberti

Turin, July 27, 2020

Summary

According to the World Health Organisation (WHO), air pollution is responsible for 7 million deaths every year, and 91% of the world population lives in places where air quality exceeds the limits mandated by WHO itself. In recent years, the research about energy waste and pollution reduction has gained a strong momentum, also pushed by European and national funding initiatives. The primary purpose of this large effort is to reduce the effects of greenhouse emission, climate change to head for a sustainable society. In this scenario, Information and Communication Technologies (ICT) play a key role in reducing energy consumption and moving forward to a more sustainable and smart society.

IoT devices are used in many contexts, to add smartness to cities, energy, and industrial processes. Their pervasiveness, combined with the recent development of machine learning techniques, allows collecting a large amount of data, enabling original opportunities to create innovative modelling and optimization approaches. Consequently, moving towards smart and sustainable energy use, my research activities have mainly focused on the design and the optimization of innovative machine learning methodologies, by exploiting primarily neural networks, for the forecasting of time-series in Smart City context. In this manuscript, I propose innovative and optimized stream data processing and machine learning methodologies, ranging from energy and environmental data and moving to data from CPS systems. I have designed and validated innovative modelling and control strategies in specific application case studies: i) Renewables, ii) Smart Building and iii) Smart Health.

In detail, I addressed the issues of prediction of GHI which is the energy component necessary for the development of photovoltaic energy, the thermal modelling of Smart Buildings and finally, I moved toward the person health by addressing the topic of blood glucose level prediction for Type I diabetic patients. In all the three cases, the studies were conducted following a bottom-up approach starting from the analysis and appropriate pre-processing of IoT data, designing neural models suitable for the type of dataset and its characteristics and finally comparing these new models with the methodologies of the literature.

In the context of renewables, I developed a methodology for photovoltaic predictions starting from the physical phenomenon of GHI. Then, I focused on the optimization of the methods for GHI forecasting in short- and mid-term. Lastly,

I also investigated how to properly exploit exogenous inputs (i.e. physical factors related to the phenomenon of solar radiation) to further improve GHI predictions. In the context of Smart Buildings, I developed a comprehensive methodology that enables thermal modelling in both new generation and historic buildings, by exploiting the possibility of creating a very reliable synthetic dataset based on BIM technology and real weather data (TMY). This configuration allowed me to develop hybrid neural models trained with synthetic data and able to exploit real data (i.e. provided by IoT devices installed in a real-world demonstrator) for indoor air-temperature predictions. Finally, by using Transfer Learning techniques, I was able to specialize the hybrid models on real data. Lastly, in the context of smart health, I addressed the problem of automated glucose level prediction leveraging multi-patient CGMS data. The aim was to learn a generalizable glucose level prediction model from a multi-patient training set, using this model to predict the future glucose values of a new patient. In practice, the objective is to create a device that can be purchased and is ready to use, without the need for initial tuning. Besides, I started to evaluate techniques to specialize this methodology by integrating real-time information to specialize the predictor system specifically on the single end-user.

Acknowledgements

I would like to express my gratitude to the Cluster on Smart Communities: Edifici a Zero Consumo Energetico in Distretti Urbani Intelligenti (EEB), funded by Italian government, for letting me be part of this incredible network. I would like to express a special thank you to my supervisor, Prof. Enrico Macii. His support, guidance and overall insights have made this an inspiring experience for me. Even to Prof. Andrea Acquaviva, who has supported me on this long journey. I am extremely grateful for our friendly chats at the end of our meetings and his personal support in my academic and business endeavours. My deepest gratitude goes to Prof. Edoardo Patti for guiding me in my research activities, publications and for the stimulating questions. The meetings and conversations were vital in inspiring me to think outside the box, from multiple perspectives to form a comprehensive and objective critique. I would like to thank Prof. Giansalvo Cirrincione for introducing me to the world of Neural Networks. From the bottom of my heart I would like to say big thank you for all the EDA research group members for their energy, understanding and help throughout my PhD journey. Finally, I would like to thank my family an my friends, near and far, for supporting me day by day. Elisa you are my sun, my moon, and all my stars.

Opus est enim ad notitiam sui experimento; quid quisque posset nisi temptando non didicit.

Lucio Anneo Seneca - De providentia

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Professor Philip N. Howard of Oxford University, in [115] has proved that in 2014 the number of connected devices surpassed the number of people on our planet. Further, as shown in the Figure 1.1, he estimated that more than 50 billion Internet-connected devices would circulate in 2020, with annual growth of more than 12 billion units.

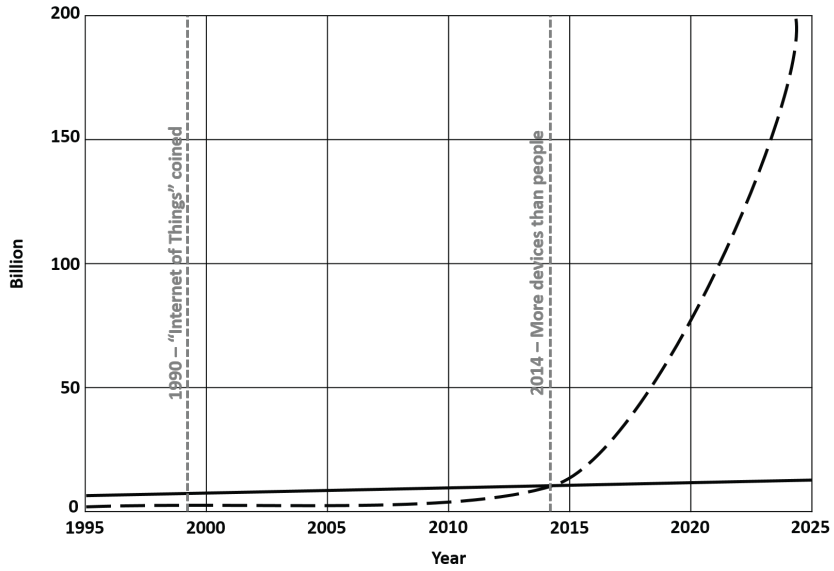


Figure 1.1: Philip N. Howard's Study of Connected Devices in relation to the population

As a result of this impetuous growth, this year IoT will impact close to 6% of the global economy, with an incremental revenue potential of $300 billion in services [31]. International Data Corporation estimates that this huge amount of smart devices will generate 79.4 zettabytes of data in 2025 [246]. As the spread of devices

and sensors grows, the amount of data that is to be managed and the number of applications that need to be developed is growing.

This combination of smart devices and a large amount of available data lays the foundation for the current and future Smart Cities.

1.1 Smart City: where everything is interconnected

International research community has been talking about *Smart Cities* for years. However, the term is often used vaguely or improperly, sometimes with an exclusively technological focus, sometimes with a science fiction approach [217]. Mark Deakin, professor at Edinburgh Napier University and luminary on Smart Cities, defines it as a *city that utilizes ICT to meet the demands of its citizens* [63]. Practically, a city can be considered "smart" when it can manage resources intelligently, becoming economically sustainable and energy self-sufficient and when it is attentive to the quality of life and needs of its citizens. It is a city in which interconnections are in the most disparate places, where objects exchange information with each other, developing intelligent infrastructures and smart communities. In a nutshell, a city that knows how to keep up with innovations and digital revolution [3].

According to Mckinsey Global Institute report [261], a Smart City, to be defined as such, must use technology and data to deliver a better quality of life. The research community agrees that a good quality of life should include improvement in the following areas:

- *economic prosperity*
  Smart Cities, through their innovative infrastructure, are fertile ground for technology start-ups and are able to attract knowledge and talent. This, consequently, attracts venture capital to the cities. Examples of cities such as London and New York have received an influx of investment capital thanks to their smart initiatives [258, 126].

- *greener environments*
  The environment is the core of every smart city strategy. Smart cities aim to reduce pollution and emissions through intelligent urban planning and transportation management, and optimized management of their resources [27, 213]. A city like Beijing has foraged projects as IBM Green Horizons [118] to identify pollution sources and to improve the city's air quality forecasting capability. The project Earthsense [74], developed in the UK, is exploring exciting new applications such as testing the impact of low emissions vehicles on reducing air pollution.

- *fast and zero-impact mobility*
  Improving the daily commuting time is one of the primary objectives for citizens and commuters to enhance their quality of life. Nowadays, cities that offer smart-mobility applications in place can reduce commuting time by as much as 20 per cent on average [87]. Multi-modal mobility solutions allow passengers great freedom by choosing between many transportation options. This, results in lower private fuel-vehicles usage since residents can opt for the transportation mode that better suits them at the moment. Among the many possible examples, the Polytechnic Institute of Beja promotes the use and the sharing of electric and regular bikes in academic communities thought the U-bike Project [38].

- *public safety*
  Together with other areas, public safety is another crucial parameter for citizens. ICT plays a key role in this application area. Indeed employing applications using, for example, real-time crime mapping [268] or statistical analysis to detect crime patterns and identify problematic zones [138] to predict the incidence of crime, the law enforcement agency can reinforce security and carry out more effective prevention. Many cities around the world are rapidly equipping themselves for such purposes. One example was the HunchLab project, a predictive technology solution in New York City, which significantly lowered crime [225].

Summarizing, many smart cities are being built all over the world. These cities pursue and implement many initiatives to promote greener and safer urban environments, fighting environmental pollution, with better mobility and efficient public services. All these initiatives are enabled by technologies like the Internet of Things (IoT) and big data analytics, that form the base of the smart city model. These technologies allow the generation of a large amount of data that could both feed more advanced and real-time services and will enable the development of new control and optimization policies.

1.1.1 Hints of IoT and Big Data in smart cities

Figure 1.2 depicts the three layers of "smartness" necessary to transform an ordinary city infrastructure in a "smart" environment [4].
In detail, the first layer, which represents the starting base, corresponds to the tech, which includes networks of sensors and connected devices which gather data. The second layer consists of the deployment of smart applications that process the raw data collected, translating it into alerts, insight, and actions. Then, the third layer involves widely adopting the system by the citizens.

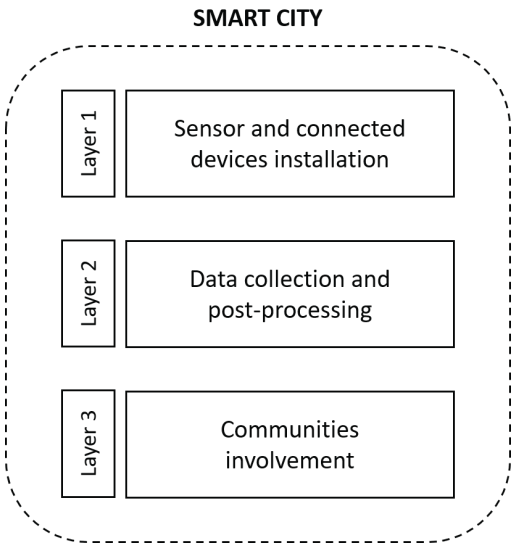


Figure 1.2: Smartness layers of Smart City

In this scenario, IoT is the essential technology on which modern smart cities base their initiatives and features. The "things" of the IoT (i.e. smart and connected devices, sensors and applications) collect the data that enables the technology solutions to be effective. In short, an interconnected network of smart devices collects a large amount of data that can be collected and processed to implement smart services. For example, in a smart city context, governance integrates ICT solutions to interconnect public services, at the same time engaging communities in local management, thus promoting cooperation. The Greater London Authority initiative [100] developed an open and universal platform to share data with local communities. In general, there are multiple examples of IoT applications in smart city contexts. Among the most important, we can find:

- *city lighting*
  Cities such as London and Quebec are installing smart street lighting [44]. This technology not only allows better management of energy consumption, but it also enables quick function as wifi hotspots, public surveillance, charging outlets for electric cars and phones, and even measure the air quality. This multitasking technology works both as a sensor and an actuator, providing services that better the quality of life of the citizens while collecting relevant data about the environment. Instruments such as streetlamps today have become multi-function devices, which interconnect to form a capillary network from which to draw numerous and valuable data.

- *waste management*
  As previously introduced, nowadays, many cities are applying advanced and

technological solutions to achieve a cleaner environment and reduce waste. For example, Songdo district in South Korea is reducing noise pollution, eliminating garbage trucks [151]. Indeed, many buildings have a smart garbage collection station where residents dispose of the trash bags, separated by organic and combustible. The station is equipped with sensors that detect when it is full. The trash is automatically sent through high-pressure pipes straight to the recycling centre.

- *public transport*
  Connected public transport—sensors send traffic data to the city transportation management software. Through the post-processing of this data, it is possible to provide real-time public services such as, for example, waiting times for the bus or train, alerting the system traffic congestion or delays [84].

As mentioned above, these are just some of the examples of using IoT technology in smart contexts.

That said, all the smart initiatives need big data analytics and post-processing to be effective. As demonstrated, the IoT devices generate massive datasets that must be analyzed and processed to implement smart city services [40]. Big data platforms, part of the city ICT infrastructure, have to sort, analyze and process the data gathered from the IoT. Smart cities are, by definition, data-driven. That is why big data and analytics play a key role. Indeed, the combination of big data analysis and smart city solutions helps cities to improve the management in critical segments, such as:

- *energy*
  To reduce pollution and energy waste, smart cities are experiencing the challenge of managing power usage efficiently [32]. This often translates into the implementation of efficient smart grids [96]. These smart grids allow city officers to analyze and administer the power consumption in real-time. Indeed, using data analytics and machine learning techniques, they can predict periods of heavy usage and plan the energy distribution accordingly. Furthermore, alternative energy production plants, such as solar energy plants, can be installed to balance energy needs. Moreover, thanks to the use of sensors, it is possible to carry out predictive maintenance and remote controls.

- *transportation*
  A smart transport infrastructure uses big data and IoT technologies to provide residents with access to faster and safer travel across the city. At the same time, it gives city authorities data about traffic flow, allowing them to manage the transit efficiently.

- *infrastructure*

Having a large amount of data available and being able to process them effectively helps cities to monitor and manage urban issues such as waste disposal, transportation, and saving resources. Improving infrastructures means being able to offer better services and thus increase the well-being of the environment, resources and ultimately citizens.

These are just some of the sectors of smart cities where the combination of IoT technologies and the management of the resulting data operate successfully. However, the lowest common denominator of all fields of application is the availability of a large amount of data. These data, adequately organized, processed and analyzed, allow to improve already existing services and expand towards new paradigms still to be explored. Undoubtedly one of the most promising tools in the field is the study of the available data to predict the future. Predicting the future means being able to implement new and more efficient control strategies.

1.2 Time-series forecasting: new control strategies

Understanding the past to discover the future. Since humans' origin, we are searching for laws that explain the behaviour of observed phenomena. There are many examples, and these range from how the heart works and the irregularity in a heartbeat to model the volatility of a currency exchange rate. In other words, humans are continually seeking to predict the future, and they do this by trying to understand and model the past [255].

Generally, if the underlying deterministic equations are well known, these can be solved to forecast the outcome of an experiment based on knowledge. On the contrary, if the equations are not known, to make a forecast, we have to find both the rules governing system evolution and the actual state of the system. For example, phenomena such as the motion of the pendulum carry within it the potential for predicting its future behaviour, based on the knowledge of its oscillations, without requiring insight into the underlying mechanism. In contrast, phenomena such as the air temperature of a building or blood glucose level are subject to many external factors that are difficult to model with conventional systems. These necessarily require the study and understanding of underlying mechanisms. Consequently, human being has developed two approaches to analyze an unfamiliar time series: i) "understanding" the series and ii) "learning" from the series. Understanding the series means to give an explicit mathematical understanding of how systems behave. Instead, learning means to adopt tools and algorithms that can emulate the structure in a time series. In both cases, the goal is to explain observations the so-called time-series analysis [255].

Usually, time series analysis has three goals [39]:

- *forecasting*
  Forecasting aims to accurately predict (in the best possible way and with the least likely error) the evolution of the system;

- *modelling*
  The goal of modelling is to find the description that accurately captures features of the behaviour of the system.

- *characterization*
  The goal of characterization, finally, attempts with little or no a priori knowledge to determine fundamental properties, such as the number of degrees of freedom of a system or the amount of randomness.

Before the 1920s, forecasting was done by merely extrapolating the series through a global fit in the time domain [255]. In 1927, Yule invented the autoregressive technique to predict the annual number of sunspots, thus beginning the time-series prediction of modern era [117]. This model can predict the next value as a weighted sum of previous observations of the series. Around 1980, two crucial developments occurred both enabled by the general availability of powerful computers that permitted much longer time series to be recorded, more sophisticated algorithms to be applied to them, and the data and the results of these algorithms to be interactively visualized. The first was the state-space reconstruction by time-delay embedding [255]. This is based on the idea from differential topology and dynamical systems to provide a technique for recognizing when deterministic governing equations have generated a time-series. The second was the emergence of the field of machine learning, typified by neural networks, that can adaptively explore an ample space of potential models [232]. With the shift in artificial intelligence (i.e. towards data-driven methods) and by means of tools able to record with orders of magnitude more data points than were available previously, time series were ready to be analyzed with machine-learning techniques requiring relatively large data sets. This has made it possible to develop more and more efficient and robust methods. Still, above all, it has made it possible to discover hidden and non-linear relationships between data, thus giving new food for thought.

Contexts such as smart cities are the first to benefit from such tools. They represent breeding grounds for a large amount of data. These data, for the most part, correspond to time series. By collecting this data and analyzing it using powerful time series analysis techniques, we are now able to discover new correlations and dependencies. This allows us to understand more and more the context (i.e. the city and all its actors) giving us the possibility to hypothesize and develop new control policies [63]. In the smart energy sector, for example, policies such as Demand/Response [226] and Demand Side Management (DSM) [93] have found room for continuous improvements. Thanks to these new paradigms, in a smart environment, energy resources can be shaped and managed according to needs. This

happens in all sectors, from energy to health and from mobility to safety. The perfect match between data availability and machine learning techniques, such as neural networks, give life to smart contexts. These, in turn, represent the ideal laboratory for continuous experimentation and technological development.

1.3 Machine learning for time-series forecasting

The desire to know and model the future has always guided the human being. Together, also the aptitude for learning tasks through experience is part of human growth. Indeed, when a human being is born, he knows nothing, and he is not able to provide for itself. But as he gains experience, he becomes more and more autonomous and able to perform even more complex activities every day. These two skills, mixed, through knowledge and technological development, gave rise to advanced techniques such as machine learning.

Machine learning includes a set of techniques based on algorithms and statistical models that allow machines to perform specific tasks without using explicit instructions, relying on patterns and inference instead [137]. Therefore these advanced techniques, combining statistics and computer science, make computers able to learn how to perform a specific task without having been programmed to do so, just like a man in his learning and growth phase. For example, for a human being, knowing how to distinguish a cat from a dog derives from the awareness of having acquired the characteristics of the entity "dog" and the entity "cat", learning its distinctive features (i.e. size, head shape, eyes, etc.) and, consequently, being able to recognize them.

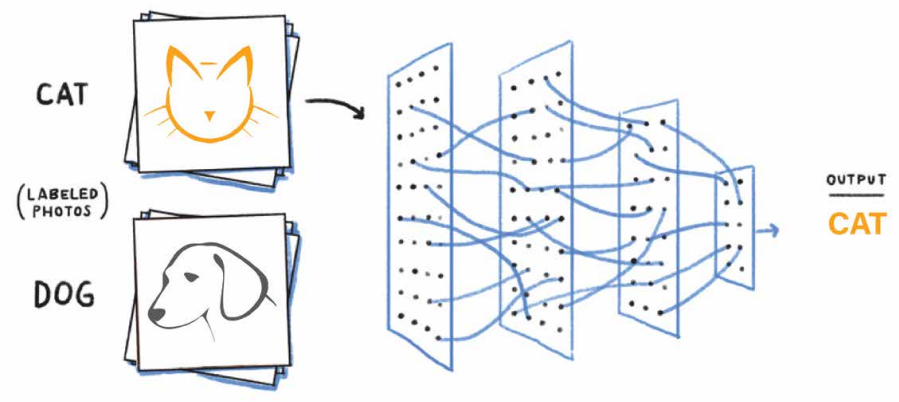


Figure 1.3: General schema of machine learning approach - Recognition of cats and dogs.

Figure 1.3 depicts a simplified example of an automatic learning approach based on the recognition of two class of animals through a series of filters. On the left are the

system inputs. In this example, the inputs are images with corresponding labels of the two animals. These images are used to train the system to discern between the two animals. The central part represents the machine learning system itself, which trough a set of tools (e.g. weights and filters) learns to recognize animals from images. Once trained, the machine learning system can discern, with a precision error, independently each animal image (i.e. cat or dog).

In an automatic learning system, a computer programmed to learn acquires statistical trends within the data. This ability allows it to recognize autonomously one class or other ones [12]. For example, by extracting knowledge from the image data of cats and dogs, the machine could understand on its own that cats have shorter muzzles and dogs appear in the wider variety of sizes. These characteristics can be represented in a dataset. A dataset is a set of data organized into essential mathematical elements such as vectors or matrices. In general, each dataset is characterized by one or more variables that describe the observable properties of an object, known as *features*. In machine learning systems, the list of features that describe an object is called *feature vector*. Moreover, the data represented by the samples, by definition, carry with them a load of information, which, however, must be optimized to avoid that more features give the same information, a phenomenon known as *redundancy*. This is disadvantageous because it can compromise the accuracy and reliability of these machine learning systems.

In the machine learning phase, the computer must identify trends within the features and establish a function by which future input will be recognized. At the same time, the programmer must address on which dataset to choose, or how to apply what he has learned, to make future decisions. In this phase, there will be some errors, but the more data the learning algorithm receives, the more effective its predictions or classifications can become. Hence, it is fundamental that the baggage of data from which the computer extracts knowledge is as large as possible to increase the generalization capabilities, but not too large to make the algorithm unable to apply the "laws" that govern it.

Machine Learning represents a momentous change in the computing paradigm. Knowledge is extracted from examples and experience, understood as the study of actual data, representing the culmination of an inductive process. There are currently three macro-categories of machine learning [173]:

- supervised learning;
- unsupervised learning;
- reinforcement learning;

Supervised learning aims to approximate the mapping function between input and output, so that, in case of new input data, is it possible to predict the output

variables for that data. Supervised learning deals with two types of problems: regression and classification [228]. Instead, in the case of unsupervised learning, the data used are not labelled, thus without prior knowledge. Consequently, data is only used to search for correlations, patterns or structures within the information itself. The most commonly used techniques are clustering or dimensionality reduction [129]. In clustering, the classification occurs through a process of discrete grouping data according to intra-class similarity and inter-class differences, while dimensionality reduction is the equivalent for continuous data. Finally, in a reinforced learning field, the system in the training phase receives feedback about its actions [240]. This type of learning methodology interacts with the environment to learn a series of combinations and activities that give the best results by storing and managing the previous experiences. Based on the actions considered virtuous or harmful, the algorithm adopts decisions (i.e. prediction or classification) according to the feedback received during the learning phase. Generally, a robust machine learning model requires:

- in-depth analysis of the available data;
- optimal data-set pre-processing;
- careful splitting of a training, test and validation set;
- choice of algorithms suitable for the task;
- model scalability.

Nowadays, the industry works with large amounts of data, considered a real economic asset [47]. Through specific machine learning techniques aimed at extracting knowledge of such data, often in real-time, companies can work and make decisions more efficiently, obtaining numerous advantages. Even high-tech players like smart cities get innumerable benefits. As introduced in the previous paragraphs, these advanced laboratories produce an enormous amount of valuable data. They provide especially time-related data. This considerable amount of data, combined with the latest machine learning techniques, opens up new horizons and possibilities for research and development. There are many fields of application of machine learning: government agencies, financial services, health care, energy, advertising, marketing and transport are some of the most developed sectors [14].

1.4 Scenarios

In recent years, the research about energy waste and pollution reduction has gained a strong momentum, also pushed by European and national funding initiatives. The primary purpose of this great effort is to reduce the effects of greenhouse

emission, climate change to head for a sustainable society. In this scenario, Information and Communication Technologies (ICT) play a crucial role in reducing energy consumption and moving forward to a more sustainable and smart society.

IoT devices are used in many contexts, to add smartness to cities, energy, and industrial processes. Their pervasiveness, combined with the recent development of machine learning techniques, allows collecting a large amount of information, enabling exceptional opportunities to create innovative modelling and optimization approaches.

My research focuses on developing stream data processing and machine learning methods, ranging from energy and environmental data and moving to data from Cyber-Physical Systems (CPS), creating and validating innovative modelling and control strategies in specific application case studies: i) Smart Building, ii) Renewables and iii) Smart Health.

Figure 1.4: Scenarios

Consequently, moving towards a smart and sustainable energy use, my research activities have mainly focused on the design and the optimization of innovative machine learning methodologies, by exploiting primarily neural networks, for the forecasting of time-series in Smart City context.

1.4.1 A common methodology

The central theme of my research is the design of innovative machine learning methodologies (i.e. neural networks), for the forecasting of time series in the Smart City scenario. Consequently, I was able to apply these methods to thermal modelling of Smart Buildings, to solar radiation forecasts for photovoltaic simulations and finally to smart health, by predicting the value of blood glucose.

Consequently, all the contexts in which I have had the opportunity to perform the experiments have a common denominator: the proposed base-line methodology. Indeed, all the studies were conducted following a bottom-up approach starting from the analysis and appropriate pre-processing of IoT data, to design neural models suitable for the type of dataset and its characteristics. Figure 1.5 shows the proposed common methodology.
The basic idea is to use real-data (i.e. from real-world demonstrators) to design

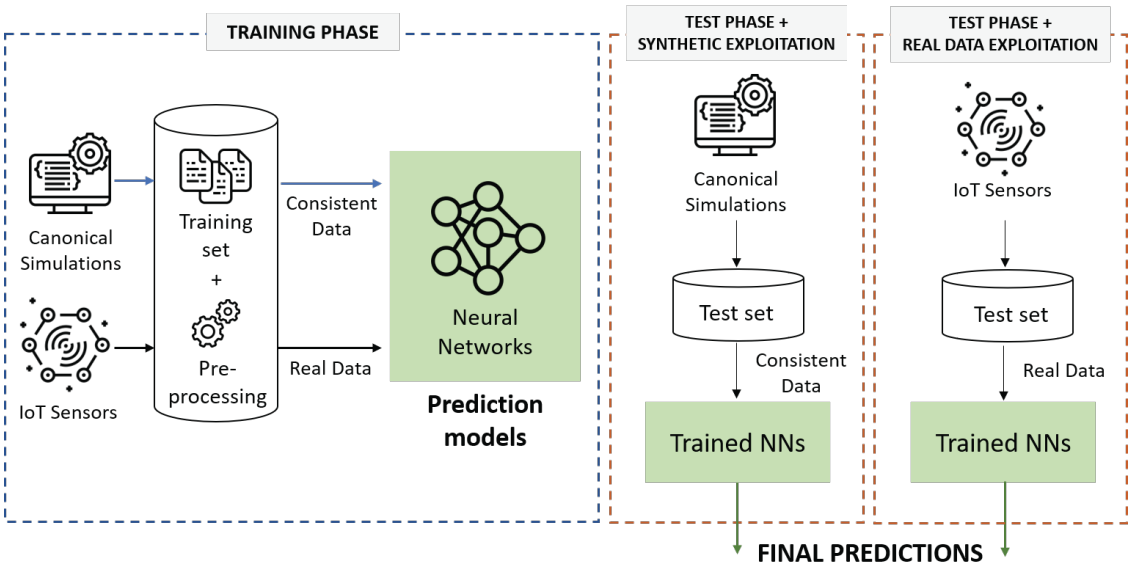


Figure 1.5: Base-line methodology

and optimize robust predictive models. When real data were not sufficient for effective characterization of the neural networks, we proposed methodologies to obtain realistic synthetic data (using the few real data available to validate the simulated ones). In this way we have been able to i) improve the canonical simulation methodologies already present and established in the literature and ii) to realize hybrid predictive models able to learn from synthetic data and to exploit the real ones (i.e. coming from IoT sensors) for the inference phase, therefore to obtain predictions. Furthermore, in cases such as historic buildings, where no historical data is present, and IoT sensors cannot be installed, we propose a full-synthetic methodology based entirely on realistic synthetic data by obtaining more accurate simulations and by overcoming the most famous simulation methodologies in literature. Besides, we investigated some Transfer Leaning techniques to specialize hybrid models on real data. This "information transfer" allows the hybrid models to transmute in a model entirely based on real data without performing the learning processes as a whole, by saving time and computational resources.

1.4.2 Smart Energy: solar radiation forecasting

Under the supervision and with the cooperation of my research team, we started by investigating neural techniques for the prediction of renewable energy production that represents a key topic in the smart city scenario. In detail, we proposed a methodology to perform short-therm photovoltaic energy predictions by producing a comparative analysis of state-of-art neural networks techniques. Then, we focus specifically on global horizontal irradiation (GHI) forecasting in short- and

mid-term, by comparing some state-of-art neural network, as deeply illustrated in Section 2.2. The experiments were conducted based on global horizontal irradiance data measured by a meteorological station located at Politecnico di Torino. The dataset, besides being used raw, has been subjected to two different pre-processing using the Clear-sky index and Tikhonov regularization respectively, to make easier the training phase of networks.

At the same time, we investigated the effectiveness of using exogenous inputs for short- and mid-term solar radiation forecasting. In detail, we identified a subset of relevant input variables for predicting GHI by applying different feature selection techniques to a broader set of variables. To assess the usefulness of the selected features, we evaluated and compared the prediction performance of the different state-of-art neural network. Finally, to demonstrate the effectiveness of using exogenous inputs for short-term solar radiation forecasting, we compared the prediction performance against models using only endogenous data. The results showed that the adoption of exogenous inputs could significantly improve forecasting performance. Figure 1.6 shows the generalization of processed flow.

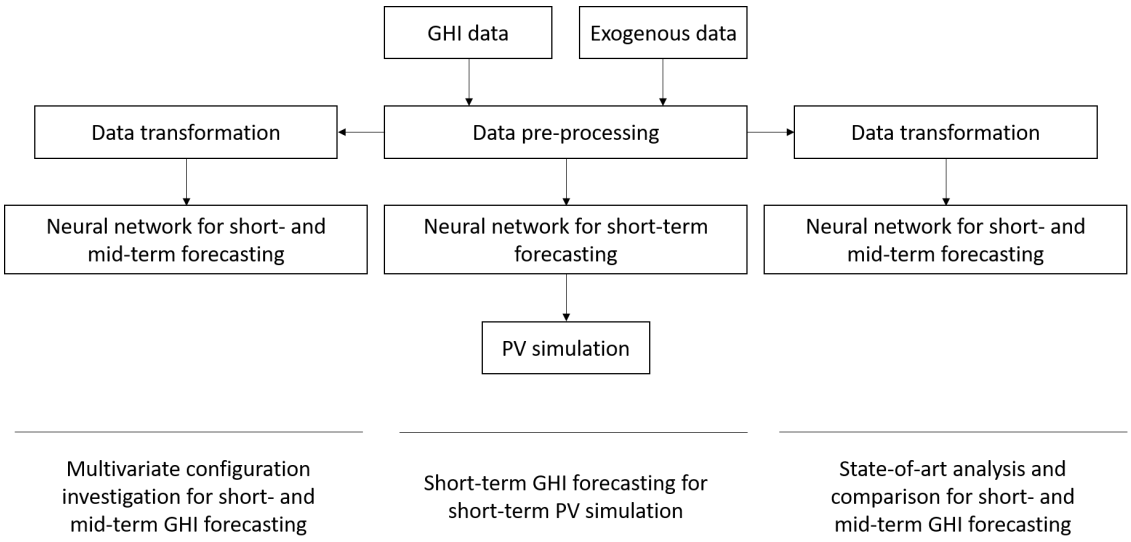


Figure 1.6: Solar Radiation forecasting flow chart overview.

1.4.3 Smart Building: indoor air-temperature forecasting

In the context of national EEB Cluster Project, we developed a comprehensive methodology to forecast indoor air-temperature trends in existing or newly constructed buildings (i.e. buildings of which no indoor air-temperature data are available). Also, in this scenario, our proposed methodology is based on the most modern and state-of-art-recognized neural networks techniques for time series prediction.

The aim is to provide a robust and generalizable methodology to enable new control policies for the thermal management of buildings w.r.t. individual rooms and the whole building. In view of this, the methodology first of all addresses the possibility of a lack of data, which are indispensable for reliable training and validation of neural networks. For this reason, we developed a technology able to produce a realistic synthetic dataset by simulating a Building Information model (BIM) of a real-world building with EnergyPlus and by exploiting real weather data instead of *Typical Meteorological Year* (TMY) data. In this way, we can propose a hybrid neural model where the training phase is completely based on simulated data, while the inference phase is conducted by exploiting real-world data (e.g. data provided by IoT devices). Therefore, our methodology is ready to be used immediately after the building is equipped with IoT devices, without needing any calibrations. Hence, it allows forecasting indoor air-temperature trends even in case of unavailable historical measurements. All this will enable us to state that this methodology can be successfully used both on existing buildings in which a historical dataset is available (using only real data) and on newly constructed buildings in which a historical dataset is not possible (using realistic synthetic data to model the construction and real data from IoT sensors for the neural network inference phase).

In parallel to this, we model, optimize and compare different neural networks to find the best architectures for short-, mid- and long-term indoor air-temperature forecast. Identified the most promising structure, we apply transfer learning techniques with the objective of further improve the prediction accuracy. In this way, buildings that do not have a historical dataset, gradually collecting data, can migrate from a model simulated on synthetic data to a model based on real data. This gradual transition allows us to avoid new simulations and computational waste by ensuring more and more accurate prediction.
Figure 1.7 shows the generalization of processed flow.

1.4.4 Smart Health: blood glucose forecasting

Lastly, I had the opportunity to apply machine learning also in the field of smart health developing an innovative multi-patient data-driven approach to blood glucose prediction addressing the problem of automated glucose level prediction leveraging multi-patient Continuous Glucose Monitoring System (CGMS) data. Specifically, the aim is to learn a generalizable glucose level prediction model from a multi-patient training set, using this model to predict the future glucose values of a new patient. This allows improving the usability of models that are solely based on the past recordings of the same patient. To achieve this, we have designed and compared different state-of-art neural network for time-series prediction.

Progressively, as second step, we focus on patient-specialized blood glucose prediction system always exploiting the data-driven approach. Thus, based on the

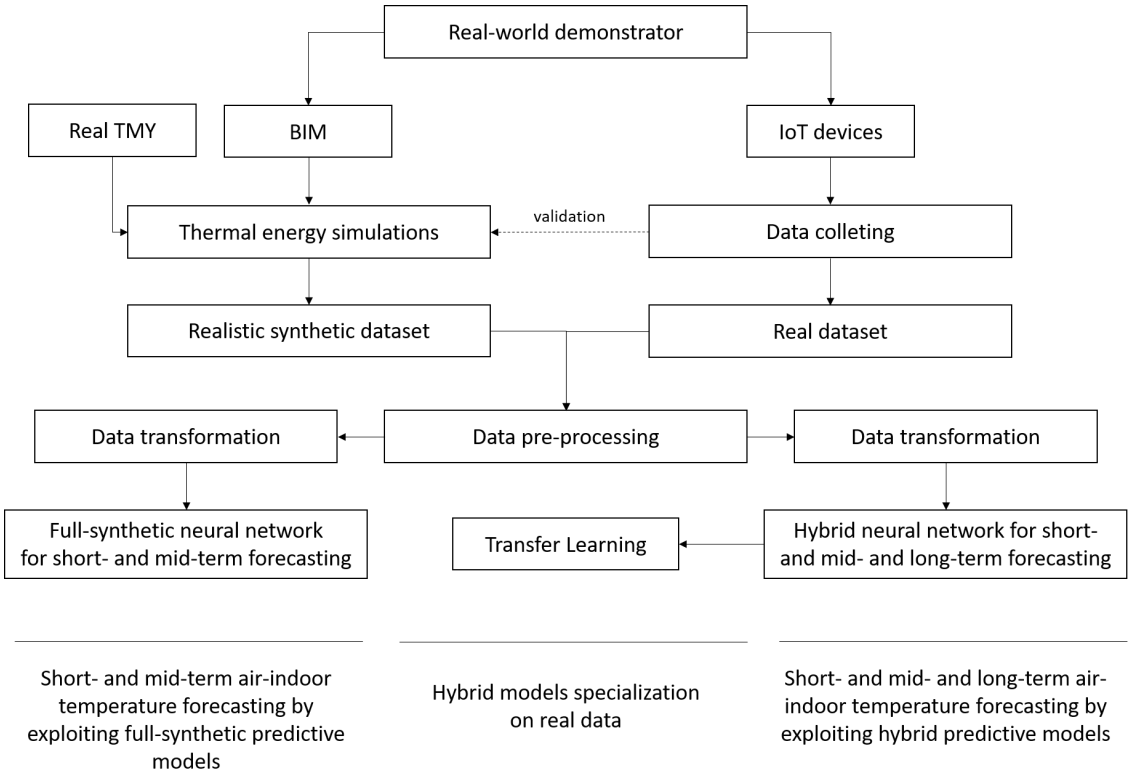


Figure 1.7: Building air-indoor temperature forecasting flow chart overview.

model previously developed, we designed and implemented our patient-specialized. Consequently, we tested and evaluated our solution capabilities and performances to improve the prediction accuracy, possibly on a much larger forecasting time horizon.

Demonstrate the potential of our model also in patient-specialized version, as future works, and we plan to apply some techniques, such as Transfer Learning, to personalize the multi-patients neural network according to glycaemia behaviours of each patient to monitor.
Figure 1.8 shows the generalization of processed flow.

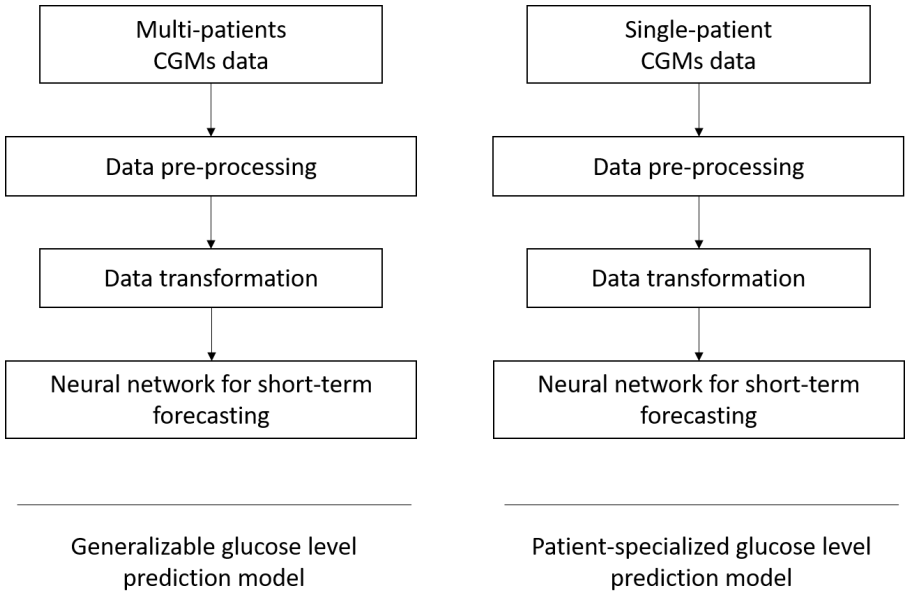


Figure 1.8: Glucose blood level forecasting flow chart overview (Type-I diabetes).

Chapter 2

Neural Networks

One of the most studied applications of machine learning is artificial neural networks (ANN). An ANN is a computing system inspired by biological animal brains [137]. Generally, an artificial neural network is composed of computational units, representing *neurons*, connected. The connections represent the *synapses*, in a biological brain. The neurons are usually organized in layers connected to each other. This structure can have a feedback connection, i.e. the possibility to propagate the error back to minimize it.

Neural Networks represent a point of contact between different disciplines such as neurology, psychology and artificial intelligence [181]. They emulate, through software and hardware devices, the behaviours of synaptic transmissions that occur in the brain during the learning and transmission processes of information (i.e. signals). Neural Networks are a branch of Machine Learning. For this reason, they have an automatic learning approach that, as the name suggests, is inspired by the fundamental element through which the brain processes information, the neuron.

A biological neuron is characterized by a structure consisting of 3 essential elements:

- *cellular body*
  Cellular body represent the core in which is present the nucleus that directs all the activity of the neuron;

- *axon*
  Single long fibre that transmits messages from the cellular body to the dendrites of other neurons;

- *dendrites*
  Fibres that receive messages from other neurons and forward these messages to the cellular body.

Figure 2.1 depicts a schematic representation of a neuron structure.

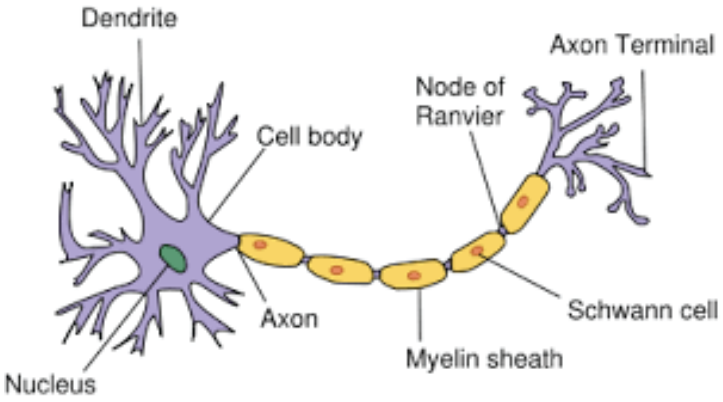


Figure 2.1: Representation of a biological neuron structure.

Neurons receive electrical inputs from dendrites, they absorb energy and release it through axons (i.e. the output channels) to other nearby neurons, by transmitting the information as an electrical impulse through specialized structures called *synapses*. The signal can favour the activation of the receiving neuron. In this case, the synapse is excitatory, and in contraction, it is defined as inhibitory. Only after a while, called *refractory time*, the neuron can generate a further impulse, clearly showing the binary nature of this critical, how simple, biological element [65].

Similarly, an artificial neural network is composed by the interconnection of simple processing units, the so-called *artificial neurons*, which have the ability to extract knowledge from data and store it through weights, the *synaptic weights*. Generally, ANNs are distinguished by two important characteristics:

- ANNs were conceived with the ability to approximate any function, both linear or non-linear. Indeed, the linearity or non-linearity depends only on the learning process and the activation functions, which can be linear or non-linear;
- ANNs can update synaptic weights during the training process. The samples are administered to the neural network, and the weights are calculated and modified to reduce the minimum distance between the target and the actual output. In this way, the ANN can build an input-output mapping of the system, starting from the examples used in the training phase.

Figure 2.2 shows a biological neuron observed trough a light microscope.
Nowadays, researchers have developed multiple types of neural networks, different in structure and purpose, from simple feed-forward networks to more complex recurrent structures. Some of these structures, especially those that specialize in predicting time series, will be described in more detail further on.

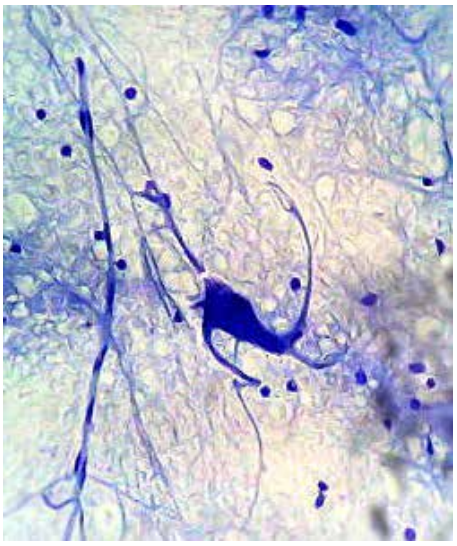


Figure 2.2: Biological neuron. Author: F.Castets

2.1 A bit of history

The history of artificial neural networks starts in 1943 when McCulloch and Pitts first proposed the concept of the *artificial neuron*, which is the basic component of every ANN [166]. In 1949 Hebb developed a theory for learning based on neural plasticity, now referred to as "Hebbian learning", which is a type of unsupervised learning [112]. Hebbian says that a synapse between two neurons is strengthened when the neurons on either side of the synapse (input and output) have highly correlated outputs. In other words, when an input neuron fires, if it frequently leads to the firing of the output neuron, the synapse is strengthened. Following the analogy to an artificial system, the tap weight is increased with a high correlation between two sequential neurons. Mathematically, this concept is described by the Equation 2.1 as following:

$$w_{ij}[n+1] = w_{ij}[n] + \eta x_i[n] x_j[n] \tag{2.1}$$

where η is a learning rate coefficient, and $x_i[n]$ and $x_j[n]$ are, respectively, the outputs of the ith and jth elements at time step n.
Based on this hypothesis, a first artificial neural network was proposed in 1954 by Farley and Clark [85]. In 1958 Rosenblatt created the perceptron [219]. In 1969 Minsky and Papert discovered that perceptrons could not process exclusive-or circuits. Also, they argued that computers were not powerful enough to handle the computations needed to simulate a large network [172]. This resulted in a setback for neural network research. Interest in artificial neural networks was revived by the development of the backpropagation algorithm, in 1974 [257]. This algorithm

allowed to train multi-layer networks efficiently. The exponential growth of computer processing power in the last decades greatly contributed to the development of this field of research.

Nowadays artificial neural networks are used for many different tasks in a wide range of fields, including facial identification, speech recognition, machine translation, image classification, game playing, and of course time series forecasting.

2.1.1 The Perceptron

Frank Rosenblatt proposed the Perceptron in 1958. In his research work [219], the author introduces the first neural network scheme, called *Perceptron*, the forerunner of current neural networks for the recognition and classification of forms, to provide an interpretation of the general organization of biological systems. The Rosenblatt probabilistic model is aimed at the mathematical analysis of functions such as information storage, and their influence on pattern recognition. Figure 2.3 depicts a perceptron and its components.

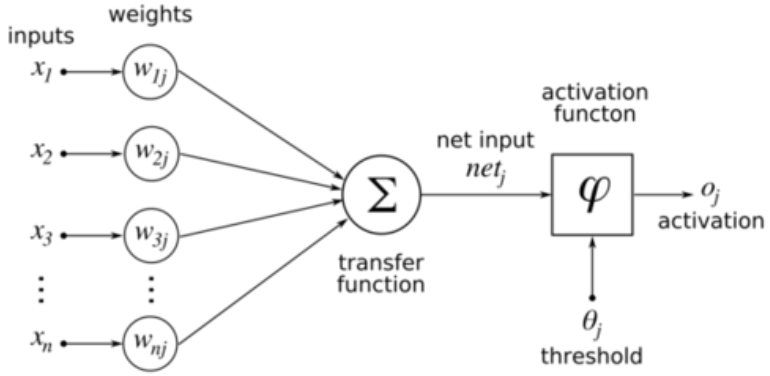


Figure 2.3: Rosenblatt perceptron schema.

The perceptron was described as an entity with an input and an output layer and a learning rule based on error minimization (i.e. error back-propagation function) which according to the evaluation of the actual output of the network concerning a given input alters the weights of the connections (synapses) as the difference between the actual and the desired output. As a linear classifier, the single-layer perceptron is the most straightforward feed-forward neural network.

Multilayer Perceptron

The Multilayer Perceptron (MLP) is an extension of the simplest perceptron. Like his predecessor, the MLP is composed of units (i.e. nodes or neurons) organized in a layer of inputs, one or more hidden layers and an output layer. Figure 2.4 depicts a general schema of a MLP.

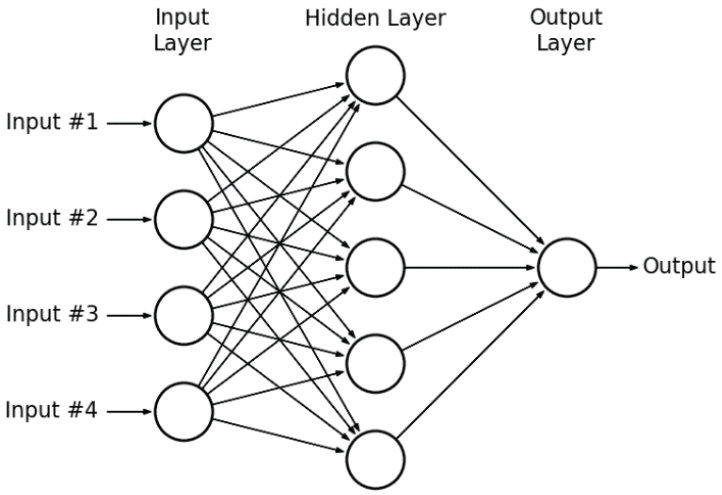


Figure 2.4: Multilayer Perceptron schema.

The MLP is a feed-forward architecture with fully connected layers. Connections between units are characterized by adjustable parameters called weights. This refers to the strength of a link between two nodes [137]. Each neuron computes a function of the sum of the weighted inputs, the *activation function*. The functional model is given by the following Equation 2.2:

$$\hat{y}_i(w, W) = F_i(\sum_{j=1}^{q} W_{ij} h_j + W_{i0}) = F_i(\sum_{j=0}^{q} W_{ij} f_j(\sum_{l=1}^{m} w_{jl} u_l + w_{j0}) + W_{i0}) \tag{2.2}$$

Weights are specified by the matrices $W = [W_{ij}]$ and $w = [w_{jl}]$; where W_{ij} scales the connection between the hidden unit j and the output unit i. w_{jl} instead scales the connection between the hidden unit j and the input unit l. The corresponding biases are W_{i0} and w_{j0}. These weights are vectorized in a vector θ. The input units are represented by the vector $u(t)$ while the vector h represents the hidden neuron outputs. The parameters are determined during the *training process*, which requires a *training set* Z^N, composed of a set of inputs, $u(t)$, and corresponding desired outputs, $y(t)$, specified by:

$$Z^N = [u(t), y(t)], \ t = 1, ..., N \tag{2.3}$$

The training phase allows to determine a mapping from the set of training data to the set of possible weights:

$$Z^N \rightarrow \hat{\theta} \tag{2.4}$$

the network can predict $\hat{y}(t)$ that can be compared to the true output $y(t)$. The

prediction error approach is instead based on the introduction of a measure of closeness in terms of a mean square error criterion, as specified by:

$$V_N(\theta, Z^N) = \frac{1}{2N}\sum_{t=1}^{N}[y(t) - \hat{y}(t|\theta)]^T[y(t) - \hat{y}(t|\theta)] \tag{2.5}$$

Weights are then found as:

$$\hat{\theta} = arg_\theta min V_N(\theta, Z^N) \tag{2.6}$$

by some kind of iterative minimization scheme:

$$\theta^{i+1} = \theta^i + \mu^i + f^i \tag{2.7}$$

where θ^i specifies the current iteration, f^i the search direction and μ^i the step size.

In the study of systems based on time-series, MLP is one of the most powerful and performing ANNs [65] family.

2.2 Neural Networks for time-series forecasting

Nowadays, one of the most effective methods for time-series prediction is based on neural networks [176]. This is due to their versatility and their ability to model a wide range of systems reducing development time and offering better performances [255]. As previously introduced, there are many different types of neural networks. These differ in architecture and purpose. Equally, there are different types of specialized neural networks for time-series prediction. In this Section will be described as the most commonly used neural network for time-series forecasting. These also correspond to the architectures adopted for the realization of the methodologies which will be presented in detail in the Chapter 3, 4 and 5.

2.2.1 Feed-Forward Neural Networks

The Feed-Forward Neural Network (FFN), or FFNN, was the first and simplest devised artificial neural network designed on history. As previously introduced, the Multilayer Perceptron is a fully-connected FFN. Generally, a FFN is composed of at least three layers: i) an input layer, ii) one or more hidden layers and iii) an output layer. Each layer is composed of nodes (or neurons). The adjective "fully connected" means that every node of each layer is connected to all the nodes of the next layer. An example of a MLP with one hidden layer is shown in Figure 2.5.

The output of each node is calculated using an activation function applied to the weighted sum of the inputs. The activation function is usually a nonlinear one (e.g. a common choice being the hyperbolic tangent (tanh)). Considering a MLP

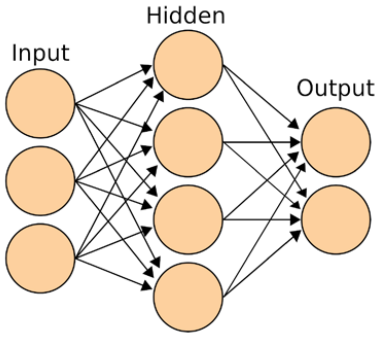


Figure 2.5: Feed-Forward Neural Networks.

with n inputs, one hidden layer with m units and one output, the output of the network can be modelled as follows:

$$\hat{y} = F(\sum_{i=1}^{m} W_i f(\sum_{j=1}^{n} w_{ij} u_j + w_{i0}) + W_0) \tag{2.8}$$

In equation 2.8, F and f are the activation functions for the output and hidden layer respectively, W_i and w_{ij} are the weights between hidden and output layer and between input and output layer, W_0 and w_{i0} are the biases, and u_j are the inputs. Generally, the training process consists in finding the W_i and w_{ij} matrices that minimize the error between the expected output y and the predicted values $\hat{y}$ (i.e. the output of the network), according to some metric. This process is called "supervised learning": the network is presented with a set of inputs (training set) and the expected outputs for each input vector. The network output is compared to the expected one, and the weights are adjusted using an algorithm of the backpropagation family. The Levenberg-Marquardt algorithm [143, 157] is one of the most widely used in the literature.

Non-linear Autoregressive Neural Network

Non-linear Autoregressive Neural Network (NAR) extends traditional linear autoregressive model [147] in that it is entirely distribution-free. Hence, it can be applied even to time series with intrinsic nonlinearities, such as sudden spikes and brief, transient periods [185].

A NAR model computes the value of a signal y at time t using n past values of y as regressors (also called feedback delays), as follows:

$$y(t) = f(y(t-1), y(t-2), ..., y(t-n)) + e(t), \tag{2.9}$$

where f is an unknown non-linear function and $e(t)$ is the model approximation error at the time t.

Function $f(\cdot)$ is computed by optimising a multi-layered neural network, whose topology is represented in Figure 2.6

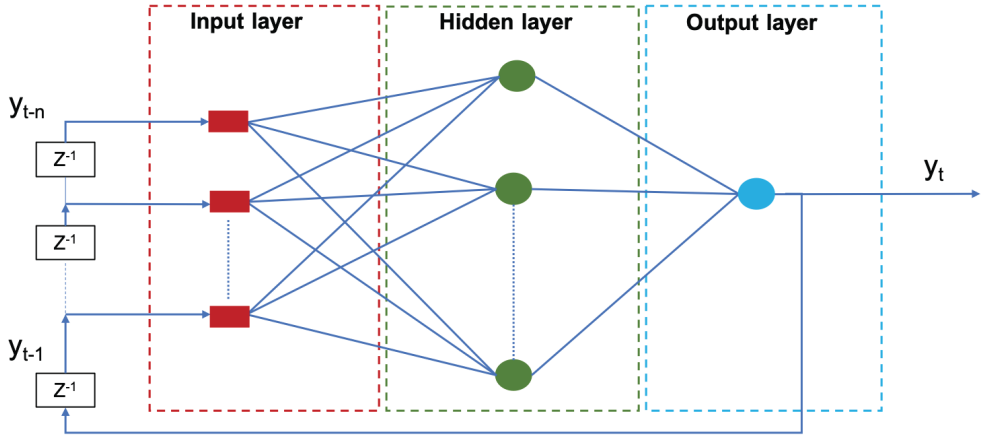


Figure 2.6: NAR topology.

At the time t, the neural network is fed with the n past values of the signal y. Such inputs are transferred through multiple layers of neurons, where each neuron is a simple computational unit characterised by a set of weights W (one per each input connection j), a bias b and an activation function h. Hence, the output of a neuron i is computed as follows:

$$out_i = h_i \left(\sum_j w_{i,j} \cdot in_{i,j} + b_i \right) \tag{2.10}$$

where the optimal values of $w_{i,j}$ and b_i are computed by back-propagation on the training set [185].

2.2.2 Recurrent Neural Networks

Unlike Feed-Forward neural networks, where the information flows just in one direction (i.e. from the input to the output) and each layer is characterised by a different set of parameters, Recurrent Neural Networks (RNN) are characterised by multiple layers of recurrent units all sharing the same parameters, with loops allowing the information to propagate back to the same computational units, as depicted in Figure 2.7). This type of neural architecture enables a mechanism in which the decision reached at time step $t-1$ affects the decision it will reach one moment later at time step t [155]. Thus RNNs have two sources of input, the *present* and the *recent past*, which combine to determine how they respond to new data. By doing so, each computational step takes into account not only the current input at time t but also what was learnt from the previous inputs. This mechanism

is called *memory* [170]. Adding memory to neural networks means extrapolate information from the sequence itself. RNNs, unlike FFNs, use this information to perform tasks. As result, this makes recurrent networks particularly suitable for time-series predictions [71].

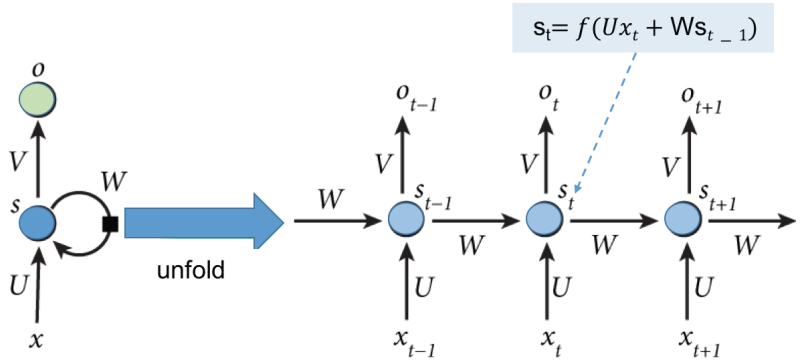


Figure 2.7: Recurrent Neural Network unit.

Figure 2.7 depicts a detail of an unfolding RNN unit. As shown, x_t is the input at time step t, s_t is the hidden state, also called memory of the network (initialised to 0), f is a non-linear activation function and, finally, o_t is the output. The process of carrying memory forward mathematically is expressed as follows:

$$s_t = f(Ux_t + Ws_{t-1}) \tag{2.11}$$

where the hidden state is a function of the input at the same time step x_t, modified by a weight matrix U added to the hidden state of the previous time step $h_t - 1$ multiplied by its own hidden-state-to-hidden-state matrix W. The weight matrices are filters that determine how much importance to accord to both the present input and the past hidden state. The error they generate will return via backpropagation and be used to adjust their weights until error cannot go any lower.

Nonlinear Auto-Regressive Moving Average Neural Network

Nonlinear Auto-Regressive Moving Average neural network (NARMA) belongs to the family of Nonlinear Autoregressive Moving Average Exogenous Model (NARMAX) [58] and represents a generalization of the NAR model. Unlike the predecessor, that realizes a feed-forward network where a predictor has feedback when the regressors are selected, the NARMAX family is characterized by the following equation:

$$y_t = F(y_{t-1}, y_{t-2}, y_{t-3}, ..., u_t, u_{t-1}, u_{t-2}, u_{t-3}, ...) + C(q^{-1}) + \varepsilon_t \tag{2.12}$$

where y_t and u_t are the variable of interest and the externally determined variable at time t, respectively; ε_t is the prediction error; C is a polynomial in the backward shift operator expressed as:

$$C(q^{-1}) = 1 + c_1 q^{-1} + ... + c_{nc} q^{-nc} \tag{2.13}$$

Consequently, the past prediction errors depend on the model output, and they are able to establish feedback. In conclusion, the significant difference between NAR and NARMA is that NARMA is a recurrent model [155]. Thus, NAR has a predictor without feedback, while NARMA has feedback through the choice of regressors. Hence, future network inputs will depend on present and past network outputs. This might lead to instability of the ANN itself, and it can be challenging to determine whether or how the predictor is stable. To avoid instability, NARMA architecture uses a linear MA-filter to filter past residuals. This is a Low Pass FIR (Finite Impulse Response) filter, commonly used for smoothing an array of sampled data/signal. The filter takes a set of inputs at the time, and it computes the average of those samples and produces a single output [153].

Long-Short Term Memory

A well-known limitation of classic RNN architectures, where the parameters of a large number of layers are learnt by backpropagation, is the instability of long-term predictions due to either vanishing or exploding gradient problems [95]. Such issues arise during the training of a deep network, when the error gradients are propagated back in time to the initial layer, going through continuous matrix multiplications. As the gradients approach the earlier layers, if they have small values (much lower than 1), they shrink exponentially until they vanish, making it impossible for the model to learn. Likewise, very large gradient values (much higher than 1) get larger and larger and eventually crash the model.

The Long Short Term Memory networks (LSTM) are specifically designed to overcome this limitation [114]. Indeed, it represents an evolution of the classic recurrent models. While the structure of the system is fundamentally very similar to RNN, a different function is used to compute the hidden state [188]. In the RNN, repeating modules consist of a single layer, typically with a tangential activation function. The memory in LSTMs is instead implemented as *cells* (see Figure 2.8), where specific gating functions decide whether the information needs to be kept or erased from memory at each time step. The key to this process is the cell state (in the diagram of Figure 2.8, the green horizontal line), which conveys information to the next cell. Gates regulate the power of either removing or adding information to the cell state (respectively, input, output, and forget gates) consisting of sigmoidal activation functions coupled with pointwise multipliers. Each sigmoid output's values in a 0 to 1 range, modulating how much of the corresponding signal should be let through [188]. Indeed, no matter how deep the network is, through its

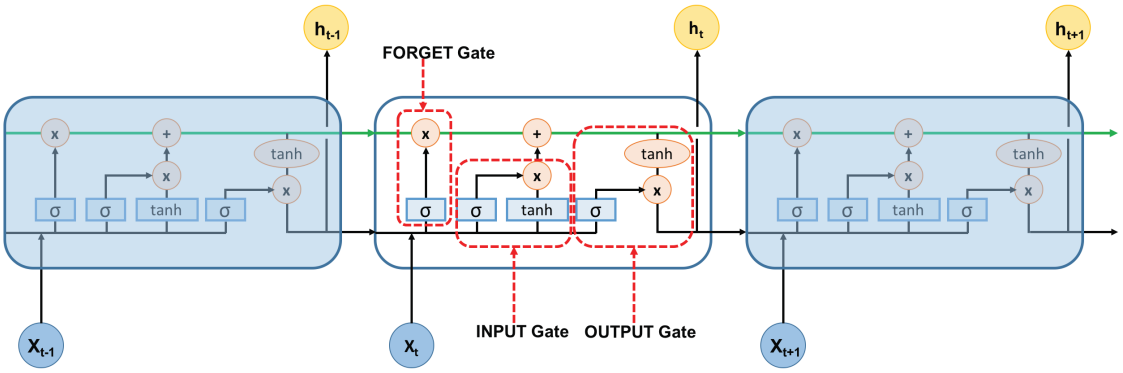


Figure 2.8: Long Short Term Memory topology, where x_t is the input and h_t is the output of a cell.

mechanisms, the LSTM network can remember values that are passed through gates all in 1 state. This makes the LSTM model intrinsically immune to vanishing and exploding gradient. In detail, the *forget gate* controls which information should be removed from the long-term state. Then, the *input gate* controls which data should be added to the long-range state. Finally, the *output gate* controls which information should be read and output at the current time step. Figure 2.9 shows

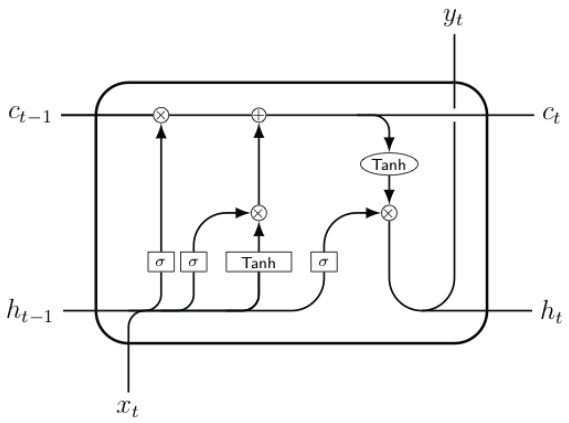


Figure 2.9: In-depth details of a LSTM cell mechanisms.

the detail of the mechanisms within an LSTM cell. Mathematically, LSTM cell computes its short-term state h_t, its long-term state c_t and its output y_t at each

time step t basing the following set of equations expressed in vectorial:

$$\begin{aligned} i_t &= \sigma(W^{xi}x_t + W^{hi}h_{t-1} + b_i) \\ f_t &= \sigma(W^{xf}x_t + W^{hf}h_{t-1} + b_f) \\ o_t &= \sigma(W^{xo}x_t + W^{ho}h_{t-1} + b_o) \\ g_t &= tanh(W^{xg}x_t + W^{hg}h_{t-1} + b_g) \\ c_t &= f_t \otimes c_{t-1} + i_t \otimes g_t \\ y_t &= h_t = o_t \otimes tanh(c_t) \end{aligned} \tag{2.14}$$

where W^{xi}, W^{xf}, W^{xo} and W^{xg} are the weight matrices of the connections to the input vector x_t, W^{hi}, W^{hf}, W^{ho} and W^{hg} are the weight matrices of the connections to the previous short-term state vector h_{t-1} and b_i, b_f, b_o and b_g are the bias terms.

Echo State Network

The Echo State Network (ESN) is a RNN that belongs to reservoir computing class [223]. ESN is composed by an input layer, a recurrent hidden layer called *reservoir* and an output layer. Figure 2.10 depicts in details the topology of a

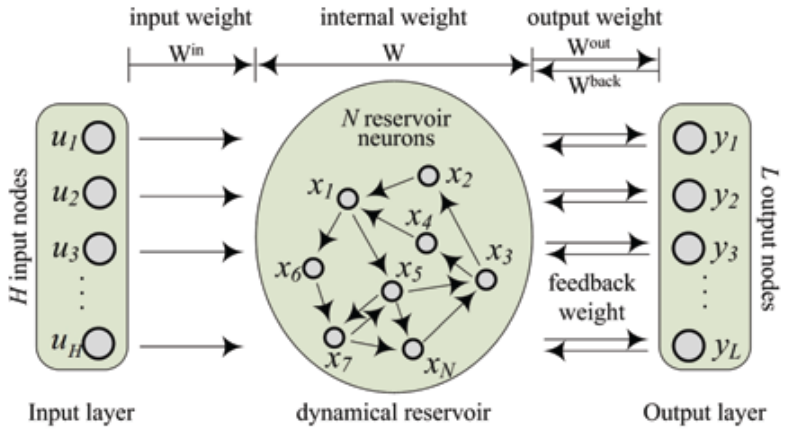


Figure 2.10: Echo State Network topology.

generic ESN [145]. The core of this model, the reservoir, consists of a large set of randomly connected neurons that constitute the hidden layer of the network. Generally, the main idea behind this model is to have a fixed, random, sparsely connected recurrent layer, and a readout layer, connecting the reservoir to the output. In a simple vanilla ESN architecture, these output connections are the only trainable ones. However, it is also possible to add direct trainable connections from input to output, bypassing the reservoir, and also feedback connections from output to reservoir [123].

The state update of a standard discrete-time ESN with N reservoir units, K inputs and L outputs is governed by the following Equation 2.15:

$$x_{t+1} = f(Wx_t + W^{bf}y_t + W^{in}u_t) \tag{2.15}$$

where f represents the state update activation function, $W \in R^{N\times N}$ is the reservoir weight matrix, x_t is the reservoir state, $W^{bf} \in R^{N\times L}$ is the feedback weight matrix, y_t is the output, $W^{in} \in R^{N\times K}$ is the input weight matrix and u_t is the input. The output of an Echo State Network is given by the following Equation 2.16:

$$y_t = g(W^{out}[x_t, u_t]) \tag{2.16}$$

where g is the output activation function, $W^{out} \in R^{L\times(N+K)}$ is the output weight matrix and $[x_t, u_t]$ is the concatenation of the reservoir and the input states. The forward connections W^{out} from the reservoir nodes to the output units are the only parameters that are learned during the training process. At the same time, the other weights are randomly initialized at the creation of the network and remain fixed.

For the ESN to work correctly, the reservoir must respect the Echo State Property (ESP), which states that the effect of initial conditions x_0 should progressively vanish as the length of the input sequence goes to infinity. The necessary condition for the ESP to hold is that the spectral radius λ, which is defined as the largest absolute eigenvalue of the matrix W, must be less than 1. In practice, it has been shown that the ESN can work properly also for spectral radius λ higher but close to 1. However, this condition is neither sufficient nor necessary. Stricter requirements have been determined in [269].

Another key feature of the ESNs models is that they are straightforward to train, unlike other recurrent neural networks. This is due to the recurrent fixed layer. Indeed, this greatly simplifies the training process, allowing a significant reduction in computational costs.

2.2.3 Convolutional Neural Networks

Firstly researched by Hubel and Wiesel [116], the Convolutional Neural Network (CNN) is a regularized version of multilayer perceptron inspired by biological process. Nowadays, this kind of ANN represents the state-of-art for image classification and pattern recognition. Indeed, it has full applications in image and video recognition, image classification and natural language processing [136].

A CNN consists of an input layer and an output layer with multiple hidden layers in between. The hidden layers are typically a set of convolutional layers with usually a Rectified Linear Unit (ReLU) as activation function, followed by a flatten layer and fully connected layers (i.e. dense layers) Figure 2.11 shows a topology of a general CNN. The convolutional layer represents the core building block of a

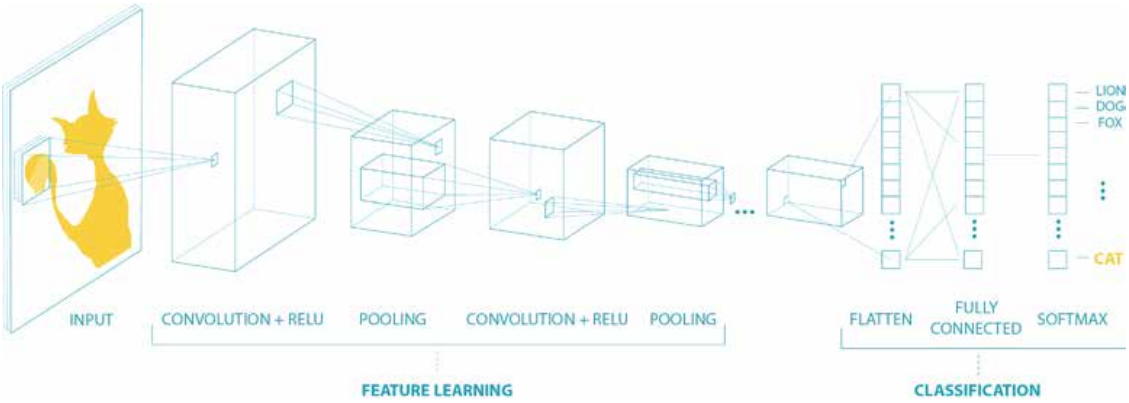


Figure 2.11: Convolutional Neural Network topology.

CNN. This layer is composed by a set of filters, applied to the full, that take a subset of the input data by sweeping over it and creating features maps. To obtain a new feature, the resulting feature maps are convolved with a learned kernel, and the results are passed into a nonlinear activation function (e.g. ReLU). The flatten layer stays between the convolutional and the dense layer. It transforms a two-dimensional matrix of features into a vector that can be fed usually into a fully connected neural network. The dense layer takes all neurons in the previous layers and connects them to every single neuron of the current layer (typically this is one or more fully connected layers). Then, an output layer follows the last dense layer.

1D-Convolutional Neural Network

As previously specified, CNNs are particularly suitable models for image processing applications as image recognition or classification. To supply applications based on time series, researchers have recently developed a special kind of CNN which was specially designed for modelling one-dimensional inputs, the 1D Convolutional Neural Network (1D-CNN) [132].

Nowadays, 1D-CNN models achieve top performance in several signal processing applications [132], thanks to its excellent capability in extracting meaningful features from sequential data. In detail, the element responsible for the feature selection process is the *filter*, which is a feature detector that learns to recognize a specific pattern in the input data. As shown in figure 2.12, the filter slides across the input sequence and activates the corresponding neuron in the feature map whenever a match for that pattern is found. The feature map also indicates where that specific pattern was found, also providing information about the feature position. The process of sliding a filter across the input data is called *convolution* and consists of a series of multiplications between the filter and the input values. In general, convolutional layers are composed of several filters working over multiple input channels, enabling the network to recognize multiple patterns in the input

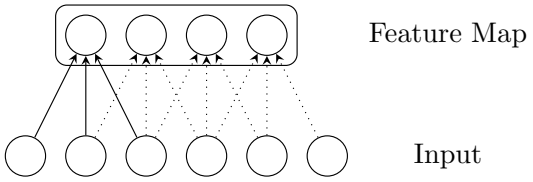


Figure 2.12: Filter representation.

data. Moreover, convolutional layers can be stacked together to form a deep architecture, where each layer builds upon the features detected by the previous layer hierarchically. Convolutional layers may be followed by a *max pooling layer*, which is responsible for reducing the amount of information received from the previous layer. The pooling layer applies a sliding window over the feature map, selecting only the max value of each block. Finally, one or more fully connected layers are added at the end of the network to interpret the features extracted. Figure 2.13 shows an example of a complete generic 1D-CNN architecture.

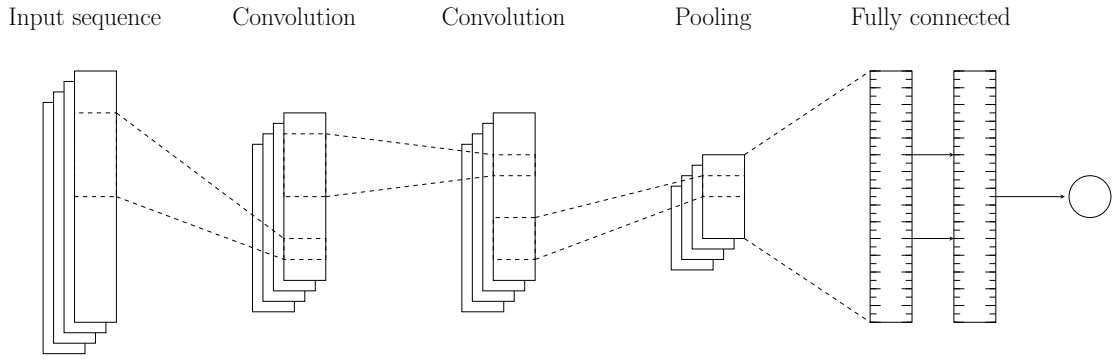


Figure 2.13: 1D-Convolutional Neural Network topology.

2.3 System Identification

Generally, when approaching applications based on neural networks, it is common practice to follow a strict and precise procedure. This is to be effective and not get lost in the meanders of the tools available today. In the book [185], the author, prof. Nørgaard suggests a comprehensive procedure to identify a dynamical system. This procedure consists of four steps as detailed in Figure 2.14.

The first step, the *Experiment*, corresponds to the problem analysis. Generally, the researcher approaches the problem by identifying the main characteristics and expected goals. At the same time, it is executed the sampling and data collection to collect a reliable data-set. Indeed, in neural network applications, once the scope

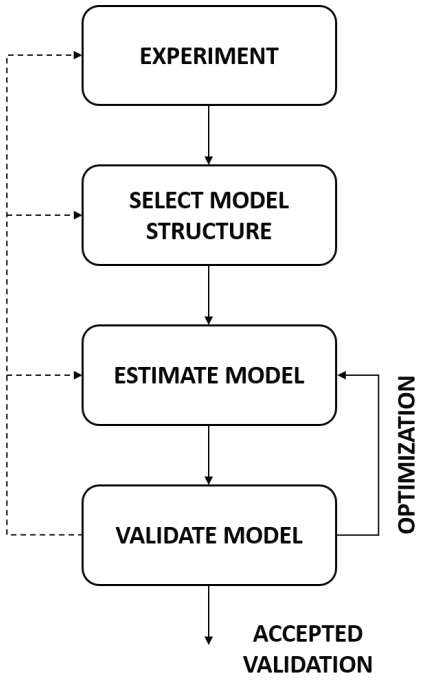


Figure 2.14: System identification procedure.

has been identified, an adequate amount of data is needed. Generally, a higher number of data allows better forecasting performances [233]. Then the available data must be divided into two different data-sets: the training set and the validation set, respectively. These data-sets are used in the training and validation phases of the neural network, which are the *Estimate Model*, and the *Validate Model* steps in Fig. 2.14, respectively.

The second step is the *Model Structure Selection*. This step allows identifying the correct architecture model to use [185]. This step is crucial because the use of the wrong instrument can affect the expected results [48]. At this aim, the system regressor must be studied. In mathematical modelling, these regressors identify independent variables able to influence the dependent variables. In time series, then, these regressors represent previous samplings concerning the predicted ones [176]. Consequently, the best neural structure can be chosen.
In this step, once the network model and the number of regressors are identified, the network is first implemented and then trained. This step is called *Model Estimation*. In time series scenario, training a neural network is needed to provide:

- the vector containing desired output data;
- the number of regressors to define the prediction;

- the vector containing the weights of both input-to-hidden and hidden-to-output layers
- the data structure containing the parameters associated with the selected training algorithm.

Finally, the training phase produces a *training error*, which represents the network performance index [233].

The *Model Validation* step validates the trained network. Generally, validating a network allows evaluating its capabilities [171]. In time-series predictions, the most common validation method consists of analyzing the residuals (i.e. prediction errors) by cross-validating the test set. This method allows performing a set of tests, including also the auto-correlation function of the residuals and the cross-correlation function between controls and residuals. This analysis provides the *test error* [233], that is an index considered as a generalization of the error estimation. This index should not be too high compared to training error. If this happens, the network could over-fit the training set.

Generally, if the network is overfitting the training set, the selected model structure contains too many weights. The structure is then subjected to the *Network optimization and final validation*. The process requires to return in the *Estimate Model* step to change and redefine some structural parameters by optimizing the whole architecture. For this purpose, the unnecessary weight must be pruned according to the Optimal Brain Surgeon (OBS) strategy, that represents one of the essential optimization strategies [104]. Consequently, once the new weights are given, the network architecture must be re-validated.

The paths going from the validation block to the previous stage (i.e. the dotted line) indicate that the whole procedure is executed iteratively. For example, it can be necessary to go back in the procedure to determinate one or more different models or, in the worst case, even redo the experiment. Generally, leading back to model estimation means that the problem has several local minima and find the global minimum is not easy. The feedback path also covers up an augmentation of the criterion called *regularization* or *weight decay*. Instead, leading back to model structure selection means that the neural structure is not fit for purpose. Indeed, this is usually oversized. Thus, it is common to apply the popular strategy of *pruning*. Generally, an initial model structure that is large enough to describe the system is determinate, and it is then reduced gradually until the optima structure is achieved. Finally, leading back to the experiment phase implies that certain regimes of the operating range are not reflected in the dataset, thus it is necessary to do additional experiments to acquire more information about the missing regimes.

2.4 Forecast horizons configuration

When studying and designing a problem based on neural networks, it is necessary to configure the output forecast time horizons accurately. Indeed, depending on the desired goals and often also on available resources, there are different configurations of neural architectures. These are generally divided into three main categories:

- Iterative prediction;
- Multi-output neural network;
- Dedicated networks for each forecast horizon.

The first category, the iterative methods, is mainly used for short-term (i.e. few steps ahead) horizons [134]. These methods are based on neural networks models trained for single-step predictions (i.e. with only one output). This means that for each step after the first one, the forecast for the previous step is used as input. Generally, this approach suffers from disadvantages due to that the prediction errors tend to accumulate.

The multi-output neural network category requires to use a single network with n outputs, where n is the number of steps ahead to predict. Unlike the previous method, during the implementation phase of the neural network architecture, it is necessary to accurately determine the number of future steps that need to be predicted. However, especially for longer forecast horizons, this method gives better performance than the iterative method [266].

Finally, it is possible to use different dedicated neural networks for each forecast horizon in the analysis. This approach is the most resource-intensive. Besides, this should have better results for short-term horizon (e.g. few steps ahead) but is outperformed by the multi-output method for longer horizons [134].

2.5 Models Evaluation

Generally, when operating in the context of the time-series predictions, to obtain a thorough evaluation of the adopted methods, it is good to perform an analysis of the prediction results. This in-depth analysis consists of:

- Analytical assessment.
  This kind of analysis assesses the validity of the predictions from a regression analysis point of view, by computing a set of metrics that are widely used to quantify the similarity of a discrete time-series with a reference ground truth.

- Qualitative assessment.
  This kind of analysis assesses the validity of the predictions from a qualitative point of view (e.g. clinical point of view for predicting blood glucose levels). Therefore, it would be useful to exploit metrics that are specifically designed to validate the qualitative outcome of measurements. These metrics, of course, change depending on the scenario and are strongly related to the type of dataset under analysis.

Usually, both assessments are performed on a test set that is entirely independent of the one that was used to design, train and optimise the prediction models.

The most commonly used analytical metrics will be detailed below. These are usually used in time series analysis. Instead, as far as qualitative metrics are concerned, these will be appropriately described in the following chapters, concerning the considered context.

2.5.1 Analytical assessment

To evaluate the prediction accuracy of the models, we exploited several metrics that are widely used in descriptive statistics and in regression analysis to quantify the similarity between predicted and observed time-series. More specifically, we focused on a list of metrics that are more often used by time-series literature [99]:

Root Mean Square Error *RMSE*, or *RMSD*, refers to the difference between the predicted and the observed values. It represents the prediction error index that is the most often used in literature. Mathematically it is expressed as follows:

$$RMSD = \frac{100}{\bar{y}_{test}} \sqrt{\frac{\sum_{i=1}^{n} (y_{pred,i} - y_{test,i})^2}{n}} \tag{2.17}$$

, as a percentage.

Coefficient of Determination R^2 is defined as square of the correlation (R) between predicted and observed values. Thus, it ranges from 0 (absence of correlation) to 1 (complete correlation). Mathematically it is expressed as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_{test,i} - y_{pred,i})^2}{\sum_{i=1}^{n} (y_{test,i} - \bar{y}_{test})^2} \tag{2.18}$$

Prediction Delay *Prediction Delay*, or *Time lag*, refers to the minimum time-shift between the predicted and observed signals which provides the highest correlation coefficient between them.

Mean Absolute Difference *MAD*, or *MAE*, is the *Mean Absolute Difference* between predicted and observed values. Mathematically it is expressed as follows:

$$MAD = \frac{100}{\bar{y}_{test}} \frac{\sum_{i=1}^{n} |y_{pred,i} - y_{test,i}|}{n} \tag{2.19}$$

FIT is computed as the ratio of RMSE and the root mean square difference between the observed signal and its mean value, as reported in the following equation:

$$FIT = \left(1 - \frac{\sqrt{\frac{1}{N}\sum (Y - \hat{Y})^2}}{\sqrt{\frac{1}{N}\sum (Y - \bar{Y})^2}}\right) \cdot 100, \tag{2.20}$$

where Y and $\hat{Y}$ are respectively the observed and predicted signals and $\bar{Y}$ is the mean value of the observed signal. FIT closer to 100% indicates better prediction accuracy.

Chapter 3

Solar radiation forecasting

Nowadays, green energy is considered as a viable solution to lower CO_2 emissions and greenhouse effects. Indeed, it is expected that Renewable Energy Sources (RES) will cover 40% of the total energy request by 2040. This will move forward decentralized and cooperative power distribution systems, also called smart grids. Among RES, solar energy will play a crucial role. However, reliable models and tools are needed to forecast and estimate with reasonable accuracy the renewable energy production in short-term periods. These tools will unlock new services for smart grid management.

3.1 Introduction

Nowadays, renewable energy is a very hustling research area. Finding viable, clean energy sources to replace fossil fuels, or at least to significantly decrease their usage in short to medium term, has become an extremely critical goal to achieve. On the one hand, air pollution, of which fossil fuels are a significant contributor, is causing a real health crisis [244]. According to the World Health Organisation (WHO), air pollution is responsible for 7 million deaths every year, and 91% of the world population lives in places where air quality exceeds the limits mandated by the WHO itself [189]. On the other hand, greenhouse gas emissions from fossil fuels are also one of the significant drivers of anthropogenic climate changes. According to a 2018 special report by the Intergovernmental Panel on Climate Change (IPCC), immediate action must be taken to limit the increase in global temperature to $1.5°C$ and avoid the worst consequences of global warming [162]. For these reasons, renewable energy sources (RES) will have a crucial role in the future of our society. Among them, an important part is played by solar energy, which can be used to produce electricity exploiting photovoltaic (PV) systems. The power output of a PV panel is directly proportional to the solar irradiance (SI), which in turn depends on various factors (e.g. latitude, season, and sky conditions).

There are different components of SI, but the most important one for PV power generation is Global Horizontal Irradiance (GHI), which is the total irradiance on a horizontal surface [262]. GHI is related to Direct Normal Irradiance (DNI), which is the irradiance on a surface perpendicular to the sun, and Diffuse Horizontal Irradiance (DHI), which is the radiation from light scattered by the atmosphere, as shown in Equation 3.1:

$$GHI = DHI + DNI \times \cos(z) \tag{3.1}$$

where z is the solar zenith angle (i.e. the angle between the sun and the zenith). GHI is measured in watt per square metre ($\mathrm{W\,m^{-2}}$).

To optimise smart grid operations and match power production, distribution and consumption efficiently and reliably, it is needed to know in advance the amount of energy produced by power plants, as well as energy consumption. However, one of the issues posed by some of the most popular renewable energy sources, like wind and solar energy, is their non-dispatchable and intermittent nature. A dispatchable energy source can be turned on and off when needed in a short amount of time, according to needs. This is obviously not true for PV power stations or wind farms. The sun only shines for a limited amount of hours during the day, depending on latitude and season, and the irradiance is also affected by clouds. When integrating non-dispatchable RES into existing power grids, this intrinsic variability must be taken into account, particularly when the share of energy from these types of sources increases [180]. An exciting possibility is to integrate RES using smart grid technologies. A traditional power grid is centralised and involves unidirectional power flows, where the power is sent from the power plant to customers. In a smart grid, on the other hand, the process becomes distributed, and the consumer can also be an active user, giving feedback on electrical use that allows the grid to tune itself to provide better performance and guarantee better reliability. An example of the application of smart grid management is Demand-Response (DR) [226]. DR refers to the changes in electricity consumption patterns by the user in response to fluctuations in power production by renewable energy sources and grid requirements, as well as for economic reasons like changes in the price of electricity.

As stated above, the most popular RES are non-dispatchable and intermittent in nature. Solar energy, in particular, is determined both by deterministic (e.g. latitude, day of the year, hour of the day) and stochastic factors (e.g. effects of the atmosphere and weather conditions like cloud coverage). In this context, the main challenge is, therefore, to find a methodology to predict the power generated by a photovoltaic system accurately. Since PV energy generation is highly correlated to solar irradiance, it makes sense to concentrate on predicting the latter, in particular GHI, and then use these predictions to calculate the expected energy production. For example, a photovoltaic simulator such as the one proposed in [33] could be employed using GHI predictions as a system input.

Since solar irradiance is a physical phenomenon, a possibility could be to develop a physical model. The main problem with this approach is its complexity, mainly when modelling the stochastic atmospheric phenomena that determine the measured GHI on the surface. A more straightforward approach is based on time-series forecasting. A time-series is a collection of data points equally spaced in time and chronological order [255]. A system that uses a sensor to measure solar irradiance at evenly spaced time intervals and stores these values generates a time-series. The idea is to use previous values of the time-series we are interested in predicting, and one or more related series, one or more future values. Several studies were proposed in the Literature to find physical and mathematical models to estimate and forecast solar radiation. Classical linear time-series models have been widely used [39]. Simple statistical models, for example, can be used but they might give sub-optimal results because solar irradiance is a complex nonlinear time-series. However, these studies have highlighted that these methodologies are not sufficient in the analysis and prediction of solar radiation due to the non-stationary and non-linearity characteristics [152]. To overcome these limits, a more robust approach is based on machine learning. One of the most used and studied applications of machine learning is that of artificial neural networks [252].

3.2 Related Works

The literature encloses several forecasting models for solar irradiance and PV power. In a 2013 review [67] these models are divided into four main categories i) statistical models; ii) cloud imagery-based models; iii) numerical weather predictions (NWP) models and iv) hybrid models. As introduced in Section 3.1, statistical models use previous values of the solar irradiance or PV power time-series to forecast the next values. For this reason, they represent the category of our interest to which we are inspired and compared. They, in turn, can be divided into linear and nonlinear models. As a result, according to the purpose of our work, this section investigates these models by highlighting merits and weaknesses.

Generally, linear methods represent the simplest forecasting model, often used as a reference to evaluate other more complex. Among them, for example, the naive methods are based on the simple assumption that the forecasted value of the time-series is the same as the current value. The following Equation characterises these methods:

$$\hat{y}_{t+1} = y_t \tag{3.2}$$

where $\hat{y}_{t+1}$ is the predicted value at time $t+1$, while y_t is the current value. It is useless to develop complex models if they cannot outperform this straightforward technique. For the solar irradiance forecasting, for example, in [141] authors propose the use of the index for persistence instead of the original solar irradiance, since it

gives better results. One of the simplest linear models is the autoregressive (AR) model [150]. An autoregressive model of order p, indicated with the notation AR(p), is defined as follows:

$$\hat{y}_t = \phi_0 + \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t \tag{3.3}$$

ϕ_i are the parameters of the model and ϵ_t white noise, representing an error term. So the current value of y is a linear combination of its p previous values. A slightly more complex model is the autoregressive moving average (ARMA), which combines autoregressive and moving-average components [150]. An ARMA(p, q) model with p autoregressive terms and q moving-average terms is defined as follows:

$$\hat{y}_t = \phi_0 + \epsilon_t + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} \tag{3.4}$$

The ARMA model can be extended, including exogenous inputs (ARMAX). AR and ARMA models can be used to forecast stationary time-series. In a stationary process, the mean and the variance remain constant over time [39]. But processes like solar irradiance are non-stationary, so they must be transformed into stationary time-series, or different models should be developed. The ARIMA model (autoregressive integrated moving average) can be used for non-stationary time-series forecasting. Reikard [215] shows that ARIMA can give good short-term solar irradiance forecasting results. His experiments evaluate forecasting horizons of 5, 15, 30 and 60 min, and the ARIMA model not only outperforms simple AR models in all cases, but it also performs better than feed-forward artificial neural networks except for the shortest time horizon. This might be caused by the difficulty to train ANNs, causing them to reach only a local optimum.

However, a limitation of linear models is that they cannot take into account the non-linearity of many real-life time-series, including solar irradiance. For this reason, nonlinear techniques for time-series forecasting have become very popular, and they have been extensively used for solar irradiance prediction [224]. In their research work [224], the authors exploit several artificial intelligence techniques to forecast mean hourly solar irradiance. Among all the models tested, the best results are given by a feed-forward neural network trained with the Levenberg-Marquardt algorithm. Their best approach, called univariate forecasting, exploits only the previous value of the time-series. In the work multivariate forecasting is also evaluated, meaning that other exogenous variables are also used as inputs. The multivariate approach is shown to improve forecasting accuracy compared to the univariate one. Martin et al. [159], forecasts half daily values of solar irradiance, i.e. "accumulated hourly global solar irradiance from solar raise to solar noon and from noon until dusk for each day". Since this time-series is non-stationary, two transformations are proposed, the clearness index, which is the ratio between the

solar irradiance measured at ground level and the extraterrestrial irradiance, and the lost component, which is the difference of the same quantities. Different feed-forward neural networks configurations, in terms of the number of hidden layers, neurons, inputs, are tested, and the best one is selected for each weather station where the prediction model is evaluated. The results show that ANNs improve the forecasting accuracy of the reference persistence model and outperform a simpler linear AR model. Pedro and Coimbra [201] evaluate different PV power forecasting techniques. Among them, a feed-forward neural network with one hidden layer of 20 neurons. The network has 13 inputs, which are fed with 13 previous values of the time-series. The forecasting is evaluated for 1 h and 2 h ahead. The neural network-based technique outperforms the other models evaluated in the study. Lauret et al. [141] instead compare several machine learning techniques. Simple persistence and AR models are used as a reference. The GHI time-series is pre-processed by transforming it in the clear-sky index.

For the shortest time horizons, the machine learning techniques perform better than the reference models for unstable conditions, while for clear-sky conditions, the AR model is also accurate. For longer time horizons, though, the machine learning models, including the feed-forward neural network, clearly outperform persistence and linear AR techniques. Rana et al. [212] use an ensemble of neural networks for short-term (from 5 to 60 min ahead) PV power forecasting for both the univariate and multivariate case. They test multiple groups E_i, where each ANN in the ensemble has i neurons in the hidden layer. Each ensemble is made of 20 networks. The final forecasting result is selected by taking the median of the 20 predictions. This method obtains better results than the reference persistence model and another machine learning technique called support vector machine (SVM). McCandless et al. [165] develop a cloud regime-dependent forecasting technique based on feed-forward neural networks. Instead of using a single "global" neural network, different ANNs are trained and used for each cloud regime. To determine the cloud regime, a k-means algorithm is applied to the clearness index time-series. For the shortest time horizons (15 min) ANNs does not improve upon the reference persistence model, except for the most unstable sky conditions. For longer horizons (from 60 to 180 min), however, the ANNs outperform persistence. It is also shown that the regime-dependent forecasting always gives better results than a single global ANN. Monjoly et al. [174] use different multi-scale decomposition techniques to pre-process the GHI time-series, after transforming it into the clear-sky index. The various time scale components are then forecasted separately using separate ANNs or using a hybrid AR-ANN model. The results show that multi-scale decomposition significantly improves forecasting results, both using ANNs and the hybrid model.

On the other hand, another widely used neural network is the recurrent neural network, where feedback connections are added. The outputs of these networks also depend on their current state (memory), not only on the current inputs. This behaviour makes them very suitable for time-series analysis and forecasting. Among

these, there are Long Short-Term Memory networks. These are recurrent neural networks using particular units (LSTM units) as nodes. These units can remember values for an arbitrarily long amount of time, and this behaviour makes them very suitable for time-series forecasting. Alzahrani et al. [13] use a deep (i.e. with more than one hidden layer) recurrent neural network with LSTM units for short-term forecasting of solar irradiance. The input time-series is sampled at a very high frequency (100 Hz). The advantage of this high-resolution time-series is that it can capture fast fluctuations. The results show that the deep LSTM has better accuracy than the reference feed-forward neural network. In [1] the authors exploit LSTM networks for 1 h ahead PV power forecasting. Different LSTM models are evaluated, and the best one, LSTM for regression with time steps, is selected and compared with other forecasting techniques, including multiple linear regression and feed-forward neural networks. The LSTM is shown to give more accurate results than the other models. Srivastava and Lessmann [234] compare LSTM with other established forecasting techniques in day-ahead GHI forecasting. They use satellite-derived GHI values and other atmospheric variables as inputs. In contrast to the majority of the studies in the literature, many different locations in several countries with different climates are taken into account, which makes it possible to assess the validity of the proposed model in different conditions. The LSTM-based approach is compared to a simple persistence model, a feed-forward neural network model and another machine learning model called "Gradient Boosting Regression" (GBR). The results show the superior performance of the LSTM compared to the other methods.

Another interesting recurrent architecture is the Echo State Network. An ESN has a sparsely connected hidden layer, called "reservoir", with fixed connections and weights. The only weights that are learned are those of the output connections. This property makes these networks easier to train compared to other recurrent architectures. Kmet and Kmetova [135] used an ESN for 24 h ahead solar irradiance forecasting, using the actual mean hourly values of irradiance and other meteorological variables like humidity and air temperature. The inputs of the network consist of 24 hourly values of the selected variables for the present day, and the outputs are the irradiance forecasting for the next day. The paper shows that this approach gives good results. In a previous study, on the other hand, Ruffing and Venayagamoorthy [221] found that in a real-world application results of an ESN-based solar irradiance forecasting model were not very promising. In a related field, Deihimi and Showkati [64] used an ESN for 1 h and 24 h ahead electric load forecasting. In this case, the results showed that the ESN has a good generalisation capability and can give very accurate results.

Contextually to the developments listed above, the correlation between solar radiation and weather conditions inspired many researchers to adopt a multi-variable approach, taking into account also potential external factors influencing the solar

radiation dynamics. Commonly used exogenous inputs are represented by meteorological variables measured on the site of interest, such as temperature, humidity, wind speed and cloud coverage. In case ground-based meteorological data are not available, an alternative source of exogenous variables are Numerical Weather Prediction (NWP) models, which provide weather forecasts for several worldwide locations with different scales and forecasting horizons. In [253] the authors compared an ANN using only endogenous inputs and an ANN using both endogenous and exogenous data for forecasting daily solar radiation. The comparison between the univariate and the multivariate methods showed that the usage of exogenous inputs produced a performance gain between 0.5% and 1% in terms of RMSE, providing a significant improvement, especially during the winter season when the solar radiation variations are more significant. In [212], the author evaluated the effectiveness of using exogenous inputs in forecasting the electrical power generated by a PV system from 5 to 60 minutes ahead. The prediction results of the study showed that the adoption of exogenous inputs does not provide any performance improvement in the short-term power forecasting. According to the authors, a possible reason for this is that PV power data already reflects the weather changes in the short-period, without the need for additional input variables. In conclusion, the authors believe that meteorological data are more likely to be useful with longer forecasting horizons.

Even though in the literature there is some evidence of the usefulness of exogenous inputs in solar radiation forecasting, it is still unclear to what extent they provide an edge over the univariate case for the different forecasting horizons. Furthermore, we noticed that despite some works collected a large number of potentially useful input variables, they ended up using only a low number of those variables after feature selection. We believe that the reason for this lies in the adoption of too stringent feature selection methods, which overlook the majority of the collected features and lead to an underutilization of the input variables. In particular, we think that the employment of the Pearson's correlation coefficient as a single feature selection criteria represents a severe limitation to the application of machine learning models since it is only able to identify linear relationships between variables [236]. Some studies overcome this limitation by using the Mutual Information criteria [174, 141], which is known for recognizing both linear and non-linear dependencies in the data. Nevertheless, the mutual information still presents the typical limitations of filter methods, which evaluate the usefulness of each variable independently of the context of other [101].

3.2.1 Contributions

Concerning presented literature solutions to forecast solar radiation, in this Chapter, we start by proposing an innovative methodology for implementing two

different non-linear autoregressive neural networks to predict Global Horizontal Solar Irradiance (GHI) in the short-term time period (i.e. from 15 to 120 min ahead). Both neural networks have been implemented, trained and validated exploiting a dataset consisting of four years of solar radiation values collected by a real weather station. We also present the experimental results discussing and comparing the accuracy of both neural networks. Then, the resulting GHI forecast is given as input to a Photovoltaic simulator to predict energy production in short-term periods. Finally, we present the results of this Photovoltaic energy estimation also discussing their accuracy.

Progressively, we focus on neural techniques for predicting GHI. The aim is to identify the best instrument with the best configuration. For this purpose, we propose an optimised methodology for the short- and mid-term GHI forecast exploiting and comparing four neural networks specifically designed and optimised (i.e. a NAR, a FFNN, a LSTM and a ESN as detailed in Section 2.2). We applied three different dataset pre-processing and filtering techniques to identify which of them has better performances by comparing the prediction results of the different architectures. As a first analysis, we gave in input to our neural networks the raw GHI dataset where we applied some basic filters to clean the dataset itself by removing possible errors, such as lack of data or storage error due to sensor sampling. Then, as a secondary analysis, we applied the Tikhonov regularisation technique to the very same raw dataset, which smooths the time-series trend-making easier the training of our neural networks. Finally, as a third analysis, we converted the raw dataset, consisting of GHI samples, into clear-sky index values, thus removing seasonal trends of the time-series. For a fair comparison, all the datasets at our disposal are used as input on all the best neural architectures identified experimentally. The use of pre-processing techniques, together with the capillary optimisation of neural structures, allows us to increase the prediction time horizon with an acceptable error rate.

Moreover, compared to state-of-art solutions, the optimisation of neural architectures from specifically transformed datasets allows us to obtain leaner structures at the computational level without affecting the prediction accuracy. The results analysis shows that the clear-sky index approach is the most successful, giving the most accurate results, particularly for mid-term predictions and the Echo State Network results to be the neural architecture that best performs in terms of prediction accuracy.

Lastly, we investigate the effectiveness of using exogenous inputs for short- and mid-term solar radiation forecasting. In this view, we propose a novel approach consisting on applying multiple feature selection techniques chosen in a way to counterbalance the limitations of each other, to achieve more robust results than those that could be obtained by using a single selection method. The predictive performance of the selected features are evaluated by feeding them into five different machine learning models: namely a FNN, an ESN, a 1D-CNN, a LSTM

and a Random Forest (RF), as detailed in Section 2.2. To conclude, we compare the obtained models against models using only endogenous inputs to evaluate the effectiveness of using exogenous data for short-term solar radiation forecasting.

3.3 Case Study

The case study is represented by a dataset of 6 years GHI measurements (i.e. from the 1st January 2010 to the 31st December 2015), sampled every 15 minutes from a meteorological station in the university campus of Polytechnic University of Turin in Italy. In addition to GHI values, the weather station also provided a set of meteorological variables measured on-site, namely cloud cover, air temperature, relative humidity, sea-level pressure and wind speed.

A dataset with additional meteorological variables was derived from Dark Sky [148], which is a software company specialized in weather forecasting and visualization. The Dark Sky API provides historical weather observations for worldwide locations by gathering information from multiple sources. The API responses come up with the distance of the nearest station that contributed to the results, together with the list of sources used for the aggregation. The nearest station for our site of interest was located at 1 km of distance. The list of references that contributed to the responses is the following: the USA NCEP's Canadian Meteorological Center ensemble model [179]; the German Meteorological Office's Icosahedral Nonhydrostatic model [187]; the USA NOAA's Global Forecast System [182]; the USA NOAA's Integrated Surface Database [183]; the USA NOAA/ESRL's Meteorological Assimilation Data Ingest System [139].

During feature engineering, we added three additional variables containing time information: i) day of the year, ii) hour of the day and iii) minute of the hour. Furthermore, we decided to replace the variables sunrise time and sunset time derived from Dark Sky by using a more compact representation called sunshine duration, which is defined as the amount of time elapsed between the sunrise time and the sunset time.

To ensure data quality, we carefully examined each variable collected and fixed any irregularity present in the dataset. The GHI variable presented a single outlier during this period, consisting of a negative observation probably due to a temporary malfunction of the pyranometer, while other variables showed some missing values. Both the outlier and the missing values were replaced by using a simple linear interpolation. The comparison between the observed GHI values and the clear-sky (I_{cs}) GHI values computed by using the Ineichen and Perez model [120] revealed that the measured GHI values exceeded several times the estimated GHI values in I_{cs} conditions. The discrepancies were probably due to some uncertainty in the measurements of the pyranometer. Therefore, since the clear-sky GHI should be the maximum value that can be assumed by the observed GHI, we decided to round

the exceeding GHI values to the corresponding clear-sky GHI values.

The final dataset obtained in this way was composed of 210333 records and 15 input variables. Table 3.1 provides a brief description of each variable present in the dataset.

Table 3.1: List of input variables

Variable	Unit
Global Horizontal Irradiance (GHI)	W/m^2
Ultraviolet (UV) index	-
Air temperature	°C
Relative humidity	%
Sea-level air pressure	hPa
Cloud cover, i.e. the percentage of sky occluded by clouds	%
Hourly precipitation intensity	mm/h
Hourly precipitation probability	%
Wind speed	m/s
Wind bearing, measured in degrees progressing clockwise from the true north	$degrees$
Dew point	°C
Sunshine duration	s
Day of the year	-
Hour of the day	-
Minute of the hour	-

Finally, to evaluate the prediction performance of the proposed models, we divided the dataset into a training set and a test set. The training set consisted of the first five years of observations in the period 2010-2014, while the test set consisted of the last year of data, i.e. 2015. It is crucial to notice that all optimizations were performed by using only the training set to avoid any look-ahead bias.

3.4 Methodology

Predicting the energy production of a PV system means being able to forecast the level of GHI. In turn, predicting the values of GHI means working with time-series information. This kind of information identifies a sequence of values chronologically ordered [103]. The study and manipulation of time-series models bring different benefits. Mainly, it allows i) understanding the underlying forces and structures that produced the observed data and ii) in fitting a model and in proceeding to forecast and monitor or even feedback and feed-forward control [186].

In this context, our methodology consists of the design of neural networks to

predict the GHI for PV energy simulation. As a first step, we focus on short-term GHI prediction. Then, using a photovoltaic simulator previously developed by my research team, we use the GHI prediction results obtained as input to the simulator. In this way, we validate GHI predictions, and we can perform robust PV simulations in the short-term. Therefore, having endorsed the GHI prediction methodology with neural networks, we focus on short- and mid-term GHI prediction. Our goal is to compare the most promising neural networks specialized in time-series forecasting, to find the best one. To this end, we design and optimize different neural networks, testing multiple configurations. Lastly, we evaluate the effectiveness of using exogenous inputs for short- and mid-term solar radiation forecasting. Again, the aim is to improve the forecast horizon by achieving increasingly accurate and robust predictions.

3.4.1 Short-term GHI forecasting for PV energy predictions

The first part of our proposed methodology consists of forecasting short-term Global Horizontal Solar Irradiance (GHI) for photovoltaic energy predictions. To deal with these time-series data, we adopted, and then compared, two ANNs: i) Nonlinear Autoregressive neural network (NAR) and ii) Nonlinear Autoregressive Moving Average neural network (NARMA), as described in Section 2.2. NAR belongs to the family of Nonlinear Autoregressive Exogenous Model (NARX) [227]. It is generally considered as one of the best tools for time-series analysis and does not suffer from stability problems [241]. This is due to its nonlinear autoregressive model, which has exogenous inputs. This neural network model bases its prediction on i) a variable range of past values and also on ii) the current and past values of the exogenous driving inputs of the time-series in the analysis. However, this process produces a prediction error that is the knowledge of the past. Indeed, the presence of this error as a result of prediction means that the future values of the time-series cannot be predicted precisely. On the other hand, NARMA belongs to the family of Nonlinear Autoregressive Moving Average Exogenous Model (NARMAX) [58]. It represents a generalization of the NAR model, as described in Section 2.2.2. However, this model realizes a feed-forward network where a predictor will have feedback when the regressors are selected.

Comparing the two proposed models, the significant difference is that NARMA is a *Recurrent Neural Network* [155], while NAR is not. Thus, NAR has a predictor without feedback, while NARMA has feedback through the choice of regressors. Hence, future network inputs will depend on present and past network outputs. This might lead to instability of the ANN itself, and it can be very difficult to determine whether or how the predictor is stable. To avoid instability, NARMA architecture uses a linear MA-filter to filter past residuals. This is a Low Pass FIR (Finite Impulse Response) filter, commonly used for smoothing an array of sampled

data/signal. It takes a set of inputs at the time, it computes the average of those samples and produces a single output [153].

To implement both NAR and NARMA, we exploit the dataset described in Section 3.3 but of only four years of real GHI values (from 2010 to 2013). These values are 15 minute time sampled by the weather station in our University Campus. Then, we divided the dataset asymmetrically into three years for the training-set (2010-2012) and one year for the validation-set (2013), according to the more recent approach described in [260]. This allows an even more accurate training phase.

Progressively, we started analyzing the number of past signals used as regressors for the prediction. Specifically, we used *Lipschiz* [211] to determinate the *lag-space*. This methodology allows identifying the order of Input-Output Models for Nonlinear Dynamic Systems. Given corresponding input and output sequences, it calculates a matrix of indices that can help determine a proper lag-space structure. However, as detailed in [110], this methodology is not always effective. Still, it represents a good starting point to define the more suitable number of regressors, which will characterize the future neural networks architectures. Consequently, we started the design of both our ANNs, considering several regressors in the range between 1 and 10. This arbitrary choice derives from our previous work [8] that exploits 10 regressors. On the other hand, ANNs proposed in this work aim at improving previous performances. Moreover, as previously described, we modified the pre-process of the input dataset, this allows designing an ANN with no more than 10 regressors. Fig. 3.1 details the result of the applied lag-space investigation methodology.

Fig. 3.1 suggests that excellent performance can be achieved with 6 regressors (i.e. past inputs). All previous values are not computationally advantageous. Even, between 1 and 4, the result diverges to infinity, and therefore they are not displayed in the plot. On the other hand, all values above 6, even if advantageous, would risk transforming ANN architecture into a more complex and less performing structure. Thus, the best configuration in the computation/performance ratio is achieved with 6 regressors (see the knee-point of the plot in Fig. 3.1). Differently to what we did in [8], in this work, we adopted also a design space exploration approach [54]. As a result, we decide to validate (or refute) the findings given by Lipschitz. For this purpose, we implemented all the possible network combinations (of both NAR and NARMA models) from 1 to 10 regressors. This allows us to evaluate and compare all the obtained architectures' performance and then find the best solutions for both NAR and NARMA. Fig. 3.2 shows the normalized sum of squared errors (NSSE) error trends of the two ANNs based on the number of regressors for each implemented architecture. NSSE error is a network performance index. The lower NSSE, the better ANN's performance.

As shown in Fig. 3.2, NAR and NARMA give the best performance with 4 and 2 regressors, respectively. This also represents the best compromise between ANN's computation and performance. It is worth noting that these NSSE results improve

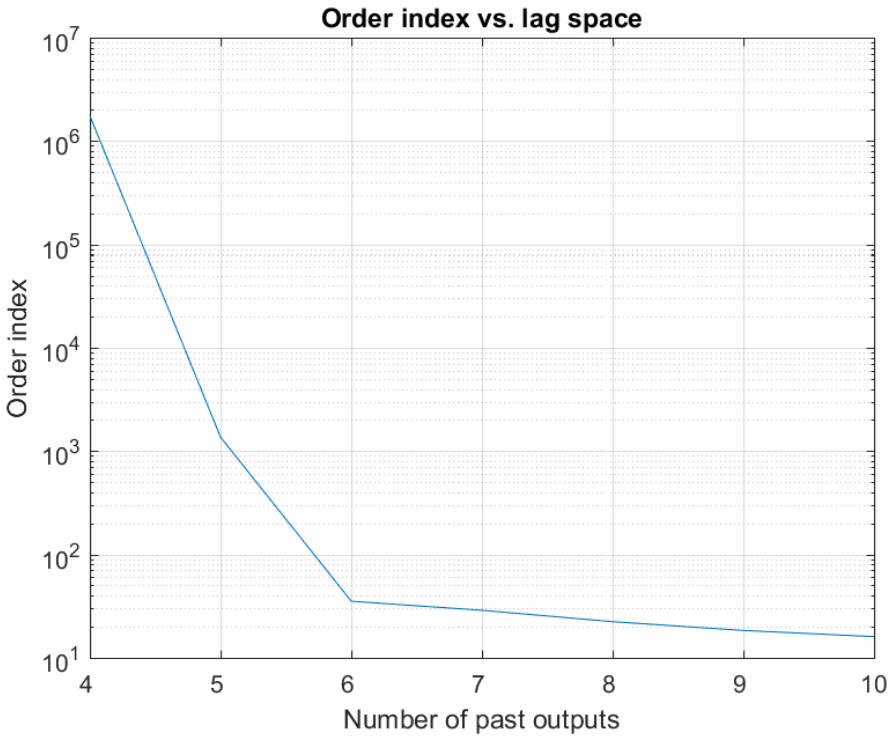


Figure 3.1: Evaluation of Order Index criterion for different lag-space

the indication given by Lipschitz methodology that suggested 6 as the best number of regressors (see Fig. 3.1).
Once we found the optimal regressors for both NAR and NARMA, we implemented the two final ANNs starting from two fully connected architectures with one hidden layer of 30 hyperbolic tangent units. This large number of units could be redundant, but it is justified by the pruning technique [242], which is used in the next phases to optimize the network architecture themselves.
Before training, the weights of both ANNs are initialized randomly. This also allows to initialize i) weights, ii) their decay threshold and iii) the maximum number of iterations. However, these parameters are overestimated during the very first training iteration. Then, we proceeded with the training phase for both NAR and NARMA networks. Training is a minimizing technique to compute the best weights. For both architectures, we used the *Levenberg-Marquardt* algorithm, which interpolates between the Gauss-Newton algorithm and the method of gradient descent using a *trust region* approach [185]. Progressively, we used the methodology illustrated in [184] for validating both ANNs. This methodology performs a set of tests, including autocorrelation function of residuals and cross-correlation function between controls and residuals to validate system outputs. The result of this process gives the NSSE error. By definition, this error should not be too large compared to

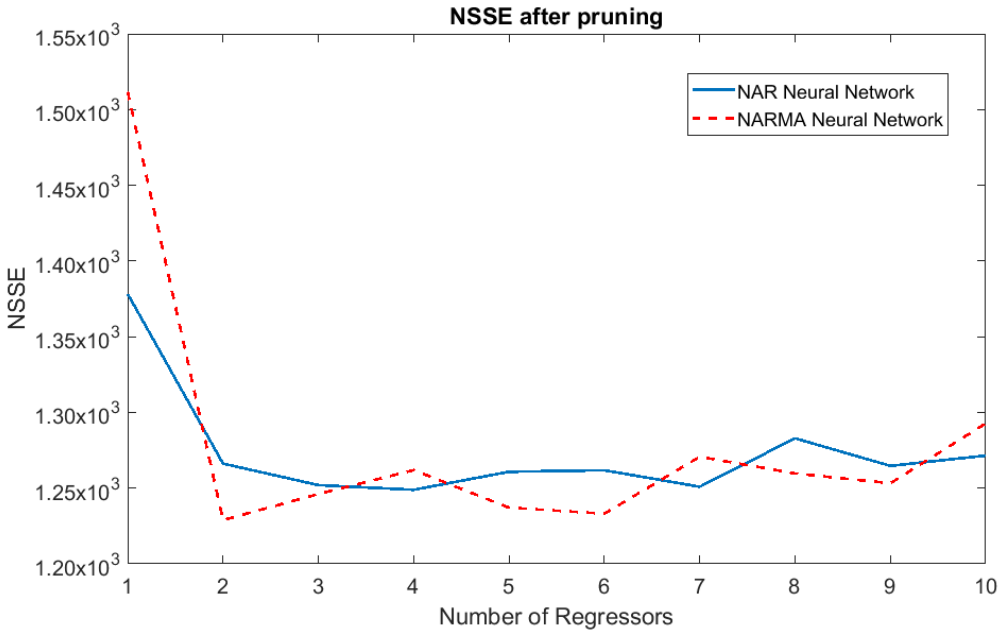


Figure 3.2: Evaluation of NSSE after pruning with regard to number of regressors

training error. If NSSE is greater than the training error, the predicted results are over-fitting the training-set. Table 3.2 illustrates the obtained results for both NAR and NARMA.

Table 3.2: NSSE comparison after the first and final validation

Neural Network	NSSE after first validation	NSSE after final validation
NAR	$1.25 \times 10^{+3}$	$1.24 \times 10^{+3}$
NARMA	$1.23 \times 10^{+3}$	$1.22 \times 10^{+3}$

A shown in Table 3.2, the validation process yields these indexes as detailed in the column *NSSE after the first validation*. The NSSE is equal to $1.25 \times 10^{+3}$ and $1.23 \times 10^{+3}$ for NAR and NARMA, respectively. These indexes will have to be compared with those obtained after the optimization of the architectures. Then, we proceeded to the optimization phase of both networks. Our purpose was to remove excess weights and obtain a smaller error than the one given during the first validation. To achieve this, we adopted the *Optimal Brain Surgeon* (OBS) [105], which is a technique to prune superfluous weights. OBS computes the Hessian matrix weights iteratively, which leads to a more exact approximation of the error function. The inverse Hessian is calculated employing recursion. This method allows for finding the smallest saliency S_i as follows:

$$S_i = \frac{w_i^2}{2[H^{-1}]_{i,i}} \tag{3.5}$$

where $[H^{-1}]_{i,i}$ is the $(i,i)th$ element of the inverse Hessian matrix and w_i represent the ith element of the vector θ containing the network weights. The saliency identifies the quality of the connection between the various network units. This methodology allows verifying the state of the saliency iteratively. If the saliency S_i is much smaller than the mean-square error, then some synaptic weights are deleted and the remaining ones are updated. The computation stops when no more weights can be removed from the network without a large increase of the mean-square error. Once the new weights are given, we re-validated both resulting pruned NAR and NARMA.

Through the same methodology used in the first validation phase, we proceeded to the validation of the final network using the new weights. The resulting NSSE error indexes for both ANNs are illustrated in the column *NSSE after final validation* in Table 3.2. NSSE error indexes are $1.24 \times 10^{+3}$ and $1.22 \times 10^{+3}$ for NAR and NARMA, respectively. In both cases, the new NSSE values are lower than the ones given after the first network validation. Thus, the optimization for both ANNs succeeded. The resulting NAR with 4 regressors and NARMA with 2 regressors are trained and validated. Hence, they are ready to forecast GHI values in short-term time-periods, and their results will be discussed in next Section 3.5.1.

3.4.2 Comparative analysis of state-of-art neural networks for GHI forecast in short- and mid-term horizons

As the second step of our methodology, we focus specifically on the forecast the values of GHI in short- and mid-term time-horizons. For this reason, we design and implement the following neural network architectures:

- Nonlinear Autoregressive Neural Network;
- Feed-Forward Neural Network;
- Long Short-Term Memory Neural Network;
- Echo State Network;

We also applied three different pre-processing techniques to the input dataset to obtain better performances in terms of prediction accuracy and computation level. Moreover, we consider also both one-step and multiple steps predictions for each selected model. Making a one-step prediction means taking GHI samples at times $t, t-1, t-2, \ldots, t-n$ to predict GHI at time $t+1$, i.e. in 15 min, since that is the

distance of two consecutive samples in the dataset. Instead, for multi-step prediction, we evaluate two different techniques, iterative and multi-output, respectively. In the iterative approach, the artificial neural network has a single output, so it can only predict one-step ahead. For subsequent steps, the predicted value for time $t+1$ is used as one of the inputs for the prediction at time $t+2$, and so on. In the multi-output approach the network has n output nodes, giving the prediction for $t+1, t+2, \ldots, t+n$ in parallel. Progressively, the networks are first trained using the raw GHI time-series, after only some basic pre-processing (i.e. removing errors due to sensor sampling). Then, we applied two different technique to pre-process this raw GHI dataset: i) Tikhonov regularization and ii) clear-sky index conversion. The Tikhonov regularization technique smooths the time-series trend avoiding possible spikes. Whilst converting GHI time-series into clear-sky index values removes all possible seasonal trends. In both cases, we trained all our neural networks again, and then we optimized their architectures. Each of the three resulting datasets has been split into training- and test-set (i.e. data never used during the training). To fair compare their performances in forecasting GHI, we trained all our neural networks with the very same training-set, and we tested them with the very same test-set. Figure 3.3 summarizes the overall process.

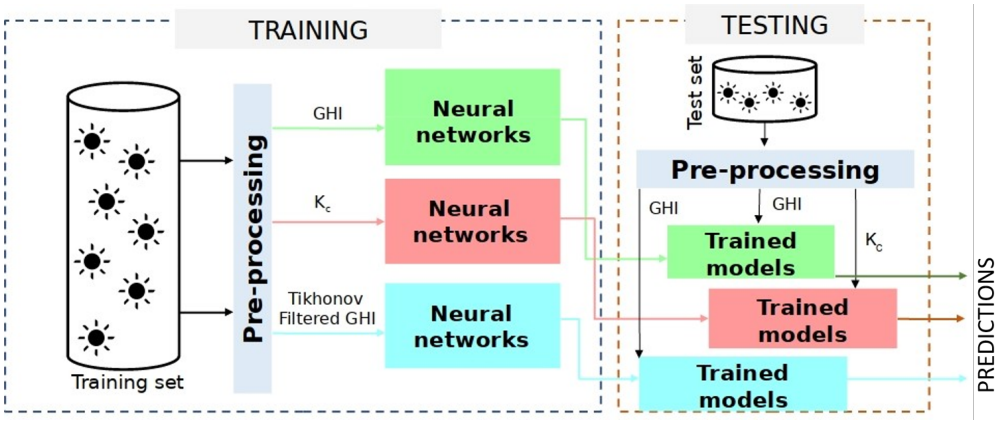


Figure 3.3: Methodology outline

Data transformation

We exploit a dataset of 6 years GHI measurements, sampled every 15 minutes from a meteorological station in Turin, as described in Section 3.3. The dataset has been subjected to necessary some pre-processing. First, GHI can never be negative (i.e. $GHI < 0$), so any negative values were set to 0. Then, comparing the raw GHI values in the dataset with the generated clear-sky values (I_{cs}), some of them were higher than the corresponding I_{cs}. This is probably due to some sampling error by the sensor or, in some cases, to some short-term cloud enhancement effects [16].

Since most of these peaks usually occur when the solar zenith angle is big [229], which is also when the reliability of the sensor is lower, it was decided to filter these anomalous peaks, so for each $GHI > I_{cs}$, GHI was set equal to I_{cs}.

After these basic pre-processing, we divide the dataset into training- and a test-set (5 years for Training (2010-2014) and 1 year for Test (2015), with 175 296 and 35 038 samples, respectively). Furthermore, we introduced two dataset transformations to improve prediction performance: i) Tikhonov regularization and ii) clear-sky index.

Tikhonov regularization Generally, raw GHI data is characterized by many sudden peaks and variations. For this reason, we decide to try smoothing the data to make the training of the neural networks easier. The disadvantage of this approach is that some information about the variability of the phenomenon will necessarily be lost. However, the potential benefit is that the networks could more easily follow the trends in the data, particularly on medium or long term predictions. This is a trade-off, meaning that the choice of this approach might depend on the required prediction horizon and the application for which the predictions are needed. Then to smooth the original data, we exploit the Tikhonov regularization [122]. This technique is used for time-series analysis and forecasts in other domains, like glucose level prediction [6]. However, it is not commonly used for solar irradiance forecasting.

The filtered signal is given by:

$$\hat{y} = U_d \omega \tag{3.6}$$

In Equation 3.6, ω is the N-dimensional first derivative of the input signal, while U_d is the integral operator matrix (Equation 3.7).

$$U_d = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & 1 & 0 & \dots & 0 & 0 & 0 \\ 1 & 1 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & \dots & 1 & 0 & 0 \\ 1 & 1 & 1 & \dots & 1 & 1 & 0 \\ 1 & 1 & 1 & \dots & 1 & 1 & 1 \end{bmatrix} \tag{3.7}$$

To calculate ω, the function $f(\omega)$ (Equation 3.8) needs to be minimized.

$$f(\omega) = \|y - U_d \omega\|^2 + \lambda_d^2 \|L_d \omega\|^2 \tag{3.8}$$

In Equation 3.8 L_d is the second derivative operator matrix, while λ_d is the regularization parameter, set to 3000, in accordance to [6]. An example of the results of the filtering can be seen in Figure 3.4.

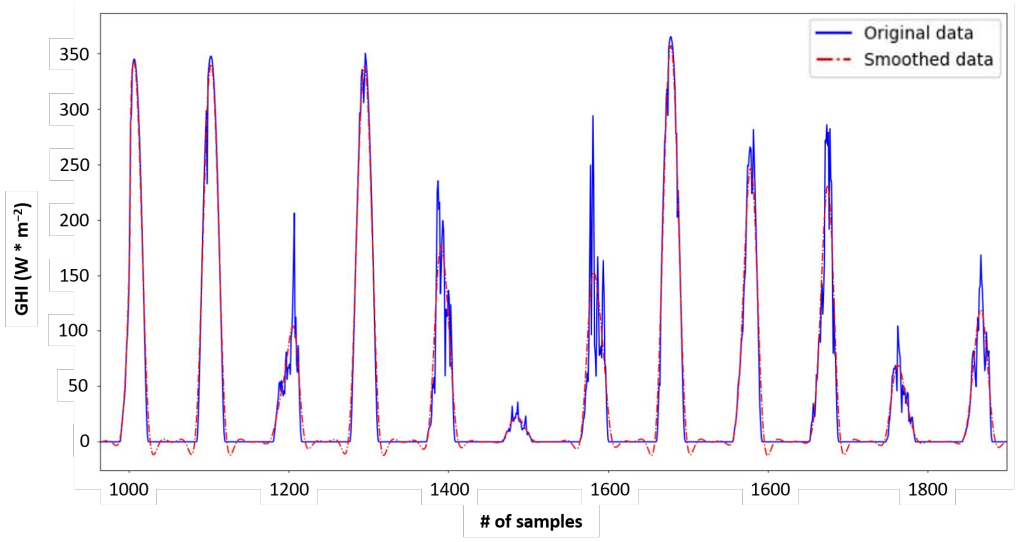


Figure 3.4: Original data vs. smoothed data

Filtered data will be used for training the networks. Since the filter eliminates some peaks and spikes in the GHI data, giving a smoother signal, it should be easier for the neural networks to approximate it, potentially increasing the generalization capability of the model. Once the neural networks are trained, the original unfiltered GHI data will be used for testing, as shown in Figure 3.3.

Clear-sky index The solar irradiance time-series exhibits a seasonality component and is therefore non-stationary. Some authors assert that neural network can work well even with non-stationary time-series, given enough training data [224]. Others prefer to transform the solar irradiance into a stationary series [201, 174]. For stationary series, statistical properties like mean and variance are constant in time. This should make it easier to predict than a non-stationary series. For this purpose, the clear-sky index (K_c) is used in literature. K_c is the ratio between the measured irradiance and the expected irradiance under clear sky conditions. For this work, after evaluating the performance of the network with the original GHI data, we chose to repeat the experiments using the clear-sky index, defined in Equation 3.9, where I_m is the measured irradiance, and I_{cs} is the calculated clear-sky irradiance.

$$K_c = \frac{I_m}{I_{cs}} \tag{3.9}$$

The K_c series was then used to train the networks. Since $I_{cs} \geq I_m$, then $0 \leq K_c \leq 1$, so it is not necessary to scale the input data. To make predictions, K_c values were used as inputs, then the predicted values (i.e. the outputs of the network were

multiplied by the corresponding clear-sky values to obtain the GHI predictions) that could then be compared with the expected values.

Prediction Model Building

Between the solutions we propose and compare, one of the most commonly used is the Feed-Forward Neural Network (FFNN). Contextually, we exploit a NAR architecture also based on the Multilayer Perceptron like the FFNN. Besides, we propose an LSTM architecture, often used for time-series forecasting and successfully applied to GHI prediction in recent studies [13], [1]. Finally, the Echo State Network that is a less traditional recurrent architecture, but it has shown promising results in time-series forecasting [135]. In the following subsections, we will present the neural architectures considered in this study. For each of them, we will describe the properties and strengths, giving particular emphasis to the hyperparameters taken into consideration and properly investigated. In Section 3.5, instead, we will present and detail all the network configuration w.r.t. the exploited dataset. This is because the optimization of the architecture strictly depends on the dataset under consideration.

Echo State Network The Echo State Network (ESN) is a recurrent neural network composed by an input layer, a recurrent hidden layer called "reservoir" and an output layer as detailed in Section 2.2.2. ESNs are an implementation of so-called "reservoir computing" [223]. The main idea is to have a fixed, random, sparsely connected recurrent layer, and a readout layer, connecting the reservoir to the output. In the most straightforward architecture, these output connections are the only trainable ones. It is also possible to add direct trainable connections from input to output, bypassing the reservoir, and feedback connections from output to reservoir [123].

One interesting feature of ESNs is that they are straightforward to train, unlike other recurrent neural networks. In ESNs, the recurrent layer is fixed, and this greatly simplifies the training process. Some hyperparameters need to be determined. Then we exploited a trial-and-error approach to identify the number of inputs (regressors). For the size of the reservoir, Lukoševičius [149] suggests that a "big" reservoir is usually better, given the sparsity of the connections between its units. It is not possible to implement an arbitrarily big reservoir since memory consumption needs to be taken into account. Keeping this in mind, the first choice was to use 500 units. Starting from that upper limit, reservoirs with 50, 100, and 200 units were also tested, to verify the assumption that "bigger is better". The reservoir density was chosen again by trial-and-error, and 0.1 was selected for its value. Another critical parameter is the spectral radius. As already discussed, a value lower than one should guarantee the echo state property. It is usually an excellent choice to choose a value close to 1, as suggested in [149], so 0.9 was selected.

Nonlinear Autoregressive Neural Network The Nonlinear Autoregressive Neural Network is an ANN that extends a traditional linear autoregressive model, as detailed in Section 2.2.1. It is particularly suitable for non-linear time-series that report unexpected spikes and fleeting, transient periods [184].

Progressively, we determined the hyperparameters of the network, in particular the number of both regressors and units in the hidden layer. Since there is no rule to determine the best amount of regressors mathematically, the choice was made by trial-and-error, going from 2 up to 20 regressors. Regarding the units in the hidden layer, we overestimate the initial number selecting 30 hidden units. This because, in this methodology, we can adopt pruning functionality that allows to eliminate superfluous weights and determine the best network configuration [8]. Once the parameters were selected, the network training was performed using the Levenberg-Marquardt algorithm [143]. Consequently, we have pruned the obtained network with the Optimal Brain Surgeon algorithm [108], and we trained the network again before making the inference for the predictions.

Feed-forward Neural Network Based on the Multilayer Perceptron, the FNN is characterized by a dense, fully connected layer, where information only moves from one side to the other. In a preliminary phase, we considered different architectures with different hidden layers. We found that the best compromise between prediction and computation accuracy is an architecture with two hidden layers. Once the model has been chosen, we determined the hyperparameters, i.e. number of regressors and activation function. For the hidden layers, we have opted for the hyperbolic tangent (tanh) activation function, since it is a common choice and gives good results [128]. For the output layer, instead, we have chosen a linear activation function. As for the number of inputs and units in the hidden layer, there is no established mathematical technique to select the best parameters. In this case, we opted for a trial-and-error approach. First, we arbitrarily decided to choose the number of units as two times plus one the number of inputs for networks with a single hidden layer. Then, we investigated the number of regressors from 2 to 20, evaluating the performance for each case. The optimization algorithm used for training is the Adaptive Moment Estimation (Adam optimizer) [131]. This algorithm is closely related to two other optimization techniques, Root Mean Square Propagation (RMSProp) and Adaptive Gradient Algorithm (AdaGrad), combining their features together. To avoid the phenomenon of overfitting, we have used the early-stopping technique [206]. In practice, during the training phase, training is stopped when there is no improvement in the validation set for a few steps. The additional benefit of early-stopping is the significant reduction of the training times.

Long Short-Term Memory Neural Network The Long Short-Term Memory Neural Network (LSTM) represents an evolution of a canonical recurrent neural network developed to solve the "vanishing gradient" problem [114], as detailed in

Section 2.2.2. This is a problem that arises during the training of such neural networks with backpropagation methods. These architectures are particularly suitable in the prediction of time-series because, thanks to their structure, they can preserve the error that can be back-propagated through time and layers. By maintaining a more constant error, they allow recurrent nets to continue to learn over many time steps. Since the LSTM was proposed, different variations of the architecture were developed [98]. The typical LSTM unit is composed of a cell, an input gate, an output gate, and a forget gate. The LSTM unit structure is shown in Figure 2.8.

For our implementation, we used an LSTM layer for the hidden part of the networks, while the output layers are simple dense layer, as for the FFNN. As already discussed for the FFNN in Section 3.4.2, the same trial-and-error approach was adopted to determine the number of regressors, hidden units and hidden layers. The Adam optimizer was again used for training, and the same early-stopping technique was also used to avoid overfitting. The benefit given by early-stopping on training times was particularly significant with the LSTM networks, whose training times are significantly longer than those of the FFNN. Even for a much smaller LSTM, with 10 hidden units vs 200 for the FFNN, each epoch took approximately 8 times longer.

Multi-step predictions

In according to Section 2.4, we evaluate the iterative prediction and multi-output neural network approach. The experimental results have proved that the multi-output approach outperforms the iterative one (see Section 3.5.3), and it was therefore chosen for all the following experiments. This does not apply to the NAR network, however, since the realized model already contains a function to perform multi-step predictions, which was therefore used.

3.4.3 In-depth analysis of Multivariate Configurations for GHI forecast

Finally, we investigated the advantages of using exogenous inputs for short- and mid-term GHI forecasting. Therefore, we exploit the whole dataset (as described in Table 3.1) by applying some statistical techniques to find the best configuration of exogenous input.

To facilitate the analysis by statistical models, we normalized the GHI values by using the clear-sky index transformation, which is a widely used transformation that introduces stationarity in solar radiation time series by removing the seasonal and daily trends, as detailed in Section 3.4.2.

Feature selection

The objective of feature selection is to find which are the most significant input variables to make the most effective use of the available information. Indeed, the selection of key features can improve the prediction performance of machine learning models and may reduce the computational costs involved in the training process. In [101], the authors classified the feature selection methods into three major categories: i) filter methods, ii) wrapper methods, and iii) embedded methods. Filters rank features based on some correlation criteria. Wrappers aim to find the best subset of features based on their empirical prediction performance. Finally, embedded methods employ learning machines that perform feature selection as part of their training process. Therefore, we selected a couple of the most commonly used methods for each category mentioned above, employing a total of six feature selection techniques. In the following, we provide a brief description of the feature selection methods adopted by our methodology, concluding with the final results of the feature selection process.

Correlation criteria The Pearson correlation coefficient measures the linear relationship between two variables. It takes values between -1 and 1, where 1 is a positive linear correlation, 0 is lack of linear correlation, and -1 is a negative linear correlation. The Pearson correlation criteria estimate between two variables x, and y is given by the following formula, where the bar notation stands for the sample mean:

$$R = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2}} \tag{3.10}$$

Since Pearson's correlation can assume both positive and negative values, we decided to adopt a different measure to compare the input variables. For this purpose, we used the coefficient of determination R^2, which is defined as the square of the correlation coefficient R and represents the fraction of variance in the target variable that is explained by individual variables. The coefficient of determination between the input variables and solar radiation constituted our ranking criteria for this method.

Information criteria In information theory, the mutual information (MI) is a measure of the mutual dependence between two random variables. More precisely, it is a measure of the statistical dependence between the density of a variable x and the density of a variable y. The following formula gives the mutual information:

$$\mathrm{MI} = \int_x \int_y p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right) dxdy \tag{3.11}$$

where $p(x)$ and $p(y)$ are the probability densities of x and y, and $p(x,y)$ is their joint probability density. In case of continuous variables, the densities $p(x)$, $p(y)$

and $p(x, y)$ are estimated by discretizing the variables or by approximating their densities with Parzen windows. For this method, the input variables were ranked based on their mutual information with solar radiation.

Sequential forward selection The sequential forward selection (SFS) is a greedy search algorithm that aims to find the best subset of features that maximizes the prediction performance. The SBS algorithm starts with an empty set and at each step adds the feature whose insertion gives the highest prediction performance. The process is iterated until the desired number of features is added. To evaluate each subset of features, we used a simple Linear Regression model. The training set consisted of 4 years of data in the period between 2010 and 2013, while the test set comprised the following year of data, i.e. 2014. The algorithm iteratively inserted the variable whose insertion obtained the lowest Mean Squared Error (MSE) on the test set. The ranking of features was obtained by sorting the input variables based on their insertion order.

Sequential backward selection The sequential backward selection (SBS) algorithm starts with the complete set of all features and at each step removes the feature whose removal gives the highest prediction performance. The process is repeated until the subset contains the required number of features. The model and the dataset splitting are the same used with the forward selection algorithm. The algorithm iteratively deleted the feature whose removal obtained the lowest MSE on the test set. Differently from the SFS algorithm, the ranking of features was obtained by sorting the input variables based on their deletion order.

LASSO regression Least Absolute Shrinkage and Selection Operator Regression (LASSO Regression) is a Linear Regression model that uses l_1 regularization. In particular, a penalty term is added to the cost function by using the l_1 norm of the model parameters. The regularized version of the cost function is defined as:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\theta^T x_i - y_i)^2 + \alpha\sum_{i=1}^{n}|\theta_i| \tag{3.12}$$

where m is the number of training instances, n is the number of input variables, x_i is the input vector, y_i is the target, θ is the parameter vector of the model, and α is the regularization hyperparameter. The peculiarity of l_1 regularization is that it tends to yield sparse feature vectors by setting to zero the weights of least significant features. In this sense, LASSO Regression can be understood as a feature selection technique. The sparsity of the feature vector can be enhanced by augmenting the regularization hyperparameter α. Here, we trained a LASSO Regression model to predict solar radiation values, eventually sorting the input variables based on the magnitude of their coefficients.

Random forest A Random Forest is an ensemble of Decision Trees, each one trained on a different random subset of the training set. In Decision Trees, essential features are generally located closer to the root of the tree, while irrelevant features are located closer to the leaves. The feature importances can be estimated by computing the average depth at which each feature appears across all the trees in the forest. Thus, features were ranked based on their relative importance.

Table 3.3 shows the final results of the feature selection process. The final ranking was obtained by computing the mean position of each feature across the different feature selection methods. The threshold between the selected features and the discarded features was chosen by using a trial-and-error approach during the model evaluation phase. In particular, we noticed a performance improvement until we inserted the day of the year variable. On the other hand, the variables pressure, wind speed, precipitation probability, precipitation intensity and minute penalized the prediction performance once inserted. In conclusion, we found that the most relevant features for solar radiation forecasting are UV index, temperature, sunshine duration, cloud cover, hour, wind bearing, dew point and humidity.

Table 3.3: Feature selection results

	Features	Methods						Ranking
		R^2	MI	SFS	SBS	LASSO	RF	
Selected	UV index	1	1	1	1	1	1	1
	temperature	3	4	6	2	2	6	3
	sunshine duration	4	7	5	4	4	5	4
	cloud cover	9	8	3	6	6	2	5
	hour	12	2	7	5	8	4	6
	wind bearing	5	3	4	8	9	9	6
	dew point	7	10	9	3	3	7	6
	humidity	2	6	2	13	14	3	6
Discarded	day	11	5	8	7	10	11	8
	pressure	13	11	10	9	5	8	9
	wind speed	6	9	14	14	12	10	10
	precipitation probability	8	13	11	10	11	13	11
	precipitation intensity	10	12	12	11	7	14	11
	minute	14	14	13	12	13	12	13

Neural Networks implementation and optimization

In the following, we present the state-of-art neural networks that we exploit to predict short-term solar radiation and investigate the exogenous input configuration. In detail, these are namely a FNN, anESN, a 1D-CNN, a LSTM and, finally, the machine learning technique of Random Forest (RF). The best architectures for the proposed models were found by using a trial-and-error approach. In particular, the models were trained by using the first four years of observations (2010-2013), while the following year (2014) was used for validation. The models were trained and tested multiple times for each configuration evaluated to minimize the performance fluctuations due to the random initialization of model parameters. Finally, the setting with the lowest error on validation data was selected.

Feedforward Neural Network As start configuration, we have set the number of epochs equal to 500, while the batch size equal to 200 samples. To prevent overfitting and reduce the training time, we adopted the early stopping criteria with the patience of 10 epochs and a validation split equal to 0.1 (10% of the training set). Thus, if the model does not show any improvement in the validation set for 10 epochs, the training is stopped. The training hyperparameters are summarized in Table 3.4, and these are the same for all the following neural networks presented in this phase work.

Table 3.4: Training hyperparameters

hyperparameter	Value
Optimizer	Adam
Loss	Mean sqaured error
Learning rate	0.001
Epochs	500
Batch size	200
Validation split	0.1
Stopping criteria	early stopping with patience equal to 10

The FNN was implemented in Python by using the Keras library with Tensorflow backend [51]. Firstly, we investigated the number of past observations to use as input by evaluating the model performance with a different number of regressors. The FNN obtained the best prediction performance by using the previous 3 observations for each input variable. The final architecture of the FNN was composed of two hidden layers of respectively 100 units and 50 units, and a final output layer of 16 units, corresponding to the number of prediction horizons of interest. The hidden layers used a hyperbolic tangent (tanh) activation function, while the output layer used a linear activation function.

Echo State Network The Echo State Network, in according to Section 2.2.2 and Section 3.4.2, was implemented in Python by using the open-source library *easyesn* [277]. As before, the best hyperparameters for the ESN were found through a trial-and-error approach. Thus, we found that the best prediction performance for the ESN was obtained by using only 1 regressor for each input variable. The final ESN configuration used a reservoir of 100 units with a reservoir density of 0.1. The spectral radius was set to 0.9, while the leaking rate was set to 0.2.

1D Convolutional Neural Network The 1D-CNN was implemented in Python by using the Keras library with Tensorflow backend [51]. The best configuration for the 1D-CNN was achieved by using the past 5 observations of each input variable. The convolutional part of our network used two 1-dimensional convolutional layers with 32 filters and a kernel size equal to 2. The activation function used for each convolutional layer was the hyperbolic tangent function. No pooling layer was inserted after the convolutional layers because the number of features was quite small and did not show a limitation in terms of computational costs. Finally, a fully connected layer with 100 units was added at the end of the network, using a hyperbolic tangent activation function. Like in the FNN, the output layer was constituted of 16 output units with a linear activation function. The full description of the 1D-CNN hyperparameters is provided in table 3.5. The hyperparameters for the training process were the same used for the FNN, and they are summarized in Table 3.4.

Table 3.5: 1D-CNN hyperparameters

Layer type	Units	Filters	Kernel size	Stride	Padding	Activation
Convolution 1d	-	32	2	1	valid	tanh
Convolution 1d	-	32	2	1	valid	tanh
Dense	100	-	-	-	-	tanh
Dense	16	-	-	-	-	linear

Long Short-Term Memory The LSTM neural network was implemented in Python by using the Keras library with Tensorflow backend [51]. The best performance for the LSTM was obtained by using the past 3 observations for each input variable. The final LSTM architecture was composed of two recurrent layers with 50 units using a tangent hyperbolic activation function. Like in the previous networks, the output layer used 16 output units with a linear activation function. The full description of the LSTM hyperparameters is provided in Table 3.6, while Table 3.4 summarize all the training hyperparameters.

Table 3.6: LSTM hyperparameters

Layer type	Units	Activation
LSTM	50	tanh
LSTM	50	tanh
Dense	16	linear

Random Forest A Random Forest is an ensemble of Decision Trees where each predictor is trained by using a different random subset of the training set, sampled via the bagging or the pasting methods. In general, when Decision Trees perform regression tasks, they are called Regression Trees, while the corresponding ensemble model is called Random Forest regressor. The prediction of a Regression Tree is given by the mean target value of the training data reaching the leaf node. The Regression Tree algorithm tries to iteratively split the training data at each node such that the MSE between the target values and mean of target values is as low as possible. The following equation describes the cost function that the training algorithm tries to minimize at each node:

$$J(k, t_k) = \frac{m_{left}}{m} \sum_{i \in left} (\bar{y}_{left} - y_i)^2 + \frac{m_{right}}{m} \sum_{i \in right} (\bar{y}_{right} - y_i)^2 \tag{3.13}$$

where k is the feature chosen for the splitting, t_k is the splitting threshold value, m is the cardinality of the training subset reaching that node, m_{left} and m_{right} are the cardinalities of the next training subsets, y_i are the training subset target values, and $\bar{y}$ is the mean target value. The prediction of a Random Forest regressor is obtained by averaging the predictions of individual trees.

Finally, we have implemented The Random Forest regressor in Python by using the *sklearn* open-source library [200]. The model showed the best prediction performance when using 10 regressors for each variable. The ensemble model was limited to 100 Decision Trees since a greater number of estimators did not show substantial improvements. To prevent overfitting, the minimum number of samples reaching a node was used as stopping criteria. By using a trial-and-error approach, we decided to stop the tree growth when the number of samples reaching a node goes below 100 instances. Table 3.7 summarize the hyperparameters:

Table 3.7: Random Forest hyperparameters

Hyperparameter	Value
Estimators	50
Minimum samples split	100

3.5 Results

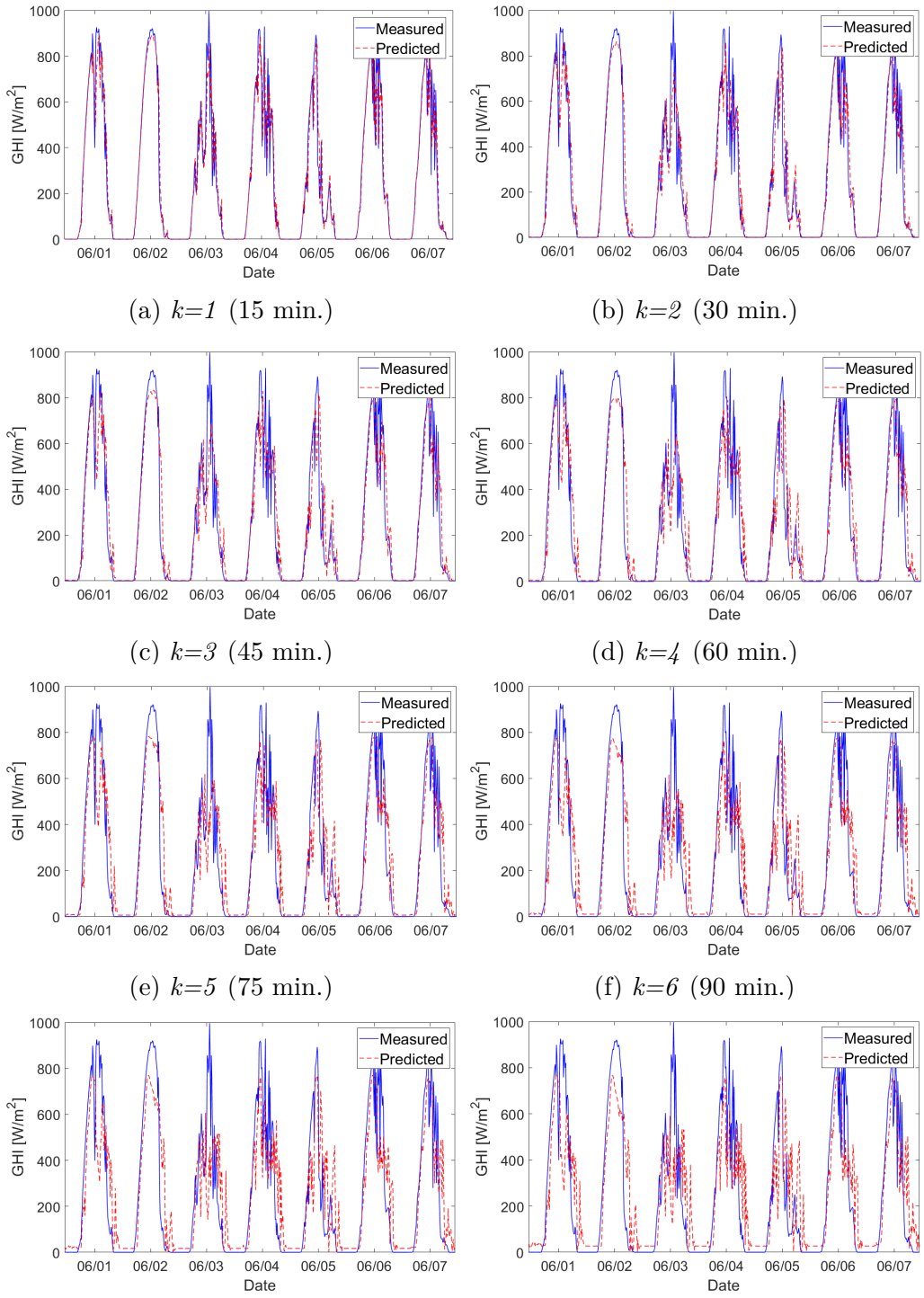
This Section presents and discusses all the experimental results.

3.5.1 Results on GHI forecast in short-term time-horizon

The first experimental results are focused on GHI forecasting in short time windows (i.e. 15 minutes). This is the minimum time interval on which many services for smart grid management work (e.g. Demand/Response [226]). However, we moved further predicting also GHI up to next two hours (again with 15 min time intervals). As described in Section 3.4.1, we implemented two Non-linear Autoregressive Neural Networks, i) NAR with 4 regressors and ii) NARMA with 2 regressors, that exploit the dataset described in Section 3.3. In this section, we present the obtained results, and we also compare and discuss the two different architectures.

To evaluate the performance of our networks, we compare the results of our predictions with real measured values. To achieve this, we used a set of indicators. In according to Section 2.5, these metrics are: i) the RMSD that represents the standard deviation of differences between predicted and observed values, ii) the MAD that represents a measure of statistical dispersion obtained by the average absolute difference of two independent values drawn from a probability distribution, iii) the MBD that measures the average squares of errors between predicted and measured values and vi) the r^2 that represents the proportion between the variance and the predicted variable. All these values are expressed in percentage. Finally, we also considered two other indicators to evaluate the overall network performance: *Willmott's Index of Agreement (WIA)* and *Legates' Coefficient of Efficiency (LCE)*. WIA represents the standardized measure of the degree of model prediction error [259]. LCE is the ratio between the mean square error and the variance in the observed data [142].

Fig. 3.5 and Fig. 3.6 show the results of predictions given by proposed NAR and NARMA compared with real measured values sampled by weather station (dashed and continuous lines, respectively). These results include predictions with eight different time-steps, from $k = 1$ (i.e. next 15 min) to $k = 8$ (i.e next 120 min). Both cases refer to the first seven days of June 2013. Prediction trends of both architectures are very similar. Indeed, they follow with good accuracy the real meteorological trends: i) clear sky, ii) cloudy and iii) rainy conditions, especially for $1 \leq k \leq 3$. Instead for $k > 3$, the prediction accuracy decreases. These aspects are better highlighted by Table 3.8 that reports the results of GHI predictions in terms of performance indicators considering the whole of 2013, which is our validation-set for both architecture. These indexes highlight that the prediction performance worsens by increasing the predictive k-steps. Indeed, GHI predictions for high values of k has a higher error compared with real measurements. In both



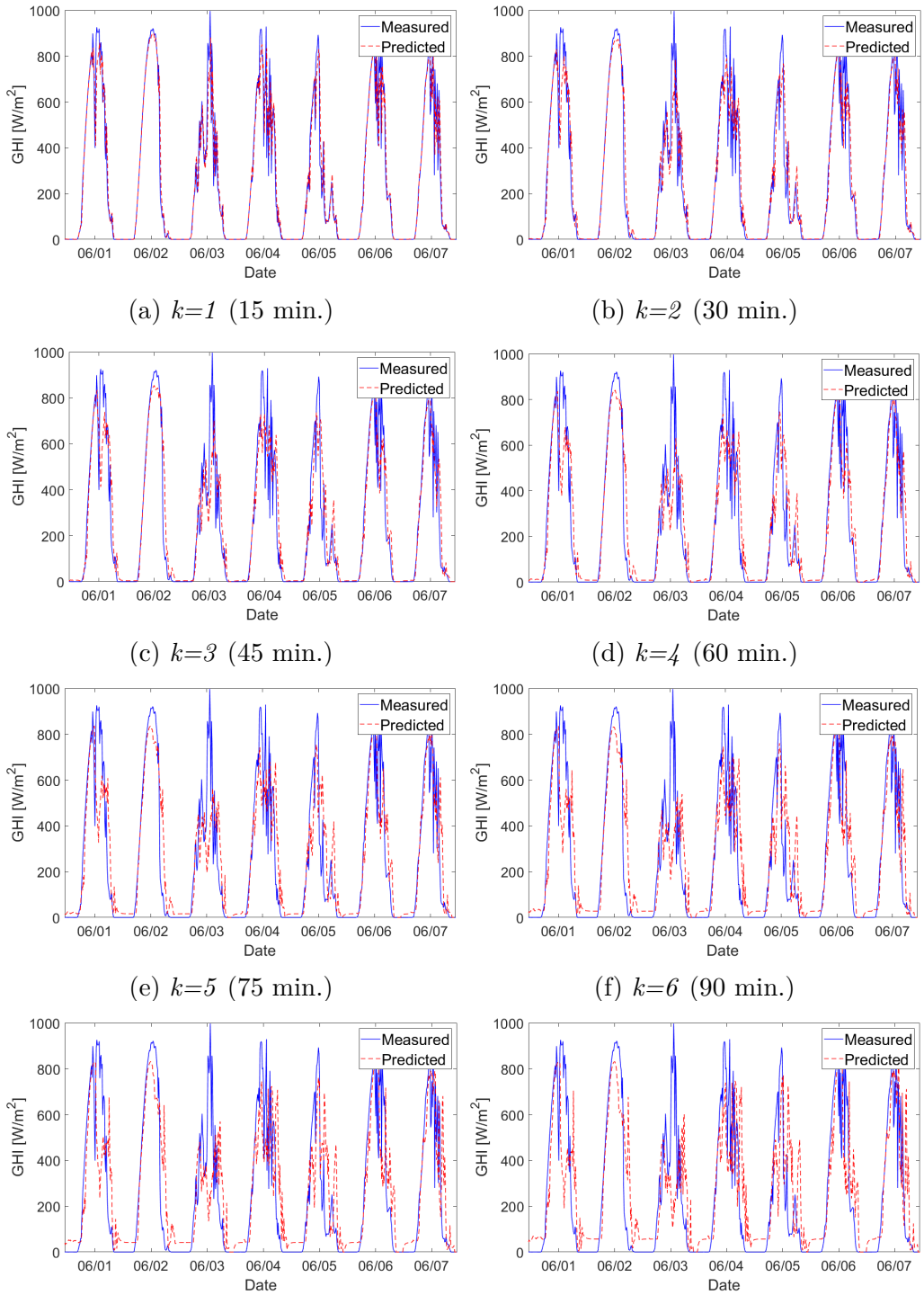
(a) *k=1* (15 min.) (b) *k=2* (30 min.)

(c) *k=3* (45 min.) (d) *k=4* (60 min.)

(e) *k=5* (75 min.) (f) *k=6* (90 min.)

(g) *k=7* (105 min.) (h) *k=8* (120 min.)

Figure 3.5: NAR GHI prediction for $1 \leq k \leq 8$ (June 2013)



(a) k=*1* (15 min.) (b) k=*2* (30 min.)

(c) k=*3* (45 min.) (d) k=*4* (60 min.)

(e) k=*5* (75 min.) (f) k=*6* (90 min.)

(g) k=*7* (105 min.) (h) k=*8* (120 min.)

Figure 3.6: NARMA GHI prediction for $1 \leq k \leq 8$ (June 2013)

cases, the analysis of indexes highlights that the best GHI predictions are given with smaller time intervals. For example, *MAD* reveals that the forecast error grows as the prediction step k increases. Indeed, for $k = 1$, the error is about 12.96% and 12.81% for NAR and NARMA, respectively. Whilst for $k = 8$, the error exceeds the 55%. Also, *RMSD* has a similar trend. A good performance of r^2 is given when its values are closer to 1. In both cases, this happens for a lower k values. For $k = 1$ and $k = 2$, r^2 is over 0.92 for both ANNs. For $k \geq 4$, it decreases down to about 0.70. This trend is also confirmed by both *LCE* and *WIA* that highlight a decreasing of the overall performance on high prediction steps. Under these circumstances, the performance indexes for $1 \leq k \leq 3$ are suitable to perform Photovoltaic energy estimations and simulation results will be discussed in the next Section 3.5.2. With this configuration, the maximum error rate for GHI prediction (expressed in *MAD*) is less than 25%.

Table 3.8: Performance Indicators for GHI predictions

		NAR Neural Network						NARMA Neural Network					
Pred. Steps	Time [min]	MAD [%]	MDB [%]	r^2	RMSD [%]	LCE	WIA	MAD [%]	MDB [%]	r^2	RMSD [%]	LCE	WIA
k=1	15	12.94	0.16	0.95	35.31	0.89	0.99	12.80	0.36	0.95	35.03	0.90	0.99
k=2	30	19.47	0.81	0.92	46.80	0.84	0.98	19.15	1.06	0.92	46.07	0.84	0.98
k=3	45	24.87	1.80	0.89	54.41	0.80	0.97	24.67	2.23	0.89	53.65	0.80	0.97
k=4	60	30.16	3.06	0.86	61.05	0.76	0.96	30.23	4.09	0.86	60.24	0.76	0.96
k=5	75	35.67	4.77	0.83	67.29	0.71	0.95	36.35	6.86	0.83	66.82	0.71	0.95
k=6	105	41.78	7.23	0.79	73.93	0.66	0.94	43.30	10.46	0.79	74.16	0.65	0.94
k=7	115	48.90	10.65	0.75	80.94	0.60	0.92	50.61	14.57	0.74	81.88	0.60	0.92
k=8	120	56.90	15.23	0.70	88.51	0.54	0.90	58.18	19.01	0.69	89.89	0.53	0.90

To train and validate the proposed ANNs, we run our simulations in a server equipped with a CPU 2x Intel Xeon E5-2680 v3 2.50 GHz and 128 Gb of RAM. Table 3.9 reports the execution time for both ANNs considering the three main phases: i) *ANN initialization before Pruning*, ii) *Pruning* and iii) *ANN initialization after Pruning*.

ANN initialization before Pruning refers to the computational time needed to initialize ANNs with random values for the first training and validation. It includes all the steps needed before carrying out the network pruning. As shown in Table 3.9, NAR with 4 regressors needs about 1 min. Whilst, NARMA with 2 regressors needs about 2:30 min because its overall architecture is more complex concerning NAR; hence, it needs more computational resources. This is highlighted during the *Pruning* in Table 3.9, which refers to computational time to evaluate and eliminate unnecessary weights in order to optimize ANNs. This procedure takes around 1 hour for NAR and about 1:48 hour for NARMA. Thus, the optimization

Table 3.9: Computation time for both NAR and NARMA

			NAR	NARMA
Execution Time [hh:mm:ss]	ANN initialization before Pruning		00:01:13	00:02:24
	Pruning		01:04:12	01:47:58
	ANN initialization after Pruning		00:01:07	00:01:22
	Total		01:06:32	01:51:44
	Prediction step	k = 1	00:00:14	00:00:14
		k = 2	00:00:13	00:00:14
		k = 3	00:00:14	00:00:14
		k = 4	00:00:16	00:00:18
		k = 5	00:00:18	00:00:16
		k = 6	00:00:19	00:00:19
		k = 7	00:00:19	00:00:20
		k = 8	00:00:20	00:00:21

process is almost doubled for NARMA concerning NAR. Finally, *ANN initialization after Pruning* is the time needed to train and validate the optimized ANNs. As highlighted in Table 3.9, it dropped to 1:22 min for NARMA concerning the previous *ANN initialization before Pruning* while it is almost constant for NAR.

Once both ANNs are pruned, the computation time to provide GHI forecasts varies between 14 and 21 seconds for $1 \leq k \leq 8$. This enables possible future applications where these ANNs are trained, validated and pruned on servers or cluster systems, since these phases need more computational resources. Then, the optimized ANNs can be deployed on embedded devices to provide GHI forecast. In a smart grid scenario, examples of application that can benefit from this forecast are: i) energy dispatching and load balancing [254], ii) battery management system [209], iii) Demand/Response services [226] and iv) vehicle-to-grid applications [210, 264].

3.5.2 PV energy estimation

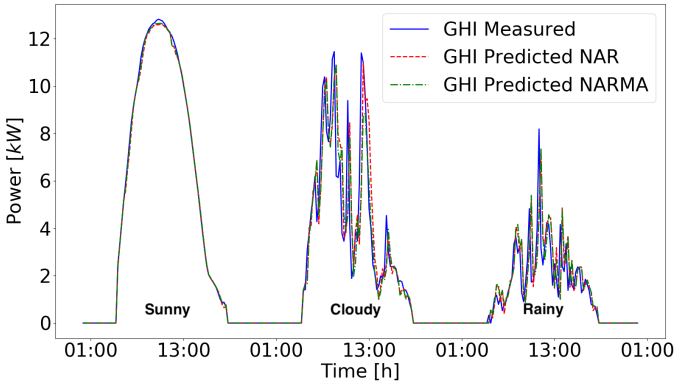
As already discussed in Section 3.5.1, the proposed ANNs forecast GHI in short-term time windows with reasonable accuracy. This allows estimating in advance energy produced by PV systems. To achieve this, we exploited the PV energy simulator (PVsim) presented in [35] that takes as input the GHI forecast resulting by

both NAR and NARMA. The combination of both ANNs and PVsim unlocks development of innovative services and control policies for better management of future smart grids [226] that can also be tested and validated exploiting the methodology in [33].

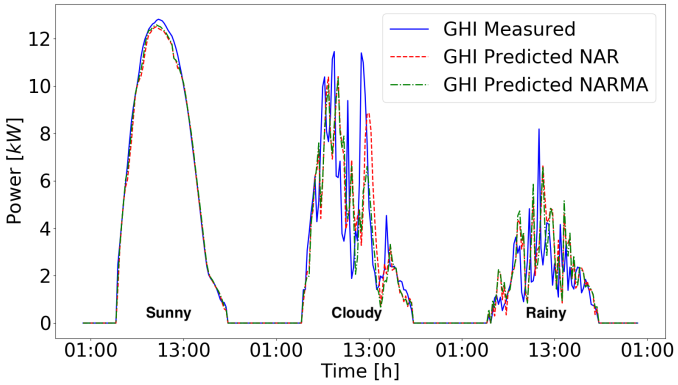
PVsim is a GIS software infrastructure that simulates PV production in real-sky conditions. The inputs for these simulations are i) a *Digital Surface Model* (DSM) and ii) GHI trends. DSM is a digital elevation model that represents terrain elevation, including all objects on it (i.e. buildings). It is used by PVsim to identify rooftops and to simulate the evolution of shadows in clear-sky conditions during the day. Then, this is combined with GHI trends to simulate solar incident radiation and, consequently, PV production in real-sky conditions with a time-resolution of 15 minutes. In a default configuration, PVsim retrieves real GHI trends from a weather station in our University Campus. To forecast PV energy production in short-term time intervals, we interpose our ANNs between weather station's data-source and PVsim. So that, both ANNs get the last real GHI measurements from the weather station and provide the resulting GHI forecast to PVsim.

As mentioned in Section 3.5.1, results of our ANNs for $1 \leq k \leq 3$ on GHI forecast are suitable to perform PV energy estimations. Hereafter, we present results obtained for these three time intervals, i.e. next 15, 30 and 45 minutes. To evaluate the error rate, we compared PVsim results of GHI forecast trends given by NAR and NARMA with those of real GHI trends retrieved by weather station. Fig. 3.7 shows PVsim results for three significant days in June 2013 with different meteorological conditions: i) sunny, ii) cloudy and iii) rainy. Blue continuous-line represents simulations given by real GHI trends, red dashed-line given by NAR GHI trends and green dashed-dotted-line given by NARMA GHI trends. As shown in Fig. 3.7, best performance is achieved when PVsim gets as input results of GHI trends from both ANNs with $k = 1$. This is also confirmed by performance indicators reported in Table 3.10 that considers the whole 2013. Indeed, the accuracy of PV energy estimations decreases by increasing the prediction step k. Regarding PVsim simulations preformed with NAR GHI trends, *MAD* increases from 10.31% to 19.22% for $k = 1$ and $k = 3$, respectively. Also *RMSD* has a similar trend, increasing from 27.96% to 44.65%. *MDB* varies from -0.61 to -2.65. r^2 for $k = 1$ is equal to 0.97. Whilst, the error increases with an $r^2 = 0.92$ for $k = 3$. Finally, *LCE* varies from 0.92 to 0.84 and *WIA* decreases from 0.99 to 0.97. Similar trends are achieved by PVsim simulations preformed with NARMA GHI trends. *MAD* increases from 10.11% for $k = 1$ to 18.47% for $k = 3$. *MDB* is -0.17 for $k = 1$, -0.74 for $k = 2$ and -1.55 for $k = 3$. r^2 varies from 0.97 to 0.92. *RMSD* rises from 27.86% to 43.97%. Finally, *LCE* varies from 0.91 to 0.85 and WIA decreases from 0.99 to 0.98.

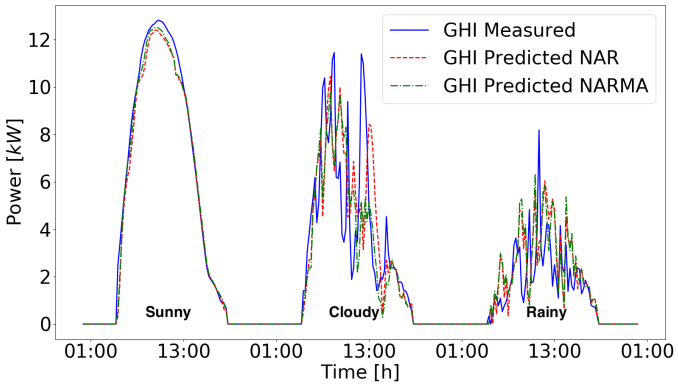
A comparison of these performance indicators highlights that NARMA GHI trends give slightly better PV energy estimations than NAR GHI trends. We can assert that both ANNs can simulate GHI trends in short-term periods with good



(a) *k=1* (15 min.)



(b) *k=2* (30 min.)



(c) *k=3* (45 min.)

Figure 3.7: Simulations of PV energy production given by real NAR and NARMA GHI trends for $1 \leq k \leq 3$ (June 2013).

Table 3.10: Performance Indicator for PV simulation with NAR and NARMA

		NAR Neural Network						NARMA Neural Network					
Pred. Steps	Time [min]	MAD [%]	MDB [%]	r2	RMSD [%]	LCE	WIA	MAD [%]	MDB [%]	r2	RMSD [%]	LCE	WIA
k=1	15	10.31	-0.61	0.97	27.96	0.92	0.99	10.11	-0.17	0.97	27.86	0.91	0.99
k=2	30	15.45	-1.58	0.94	38.15	0.87	0.98	14.93	-0.74	0.94	37.66	0.88	0.98
k=3	45	19.22	-2.65	0.92	44.65	0.84	0.97	18.47	-1.55	0.92	43.97	0.85	0.98

accuracy. This is also confirmed when we use ANNs' results to estimate PV energy production.

3.5.3 State-of-art Neural Network implementation and optimization for GHI forecast in short- and mid-term time-horizon

As second analysis, we focused on how to improve predictions of GHI values. For this purpose, we selected some of the state-of-art neural networks optimizing the implementation process concerning the case study. Also, as a detail in Section 3.4.2, we investigated some pre-processing techniques to obtain better performances in terms of prediction accuracy and computation level. Therefore, in this Section, we present our experimental results. First, we analyze and compare the prediction models. Then, we prove that by using a multi-output artificial neural network for predictions with many steps ahead, we can obtain better results than the iterative method, justifying the choice of the former approach. Finally, we describe the prediction performances obtained with raw GHI data, Tikhonov regularisation and clear-sky index.

Iterative vs. multi-output networks

As introduced in section 2.4, we have applied two different methodologies for multi-step predictions, iterative and multi-output. The former uses a single-step model and iteratively generates multiple predictions; the latter gives the desired n predictions exploiting a single step of calculation (i.e. n steps in the future). Figure 3.8 shows the comparison for the FFNN exploiting raw GHI data with time horizon from 15 min to 2 h.

The performance is similar for the first 30 minutes, but the multi-output network starts improving for mid-term time horizons. Increasing the time horizon will result in the accumulation of the error, further widening the gap between the two approaches. Figure 3.9 depicts the same experiment for the LSTM network still exploiting raw GHI. Again, the experimental results report similar behaviour with the LSTM network for the first 30 min. Instead, we do not apply our iterative approach to NAR and ESN because their models already embed a feature to

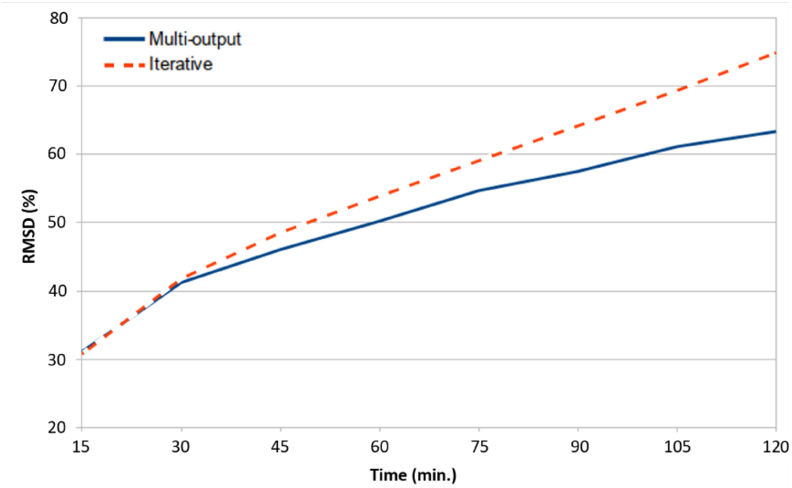


Figure 3.8: Comparison of RMSD for iterative and multi-output predictions using a Feed-Forward Neural Network (raw GHI)

perform multi-step predictions. These results justify our choice of using networks with multiple outputs to predict GHI many steps ahead, in accordance with [134]. Consequently, all the results illustrated in the following sections are based on the multi-output approach.

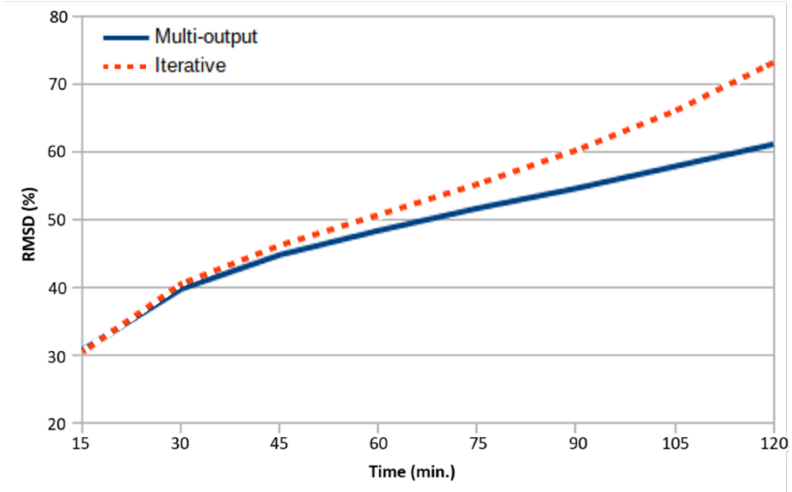


Figure 3.9: Comparison of RMSD for iterative and multi-output predictions using a Long Short-Term Memory (raw GHI).

Table 3.11: MAD (%), R^2, and RMSD (%) of NAR predictions

Model	Indicator	15 min	30 min	45 min	60 min	75 min	90 min	105 min	120 min
NAR-7 (7 regressors)	MAD	10.79	16.95	21.64	26.12	30.79	35.79	41.32	47.48
	R^2	0.96	0.93	0.91	0.89	0.86	0.83	0.80	0.76
	RMSD	30.50	41.29	47.78	53.30	58.79	64.60	70.78	77.26
NAR-12 (12 regressors)	MAD	10.85	16.74	21.03	24.90	28.86	32.84	36.96	41.39
	R^2	0.96	0.93	0.91	0.89	0.87	0.85	0.83	0.81
	RMSD	30.43	40.62	46.62	51.41	56.14	60.65	65.05	69.55

GHI prediction exploiting raw GHI data

In this section, we present the performance of our neural networks optimised to use the raw GHI dataset for both training and test (see Section 3.4.2).

Starting from NAR, we used as reference the architecture deeply described in [8]. Consequently, we design the same network (with 7 regressors and 30 neurons, before pruning), exploiting the very same dataset (for both training and test) to guarantee a fair comparison. Then, as described in Section 3.4.2, different numbers of regressors were evaluated, trying to improve the performance and find the best network. Table 3.11 reports the comparison of raw GHI predictions between the reference model with 7 regressors [8] (hereinafter NAR-7) and our best model with 12 regressors (hereinafter NAR-12) in terms of MAD, R^2 and RMSD.

The results report that NAR-12 performs slightly better than NAR-7 for all three statistical indicators, particularly for longer time horizons. Up to 45 min, the performance of predictions is quite similar. The improvements can be noted from 60 min onward. At 120 min, NAR-12 performs better than NAR-7 with improvements of about 6%, 0.05 and 8% for MAD, R^2 and RMSD, respectively. Consequently, we chose this new model to be compared with the other networks.

For FFNN, LSTM and ESN, we have tested different configurations, as described in Section 3.4.2, and we compared their RMSD values to decide which parameters give the best results.

In the following, we report the configuration of each neural network:

- **FFNN:** 19 regressors and 2 hidden layers; the first layer with 200 neurons and the second with 100 neurons.
- **LSTM:** 16 regressors with 2 hidden layers; the first layer with 10 neurons and the second with 5 neurons.
- **ESN:** 3 regressors with a reservoir of 500 neurons.

Looking at these parameters, it can be noted that both FFNN and LSTM work better with a higher number of regressors, while the ESN has the best performance with just 3 regressors. As for the size of the networks, the ESN works well with a big reservoir, as recognised in literature [149]. The FFNN also has better results

using a significant amount of units. Whilst, the LSTM works better with a smaller architecture.

Our first expectation was that the multi-output architecture used for multi-step prediction improves the results of the NAR, particularly for longer time horizons. We also expected that the LSTM would perform better than the FFNN. In fact, its recurrent structure should be more suitable to model the temporal behaviour of GHI time-series. Moreover, LSTM has shown excellent performance in many tasks, including time-series forecasts [98].

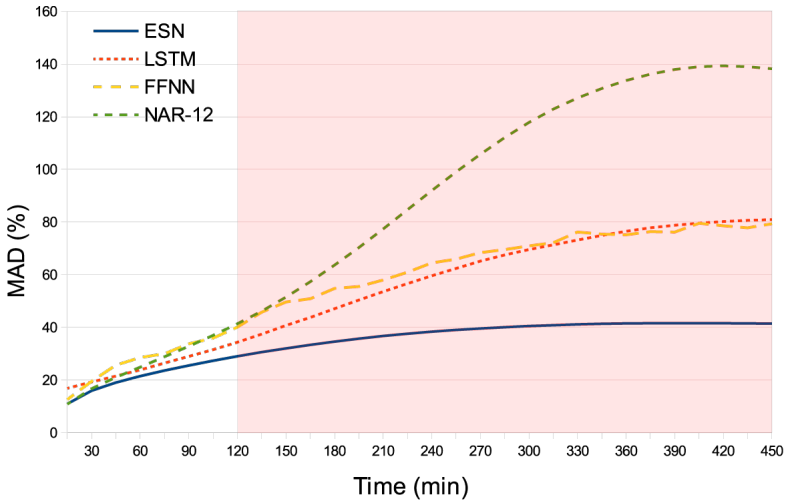


Figure 3.10: MAD over time for GHI prediction.

Figures 3.10 to 3.12 show plots of MAD, R^2 and RMSD over the time for all the proposed neural network architectures. These plots show that the results are very similar for short-term predictions (i.e. up to 60 min). Then, the four trends start diverging each other for all three indicators. Performance of NAR-12 (see the green dotted line) rapidly decreases compared to the other networks. Contrary to our expectations, LSTM does not outperform the FFNN (see red and yellow dotted-lines, respectively). Both architectures have very similar trends with a few negligible differences. These plots strongly highlight the outstanding results of the ESN for mid-term forecasts, which outperforms the other neural networks. Table 3.12 details the values of MAD, R^2 and RMSD for the four networks up to 120 min ahead predictions. In our view, this is a good time horizon in which the forecast error is still acceptable. As reported in Table 3.12, at 120 min ESN outperforms the other neural networks with $MAD = 29.03\%$, $R^2 = 0.88$ and $RMSD = 55.22\%$, clearly improving the performance of about 12%, 0.07 and 14% (for MAD, R^2, and RMSD, respectively) w.r.t. NAR-12.

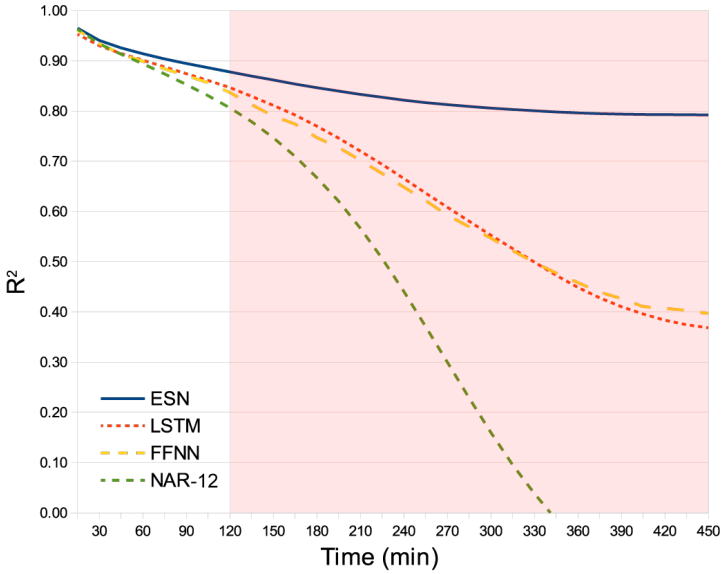


Figure 3.11: R^2 over time for GHI prediction.

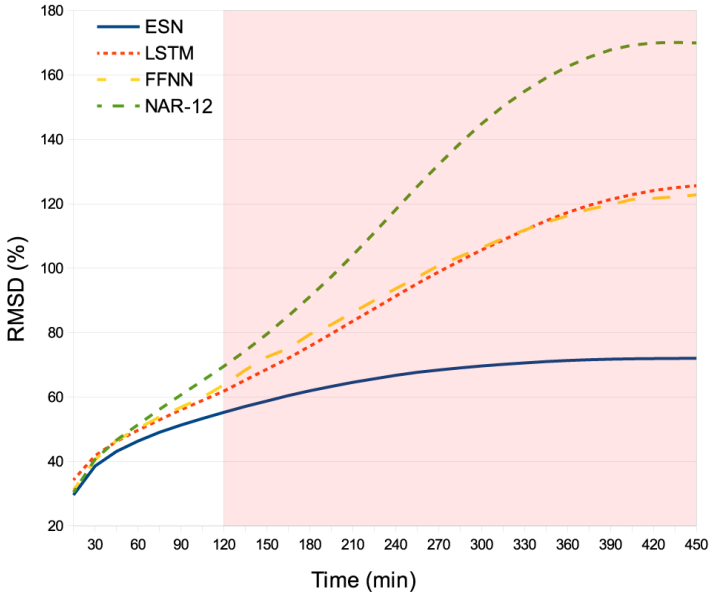


Figure 3.12: RMSD over time for GHI prediction.

GHI prediction exploiting Tikhonov regularisation

As a second analysis, we focused on predictions based on the Tikhonov regularisation applied to the raw GHI dataset, as described in Section 3.4.2. As discussed in [6], Tikhonov regularisation does not apply to real-time data. Thus, we exploited

Table 3.12: MAD (%), R^2, and RMSD (%) for GHI predictions with raw data

Network	Indicator	15 min	30 min	45 min	60 min	75 min	90 min	105 min	120 min
NAR-12	MAD	10.85	16.74	21.03	24.90	28.86	32.84	36.96	41.39
	R^2	0.96	0.93	0.91	0.89	0.87	0.85	0.83	0.81
	RMSD	30.43	40.62	46.62	51.41	56.14	60.65	65.05	69.55
FFNN	MAD	12.62	19.53	25.66	28.55	30.00	33.76	35.94	40.10
	R^2	0.96	0.93	0.91	0.90	0.88	0.87	0.86	0.84
	RMSD	31.15	40.54	46.49	50.20	53.81	56.81	59.85	63.78
LSTM	MAD	16.87	19.29	21.51	23.86	26.47	28.98	31.54	34.36
	R^2	0.95	0.93	0.91	0.90	0.89	0.87	0.86	0.85
	RMSD	34.27	41.75	46.20	49.67	52.91	56.03	58.92	61.84
ESN	MAD	10.96	15.97	19.05	21.49	23.62	25.53	27.34	29.03
	R^2	0.96	0.94	0.93	0.91	0.90	0.89	0.89	0.88
	RMSD	29.60	38.55	43.10	46.31	49.03	51.28	53.31	55.22

the "hybrid" approach presented in [6] in which the Tikhonov regularisation is applied only on the training set used for training our ANNs. Instead, the test-set, used to asses the performance of our ANNs, consists of raw GHI data.

Since the dataset is different w.r.t our previous NAR-7, we applied the very same methodology described in [8] to design the best NAR suitable for this training-set pre-processed with Tikhonov regularisation. The resulting architecture (hereinafter NAR-10) is characterised by 10 regressors and 21 neurons, before pruning. To design and train the best network architecture for FFNN, LSTM and ESN, we applied the same trial-and-error approach described in Section 3.5.3. In the following, we report the configuration of each neural network:

- **FFNN:** 10 regressors and 2 hidden layers; the first with 50 neurons and the second with 25 neurons.
- **LSTM:** 16 regressors with 2 hidden layers; the first with 10 neurons and the second with 5 neurons.
- **ESN:** 3 regressors with a reservoir of 100 neurons.

Table 3.13 reports the values of MAD, R^2 and RMSD for the four ANN up to 120 min ahead predictions. Comparing these results with those in Table 3.12 (i.e. prediction performance exploiting raw GHI data), we can notice that the performance of all the ANN gets worse when we exploit this "hybrid" approach. In general, the error on predictions is too high, and it is not acceptable. This is also confirmed by the trends reported in Figures 3.13 to 3.15. Thus, this "hybrid" approach is not suitable for this application scenario.

If we apply the Tikhonov regularisation also on the test-set, performances significantly improve in average of about 10%, 1 and 25% for MAD, R^2 and RMSD,

Table 3.13: MAD (%), R^2, and RMSD (%) for networks trained with Tikhonov filtered data, inference using raw GHI

Network	Indicator	15 min	30 min	45 min	60 min	75 min	90 min	105 min	120 min
NAR-10	MAD	13.83	25.01	35.07	45.13	55.61	66.64	78.17	89.46
	R^2	0.93	0.81	0.66	0.49	0.27	0.01	−0.30	−0.61
	RMSD	41.65	68.45	91.20	113.03	135.10	157.53	180.69	200.80
FFNN	MAD	16.66	23.37	30.35	38.41	47.06	55.64	64.08	72.56
	R^2	0.93	0.89	0.82	0.72	0.58	0.41	0.21	−0.01
	RMSD	42.20	53.55	67.29	84.05	102.96	121.84	140.40	158.60
LSTM	MAD	15.65	19.16	23.41	28.23	33.00	37.93	42.90	47.80
	R^2	0.95	0.92	0.88	0.84	0.79	0.74	0.69	0.63
	RMSD	36.72	45.71	54.47	63.18	71.78	79.97	87.76	95.53
ESN	MAD	12.17	21.10	29.30	37.80	46.68	55.50	63.98	72.47
	R^2	0.95	0.87	0.78	0.67	0.54	0.39	0.22	0.05
	RMSD	35.73	56.01	73.69	90.94	107.73	123.66	139.19	153.80

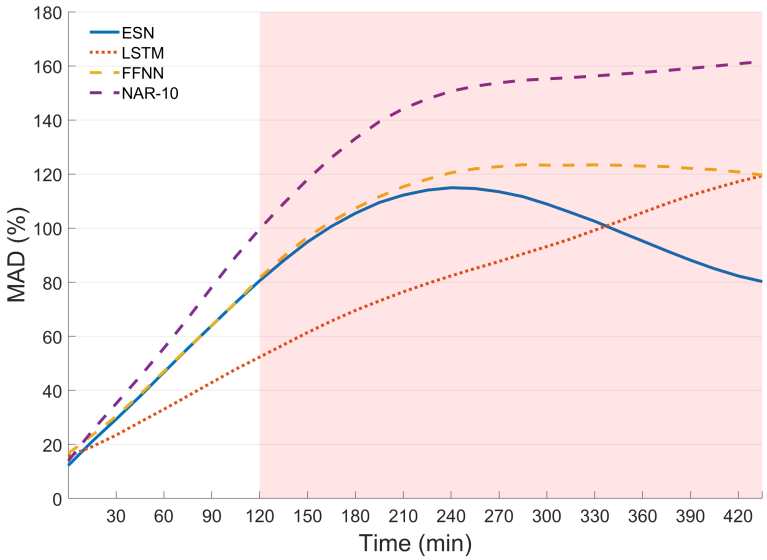


Figure 3.13: MAD trend evolution for predictions with "hybrid" approach based on Tikhonov regularisation.

respectively (see Table 3.14 and Figures 3.16, 3.17 and 3.18. However, as previously anticipated, the Tikhonov regularisation should be rethought to work in real-time. This because, generally, the Tikhonov filter methodology should be applied to the whole data set. Instead, in a real application, exploiting real-time data, this is not applicable since new data would have to be filtered when it becomes available.

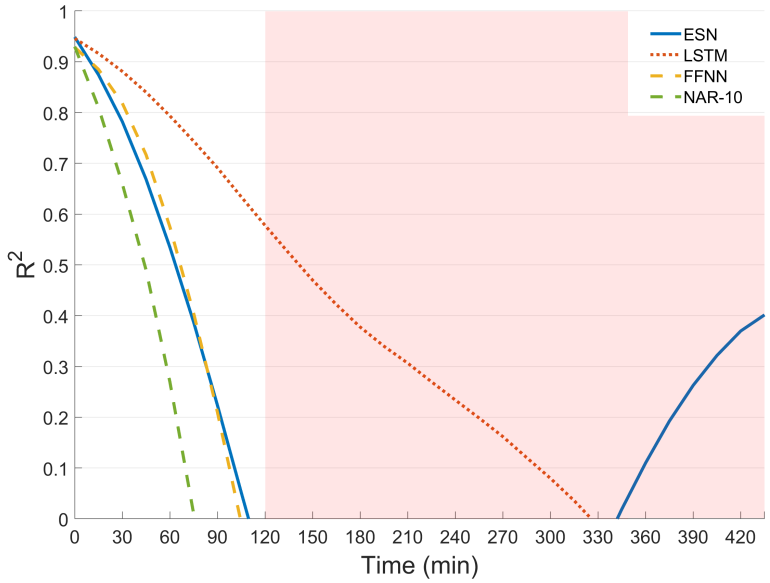


Figure 3.14: R^2 trend evolution for predictions with "hybrid" approach based on Tikhonov regularisation.

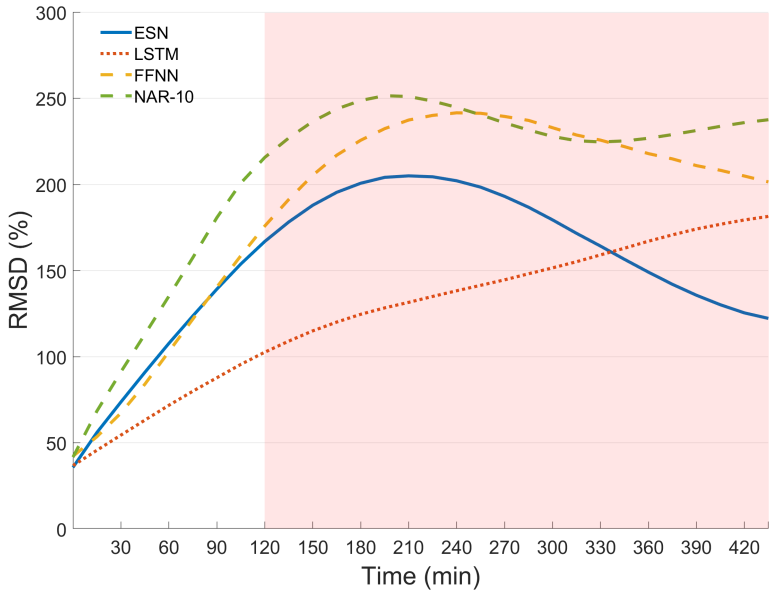


Figure 3.15: RMSD trend evolution for predictions with "hybrid" approach based on Tikhonov regularisation.

Clear-sky index prediction

As a third analysis, we focus on predictions exploiting the clear-sky index (K_c). The same network configurations for the NAR, FFNN, LSTM and ESN used for

Table 3.14: MAD (%), R^2, and RMSD (%) for networks trained with Tikho. filtered data, inference using Tikho. filtered GHI

Network	Indicator	15 min	30 min	45 min	60 min	75 min	90 min	105 min	120 min
NAR-10	MAD	0.95	2.56	4.72	7.39	10.56	14.23	18.44	23.24
	R^2	1.00	1.00	1.00	0.99	0.99	0.98	0.97	0.95
	RMSD	3.28	5.49	8.37	11.92	16.14	21.02	26.64	32.90
FFNN	MAD	5.15	6.78	8.17	9.27	10.06	10.50	11.09	12.08
	R^2	1.00	1.00	1.00	0.99	0.99	0.99	0.98	0.98
	RMSD	6.21	8.47	10.72	12.88	14.87	16.85	19.33	22.42
LSTM	MAD	3.38	2.85	3.12	3.82	4.75	5.83	7.07	8.50
	R^2	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.99
	RMSD	4.99	4.68	5.40	6.77	8.56	10.67	13.05	15.68
ESN	MAD	0.65	1.46	2.51	3.79	5.31	7.06	9.00	11.13
	R^2	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.98
	RMSD	2.76	4.35	6.26	8.47	10.99	13.81	16.90	20.21

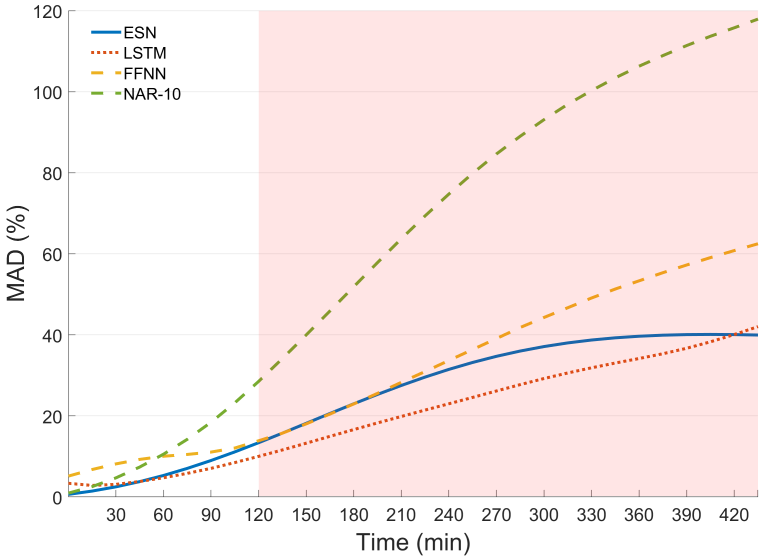


Figure 3.16: MAD trend evolution for predictions with Tikhonov regularisation applied also to the test-set.

the GHI prediction exploiting Tikhonov regularisation (see Section 3.5.3) were also applied to this new dataset.

The transformation of GHI into K_c removes the seasonal trends due to the changing position of the sun during the year, and the clear-sky model already takes into account the atmosphere turbidity. So the networks only have to predict the stochastic component due to clouds. The expectation is that the results will be better than those described in the previous sections, because these filtered data

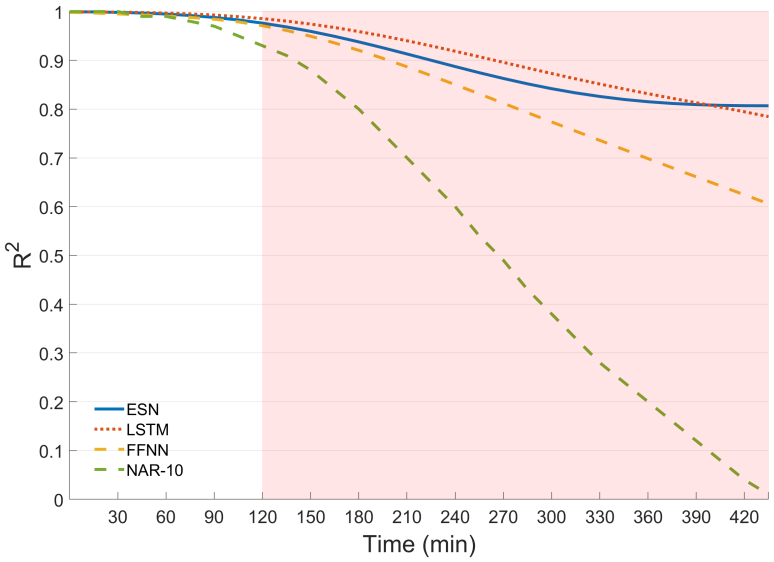


Figure 3.17: R^2 trend evolution for predictions with Tikhonov regularisation applied also to the test-set.

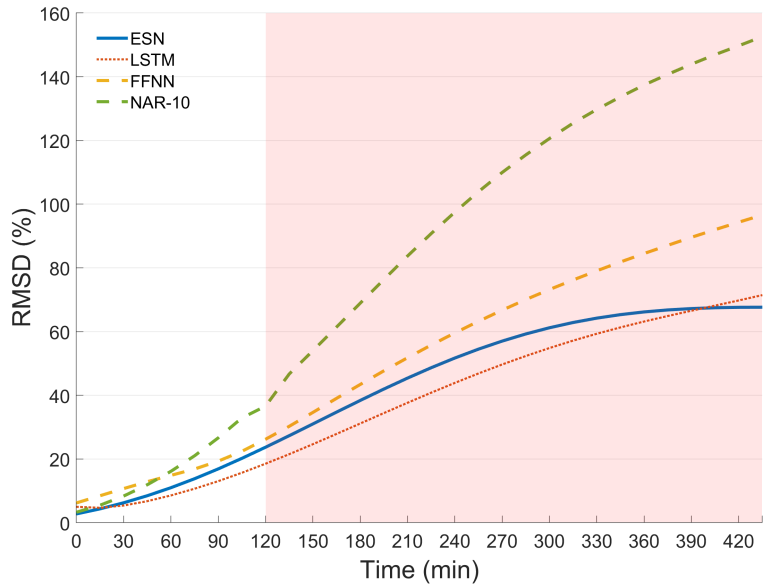


Figure 3.18: RMSD trend evolution for predictions with Tikhonov regularisation applied also to the test-set.

should be easier to model for the neural networks. The analysis of the obtained results confirms this assumption. As shown in Figures 3.19 to 3.21, the gap between ESN and the other three neural networks is reduced in terms of MAD, R^2, and RMSD. The ESN has only a slight improvement but still is more performing than

the others. It appears that the ESN was already able to extract the seasonal trends from the raw GHI data better than the other networks. When only the stochastic component needs to be predicted and the rest is handled by the clear-sky model, the differences among the networks greatly decreases. As reported in Table 3.15, at 120 min ESN is still outperforming the other neural networks with $MAD = 25.47\%$, $R^2 = 0.88$ and $RMSD = 55.02\%$. Comparing the ESN with the NAR-10, which gives worst performances among the four neural networks, there is an improvement of about 0.6%, 0.03 and 5.4% (for MAD, R^2, and RMSD, respectively).

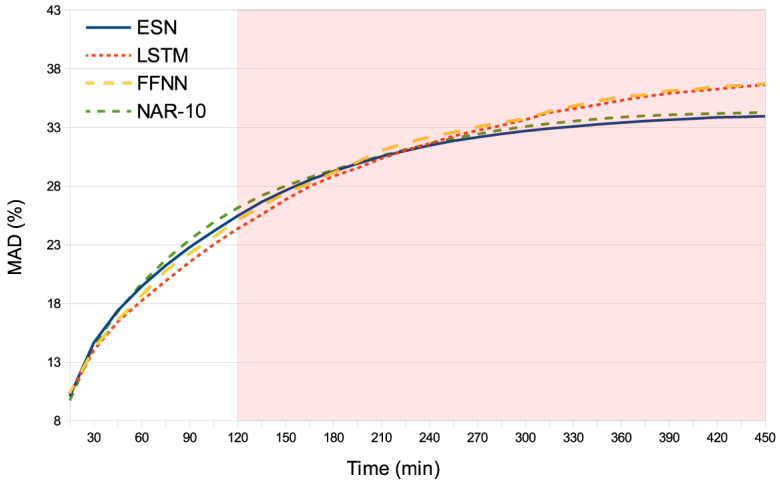


Figure 3.19: MAD over time for K_c prediction

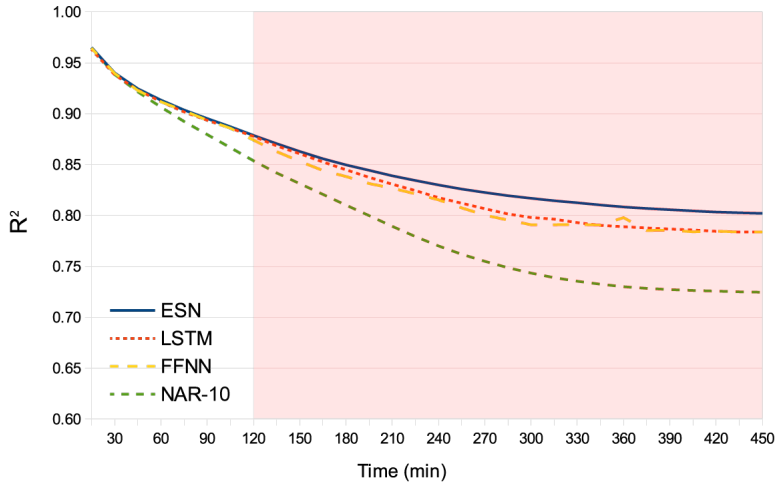


Figure 3.20: R^2 over time for K_c prediction

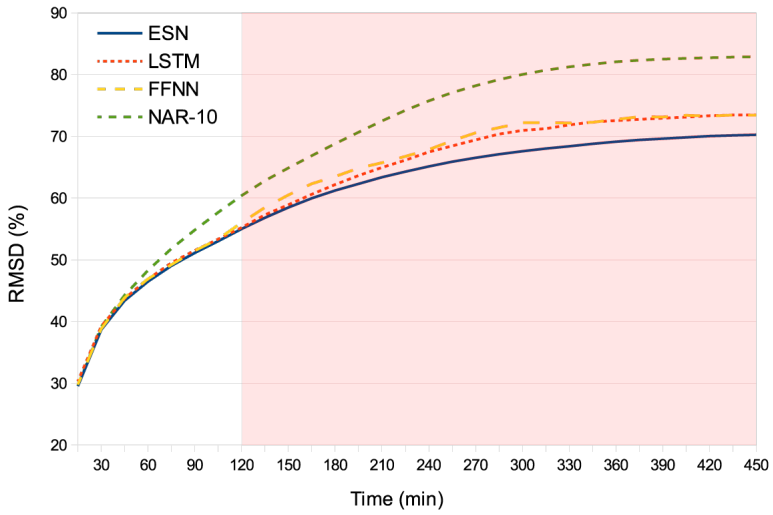


Figure 3.21: RMSD over time for K_c prediction

Table 3.15: MAD (%), R^2, and RMSD (%) for K_c prediction

Network	Indicator	15 min	30 min	45 min	60 min	75 min	90 min	105 min	120 min
NAR-10	MAD	9.75	14.43	17.31	19.71	21.72	23.44	24.93	26.15
	R^2	0.96	0.94	0.92	0.91	0.89	0.88	0.87	0.85
	RMSD	29.61	39.18	44.28	48.32	51.83	54.85	57.71	60.42
FFNN	MAD	10.38	14.20	16.59	18.71	20.80	22.28	23.66	25.14
	R^2	0.96	0.94	0.92	0.91	0.90	0.89	0.89	0.87
	RMSD	29.81	38.83	43.89	46.93	49.21	51.49	53.39	56.04
LSTM	MAD	10.30	14.02	16.46	18.24	19.92	21.54	23.03	24.38
	R^2	0.96	0.94	0.92	0.91	0.90	0.89	0.89	0.88
	RMSD	30.34	39.25	43.75	46.90	49.52	51.54	53.37	55.24
ESN	MAD	10.10	14.66	17.40	19.47	21.24	22.80	24.17	25.47
	R^2	0.97	0.94	0.92	0.91	0.90	0.90	0.89	0.88
	RMSD	29.56	38.71	43.42	46.51	49.05	51.14	53.05	55.02

Final remarks on prediction results

Considering the results discussed in the previous sections and reported in Figure 3.22, the main findings can be summarised as follows:

- designing ANNs following the multi-output approach to forecast GHI provides better performances than the iterative approach;
- the Echo State Network is the ANN architecture that better performs among those tested;

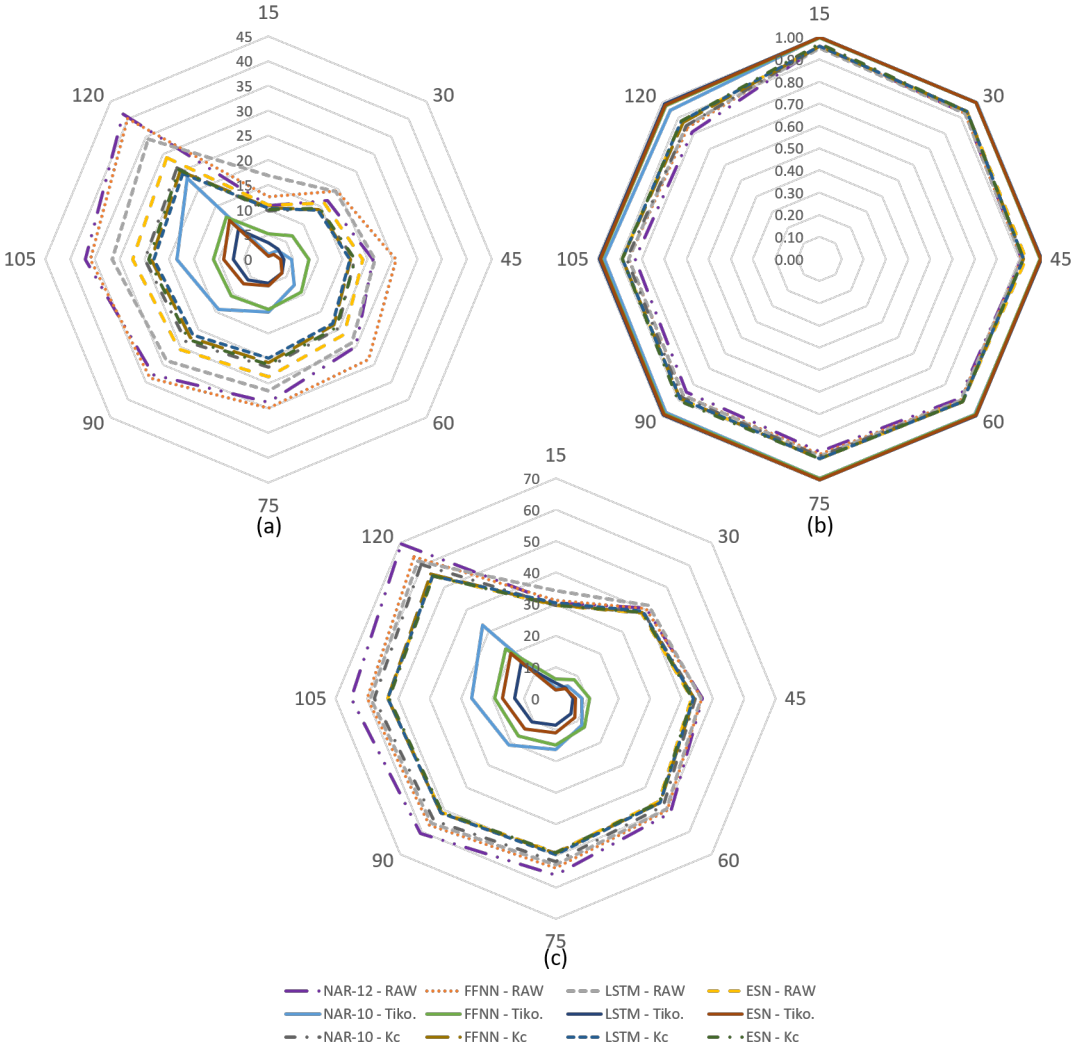


Figure 3.22: Performance comparison of all the presented ANN: (a) MAD, (b) R^2 and (c) RMSD.

- using the clear-sky index the prediction accuracy significantly improves and allows to use smaller networks with fewer regressors;
- best performances are achieved when Tikhonov regularisation is applied. However to be suitable for GHI forecasts, it should be rethought to work in real-time;
- 120 min is the maximum time horizon reached by our ANNs in which the forecast error is still acceptable.

3.5.4 Multivariate configurations for GHI forecast in short- and mid-term time-horizon

Finally, we start our investigation on the exploitation of exogenous inputs in the training phase of neural networks for GHI forecast in short- and mid-term time-horizon. Therefore, in this section, we present the prediction results obtained by using the machine learning models illustrated in Section 3.4.3. In this way, we demonstrate the effectiveness of using exogenous inputs for solar radiation forecasting by comparing the obtained results against a model using only endogenous inputs, detailing the best multivariate configuration found.

Models evaluation

To evaluate how far ahead our models can still achieve acceptable forecasting errors, we performed multi-step ahead predictions with a time step of 15 minutes. Therefore, we adopted a Multiple-Input Multiple-Output (MIMO) approach [26], where a single model with multiple outputs allows to predict multiple values at once.

In according to Section 2.5, we exploit some analytical indexes to evaluate and compare the prediction performances of our predictive models. To overcome the performance variations due to the random initialization of parameters, each model was trained multiple times, using the final 10% of the training set for validation. Thus, the model with the lowest validation error was finally evaluated on the test set. Tables 3.17, 3.18 and 3.19 show the forecasting error for each prediction horizon in terms of MAD, RMSD and R^2, respectively. The underlined values indicate the model with the lowest error for that specific prediction horizon. In case of similar errors, the model with fewer parameters was selected. Figures 3.23, 3.24 and 3.25 provide a visual comparison of the prediction errors, where the red area indicates an excessive degradation in the prediction performance. Overall, the LSTM demonstrated the best prediction performance among the five models, producing acceptable forecasting errors up to 4 hours ahead. The FNN and the 1D-CNN also demonstrated excellent prediction performance, comparable to those of the LSTM for prediction horizons shorter than 2 hours. The ESN presented the highest forecasting errors, revealing poor prediction performance in modelling multivariate time series. The RF performed slightly better than the ESN, showing promising results. The figures between 3.30 and 3.29 compare the GHI values predicted by the LSTM against real observed GHI values for three days with different weather conditions, from 1-h to 4-h ahead.

The experiments were run on a computer equipped with an Intel Core i5 processor with 1.4 GHz and 4 GB of main memory. The execution times for training and testing each model are reported in table 3.16.

Table 3.16: Training and test execution times.

Model	Training time [mm:ss]	Test time [mm:ss]
RF	00:45	00:11
FNN	01:01	00:01
ESN	02:51	00:12
LSTM	05:05	00:06
CNN	04:31	00:03

Table 3.17: MAD results.

Pred. horizon	Models				
	RF	FNN	ESN	LSTM	CNN
15 mins	11.40	11.15	**10.39**	10.74	11.00
30 mins	14.94	14.61	14.79	14.11	**13.90**
45 mins	16.90	16.04	17.36	15.96	**15.71**
60 mins	18.37	17.47	19.32	17.25	**17.22**
75 mins	19.47	18.62	20.89	18.36	**18.28**
90 mins	20.43	19.54	22.21	**19.16**	19.40
105 mins	21.32	20.54	23.36	**19.98**	20.32
120 mins	22.12	21.20	24.32	20.80	**20.78**
135 mins	22.87	21.98	25.10	**21.28**	21.51
150 mins	23.57	22.74	25.76	**22.03**	22.44
165 mins	24.17	23.44	26.31	**22.66**	22.90
180 mins	24.77	23.65	26.74	**22.95**	23.58
195 mins	25.31	24.48	27.15	**23.37**	24.89
210 mins	25.82	24.84	27.57	**23.83**	25.07
225 mins	26.33	25.51	27.97	**24.09**	24.75
240 mins	26.84	26.24	28.34	**24.51**	25.02

Improvement of exogenous inputs

To demonstrate the effectiveness of using exogenous inputs for short-term solar radiation forecasting, we compared the five multivariate models with their univariate counterparts trained by using only endogenous inputs. Figures 3.30, 3.31 and 3.32 provide a visual comparison between the univariate and multivariate models in terms of MAD, RMSD and R^2, respectively. According to the results, the models using both endogenous and exogenous inputs demonstrated better prediction performance with respect to the models using only endogenous inputs. In particular, the performance improvements due to exogenous inputs are more pronounced for longer prediction horizons, while for shorter prediction horizons the

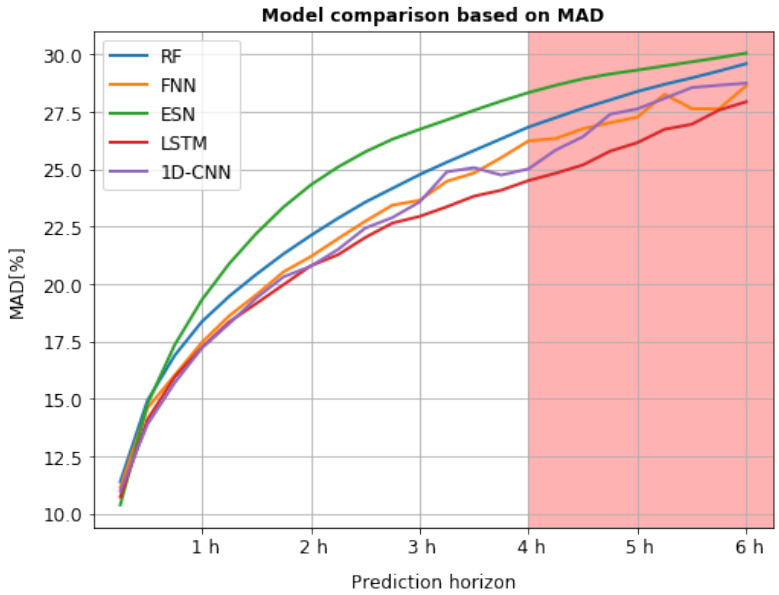


Figure 3.23: Model comparison based on MAD.

Table 3.18: RMSD results

Pred. horizon	Models				
	RF	FNN	ESN	LSTM	CNN
15 mins	30.39	29.92	**29.22**	29.81	29.73
30 mins	37.95	37.95	37.56	37.54	**36.91**
45 mins	41.63	41.16	41.71	41.10	**40.79**
60 mins	44.14	43.55	44.51	43.31	**43.18**
75 mins	45.87	45.35	46.68	**44.96**	45.26
90 mins	47.29	46.51	48.42	**46.28**	46.46
105 mins	48.65	47.83	49.96	**47.45**	47.51
120 mins	49.90	48.99	51.35	**48.55**	48.75
135 mins	51.08	49.90	52.54	**49.51**	49.75
150 mins	52.11	50.90	53.65	**50.36**	50.56
165 mins	53.07	52.01	54.58	51.35	**51.21**
180 mins	53.99	52.94	55.29	**52.11**	52.26
195 mins	54.82	53.82	56.01	**52.82**	53.71
210 mins	55.62	54.58	56.70	**53.66**	54.16
225 mins	56.37	55.32	57.39	**54.44**	54.60
240 mins	57.14	56.31	58.09	**55.19**	55.75

performance of the two approaches are very similar. Tables 3.20, 3.21 and 3.22

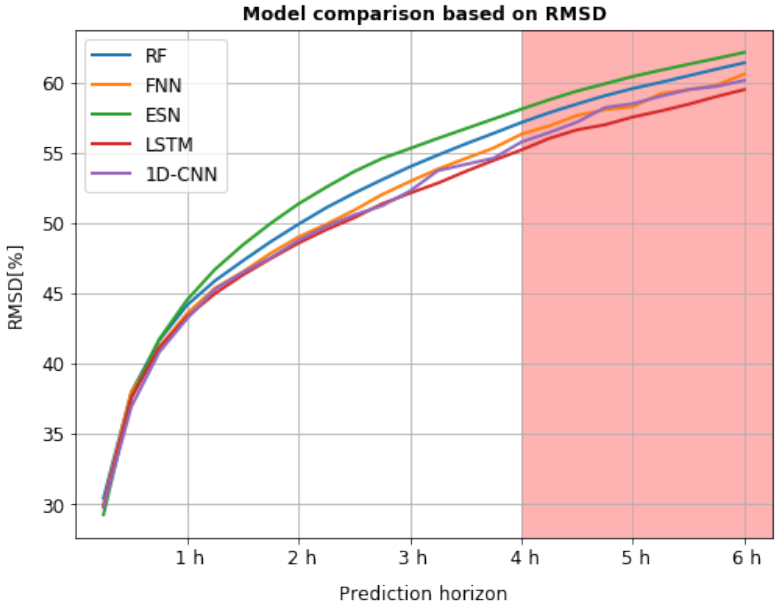


Figure 3.24: Model comparison based on RMSD.

Table 3.19: R^2 results

Pred. horizon	Models				
	RF	FNN	ESN	LSTM	CNN
15 mins	0.96	0.96	**0.97**	0.96	0.96
30 mins	0.94	0.94	0.94	0.94	**0.95**
45 mins	**0.93**	0.93	0.93	0.93	0.93
60 mins	0.92	0.92	0.92	0.92	**0.93**
75 mins	**0.92**	0.92	0.91	0.92	0.92
90 mins	**0.91**	0.91	0.91	0.91	0.91
105 mins	**0.91**	0.91	0.90	0.91	0.91
120 mins	0.90	0.90	0.89	**0.91**	0.90
135 mins	**0.90**	0.90	0.89	0.90	0.90
150 mins	0.89	**0.90**	0.88	0.90	0.90
165 mins	**0.89**	0.89	0.88	0.89	0.89
180 mins	0.88	**0.89**	0.88	0.89	0.89
195 mins	0.88	0.88	0.87	**0.89**	0.88
210 mins	**0.88**	0.88	0.87	0.88	0.88
225 mins	0.87	**0.88**	0.87	0.88	0.88
240 mins	0.87	0.87	0.86	0.88	**0.88**

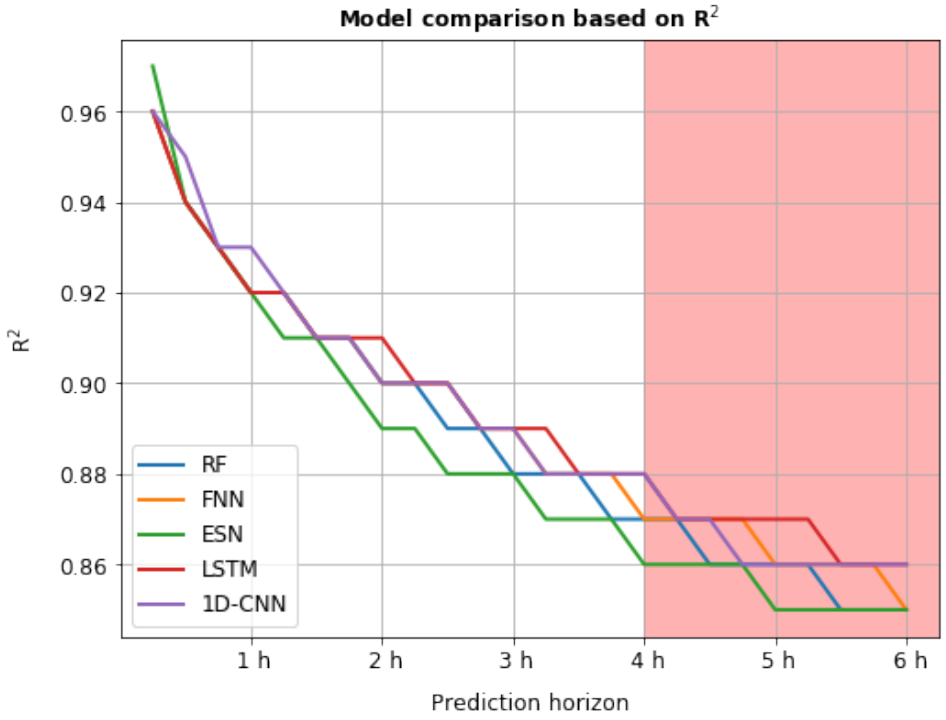


Figure 3.25: Model comparison based on R^2.

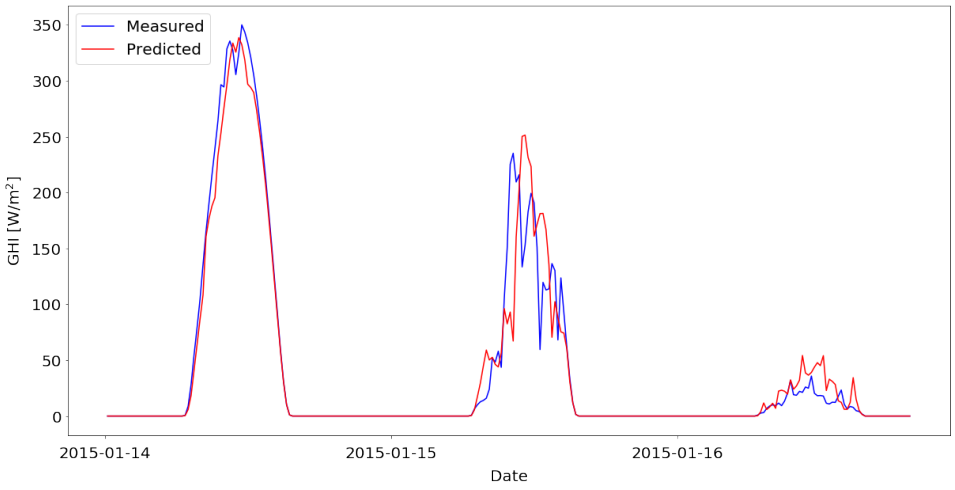


Figure 3.26: GHI prediction with LSTM for 1-h ahead.

compare the models using exogenous inputs with the models using only endogenous inputs in terms of MAD, RMSD and R^2, respectively. The underlined values indicate the model with the lowest error for that specific prediction horizon. In case of similar errors the model with less parameters was selected. In the third column we

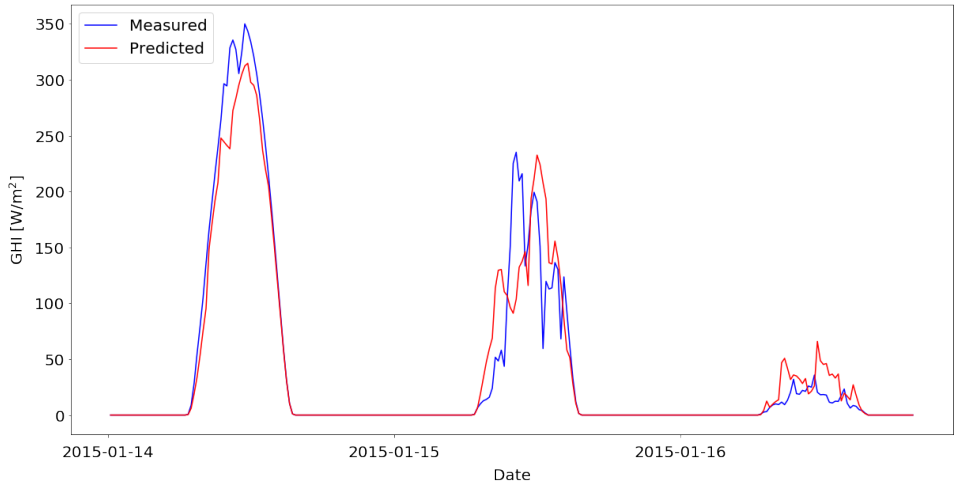


Figure 3.27: GHI prediction with LSTM for 2-h ahead.

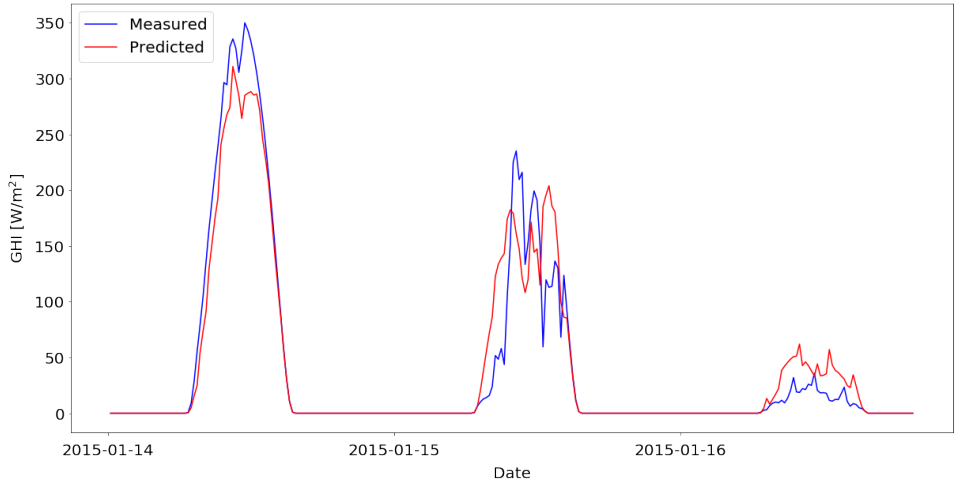


Figure 3.28: GHI prediction with LSTM for 3-h ahead.

reported the percentage improvement of each multivariate model with respect to its univariate implementation, where a positive value indicates a beneficial impact of exogenous inputs in prediction performance. The results clearly show that the multivariate approach outperforms the univariate approach for almost every prediction horizon. Indeed, all statistical metrics present a certain improvement in the multivariate scenario from 30 min to 240 min ahead, showing higher improvements for longer forecasting horizons. For example, the LSTM trained by using both endogenous and exogenous inputs achieved an impressive performance improvement of 22.14% in terms of MAD for 4 h ahead predictions. The RSMD and the R^2 indices also show an outstanding performance improvement of 18.99% and 8.64% for 4 h ahead forecasts, respectively. On the other hand, the univariate approach

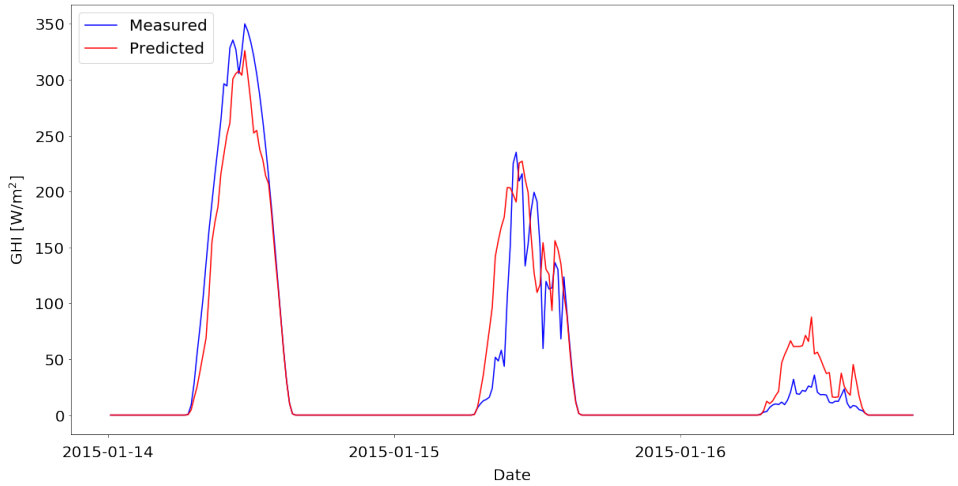
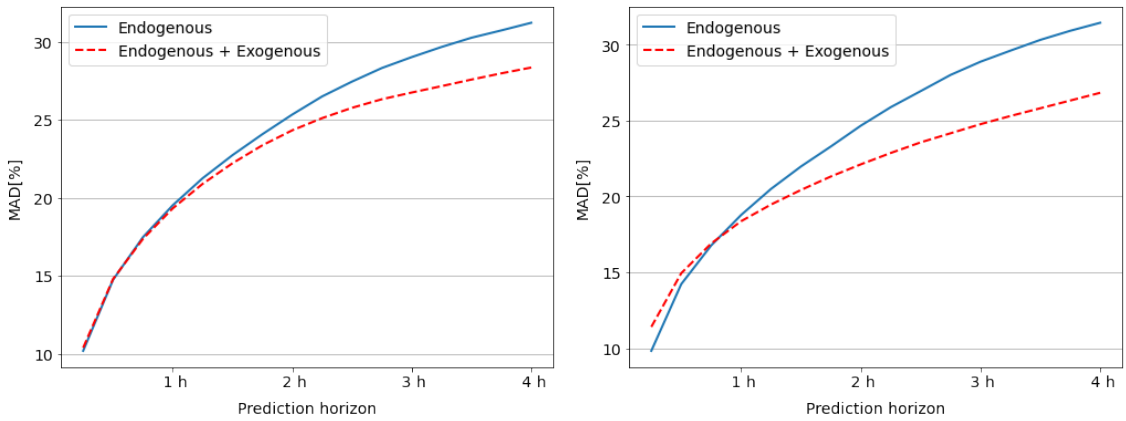


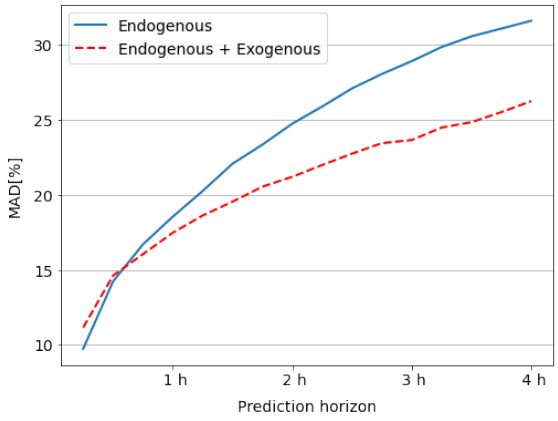
Figure 3.29: GHI prediction with LSTM for 4-h ahead.

surpassed the multivariate solution only for 15 min ahead forecasts, where the 1D-CNN reported the best results in terms of MAD with 9.61%. As a matter of fact, the MAD index shows that there are no advantages in the multivariate approach for 15 min ahead predictions, reporting only negative performance improvements for all models in that prediction horizon. To conclude, the results show that the adoption of exogenous inputs can significantly improve the forecasting performance for prediction horizons greater than 15 min, while for very short prediction horizons the performance improvement due to exogenous inputs can be considered as negligible. On these bases, we suggest to adopt a multivariate approach only for longer forecasting horizons, where the benefits provided by exogenous inputs give reasons for the higher computational costs required by more complex models. On the contrary, we believe that endogenous inputs can suffice for very short-term solar radiation predictions, since there is no evidence that exogenous inputs can provide performance improvements for very short prediction horizons.

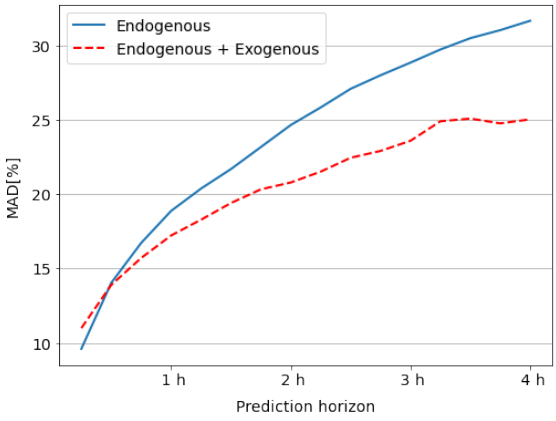


(a) ESN

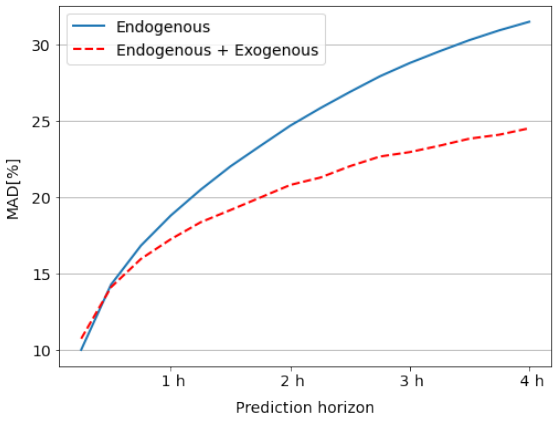
(b) RF



(c) FNN

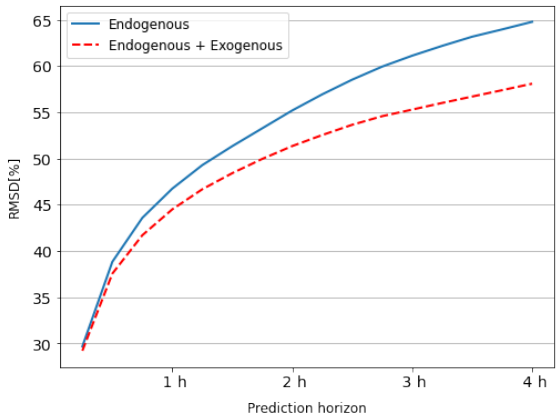


(d) 1D-CNN

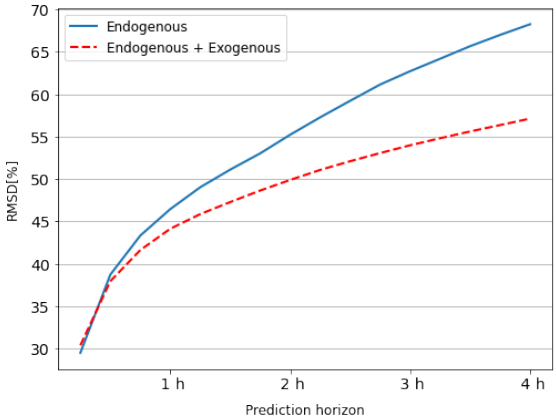


(e) LSTM

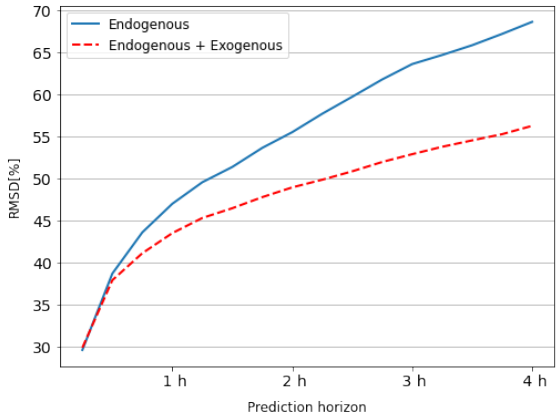
Figure 3.30: Improvement of exogenous inputs in terms of MAD



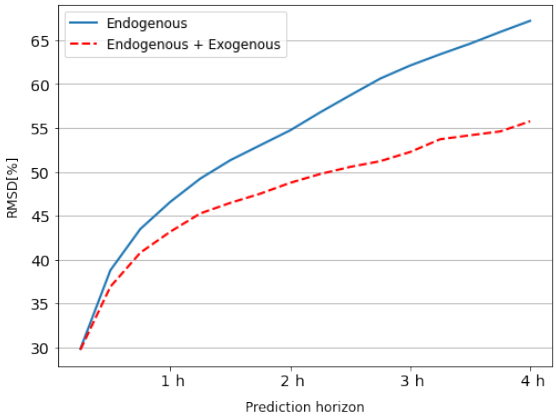
(a) ESN



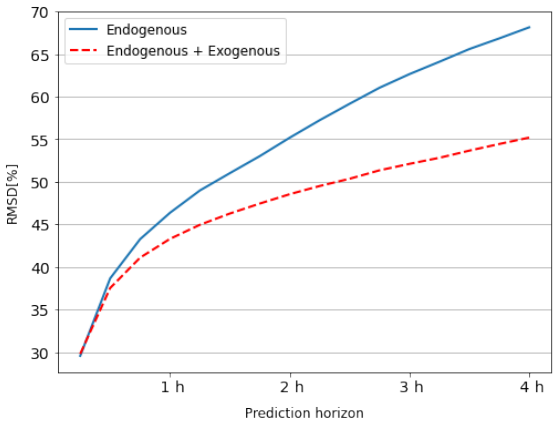
(b) RF



(c) FNN

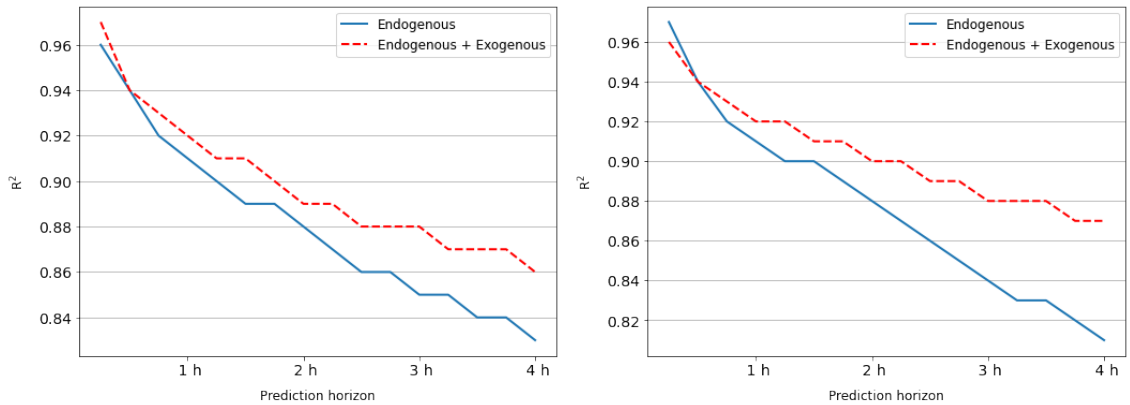


(d) 1D-CNN



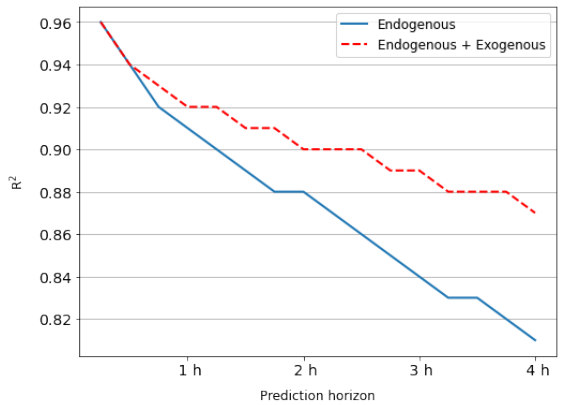
(e) LSTM

Figure 3.31: Improvement of exogenous inputs in terms of RMSD

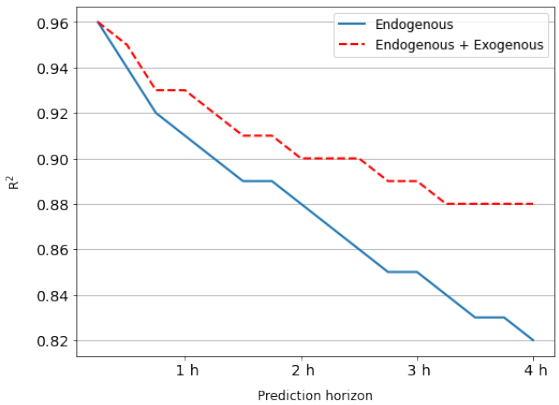


(a) ESN

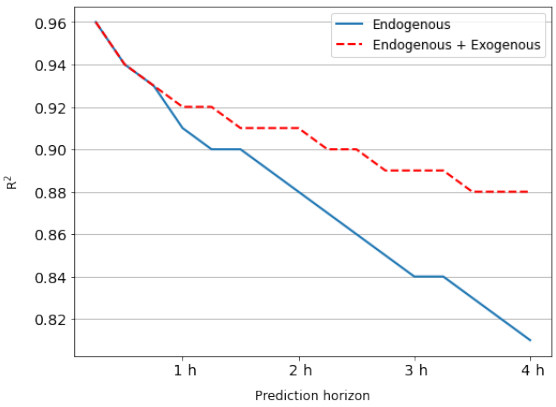
(b) RF



(c) FNN



(d) 1D-CNN



(e) LSTM

Figure 3.32: Improvement of exogenous inputs in terms of R^2

Table 3.20: Improvement of exogenous inputs in terms of MAD

	Endogenous + Exogenous					Endogenous					Improvements				
Time (min)	RF	FNN	ESN	LSTM	CNN	RF	FNN	ESN	LSTM	CNN	RF	FNN	ESN	LSTM	CNN
15	11.40	11.15	10.39	10.74	11.00	9.82	9.73	10.19	10.01	**9.61**	-16.09%	-14.59%	-1.96%	-7.29%	-14.46%
30	14.94	14.61	14.79	14.11	**13.90**	14.22	14.22	14.74	14.28	14.04	-5.06%	-2.74%	-0.34%	1.19%	1.00%
45	16.90	16.04	17.36	15.96	**15.71**	16.77	16.69	17.47	16.83	16.72	-0.78%	3.89%	0.63%	5.17%	6.04%
60	18.37	17.47	19.32	17.25	**17.22**	18.78	18.54	19.52	18.80	18.87	2.18%	5.77%	1.02%	8.24%	8.74%
75	19.47	18.62	20.89	18.36	**18.28**	20.50	20.25	21.25	20.49	20.37	5.02%	8.05%	1.69%	10.40%	10.26%
90	20.43	19.54	22.21	**19.16**	19.40	21.98	22.07	22.73	22.02	21.68	7.05%	11.46%	2.29%	12.99%	10.52%
105	21.32	20.54	23.36	**19.98**	20.32	23.30	23.34	24.07	23.36	23.16	8.50%	12.00%	2.95%	14.47%	12.26%
120	22.12	21.20	24.32	20.80	**20.78**	24.67	24.74	25.33	24.68	24.64	10.34%	14.31%	3.99%	15.72%	15.67%
135	22.87	21.98	25.10	**21.28**	21.51	25.89	25.88	26.49	25.82	25.82	11.66%	15.07%	5.25%	17.58%	16.69%
150	23.57	22.74	25.76	**22.03**	22.44	26.95	27.09	27.44	26.89	27.07	12.54%	16.06%	6.12%	18.07%	17.10%
165	24.17	23.44	26.31	**22.66**	22.90	28.02	28.05	28.31	27.91	27.98	13.74%	16.43%	7.06%	18.81%	18.16%
180	24.77	23.65	26.74	**22.95**	23.58	28.89	28.91	29.01	28.78	28.83	14.26%	18.19%	7.82%	20.26%	18.21%
195	25.31	24.48	27.15	**23.37**	24.89	29.62	29.85	29.66	29.55	29.71	14.55%	17.99%	8.46%	20.91%	16.22%
210	25.82	24.84	27.57	**23.83**	25.07	30.33	30.56	30.25	30.28	30.47	14.87%	18.72%	8.86%	21.30%	17.72%
225	26.33	25.51	27.97	**24.09**	24.75	30.93	31.08	30.72	30.92	31.01	14.87%	17.92%	8.95%	22.09%	20.19%
240	26.84	26.24	28.34	**24.51**	25.02	31.46	31.60	31.21	31.48	31.64	14.69%	16.96%	9.20%	22.14%	20.92%

Table 3.21: Improvement of exogenous inputs in terms of RMSD

	Endogenous + Exogenous					Endogenous					Improvements				
Time (min)	RF	FNN	ESN	LSTM	CNN	RF	FNN	ESN	LSTM	CNN	RF	FNN	ESN	LSTM	CNN
15 min	30.39	29.92	**29.22**	29.81	29.73	29.50	29.65	29.69	29.58	29.83	-3.02%	-0.91%	1.58%	-0.78%	0.34%
30 min	37.95	37.95	37.56	37.54	**36.91**	38.73	38.74	38.84	38.67	38.78	2.01%	2.04%	3.30%	2.92%	4.82%
45 min	41.63	41.16	41.71	41.10	**40.79**	43.34	43.64	43.59	43.27	43.49	3.95%	5.68%	4.31%	5.02%	6.21%
60 min	44.14	43.55	44.51	43.31	**43.18**	46.46	47.04	46.75	46.37	46.58	4.99%	7.42%	4.79%	6.60%	7.30%
75 min	45.87	45.35	46.68	**44.96**	45.26	49.04	49.60	49.29	48.99	49.23	6.46%	8.57%	5.30%	8.23%	8.06%
90 min	47.29	46.51	48.42	**46.28**	46.46	51.12	51.42	51.34	51.02	51.32	7.49%	9.55%	5.69%	9.29%	9.47%
105 min	48.65	47.83	49.96	**47.45**	47.51	53.04	53.69	53.27	53.02	53.03	8.28%	10.91%	6.21%	10.51%	10.41%
120 min	49.90	48.99	51.35	**48.55**	48.75	55.24	55.56	55.20	55.18	54.73	9.67%	11.83%	6.97%	12.02%	10.93%
135 min	51.08	49.90	52.54	**49.51**	49.75	57.29	57.75	56.94	57.23	56.78	10.84%	13.59%	7.73%	13.49%	12.38%
150 min	52.11	50.90	53.65	**50.36**	50.56	59.24	59.77	58.53	59.16	58.72	12.04%	14.84%	8.34%	14.87%	13.90%
165 min	53.07	52.01	54.58	51.35	**51.21**	61.16	61.83	59.94	61.04	60.62	13.23%	15.88%	8.94%	15.87%	15.52%
180 min	53.99	52.94	55.29	**52.11**	52.26	62.73	63.66	61.12	62.64	62.11	13.93%	16.84%	9.54%	16.81%	15.86%
195 min	54.82	53.82	56.01	**52.82**	53.71	64.20	64.74	62.18	64.09	63.39	14.61%	16.87%	9.92%	17.58%	15.27%
210 min	55.62	54.58	56.70	**53.66**	54.16	65.68	65.90	63.17	65.58	64.60	15.32%	17.18%	10.24%	18.18%	16.16%
225 min	56.37	55.32	57.39	**54.44**	54.60	67.01	67.25	63.96	66.83	65.93	15.88%	17.74%	10.27%	18.54%	17.18%
240 min	57.14	56.31	58.09	**55.19**	55.75	68.27	68.68	64.78	68.13	67.20	16.30%	18.01%	10.33%	18.99%	17.04%

Table 3.22: Improvement of exogenous inputs in terms of R^2

	Endogenous + Exogenous					Endogenous					Improvements				
Time (min)	RF	FNN	ESN	LSTM	CNN	RF	FNN	ESN	LSTM	CNN	RF	FNN	ESN	LSTM	CNN
15 min	0.96	0.96	0.97	0.96	0.96	**0.97**	0.96	0.96	0.96	0.96	-1.03%	0.00%	1.04%	0.00%	0.00%
30 min	0.94	0.94	0.94	0.94	**0.95**	0.94	0.94	0.94	0.94	0.94	0.00%	0.00%	0.00%	0.00%	1.06%
45 min	0.93	0.93	0.93	0.93	0.93	0.92	0.92	0.92	**0.93**	0.92	1.09%	1.09%	1.09%	0.00%	1.09%
60 min	0.92	0.92	0.92	0.92	**0.93**	0.91	0.91	0.91	0.91	0.91	1.10%	1.10%	1.10%	1.10%	2.20%
75 min	**0.92**	0.92	0.91	0.92	0.92	0.90	0.90	0.90	0.90	0.90	2.22%	2.22%	1.11%	2.22%	2.22%
90 min	**0.91**	0.91	0.91	0.91	0.91	0.90	0.89	0.89	0.90	0.89	1.11%	2.25%	2.25%	1.11%	2.25%
105 min	**0.91**	0.91	0.90	0.91	0.91	0.89	0.88	0.89	0.89	0.89	2.25%	3.41%	1.12%	2.25%	2.25%
120 min	0.90	0.90	0.89	**0.91**	0.90	0.88	0.88	0.88	0.88	0.88	2.27%	2.27%	1.14%	3.41%	2.27%
135 min	**0.90**	0.90	0.89	0.90	0.90	0.87	0.87	0.87	0.87	0.87	3.45%	3.45%	2.30%	3.45%	3.45%
150 min	0.89	**0.90**	0.88	0.90	0.90	0.86	0.86	0.86	0.86	0.86	3.49%	4.65%	2.33%	4.65%	4.65%
165 min	**0.89**	0.89	0.88	0.89	0.89	0.85	0.85	0.86	0.85	0.85	4.71%	4.71%	2.33%	4.71%	4.71%
180 min	0.88	**0.89**	0.88	0.89	0.89	0.84	0.84	0.85	0.84	0.85	4.76%	5.95%	3.53%	5.95%	4.71%
195 min	0.88	0.88	0.87	**0.89**	0.88	0.83	0.83	0.85	0.84	0.84	6.02%	6.02%	2.35%	5.95%	4.76%
210 min	**0.88**	0.88	0.87	0.88	0.88	0.83	0.83	0.84	0.83	0.83	6.02%	6.02%	3.57%	6.02%	6.02%
225 min	0.87	**0.88**	0.87	0.88	0.88	0.82	0.82	0.84	0.82	0.83	6.10%	7.32%	3.57%	7.32%	6.02%
240 min	0.87	0.87	0.86	0.88	**0.88**	0.81	0.81	0.83	0.81	0.82	7.41%	7.41%	3.61%	8.64%	7.32%

3.6 Discussion and Remarks

In this Chapter, we presented a methodology to forecast solar radiation values in short- and mid-term time-periods exploiting neural networks. In a smart grid scenario, this forecast is needed to estimate in advance the energy production of renewable energy sources enabling novel control strategies and services for grid management, such as Demand/Response policies [226].

The first part of our methodology presents a newel solution to forecast GHI in short-term period by implementing two different non-linear autoregressive neural networks, NAR and NARMA, respectively. Both neural networks have been implemented, trained and validated exploiting a dataset consisting of four years of solar radiation values collected by a real weather station. Then, the results of these ANNs have been given as input to a Photovoltaic simulator (PVsim) [35] to estimate the energy production of photovoltaic systems. The accuracy of these results is also acceptable, especially for time horizon from future 15 to 45 minutes.

In the second part, we focus specifically on short- and mid-term GHI forecast based on state-of-art artificial neural networks. The goal is to identify the most performing tool that allows us to make more accurate GHI predictions. To achieve this, we designed, optimised and compared four ANN architectures based on i) NAR, ii) FFN, iii) LSTM and iv) ESN. Contextually, using these architectures, three different approaches were evaluated: i) the raw GHI was directly used to train the networks and to make predictions; ii) the training set of the GHI series was filtered with Tikhonov regularisation before performing the training procedure; iii) the GHI time-series was transformed into the clear-sky index series, which was then used to train the networks and make predictions. The obtained results suggest a few interesting considerations. First of all, using a multi-output approach significantly improves the accuracy of multi-step predictions when more than 60 min in the future is required. In our case, this means that for forecast horizons of 45 or 60 min and longer this approach is to be preferred over a single-output model used iteratively. Considering that a single-output model is simpler, it may be better to use the iterative approach when a long-term prediction is not needed for the application.

The ESN gives very good results compared to other models, mainly when directly predicting GHI. Moreover, compared to the other models, it needs a smaller number of regressors to give very accurate results. This can be important in terms of availability. Given the lack of research in the literature on GHI forecasting with ESN, these findings are genuinely new results worthy of further studies.

The proposed model that uses Tikhonov regularisation to filter the training data and uses the unfiltered GHI for the testing part, which is used successfully in other fields involving time-series forecasting, like blood glucose predictions, does not appear suitable for GHI forecasting. Therefore, filtered data was used in input for the testing part, too. This is not ideal since this method would require, when the system is used for actual predictions "in the field", to filter the data every time

a new forecast is requested, which might limit the applicability of the method. Moreover, the Tikhonov filter was applied to the whole testing set, divided into long segments, but in a real application, using real-time data, this is not possible, because new data would have to be filtered when it becomes available (e.g. in our scenarios every 15 min). The algorithm would have to be modified accordingly, and the way it might affect the results needs to be studied more in detail. However, with this approach (i.e. exploiting filtered data in the entire process, from training to inference) the results were more interesting, showing that potentially, filtered data allows maintaining a better accuracy for short- and mid-term forecast.

The clear-sky index K_c significantly improves prediction accuracy when predicting many steps ahead, particularly for NAR, FFNN and LSTM networks. As already stated, the improvement for ESN is small. However, using K_c allowed the ESN to give better results with a smaller reservoir, which is important in terms of memory usage.

Finally, we evaluated the effectiveness of using exogenous inputs for short-term solar radiation forecasting. In detail, we identified a subset of relevant input variables for predicting GHI by applying different feature selection techniques to a broader set of variables. The results of feature selection revealed that the most significant input variables for predicting solar radiation are: i) UV index, ii) cloud cover, iii) temperature, iv) humidity, v) dew point, vi) wind bearing, vii) sunshine duration and viii) hour of the day.

To assess the usefulness of the selected features, we evaluated and compared the prediction performance of five different machine learning models, namely i) a FNN), ii) an ESN), iii) a 1D-CNN, iv) LSTM and v) a Random Forest. Overall, the LSTM demonstrated the best prediction performance among the five models, producing acceptable forecasting errors up to 4 hours ahead. The FNN and the 1D-CNN also demonstrated excellent prediction performance, comparable to those of the LSTM for prediction horizons shorter than 2 hours. The ESN presented the highest forecasting errors, revealing poor prediction performance in modelling multivariate time series. The RF performed slightly better than the ESN, showing promising results.

Finally, in order to demonstrate the effectiveness of using exogenous inputs for short-term solar radiation forecasting, we compared the multivariate models with their univariate counterparts. The results showed that the adoption of exogenous inputs can significantly improve the forecasting performance for prediction horizons greater than 15 min, while for shorter prediction horizons the performance improvement due to exogenous inputs can be considered as negligible. Overall, the results demonstrated the effectiveness of using exogenous inputs for short-term solar radiation forecasting.

Chapter 4

Indoor air-temperature forecasting

Climate change is having devastating effects on our planet. Consequently, the contrast against energy waste and pollution has become mandatory and widely endorsed. Strictly connected to pollution and CO_2 emissions issues, the energy sector is one of the main actors. Therefore energy management has become mandatory in every field of our society. Many countries are promoting different incentives to foster low-carbon and sustainable initiatives, especially in the building sector. Indeed, the building sector energy management is one of the most critical. Only in Europe, buildings are responsible for 40% of total energy consumption, affecting more than a third of the total pollution produced. Therefore, energy control policies of buildings (e.g. forecast-based policies such as Demand Response and Demand Side Management) play a decisive role in reducing energy waste.

4.1 Introduction

Nowadays research in the framework of green energy consists in reducing energy consumption and CO_2 emissions to contrast global warming, as highlighted during the last international conference on climate changes (COP21) [247]. At a global level, it has therefore been decided to promote many initiatives in order to move towards a more sustainable society. This is having a substantial impact on our cities, especially in the building sector. Indeed, buildings are responsible for about 40% of the total energy consumption and about 36% of the total pollution in Europe, as reported by the European Union Directive on the *Energy Performance of Buildings* [79]. Consequently, the scientific community agrees that to reduce this energy waste, novel tools are needed to model, monitor and control building energy behaviours. The European Union declared: *"Information and Communication Technologies (ICTs) have an essential role to play in reducing*

the energy intensity and increasing the energy efficiency of the economy, in other words, in reducing emissions and contributing to sustainable growth" [57]. Contextually, the recent improvements in ICTs offer an archipelago of devices, software architectures and communication paradigms that can enable the deployment of real smart-buildings and cities. Devices, such as low-power Wireless Sensor Networks (WSNs) for environmental monitoring, and novel smart-meters for electric load profiling and recognition, give the possibility of monitoring and characterisation of energy consumption behaviour of buildings and dwellings. On the other hand, recent improvements in Building Information Models (BIMs) [75] makes possible to physically model the buildings considering their construction materials, sub-systems and usage behaviour.

Rising ICTs such as IoT technologies, BIM, and Machine Learning are becoming key players to design and develop new control strategies based on systematic knowledge and prediction of energy behaviour. Hence, all existing buildings should deploy novel technologies to convert into Smart Buildings [124] that can interoperate in the Smart Cities of the future [43]. Indeed, existing buildings, especially the public ones, are often equipped with *building management systems* to allow monitoring and control of Heating Ventilation and Air Conditioning (HVAC). However, a Smart Building has to react in (near-) real-time to guarantee the right level of comfort to inhabitants and save energy. In this view, existing HVAC can be enhanced with pervasive and heterogeneous IoT devices with a minimum construction impact by exploiting distributed software platforms [198]. This allows exhaustive fine-grained monitoring of each room in the building and fosters the development of new control strategies both at the building and at the district level.

4.2 Enabling technologies for Smart Energy Management

As previously introduced in Section 4.1, the relentless rise of ICTs is paving the way for new green and sustainable paradigms in Smart Building context. Technologies such as IoT, BIM and machine learning are key players to design and model new energy control strategies. According to what is already abundantly detailed in Section 1.1, this section presents the latest enabling technologies to be implemented for smart energy management in buildings. Specifically, it provides an overview of the latest IoT and BIM technologies that can be applied to move forward the Smart Building and Smart City views.

4.2.1 IoT for Energy Monitoring

Contextually to the development of Smart Cities and Smart Buildings, the scientific community has addressed many challenges to facilitate the deployment of

IoT technologies such as *Smart Meters* and *Wireless Sensor Networks* (WSN). This in order to realize seamless, energy-efficient, reliable and low-cost remote monitoring and control solutions.

Smart meters are internet-connected devices able to provide information for billing, monitoring and controlling purposes [22]. The information produced by these devices, in turn, enable the development of supervise systems able to monitor and prevent contingencies, faults and unconventional behaviour. Generally, the main functionalities of smart meters can be summarized as following [24]:

- data collection/recording;
- two-way communication;
- programming capabilities;
- load control.

Instead, the most common communication technologies are the Radio Frequency (RF) and the Power Line Communication (PLC) [275].

WSNs are networks of devices (called nodes) working cooperatively to communicate gathered information from a monitored area [214]. As the name suggests, this network works through wireless links. For that reason, the gathered data are sent to a hub that uses these data locally or sends them to other networks (e.g. Internet through a gateway). In the Smart City domain, WSNs can be divided into three main subsets [81]:

- Home Area Networks (HANs);
- Neighborhood Area Networks (NANs);
- Wide Area Networks (WANs).

HANs establish a communication path among smart meters and home appliances through low-cost communication technologies and protocols (e.g. Wi-Fi, Bluetooth, ZigBee and Spirit). These networks enable end-users and energy manager to collect information on the environment. Differently, *NANs* are established between data collectors and smart meters in a neighbourhood area through Wi-Fi and RF mesh technologies. Indeed, these networks are short-range communication typologies, and they are used to collect data from smart meters and transmit them to a data concentrator. Lastly, *WANs* create a communication path between a service provider data centre and data concentrators. In general, these networks are based on LTE, mobile networks, fibre or power line communication networks. The deployment of WSN technologies offers numerous advantages such as lower costs, scalability, reliability, accuracy, flexibility in a wide range of applications [107].

4.2.2 Building Information Models

The increasing energy efficiency of cities and buildings is one of the most challenging research areas to test the potential of Building Information Modelling (BIM) methodology.

Generally, BIM is intended as a digital three-dimensional representation of a building, based on objects with associated information whereby it is possible develop a working method based on cooperation between different professionals involved in the construction process from a shared database. The BIM is a key element for the building life-cycle:

- design;
- construction;
- management;
- deconstruction.

In a Smart Building framework, BIM allows analyzing the existing building stock to promote better management and retrofitting actions. Indeed, the BIM model can be used to generate the geometry of the energy model, minimizing misinterpretations and improper approximations encountered in practice [80]. This is the main advantage of using BIM for energy modelling. BIM provides the opportunity to easily simulate different optimization scenarios, both in the design and refurbishment phases. This aspect is guaranteed by the possibility to validate the building energy model based on the real data gathered from monitoring activities and energy bills. Furthermore, exploiting BIM for energy simulations allows the interoperability and data exchange among different software (i.e. modelling, management and simulation software).

4.3 Related Works

In the last few years, due to the increase in greenhouse gas emissions and energy demand, many resources have been allocated to study and develop efficient solutions to reduce energy waste [56]. The building sector is one of the main responsible for global pollution due to its intrinsic construction characteristics [11] and for its inefficient use [193]. Consequently, both the scientific and political communities are focused on making the buildings (either already existing or new ones) more energy-efficient, moving forward to the Smart Building concept [198].

The literature presents many studies providing methodologies to model buildings and enable in-depth analysis and simulations [92, 158]. Furthermore, the interest in this topic is plentifully confirmed by the success obtained by different commercial software such as EnergyPlus [60] and TRNSYS [243]. These software

provide very robust and accurate results, consequently, they have become milestone as simulation tools for building- and energy-managers that need to evaluate thermal energy performance in buildings. However, they are extremely demanding in terms of computational resources [34], which makes them unfeasible for Model Predictive Control (MPC) systems. Consequently, to overcome such limitations, researchers have developed new methodologies to provide a better compromise between computational costs and thermal estimations accuracy [55]. These start from an accurate model of the building and then obtain a more compact approximated representation via i) model order reduction, ii) model aggregation or iii) ad-hoc dynamics extraction. Therefore, the thermal behaviours of buildings have been modelled as Resistor–Capacitor (R/C) circuits [66], exploiting an aggregation-based reduction approach to perform localized attenuation preserving relevant properties. In [216], the authors have developed a reduction methodology able to extract linear dynamics of thermal behaviours in buildings starting from simulation software (like EnergyPlus). However, these kinds of methodologies require very detailed structural information as thermal equations that often are not available (particularly for already existing building). Moreover, the reductions frequently introduce very significant losses on accuracy. R/C circuits for building modelling have also been exploited in [161], where authors presented a methodology based on Unscented Kalman Filter and thermal network representation to estimate thermal dynamics in buildings. However, the complexity of the model increases with increasing buildings complexity (for example, buildings with many rooms), making this approach suitable for small constructions.

Simultaneously, researchers have investigated another approach based on Very Large Scale Integration (VLSI) techniques to build a compact thermal model. Generally, VLSI-based solutions exploit matrix pencil [144] and subspace identification [76] that do not consider physical restrictions and hence make the compact model very flexible. These solutions take advantage of a detailed analysis of numerical simulations or real-world sampled data, making the training phase of the whole model very accurate. However, VLSI-based solutions are generally not suitable to deal with the non-linearity of a whole building thermal system. As a result, the scientific community has approached to machine learning techniques, such as Artificial Neural Networks (ANNs), that efficiently address non-linear problems. The literature presents several attempts to use ANN-based solution [220, 69, 177]. All these models, generally, have improved accuracy compared to state-of-the-art physical models. However, although the improved accuracy compared to the traditional physical models, all these ANN-based solutions exploit very limited data-sets consisting of real-world measurements (that are typically difficult to obtain and often incomplete) for both the training and the validation. As result, the ANNs performances have a very negative impact in terms of accuracy and prediction horizon, as well as on their generalization capabilities. To address this problem, the authors in [274] have used a simplified BIM of a fictitious building to create with EnergyPlus

a synthetic data-set of indoor air-temperature trends and then used this data-set to train two Recurrent Neural Networks. However, this solution over-simplifies the model, which makes it very distant from representing the thermal dynamics of a real-world building. Most recent works generally have a better capability of dealing with complex models. For example, in [86], authors proposed an overall framework for energy consumption prediction in buildings, comparing three different machine learning algorithms based on deep extreme learning machine, adaptive neuro-fuzzy inference and artificial neural networks. More recently, in [2] authors presented a big-data platform for predicting and characterizing energy consumption of buildings connected to the district heating system, exploiting multi-regression method between power consumption and environmental conditions. However, a general limitation of these models is that they do not provide any prediction of internal temperature conditions of the building.

In recent years, more and more authors have exploited ANNs to predict indoor temperature. In [164], the authors have developed an ANN to predict hourly temperature profile and humidity values of a room in humid regions. Nonetheless, the authors did not provide a thorough numerical evaluation of the performance of their work in terms of errors, but only provided correlation coefficients w.r.t the ground truth and a graphical validation of their model. In [20], the authors described an ANN for indoor temperature forecasting in smart buildings. This solution consists of a methodology for the selection of the input parameters of the ANN and concluded that the best combination is achieved by using outdoor and building facade temperature sensors. In [270], the authors have compared the performance of two different neural network models for predicting indoor temperature profiles, exploiting thermostat data together with outdoor weather information. Finally, in [265], authors recently proposed a modified version of a long short-term memory model for the prediction of the indoor temperature in a smart building. The data-set is composed of 5-minute samples, with a maximum prediction horizon of 30 minutes. However, the limits of these models are extremely short forecasting horizons.

4.3.1 Contributions

In order to overcome the limits listed above in the literature, this thesis presents an innovative methodology to forecast indoor air-temperature trends in existing or newly constructed buildings (of which, therefore, no indoor air-temperature data are available). This methodology is based on the most modern and state-of-art-recognized neural networks techniques for time series prediction. The aim is to provide a robust and generalizable methodology to enable new control policies for the thermal management of buildings w.r.t. individual rooms and the whole building. Given this, the methodology first of all addresses the possibility of a lack of data, which are indispensable for reliable training and validation of neural networks. Indeed, differently from standard ANN-based literature solutions, our

methodology exploits a huge synthetic dataset for training the model. This dataset is obtained by simulating a BIM model of a real-world building with EnergyPlus exploiting real weather data instead of *Typical Meteorological Year* (TMY) data. As the training is entirely based on simulated data, the methodology is ready to be used immediately after the building is equipped with IoT devices, without needing any calibrations. Hence, it allows forecasting indoor air-temperature trends even in case of unavailable historical measurements. All this allows us to state that this methodology can be successfully used both on existing buildings in which a historical dataset is available (using only real data) and on newly constructed buildings in which a historical dataset is not available (using realistic synthetic data to model the building and real data from IoT sensors for the neural network inference phase). In parallel to this, we model, optimize and compare different neural networks in order to find the best architectures for short-, medium- and long-term indoor air-temperature forecast. Identified the most promising architecture, we apply transfer learning techniques with the objective of further improve the prediction accuracy. In this way, buildings that do not have a historical dataset, gradually collecting data, can migrate from a model simulated on synthetic data to a model based on real data. This gradual transition allows avoiding new simulations and computational waste by ensuring more and more accurate prediction. In all the phases that make up the methodology, the intermediate and final results are validated using both analytical and qualitative metrics. In particular, concerning qualitative analysis, the proposed methodology uses the Fanger analysis, a method for evaluating thermal comfort in a building, based on environmental and occupants variables.

4.4 Methodology

This Section describes in detail the proposed methodology to forecast indoor air-temperature values in existing or newly constructed buildings (i.e. buildings without a consistent set of historical data). The main steps of the solution are in the following:

1. design and development of realistic artificial indoor-air temperature trends by exploiting BIM model of our demonstrator;

2. validation of the thermal energy simulation by exploiting real data provided by IoT devices installed in our demonstrator;

3. design and development of a NAR neural network for the comparison and validation of models based entirely on synthetic data and hybrid models (i.e. trained with syntetic data and inferred with real data);

4. design and development of the most modern and performing neural networks for time series prediction identify the most promising model to be adopted in the indoor-air temperature forecast context;

5. application of Transfer Learning methodologies to specialize the hybrid model on real data.

In Figure 4.1, summarizes the main steps of our proposed solution.

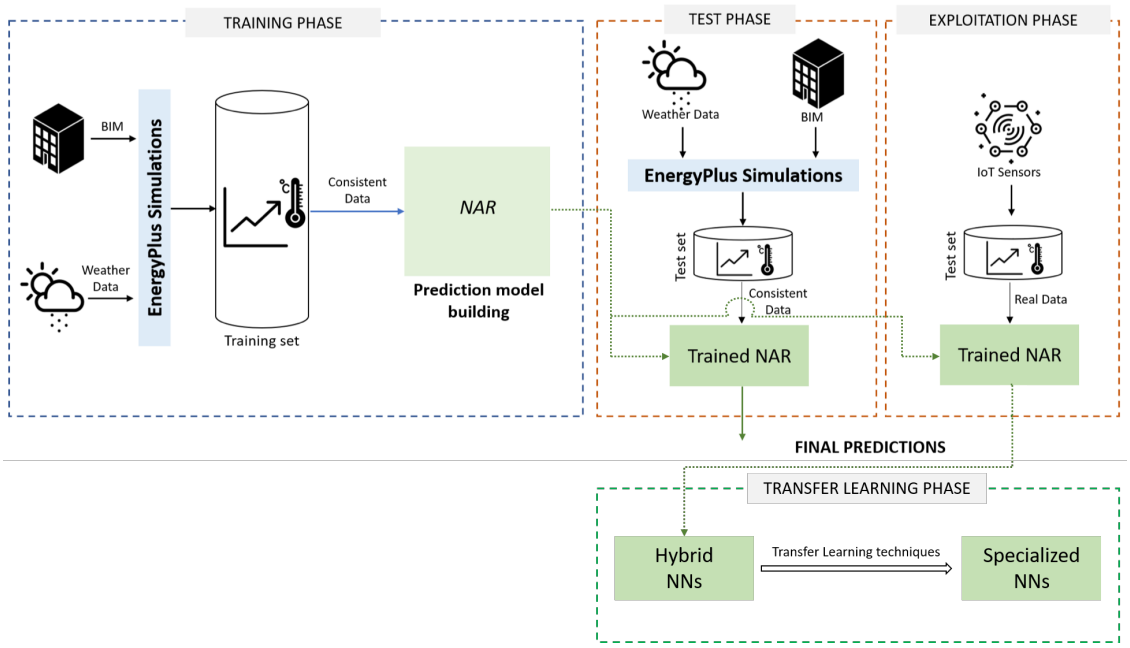


Figure 4.1: Main steps of the proposed system.

During the training phase, time-series data consisting of realistic artificial indoor-air temperature trends (see Section 4.5) are given as input to build a prediction model based on NAR. In the test phase, new unseen data from the realistic artificial test set are fed into the trained models to obtain the temperature predictions. Finally, in the exploitation phase, new unseen data sampled by IoT devices (deployed in the real-world building) are given as input to the trained models to evaluate the final predictions against real indoor air-temperature values. The rationale for splitting the experimental assessment into two different phases (test and exploitation) is the following. In the test phase, the model is tested on a large synthetic data-set obtained with EnergyPlus simulations, as for the training. Hence, the aim of this test is assessing the prediction capabilities of the model in a significant number of examples. In the exploitation phase, the NAR is assessed on a much smaller real-world data-set, which is less representative in terms of size but on the other hand, provides better insights into the generalization capabilities of the model in real-life

conditions. On top of that, it provides an indirect assessment of the reliability of the synthetic data that were used for training the model. Once we consolidated the methodology, we start exploring different state-of-art neural networks for the time-series forecasting. Our goal is to identify and optimize the most promising neural model, using the hybrid methodology (i.e. training with simulated data and inference with real data). To this purpose, we designed and optimized three neural networks: i) an Echo State Neural Network (ESN), Long Short-Term Memory (LSTM), One-dimensional Convolutional Neural Network (1D-CNN). Once we have identified the best model, we apply Transfer Learning techniques in order to specialize the hybrid model entirely on real data.

In the following, we report in details the main phases of the system, describing: i) the thermal energy simulation process to realize a consistent synthetic data-set, ii) the validation process of a neural model based entirely on synthetic data and of a hybrid neural model trained with synthetic data and inferred with real ones, iii) the design and optimization of different state-of-art neural networks for time series prediction and iv) the application of Transfer Learning techniques on the most promising model to further specialize it on the real data provided by our demonstrator.

4.4.1 Thermal energy simulation

This Section describes the followed methodology to perform energy simulations starting from BIM and correlating IoT data within an integrated process. The building energy modelling and monitoring approach is one of the most challenging topics in Smart City scenario. In this context: i) BIM establishes a proper knowledge of the buildings; ii) technical investigations aimed at energy efficiency are required by EU Energy Performance of Buildings Directive [79]; iii) IoT links different domains and provides real data from the field. These factors constitute the key issues for this research development.

To achieve it, BIM models have been developed with *Autodesk Revit 2016* [21] starting from on-site surveys. They include i) accurate building envelope characterizations in terms of correct stratigraphy, thermal and physical properties; ii) facility management information (e.g. room type and occupants); iii) materials nomenclature standards. Thus, they become a significant repository of graphical and alphanumeric information useful for energy analysis. To properly set the model to perform energy simulations, the BIM needs simplifications by removing excessive details in the architectural model, such as decorations and staircases (see Figure 4.2). These details are unnecessary and slow down the simulation or can even include inaccuracies in final results. Figure 4.3 shows the *Energy Analysis Model* (EAM) that consists of rooms and analytical surfaces generated from the BIM model and exported by *Revit* in gbXML data-format. Figure 4.4 reports the proposed energy modelling optimization process. The *EAM Simulation Engine*

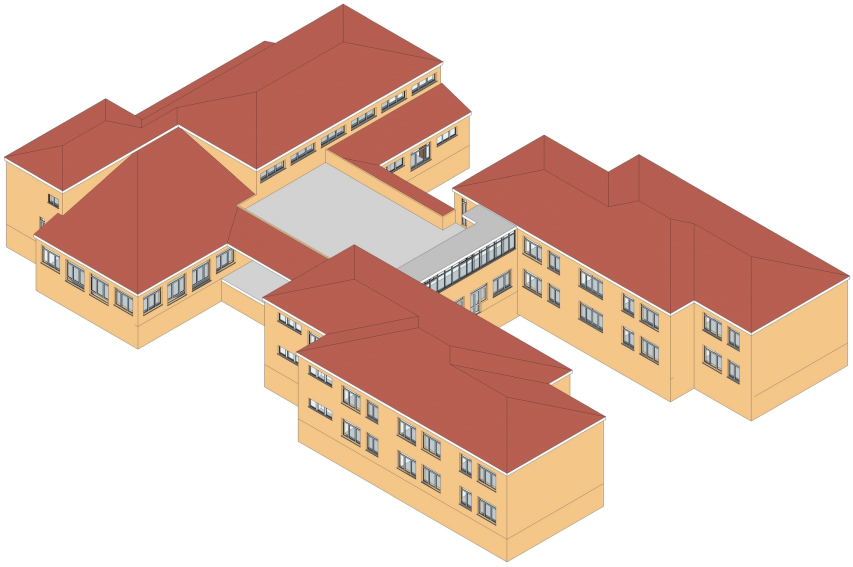


Figure 4.2: BIM model.

block performs building simulations using EnergyPlus [245]. It needs the following inputs:

- *Geometry and materials* of building components (e.g. stratigraphy and shades) and their thermal and physical properties, these come from BIM models;
- *Real weather data* such as i) air dry-bulb temperature, ii) solar radiation and iii) average air temperature;
- Data retrieved from *Heating Ventilation and Air Conditioning systems* such as i) nominal power and flow rate of radiators, ii) nominal power and efficiency of boiler, iii) climate control unit, iv) on/off profile of the heating system;
- *Occupancy* of rooms, including number of users and time-shifts.

The outputs of the *EAM Simulation Engine* block are radiant, operating and indoor temperature. It also provides the energy consumption profiles of the building. Traditionally energy simulations with EnergyPlus are performed using Typical meteorological year (TMY). TMY is obtained by averaging hourly meteorological measurements collected for 10 years. Thus, it does not represent real weather conditions. As a strong point of our simulations, we integrated in our software platform third-party weather data-source from the nearest weather station. Hence, real weather information (i.e. solar radiation, outdoor air temperature and humidity) are considered in the simulation process replacing the default TMY.

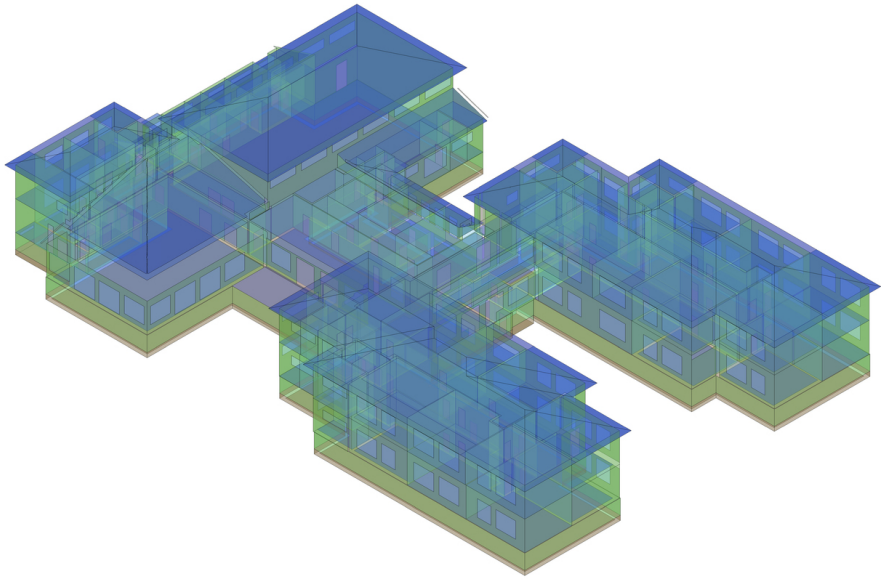


Figure 4.3: Energy Analysis Model (EAM).

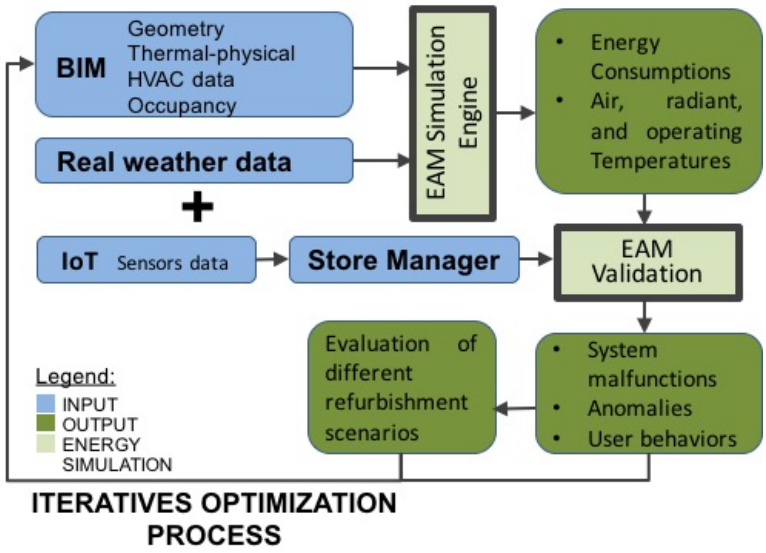


Figure 4.4: Proposed energy modeling optimization process.

IoT devices send indoor air temperature and humidity every 15 minutes. Such data are needed by the *EAM Validation* block in Figure 4.4 to validate the performed simulations. This validation is done by comparing the results of the *EAM Simulation Engine* with the real measured values coming from the deployed IoT devices. Analyzing temperature and consumption trends, factors that may affect the

energy model can be identified, such as user behaviours, malfunctions and anomalies in the system. For instance, user-awareness applications can help in minimizing not energy-efficient behaviours. Whilst maintenance activities can be planned to monitor and solve identified anomalies (e.g. by comparing measured and simulated data, it is possible to discover irregular trends of real indoor temperatures due to faults in on/off schedules of the heating system or efficiency losses of the building-system).

In addition, BIM models can be used to evaluate different design and/or refurbishment scenarios (see Figure 4.4) (e.g. external/internal coat application, fixtures replacement and power peaks regulation). Thus, this updated BIM model is a new input for the energy modelling optimization process. This process is iterative and can help building- and energy-managers in evaluating the best solution for both energy performances and Return of Investment.

4.4.2 Hybrid predictive model validation

The process of Thermal energy simulation allows us to realize a consistent synthetic data-set. Consequently, we use this near-realistic data-set to design two different NAR models in order to validate the exploitation of syntactic data in the framework of indoor air-temperature forecasting in existing or newly constructed buildings. In detail, we implement:

- a model completely based on syntetic data (i.e. design, training, validation and inference phases);
- a hybrid model trained and validated with synthetic data but inferred with real-world data;

As a result, for these models we compare prediction performance by highlighting their strengths and weaknesses.

As introduced in Chapter 2, NAR is an ANN that extends a traditional linear autoregressive model [147] to be completely distribution-free. Thus, NAR is suitable for non-linear time-series that report, for instance, unexpected spikes and fleeting transient periods [185]. Consequently, we base the design phase on the assumption detailed in Section 2.2.1. To design the NAR model, the starting point is the selection of the *lag-space*, that in our application is the optimal number of past air-temperature values to be used as regressors. For this purpose, we applied Lipschitz methodology [211], that is a well-known approach in the analysis of input-output models' orders in non-linear dynamic systems that allows determining the number of regressors of a system empirically. By applying this method as described in [6], we found $n = 13$ as the best candidate. This value is not the final optimum of our system, but a reference point for a more in-depth empirical analysis. Thus, we implemented a *grid search* to find the optimal value, testing the performance of

different ANNs varying the number of regressors within a 20-dimensional range of values centred in $n = 13$. This approach was implemented for each selected room and the whole building, respectively. In each configuration, we started with fully-connected NAR ANNs, as shown in Figure 2.6, with the following characteristics:

1. one input layer with a variable number of regressors decided by our *grid search*;
2. one hidden layer with 30 neurons;
3. one output layer with one neuron.

This architecture is the result of our preliminary experiments with different network structures, where we found that increasing the number of hidden layers (and hence the complexity and computational costs of the training) did not provide significant benefits in terms of prediction performance. For all the configurations, we used hyperbolic tangent activation functions for the hidden neurons and a linear activation function for the output neuron, as shown in Figure 4.5.

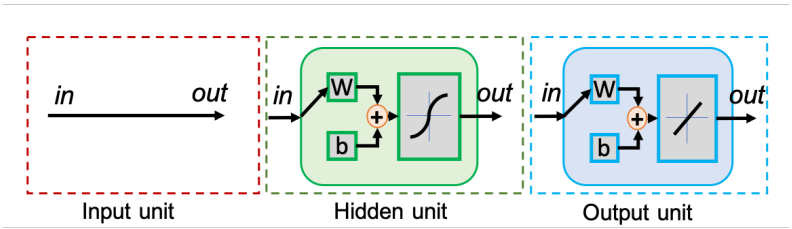


Figure 4.5: Nonlinear autoregressive neural network: neuron models of the input, hidden and output layers.

Then, we chose and implemented the Levenberg-Marquardt back-propagation procedure (LMBP), which is a learning paradigm widely applied to NAR ANNs in literature [271]. LMBP reduces the training speed compared to other back-propagation techniques because it approximates second-order derivatives leveraging a *trust region* approach [185] without computing the Hessian matrix. These models were trained on the training set described in Section 4.5. We found that the best training performance was obtained with 20, 19, 16 and 20 regressors for respectively the classroom facing East, the classroom facing West, the corridor and the whole building (see Section 4.5). The second column of Table 4.1 reports the normalized sum of squared errors (nSSE) on the validation set obtained after the training phase for all the different networks.

As it is widely known, ANNs with too many connections have longer training procedure and may easily lead to over-fitting. To overcome this issue, we pruned the initial fully-connected structures adopting the Optimal Brain Surgeon (OBS)

Table 4.1: Validation error (nSSE) before and after OBS pruning.

NAR Network Type	**nSSE before pruning**	**nSSE after pruning**
Classroom facing East (20 regressors)	9.95E-03	9.51E-03
Classroom facing West (19 regressors)	1.59E-02	1.54E-02
Corridor (16 regressors)	1.87E-02	1.85E-02
Whole building (20 regressors)	1.19E-02	1.17E-02

methodology [105]. The rationale of this operation is to remove redundant connections between neurons to obtain more efficient and compact models than the initial ones, not affecting or possibly improving their prediction capability. OBS estimates the increase in the training error when deleting weights, leveraging information in the second-order derivatives of the error surface. This procedure works towards the minimization of the error variation, computing the inverse Hessian matrix recursively from the training data to achieve better approximations of the error function (more details are provided in [105]). Dong et al. [70] demonstrated that OBS performs better than other pruning techniques by removing more redundant neuron connections.

After pruning, the four different ANNs were trained again with LMBP. The new nSSE values provided by this second round of training are reported in the third column of Table 4.1. The values if this table highlight that OBS pruning was successful, as it further reduced the validation error of the four NAR models.

4.4.3 State-of-art neural network for time series forecasting design

Once we consolidated the methodology based on the hybrid model, we designed three other neural networks that represent the state-of-art time series forecasting: i) an Echo State Neural Network (ESN), Long Short-Term Memory (LSTM), One-dimensional Convolutional Neural Network (1D-CNN) (see Chapter 2).

First of all we split our data-set as shown in Table 4.2 In according to Kline [133],

Table 4.2: Data-set splitting.

	Neural Networks implementation		
Phase	Training	Validation	Inference
Data	70% Synthetic	30% Synthetic	100% Real

we opted for the implementation of multi-outputs models which is the best configuration to achieve better accuracy of the prediction, as detailed in 2.4. Thus, to use the data-set as the input of our network, we have re-framed the data available

(see Section 4.5) to be compatible with supervised learning algorithms, which have to be transformed in a pair of input and output sequences. Since we use the multi-outputs, for each sequence of input, we associated a sequence of outputs using the sliding window algorithm as shown in Figure 4.6). Once the data are prepared

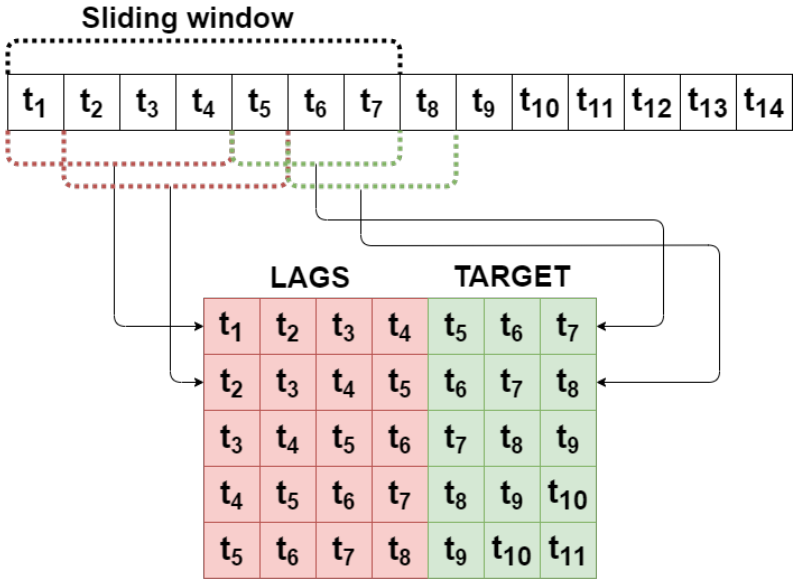


Figure 4.6: Re-framing samples methodology.

for training, we start defining all the the network structure and the associated hyper-parameters.

1D-CNN

We exploit a 1D-CNN architecture based on [167]. Figure 4.7 details the architecture structure.
In detail, the structure is composed as following:

1. *Input layer*: it accepts an input of dimension *(steps,features)*, where *steps* is the prediction lag, which is the number of previous temperature values that are used as regressors and features that in this case is one, the temperature.

2. *Convolutional layer*: it has a filter of dimension 2, than after the convolution operation we have an output of length equal to input dimension minus 1. The second dimensionality is equal to the number of filters, that is chosen based on different trials.

3. *Pooling layer*: The pooling strategy used is the Max Pooling with pool size equal to 2, which means that the dimension of the output is halved.

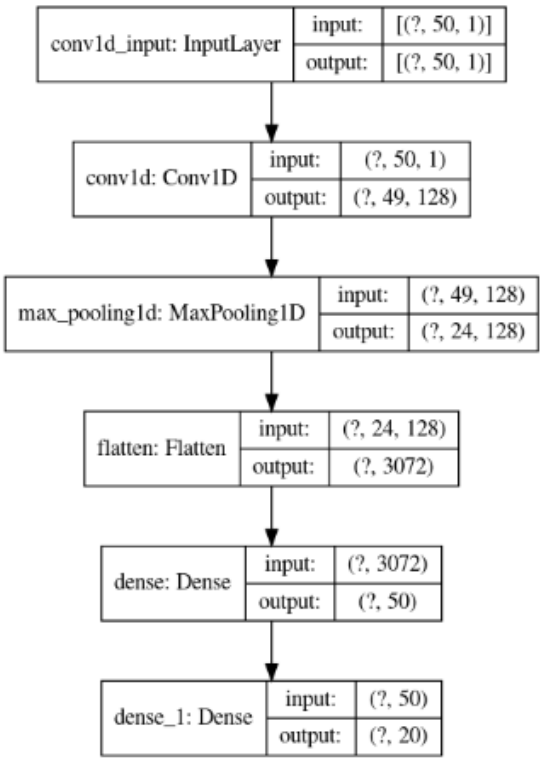


Figure 4.7: 1D-CNN structure.

4. *Flatten layer*: a flatten layer is used because we have a multidimensional output due to the multiple filers used. Than this layer is used to "unroll" the data, in order to be fed to the dense layer.

5. *Dense layer*: an intermediate dense layer is used to improve the learning ability of the network. The number of units of this layer is decided in the tuning phase.

6. *Output layer*: finally, the output layer with a number of neurons equal to the number of forecasting horizons.

In order to avoid overfitting, we adopt the *Early stopping* technique [73] with a patience of 8 epochs and a delta parameter equal to 0.003. For the activation, we opted for ReLU function for both Convolutional and Dense layers, while linear function for the Output layer.

Tuning of hyper-parameters In order to find the best set of parameters of the network, we adopt the trial-and-error strategy tuning the parameters in the following order: i) prediction lag, ii) number of hidden neurons of the Dense layer,

iii) batch size, iv) learning rate and finally v) number of filters. All experiments are evaluated for each room as detailed in Section 4.5.

LSTM

Later, we implement a stateful vanilla LSTM, that is composed by a single layer of LSTM cells. The structure is represented in Figure 4.8.

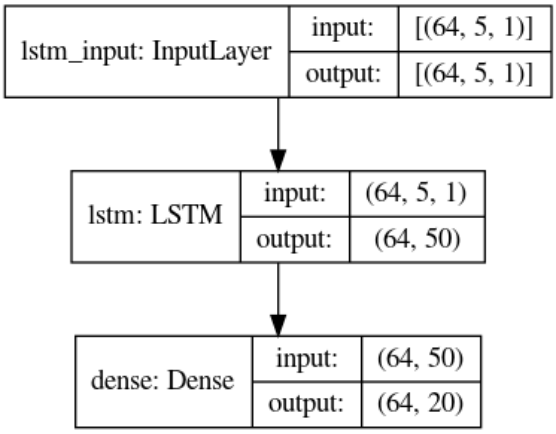


Figure 4.8: LSTM structure.

The network is composed by the following layers:

- *Input layer*: it accepts a shape of *(batch size, lags, number of features)*.
- *LSTM cell*: it is composed by LSTM cells, whose number is decided in the tuning phase.
- *Output layer*: it let the dimensionality of the LSTM layer to match the length of the forecasting horizons. Its activation function is the linear function.

Tuning of hyper-parameters In order to find the best set of parameters of the network, we staring set: i) the batch size equal to 256, ii) training epochs equal to 100; ii) the number of LSTM cells equal to 50. We adopt this starting configuration for each room (see Section 4.5), then we exploit a trial-and-error approach by evaluating different inputs, finally choosing the most performing. Then, we tune the number of epochs by analyzing the behaviour of loss during the training phase and choosing. Finally, the batch size and the learning rate are optimized together, due to their relationship, by creating a comparison grid with the couples of values *(learning rate, batch size)*.

ESN

Finally, we design and implement a vanilla Echo State neural network composed by an input layer, a hidden layer (the reservoir) and an output layer according to Section 2.2.2.

Tuning of hyper-parameters As with previous models, we start optimizing some hyper-parameters to find the best configuration. As far as the ESN model is concerned, the key parameters are: i) the reservoir size, ii) the spectral radius, iii) the state noise and iv) the sparsity of the reservoir connectivity. In detail, the spectral radius is the largest eighenvector[1] of the weights matrix of the reservoir. The state noise is a kind of noise, added to the reservoir states, able to stabilize the solution. The connectivity of the reservoir determines the number of internal connection between its nodes, it is the proportion of recurrent weights set to zero. The strategy we adopted was first to optimize the number of regressors, than conjunctively optimize the spectral radius and the noise, and finally, the sparsity with the reservoir size.

Lastly, the output activation function adopted is the identity, while the hidden-to-hidden layer has a *Tanh* activation.

4.4.4 Exogenous inputs investigation

All the models previously described in the Sub-section 4.4.3 are implemented considering only the indoor-air temperature values as input.

Consequently, we decided to investigate how the models would react by inserting some exogenous inputs: in detail, labels showing i) the hour of the day and ii) the day of the week, as depicted in Table 4.3 and Table 4.4 respectively.

Table 4.3: Time slots identification.

Hour	4:30 - 6:00	6 - 19:30	19:30 - 22	22 - 4:30
Label	0	1	2	3

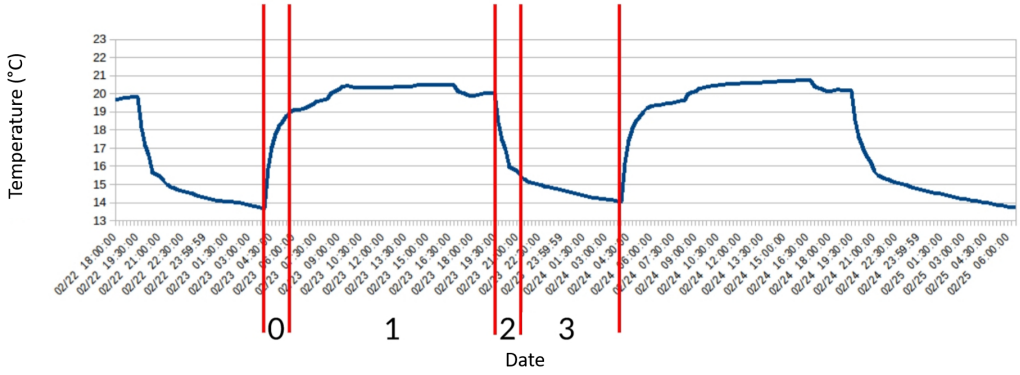
Table 4.4: Day of the week identification.

Day	Mon	Tue	Wed	Fri	Thu	Sat	Sun
Label	0	1	2	3	4	5	6

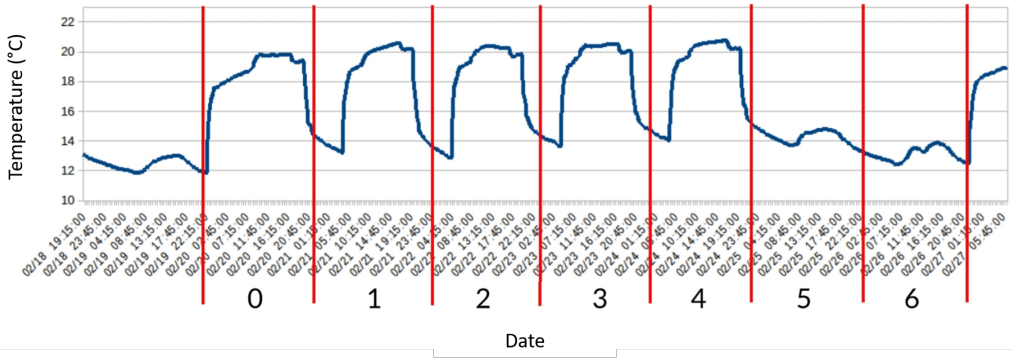
[1]In algebra, an *eigenvector* is a nonzero vector that changes at most by a scalar factor when that linear transformation is applied to it.

We assume that by looking at the temperature behaviour (Figure 4.9), the impact of the heating system during working hours is easily identifiable in cyclic patterns. Figure 4.9a clearly shows the switch-on phase of the heating system (label 0), the period of temperature maintenance during the hours of use of the spaces by the users (label 1), the shutdown phase (label 3) and lastly the phase of non-use of the heating system. Figure 4.9 shows, instead, that these behaviours are repeated throughout the working days (except on Saturdays and Sundays when the building is not in use).

The main goal is to try helping the neural network by feeding it with additional information. In detail, labels that identify recurring patterns. For this aim, we decide to exploit the 1D-CNN and LSTM architectures, with the same set of parameters identified for the single-input scenario. We decided to use only these two models since we are interested in discovering the impact of exogenous inputs purely and have a fair comparison of the results.



(a) Hour of the day labeling.



(b) Day of the week labeling.

Figure 4.9: Second feature extraction

Table 4.5: Dataset exploitation in the different phases.

	Neural Network implementation		Transfer Learning	
Phase	Training	Validation	Tuning	Inference
Data	70% Synthetic	30% Synthetic	80% Real	20% Real

4.4.5 Transfer Learning exploiting real data

Finally, after finding the best performing model, we introduce some Transfer Leaning approach in order to specialize our network on real-world data provided by our demonstrator. Consequently, we re-train the most promising model following three different paths:

- re-training all the layers (see Figure 4.12);
- re-training only the input layer (see Figure 4.10);
- re-training only the output layer (see Figure 4.11).

This procedure shall be carried out in accordance with the procedure described in Section 4.4.3.

Then, the real data-set, previously adopted entirely for the inference phase of the hybrid models, now is split into 2 parts: i) one used to fine-tune the network and ii) the other one to test it, as shown in Table 4.5).

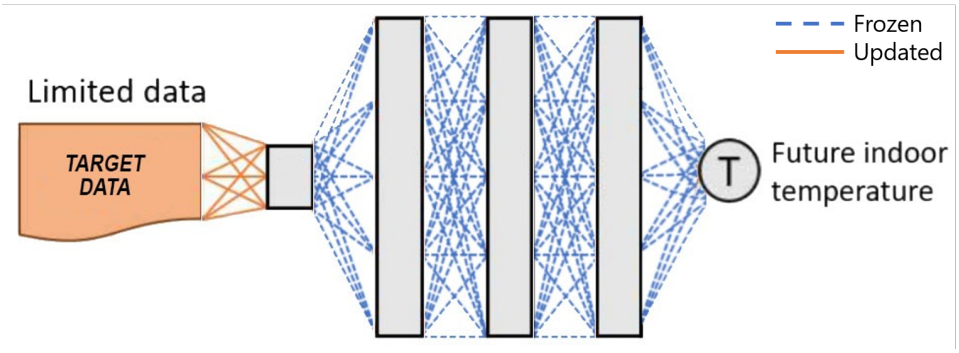


Figure 4.10: Freezing Layers technique - First layer re-trained.

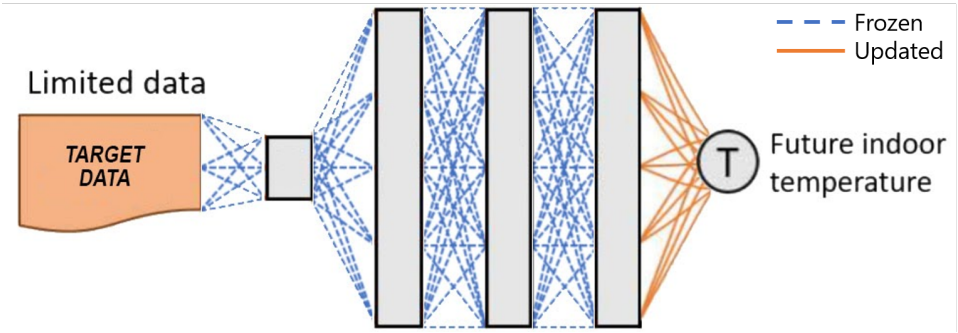


Figure 4.11: Freezing Layers technique - Last layer re-trained.

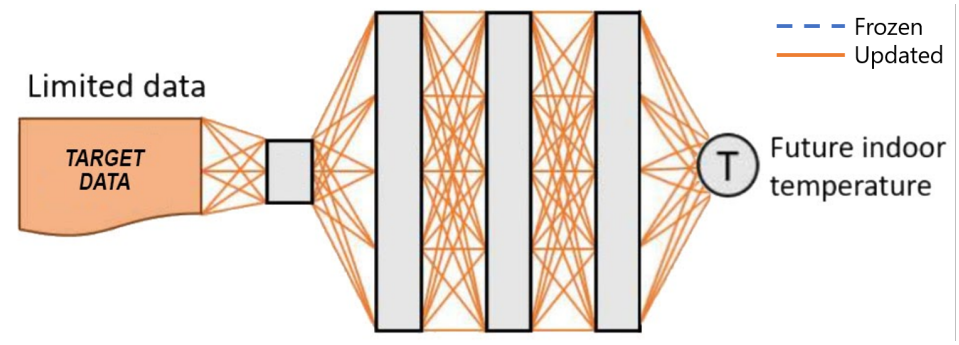


Figure 4.12: Soft Start technique.

4.5 Case Study

All the steps of this methodology are based on a real-world demonstrator. The building under analysis is a secondary school located in Turin, north-west of Italy. It is composed of about $14'500\ m^2$ and two floors. The interest in this building is given by the fact that it is connected to the district heating distribution network and is not equipped with a conditioning system. Windows on brick walls facades are double glazed. Both east- and west-oriented facades receive substantial contributions of thermal energy due to solar radiation. Besides, this building presents the structural information stored at the real estate registry office but does not have a system for the collection of indoor air-temperature values, this results in a lack of a historical data-set.

Following our methodology, to obtain a suitable data-set for the study, we first analyzed the structural information of the building (i.e. geometry, materials, thermal and physical properties of building components) and then we built its BIM model, that is reported in Figure 4.13. To provide a thorough analysis, we decided

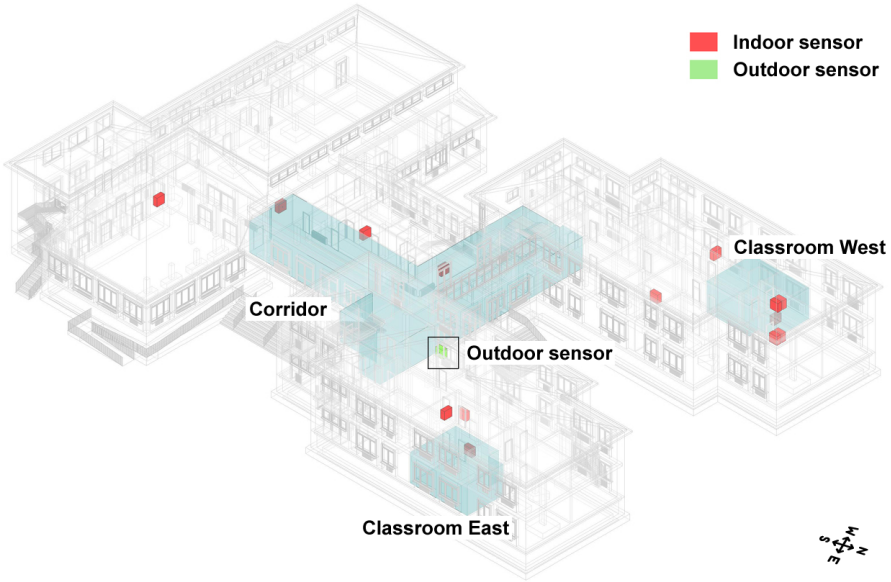


Figure 4.13: Case-study building - BIM model (reference rooms are highlighted in light-blue).

to focus our study on the building as a whole, as well as three representative rooms, chosen based on symmetrical shapes and regular internal distribution:

- a classroom facing west;
- a classroom facing east;
- the corridor at the main entrance.

Both classrooms are comparable in terms of size, internal characteristics, use and occupants and differ only in the orientation. Hence, they reasonably represent two opposed thermal conditions of the same type of classroom. A constant occupancy does not characterize the corridor during working hours. Nonetheless, it is considered significant for this study because it is a vast environment located in a central position of the building, with many openings and glazed windows.

In order to monitor and collect the air-temperature trends, in the real-world building, we deployed 13 IoT devices based on ST-Microelectronics STM32 Nucleo-64 boards [238] equipped with a low power transceiver module SPIRIT1 [237] with a sampling rate of 15 minutes. In Figure 4.13 shows the arrangement of the internal sensors (in red) and the external sensor (in green). This configuration allowed us to collect a data-set of 7777 samples in total. However, this is not large enough to exploit a prediction model efficiently based on neural networks. Thus, we generated an enlarged data-set by simulating the thermal energy behaviour of the building with EnergyPlus, exploiting the BIM model together with real weather data of about six years (TMY data), from 2010 to 2015. Indeed, by exploiting real weather information, the methodology can provide indoor air-temperature trends (in the form of time-series) with a lower error rate. In this way, we obtain a realistic synthetic data-set with values sampled every 15 minutes. In detail, we considered all the values between November and March that is the operational period of the building heating systems in the north of Italy. Then, the synthetic data was split into two independent subsets for training and testing purposes, containing about 70% and 30% of the initial samples, respectively. The training set (71901 samples in total) was used to train the prediction models and optimize their parameters, while the test set (14303 samples) was used to assess the prediction performance. The data-set with the real measurements (7777 samples at 15 minutes sampling rate) was then employed, for inference purposes and fine-tuning processes of the most promising neural architecture found.

4.6 Results

This Section shows all the characterizing results obtained for all the phases that compose the methodology presented. At first, how we obtain the realistic synthetic data-set is described. Subsequently, the Section highlights the advantages of using a consistent synthetic data-set in the training phase and real data from IoT sensors, installed in the building, for the inference phase. For this purpose we model and optimize a Non-linear Regressive Neural Network. Once validated the advantages of using a synthetic data-set through the results, the Section describes the design and optimization processes performed on state-of-art neural networks for time-series prediction exploited. At the same time, we try to introduce time labels (i.e. time of day and day of the week) within the data-set, to improve the

time prediction performance of the models implemented. Finally, we identify the most promising neural model and apply different fine-tuning methodologies. In this way, we demonstrate how to gradually change the model trained with the realistic synthetic data-set into a model based entirely on a real data-set. All without negatively affecting prediction performances. All this managing to improve the prediction time horizons present in the Literature.

4.6.1 Design of a indoor air-temperature consistent synthetic data-set

As case study, the methodology described in Section 4.4.1 has been applied on a primary school. It is a public building of about 14,500 m^2 spread on two floors (see Figure 4.2 and Figure 4.3) with brick walls facades, double glazed windows and pitched roofs. The building is connected to the district heating distribution system. During working-days, the heating system cycle is from 4:00 a.m. to 7:30 p.m.. To ensure a comfortable environment for users the ignition of Monday is anticipated on previous Sunday at 11:00 p.m.

The building has been equipped with 13 IoT devices, 12 indoor and 1 outdoor (see Figure 4.13), to send air temperature and relative humidity. Sensors have been installed in the most meaningful building zones according to its intended use, construction type and floors number (i.e. main entrance, classrooms, gym and student canteen). The Wireless Sensor Network has been evaluated to optimize the employed IoT devices with respect to the good result of the energy simulation. In this study, indoor devices have been placed in comparable rooms in terms of use and dimension characterized by a different orientation. Instead the outdoor device has been placed at the worst solar exposure to detect the minor outdoor temperature. The symmetrical shapes and the regular internal distribution of the building have allowed us to select some reference rooms to collect enough data for energy analysis.

The energy simulation has been performed for the whole heating season. In the following, we present the results from January 9^{th} to January 15^{th} 2017. The validation model is achieved by comparing the Temperature trends, as described in Section 4.4.1. For this purpose, we have analysed three selected rooms in the building (see Figure 4.13). These rooms have been chosen in relation to building shape and their occupancy during the week, as described in the following:

- **Room 1** is a classroom in the east part of the building occupied by 21 people. It is located in correspondence of thick trees that act as solar shield for the building.
- **Room 2** is a classroom in the west part of the building occupied by 22 people.

- **Corridor** is at the entrance of the school in a central position of the building. It is characterized by a very large environment with many openings and glazed windows. It does not have a constant occupancy during the day.

Both east- and west-oriented facades receive substantial contributions of thermal energy due to solar radiation. This is an advantage during winter season. Vice-versa, this translates into increased heat load during summer season, which would necessitate air conditioning. As the school is not equipped with such conditioning system, our simulations cover only the winter period.

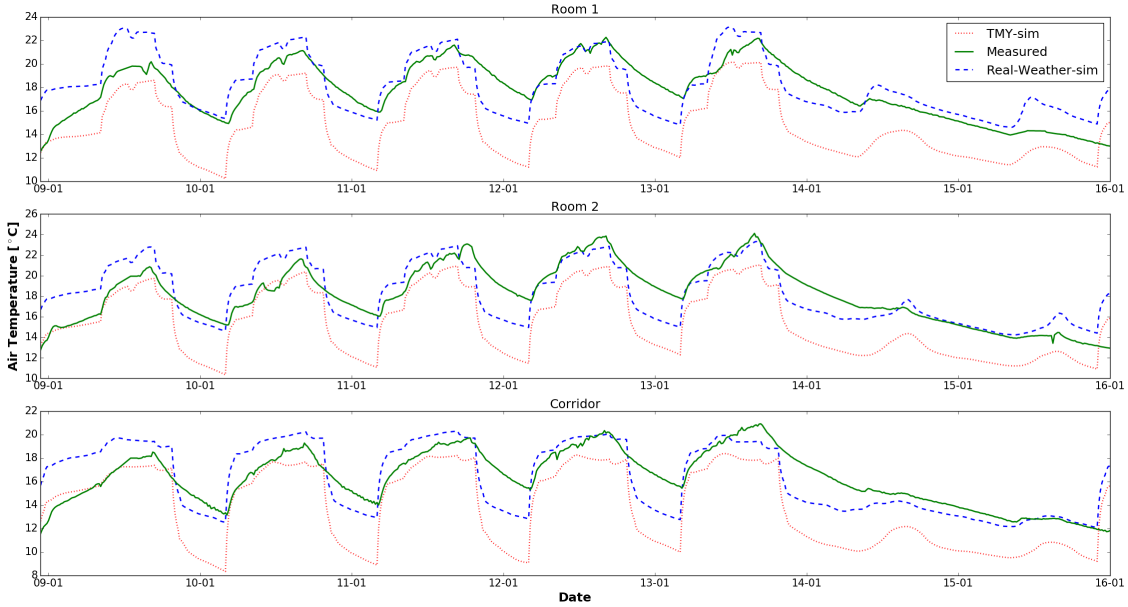


Figure 4.14: Simulated and measured indoor air temperature trends between 9$^{\text{th}}$-15$^{\text{th}}$ of January 2017.

Figure 4.14 reports three air temperature trends for the observation period: *i)* measured data coming from IoT devices (green line), *ii)* simulated with TMY Weather condition (red doted line) and *iii)* simulated with real-weather conditions (blue dashed line). The daily trends identifies the different phases of the heating cycle: *i)* ignition of the heating system (04:00 a.m.); *ii)* school entering (8:30 a.m.); *iii)* lunch break with opening windows for air circulation (12:30 a.m.); *iv)* school exiting (4:30 p.m.); *v)* shut-down of the heating system (07:30 p.m.). The air temperature chart highlights that measured data and simulation results with real-weather conditions have similar trends. On the contrary, the trend of TMY simulation results has the worst correlation with real samples. Especially during night hours, the temperature trend decreases to around 10 $°C$ with TMY simulations, while both measured and real-weather trends reaches about 16 $°C$. This because TMY refers to meteorological conditions, in terms of temperature and solar radiation, significantly different to daily weather samples. Both simulations with real-weather data

and TMY show a quicker slope of increase and decrease in the temperature trend when the heating system is switched *on* and *off*. This quicker response is related with the modelled heating capacity of the building. Indeed in the development of the BIM model the stratigraphy of the walls has been hypothesized following the suggestions in [207]. Those hypothesis were necessary due to a lack of information on real wall stratigraphy data in the building documentation.

To evaluate the performance of our simulations three indicators of dispersion have been used:

- *Mean Bias Difference* (MBD) measures the average squares of errors between simulated and measured values;
- *Root Mean Square Difference* ($RMSD$) represents the standard deviation of differences between simulated and measured values;
- *Mean Absolute Difference* (MAD) is defined as the average of the absolute difference of two variables X and Y.

Table 4.6 details the error rates given comparing measured data with simulations performed with both real-weather and TMY conditions. As shown in Table 4.6,

Table 4.6: Dispersion indicators of simulated indoor temperature against real measured values

Rooms	**Indicator [%]**	**Real-weather Sim vs Measured**	**TMY Sim vs Measured**
Room 1	MAD	8.02	16.82
	MBD	2.18	-16.64
	RMSD	9.78	19.01
Room 2	MAD	9.07	18.55
	MBD	0.10	-18.34
	RMSD	10.83	20.74
Corridor	MAD	9.35	16.94
	MBD	-0.17	-16.06
	RMSD	11.52	20.85

real-weather information improves the simulation results drastically with respect to TMY. Indeed, MAD, MBD and $RMSD$ have lower values with real-weather conditions. In particular in *Room 1*, we obtain a MAD of 8.02% against 16.82%; a MBD of 2.18% against -16.64%; a $RMSD$ of 9.78% against 19.01%. Similar results have been obtained for the other two rooms.

4.6.2 Synthetic data-set exploitation

The purpose of our methodology is to make predictions of indoor air temperature values in buildings in order to enable new energy policies. To achieve this target, the predictions need to be as accurate as possible while providing the longest time horizon possible. To assess the ability of our system to achieve this goal, we split the experimental assessment into two different phases, as detailed in Figure 4.1: i) *test*, where the model was tested on a large set of simulated data in order to obtain a reliable estimate of the prediction accuracy and prediction window; ii) *exploitation*, where the same model trained on synthetic data was tested on a smaller data-set of real measured data. This second assessment provides information about the robustness of the model in real-life conditions, as well as on the reliability of the simulations that were used for the training. For the two phases, we used the data-sets described in Section 4.5 that are both completely independent from the one used to train and optimize the model.

In both cases, the goodness of the predictions was first established by measuring the similarity between the predicted and the observed values, used as the ground truth. For this purpose, we adopted metrics that are widely used in statistical analysis and more specifically in time-series analysis literature [99]:

- *Mean Absolute Difference* (MAD), defined as the average absolute difference between predicted and observed values;
- *Root Mean Square Difference* (RMSD), defined as the standard deviation of differences between predicted and observed values.

Test on Simulated Data

In our first set of experiments, the model trained on simulated data was tested on an independent data-set, even in this case obtained by simulations. As already discussed in Section 4.5, our overall simulations included 6-year indoor air-temperature values at 15 min intervals, obtained with the strategy presented in [34]. As the test-case building contains a total number of 115 rooms, including uninhabited areas such as basements and attic, we decided to focus our study on three most representative rooms (facing East, facing West and Corridor) as well as on the building as a whole, obtained as the average of all the 115 time-series. This implies that the four NAR models described in Section 4.4.2 were trained on a training set containing the temperature time-series corresponding to the four different environments and then tested on the corresponding test sets. We made experiments at different prediction windows, up to a maximum of 270 min.

In Figure 4.15 we report the values of MAD and RMSD obtained at different prediction windows (see first column of the figure), separately for the three rooms of interest and for the whole building. As the building of our case-study is a public school, we focused our analysis on the only working hours.

By analyzing the values reported in the figure, we can make the following considerations.

- As expected, the prediction performance worsens as the prediction horizon increases with more or less the same trend for the four different scenarios.
- The prediction accuracy is comparable for the three individual rooms with MAD and RMSD values differing by few fractions of degree at best, which is a variation that would be hardly perceived by the human occupants. This is quite remarkable, if we consider that the three rooms are very different from each other in terms of thermal conditions.
- When considering the whole building, the prediction accuracy is better than the one achieved on the three individual rooms. This can be easily explained if we consider that the temperature values of the building were obtained by averaging the temperatures of all the 115 rooms. The averaging smoothes off temperature spikes that might be present in the individual rooms, especially if these rooms are at the extremes of the temperature distributions of the whole building, like the three ones that were analyzed in our study.

Time [min]	Classroom Facing East		Classroom Facing West		Corridor		Whole Building	
	MAD [°C]	RMSD [°C]	MAD [°C]	RMSD [°C]	MAD [°C]	RMSD [°C]	MAD [°C]	RMSD [°C]
15	0,06	0,39	0,07	0,43	0,07	0,39	0,06	0,43
30	0,13	0,45	0,15	0,50	0,14	0,49	0,12	0,48
45	0,21	0,52	0,24	0,59	0,22	0,60	0,17	0,53
60	0,30	0,60	0,33	0,67	0,30	0,71	0,23	0,59
75	0,39	0,70	0,41	0,76	0,37	0,82	0,29	0,65
90	0,48	0,80	0,49	0,84	0,45	0,97	0,35	0,72
105	0,58	0,91	0,56	0,92	0,52	1,08	0,40	0,79
120	0,67	1,04	0,63	1,00	0,58	1,18	0,45	0,85
135	0,77	1,17	0,69	1,07	0,65	1,31	0,50	0,91
150	0,87	1,31	0,75	1,13	0,72	1,44	0,55	0,99
165	0,96	1,43	0,80	1,19	0,77	1,46	0,60	1,03
180	1,04	1,55	0,85	1,25	0,85	1,57	0,65	1,11
195	1,13	1,66	0,90	1,31	0,93	1,78	0,71	1,19
210	1,20	1,77	0,94	1,37	1,02	1,89	0,77	1,26
225	1,27	1,86	0,99	1,41	1,11	2,03	0,83	1,37
240	1,34	1,94	1,03	1,46	1,20	2,24	0,89	1,48
255	1,40	2,03	1,06	1,51	1,28	2,31	0,95	1,55
270	1,47	2,13	1,11	1,59	1,38	2,51	1,01	1,65

Figure 4.15: Prediction performance on the simulated data-set. Blue and green lines correspond to the maximum prediction horizons that guarantee acceptable thermal comfort, respectively for the individual rooms and the whole building models.

According to most standards and literature studies, to guarantee no impact on the thermal comfort perceived by the occupants, the operative temperatures should never fluctuate more than 1.1 °C (2.0 °F) within 15 min, nor change more than 2.2 °C (4.0 °F) within 1 h [83, 17]. Upon these considerations, in our study we established a value of MAD of about 1 °C as the maximum acceptable threshold for our temperature predictions. Based on this conservative threshold, we assessed the maximum prediction horizons that can be guaranteed by our models. As it can be seen in Figure 4.15, this prediction horizon is 180 min when considering the individual rooms (see blue-coloured line) and 270 min for the whole building (see green-coloured line), which is quite a remarkable time-window. Compared to our previous work [9], we were able to improve MAD and RMSD performance index on average by 21% and 13%, respectively.

Exploitation on Real Data

In our second set of experiments, the models trained on the simulated data were exploited on a small data-set of real temperature measurements that was described in Section 4.5. This data-set was sampled by temperature sensors operating in a range between −40 °C and +120 °C with a temperature sensitivity and accuracy of 0.016 °C and ±0.5 °C respectively. The sampling frequency is 15 min and there are no missing values. Even though this second data-set is very limited in terms of number of data samples, and hence less significant for the performance evaluation, it can still provide very meaningful insights into the robustness of the predictions in real-life conditions, as well as into the reliability of the simulations that were used to generate the training data. The overall results obtained on the real data-set are shown in Figure 4.16 with the same content and format of the ones obtained on the simulated data.

The prediction performance obtained on the real data-set is lower than the one obtained on the simulated data-set. This is reasonably due to some intrinsic differences between the real measurements data and the simulated ones. For example, due to some unpredictable actions of the human occupants (e.g., opening/closing windows), which might considerably change some temperature values.

If we establish again a value of about 1 °C as the maximum acceptable threshold on MAD, we obtain that the maximum prediction horizons of our models on the real data are 105 and 180 min, respectively for the individual rooms (blue line in Figure 4.16) and the whole building (green line in Figure 4.16). While these horizons are lower than the ones estimated on the simulated data, they are still remarkably long, which confirms the wide usability of the model in Demand Response applications [226, 61, 272].

All the other considerations made on the simulated data-set are still valid.

Time [min]	Classroom Facing East		Classroom Facing West		Corridor		Whole Building	
	MAD [°C]	RMSD [°C]	MAD [°C]	RMSD [°C]	MAD [°C]	RMSD [°C]	MAD [°C]	RMSD [°C]
15	0,15	0,23	0,13	0,30	0,11	0,16	0,08	0,12
30	0,29	0,43	0,26	0,51	0,21	0,32	0,17	0,23
45	0,42	0,61	0,39	0,71	0,34	0,49	0,25	0,33
60	0,54	0,75	0,51	0,90	0,49	0,66	0,35	0,46
75	0,64	0,89	0,62	1,06	0,69	1,26	0,45	0,58
90	0,75	1,03	0,73	1,16	0,86	1,34	0,54	0,70
105	0,86	1,17	0,84	1,30	1,01	1,50	0,63	0,82
120	0,98	1,31	0,95	1,50	1,17	1,80	0,73	0,95
135	1,10	1,45	1,05	1,73	1,34	2,18	0,82	1,08
150	1,20	1,58	1,11	1,64	1,48	2,46	0,90	1,19
165	1,30	1,68	1,20	1,92	1,59	2,82	0,95	1,25
180	1,38	1,78	1,26	1,87	1,79	3,52	1,03	1,35
195	1,45	1,86	1,35	2,06	1,91	3,66	1,13	1,46
210	1,51	1,94	1,41	2,16	2,02	3,68	1,21	1,57
225	1,58	2,03	1,48	2,20	2,26	4,44	1,28	1,67
240	1,65	2,11	1,56	2,43	2,40	4,28	1,33	1,75
255	1,70	2,17	1,59	2,36	2,54	4,51	1,39	1,83
270	1,76	2,25	1,73	2,81	2,80	4,86	1,46	1,93

Figure 4.16: Prediction performance on the real-measurements data-set. Blue and green lines correspond to the maximum prediction horizons that guarantee acceptable thermal comfort, respectively for the individual rooms and the whole building models.

Besides the traditional assessment based on prediction performance, the goodness of a temperature prediction model can be evaluated indirectly by estimating the impact that a temperature change w.r.t. the observed values would eventually have on the well-being of the occupants. To do so, in our work we exploited the method described in [83], which is implemented in most international standards for the design, operation, and commissioning of occupied spaces [121, 17], on our real measurements data-set.

This method leverages upon the quantification of the following two metrics:

- *Predicted Mean Vote* (PMV), a -3 to $+3$ index estimating the state of well-being of a group of individuals, where -3 means feeling too cold, $+3$ means feeling too hot and 0 represents a perfect thermal well-being.

- *Percentage of Person Dissatisfied* (PPD), a 0 to 100 value estimating a percentage of people dissatisfied by the thermal conditions of the environment.

More specifically, in our analysis we exploited the well-known sensation scale defined in [83], which puts in relation the values of PMV and PPD and defines the thermal comfort area as the range of values for which $-0.5 < PMV < +0.5$. As reported by [83], this range is associated to the maximum probability of having at least 90% of the population of occupants completely satisfied by the thermal conditions of the environment.

More specifically, we applied the following procedure:

1. We computed PMV/PMD indices for all the target environments of our real-world demonstrator. Again, we focused only on the working hours, which are the ones that are significant for the temperature predictions.

2. For each prediction horizon, we computed the percentage of predicted values that are within the ±0.5 PMV thermal comfort area.

The obtained results are reported in Figure 4.17, separately for the three individual rooms and the whole building models. The first row of the table in figure (at time 0, red coloured) shows the thermal comfort values obtained on the observed data, which can be used as a reference. The following rows of the table in figure report the thermal comfort values obtained on the predicted data at increasing prediction horizons. The rationale of the experiment is: the closer the thermal comfort values to the corresponding reference values at time 0, the better the prediction. The blue and green areas in the figure correspond to the prediction windows that were identified as reliable in our previous prediction performance analysis on the same data (respectively up to 105 min for the individual rooms and 180 min for the whole building).

	Values within the ±0.5 thermal comfort area [%]			
Time [min]	Classroom Facing East	Classroom Facing West	Corridor	Whole Building
0	77,5	90,5	91,3	90,7
15	76,9	90,5	91,5	95,8
30	75,8	89,1	92,1	94,7
45	74,0	88,7	91,3	93,9
60	74,0	88,9	90,9	93,3
75	72,8	88,1	89,6	92,7
90	72,9	87,4	87,8	92,5
105	73,1	86,0	85,6	91,5
120	72,0	84,2	83,3	90,5
135	71,4	82,5	81,8	90,7
150	69,8	80,7	80,6	89,9
165	67,9	80,2	80,1	89,1
180	66,1	79,2	79,1	87,7
195	64,2	78,2	77,2	86,7
210	61,9	76,9	74,0	83,6
225	59,9	75,3	71,2	81,4
240	58,4	74,8	67,3	77,8
255	56,4	74,0	64,8	74,5
270	55,2	73,1	62,1	71,7

Figure 4.17: PMV/PPD evaluation on real data: percentage of predicted values within the ±0.5 thermal comfort area. The first row (in red) represents reference percentages, computed on the observed temperature values. The following rows report percentages computed on the predicted values, at increasing prediction horizons.

If we look at the blue and green areas, we can observe that the values are generally high, with a difference with respect to the reference values that is always below 6% for the individual rooms and below 3% for the whole building. As for the prediction performance metrics, the percentage of predicted values within the thermal comfort zone tends to decrease with the prediction horizon, even though with some minor oscillations. Figure 4.18 clearly shows this trend.

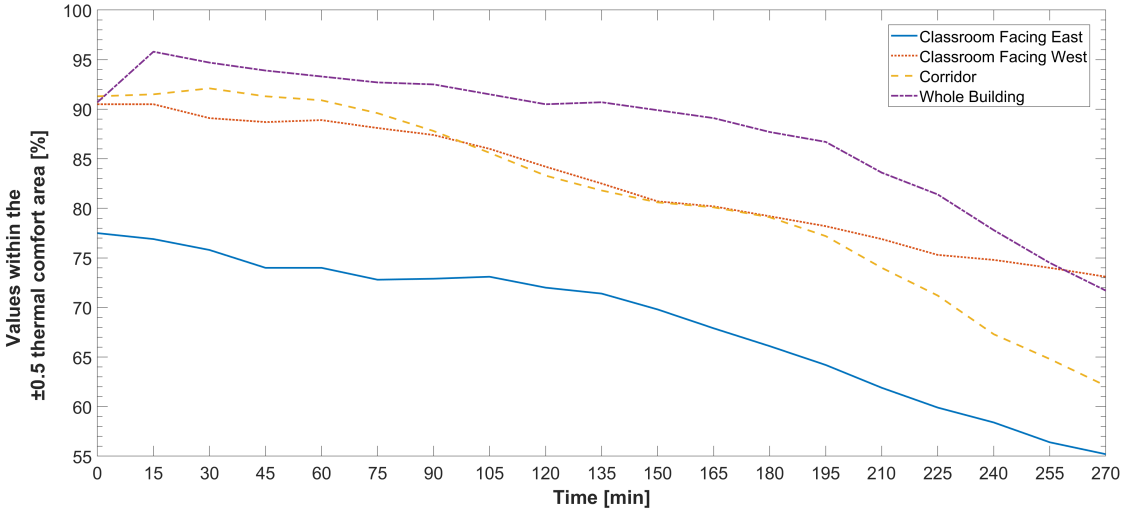


Figure 4.18: PMV/PPD evaluation graph on real data.

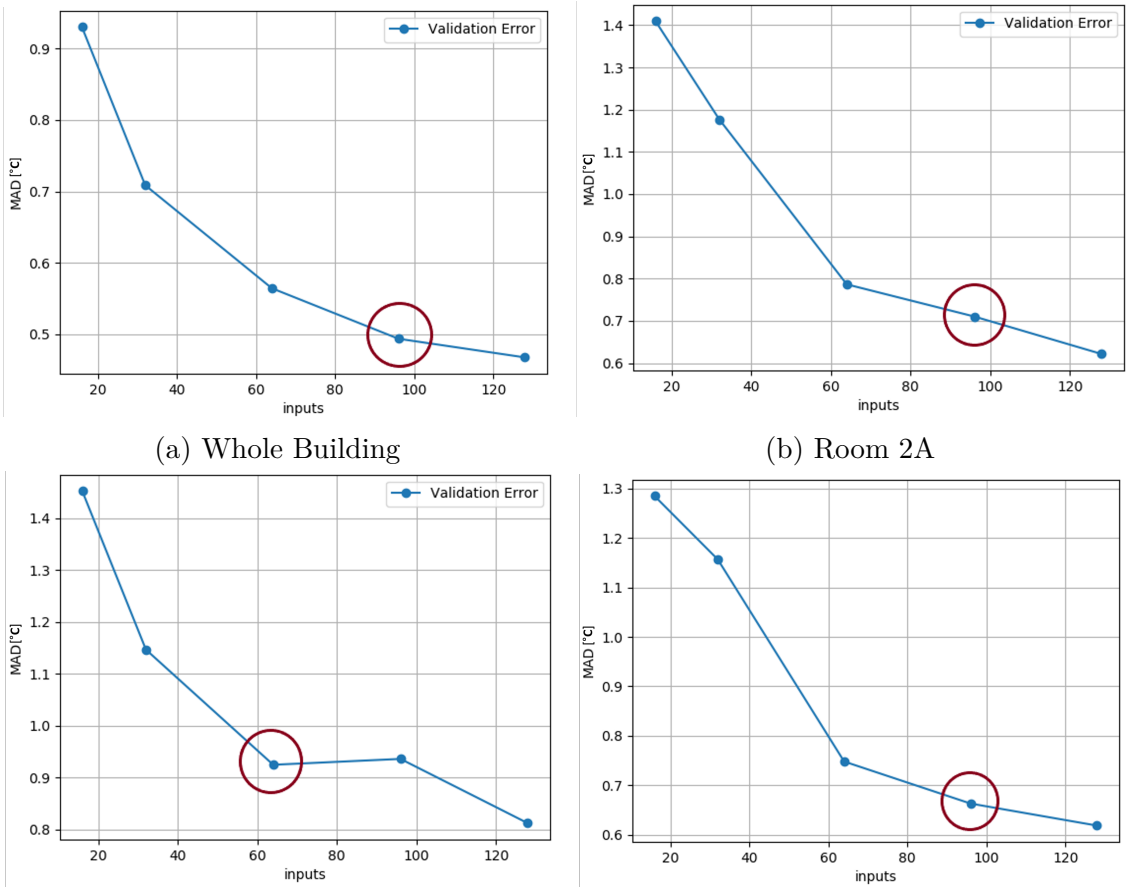
Nonetheless, the values do not have a sudden drop even outside the nominal prediction windows of 105 and 180 min, which confirms that our thresholds were conservative enough. Again, the performance of the models in different types of environments are comparable.

4.6.3 Neural Networks implementation and optimization

The results presented in Section 4.6.2 allow us to state that the exploitation of a hybrid model in the context of the prediction of indoor air-temperatures in buildings is valid both quantitatively and qualitatively. Consequently, we start designing and implementing the most promising state-of-art neural network for time-series prediction to find the best model for this framework.

1D-CNN

Inputs We start evaluating the 1D-CNN with different time-steps values (i.e. 16,32,64,96 and 128). Then we investigate others hyper-parameters: i) the batch size: 128, ii) the filters: 64, iii) the forecasting horizon: 7,5 hours and iv) the hidden neurons: 100. Figure 4.19 shows the results obtained results. Clearly as much input we fed, the most accurate is the prediction, thus we decide to test until 120 inputs, that are 30 hours before. We believe that this value represents the maximum limit applicable because, generally, exploiting bigger value would add complexity, increasing both the training time and the risk of overfitting. The number of regressors chosen is for all the characterizing rooms 96, except for Room 1D that is 64. Note that even though 120 inputs guarantee a smaller error, the negligible improvement it adds does not worth the increased training time.



(a) Whole Building

(b) Room 2A

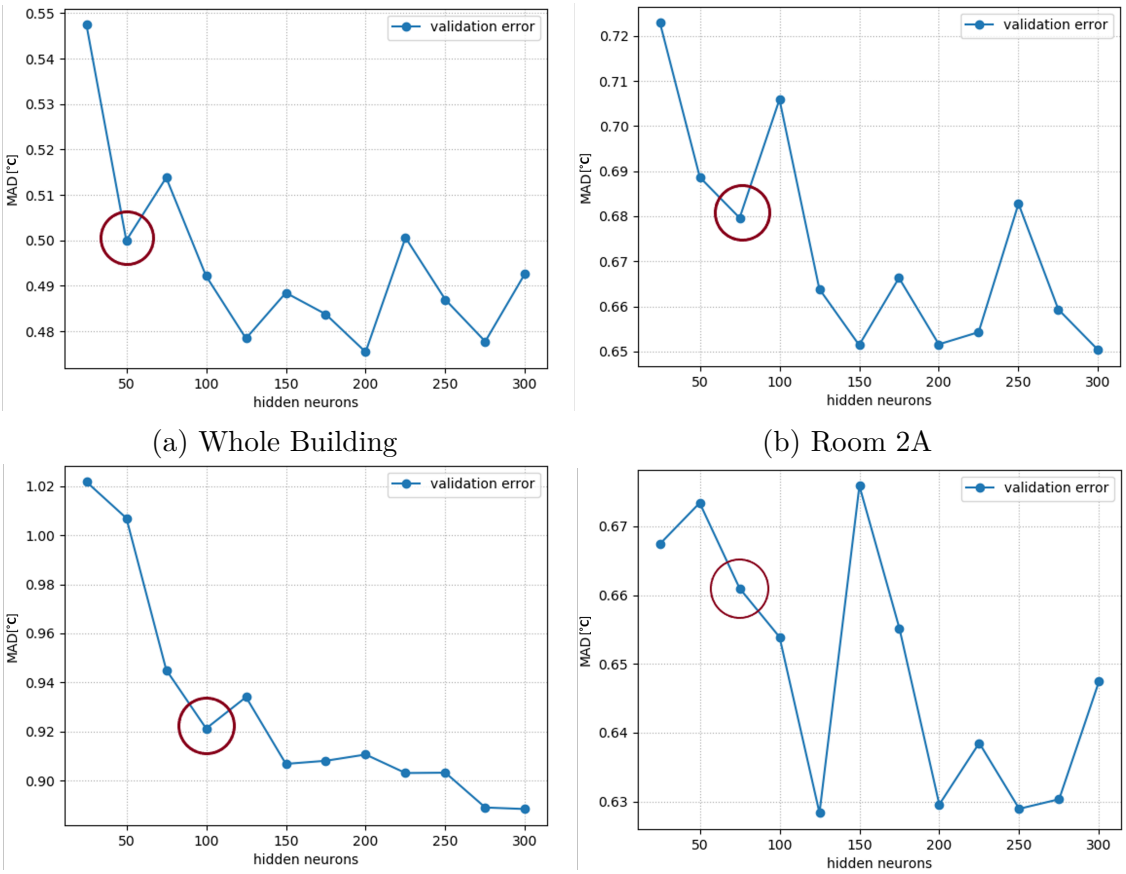
(c) Room 1D

(d) Corridor

Figure 4.19: 1D CNN - MAE variation against regressors number.

Neurons After having identified the number of regressors, we evaluated the effect of changing the number of hidden neurons in the first Dense layer. Therefore, all the other parameters are unchanged. Figure 4.20 shows the results. The red circles highlight the best number of neurons to be adopted. We can see that increasing the number of neurons does not produce a considerable reduction of the error, whose value is fluctuating within the very small range of 0.03 $°C$, hence we chose a number of neuron that is a trade-off between training time and accuracy.

Batch and learning rate Then we start optimizing the batch and the learning rate, doing it jointly, due to their relationship. Indeed, the former affects the number of iteration while the latter the "speed" of convergence. Therefore, we expect that a smaller batch size with a lower learning rate can achieve comparable results with large batch and learning rate. Figure 4.21 depicts the obtained results. Consequently, we choose the parameters that allow us to achieve better accuracy,



(a) Whole Building

(b) Room 2A

(c) Room 1D

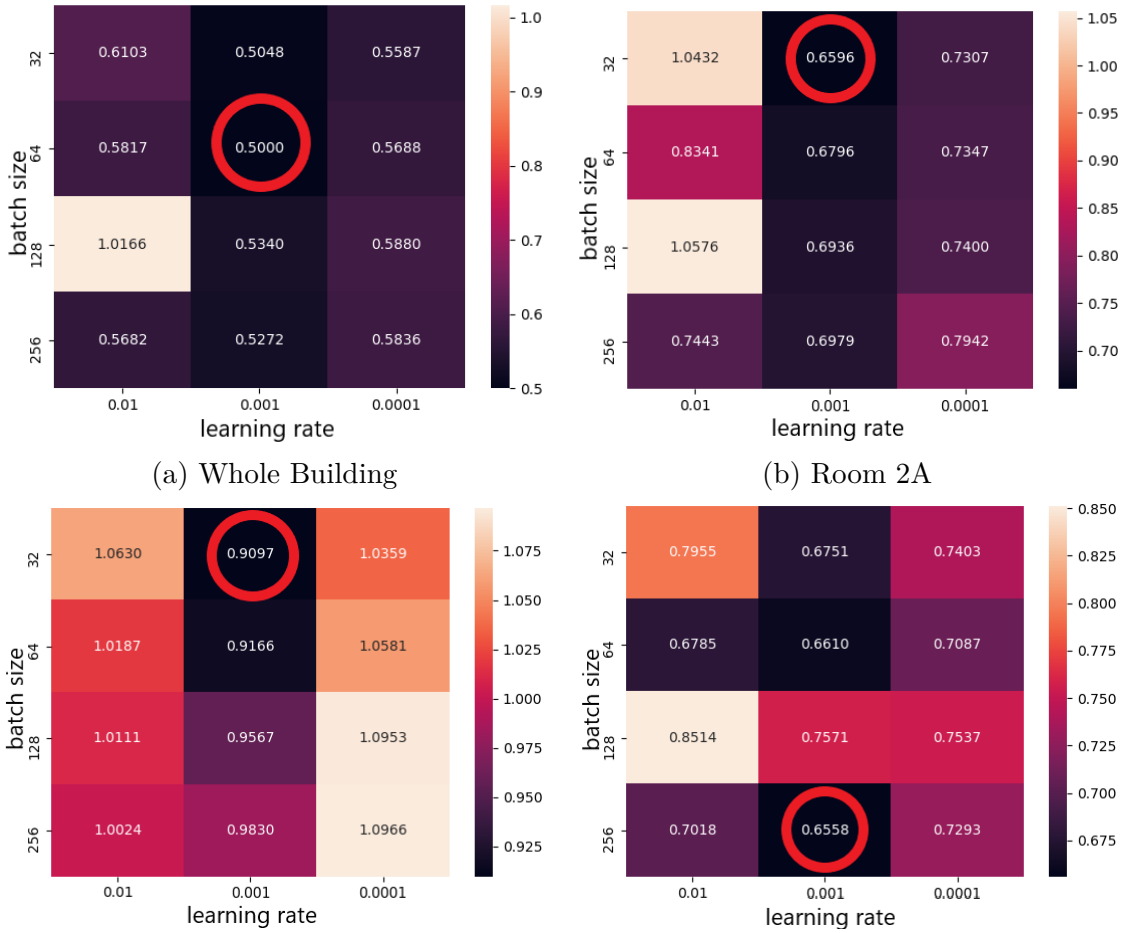
(d) Corridor

Figure 4.20: 1D-CNN - MAE variation against number of neurons.

but as we can see the values are all comparable, which could be a reasonable choice also to choose the more straightforward (and faster trainable) model, which is the one with the bigger batch size and learning rate).

Filter optimization Finally, the number of filters used in the convolution has been optimized. With all the parameters chosen, as explained in the previous paragraphs, we evaluate the network with a different number of filters. The results obtained are shown in Figure 4.22. As in the neurons identification case, we select several filers that produce an excellent performance without increasing the complexity of the network. In Room 1D and Whole building, we notice that the increased number of filters degraded the performance, maybe a principle of overfitting occurs.

Summarizing, in Table 4.7 reports the best configuration of hyper-parameters for all the characterizing rooms and the whole building.



(a) Whole Building

(b) Room 2A

(c) Room 1D

(d) Corridor

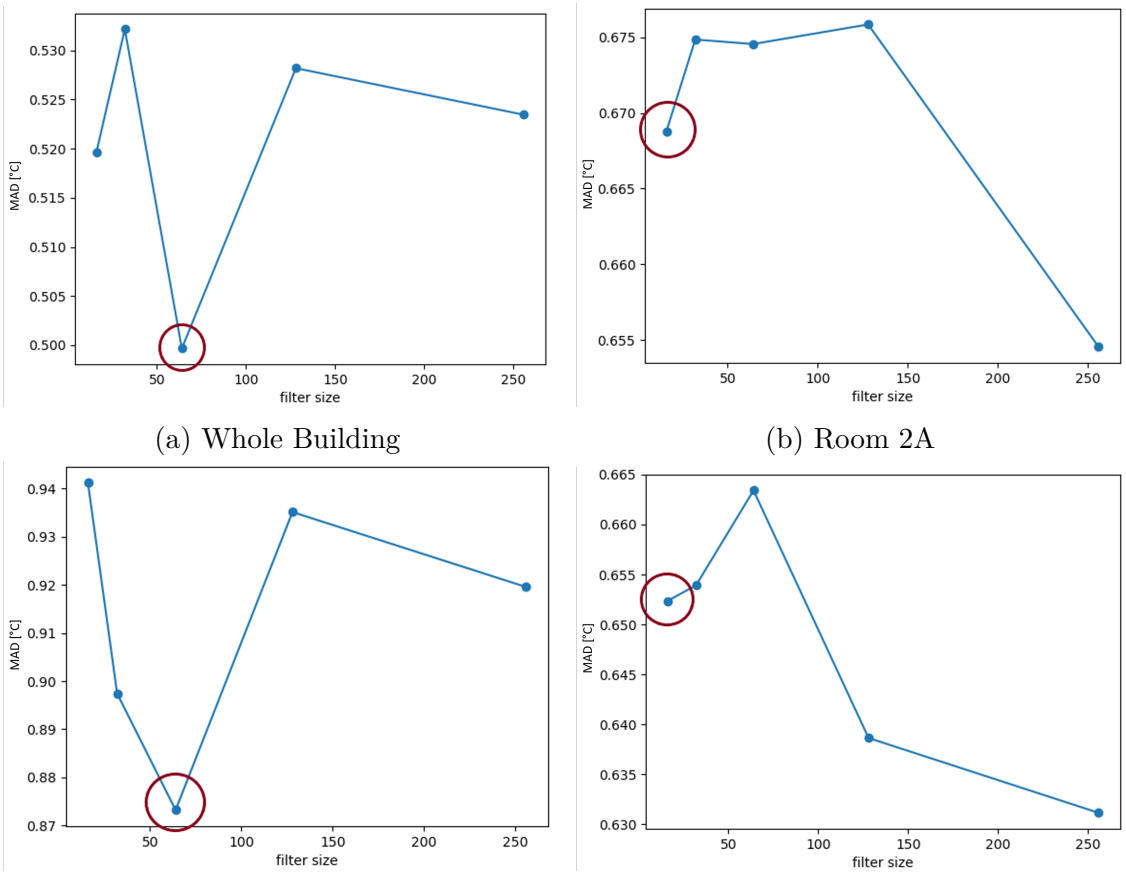
Figure 4.21: 1D-CNN - MAE variation against batch and learning rate.

Table 4.7: 1D-CNN hyper-parameters configurations.

Zone	Regressors	Neurons	Batch	Learning rate	Filters
Corridor	96	75	256	0.001	16
Whole Building	96	50	64	0.001	64
Room 2A	96	75	32	0.001	16
Room 1D	64	100	32	0.001	64

LSTM

In according to the considerations of the previously discussed 1D-CNN case, we start optimizing the i) inputs, ii) batches and iii) learning rate for the LSTM.



(a) Whole Building

(b) Room 2A

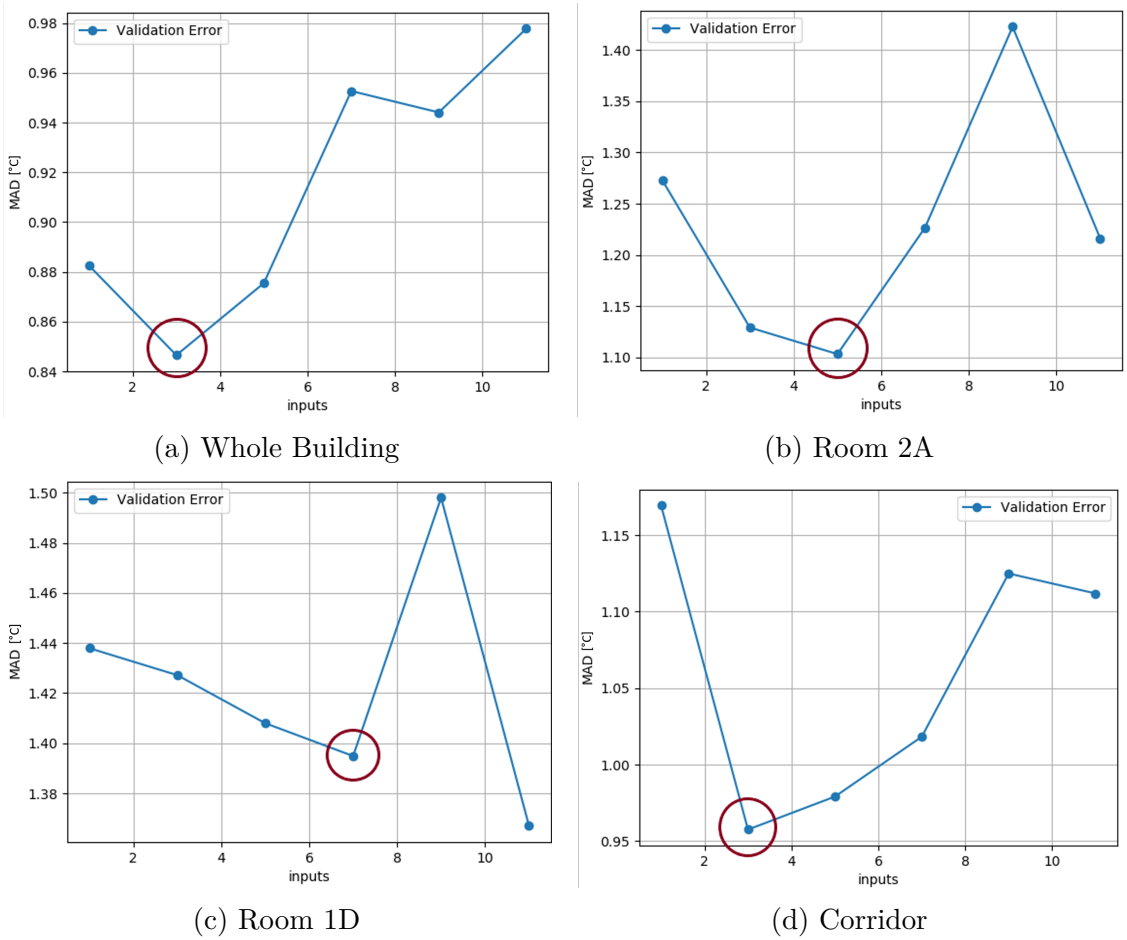
(c) Room 1D

(d) Corridor

Figure 4.22: 1D-CNN - MAE variation against filters number.

Inputs The first parameter we optimized is the number of regressors. In this case, the number of inputs can be smaller than in the 1D-CNN, thanks to the cell mechanism that allows to "memorize" the state of the previous batches. Therefore, we evaluated the network for 1, 3, 5 , 7, 9 and 11 timesteps. The other parameters used are characterized as following: i) batch size: 256, ii) forecasting horizon: 7,5 hours, iii) training epochs: 100; iv) number of LSTM cells: 50. With this configuration for each room and the whole building, we tried different inputs and chose the most performing in terms of prediction accuracy. Figure 4.23 shows the experimental results.

Epochs In order to identify when stopping the training phase, to avoid the risk of overfitting, we investigate the loss values at each epoch. This allows us to understand when learning capacity becomes negligible, and it is reasonable to stop it. The results are shown in Figure 4.24, where the yellow line is the epoch we chose as the limit. This experiment was carried out using a batch size of 256, which is the



(a) Whole Building

(b) Room 2A

(c) Room 1D

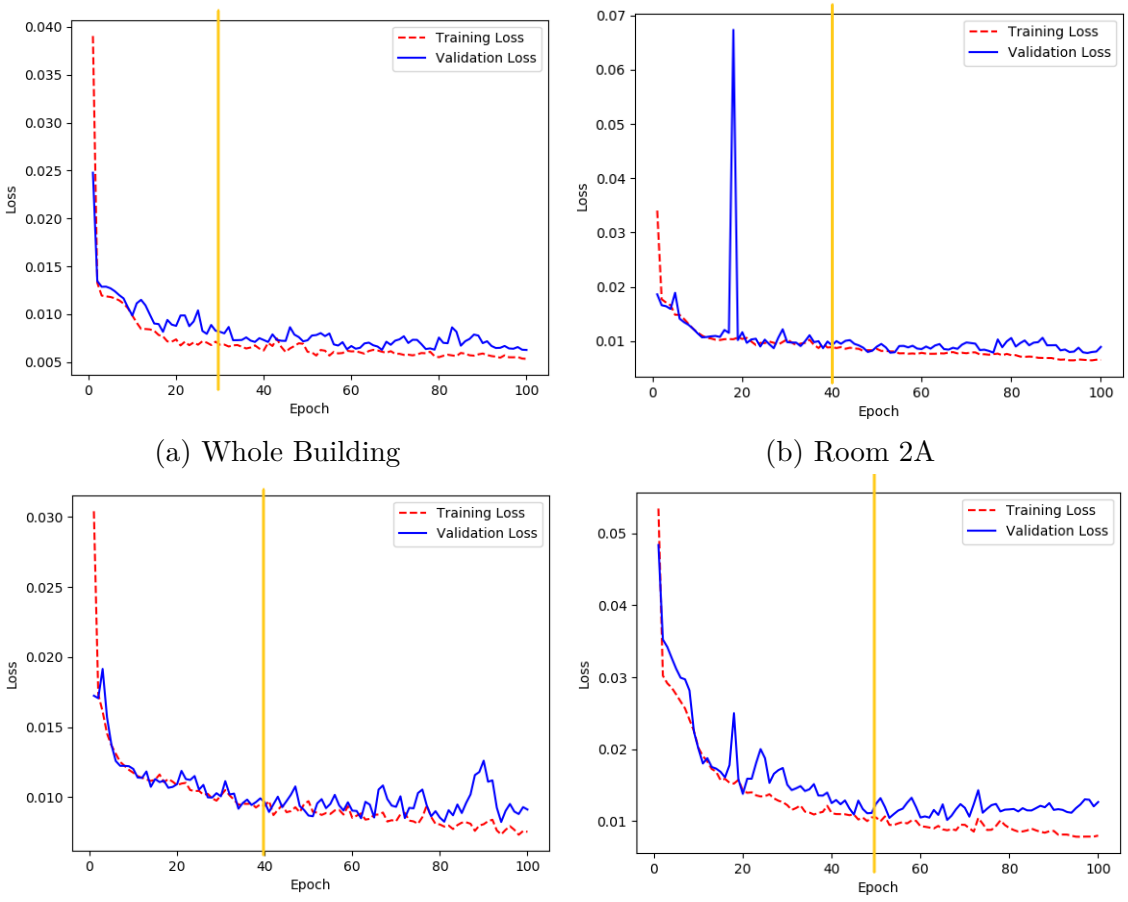
(d) Corridor

Figure 4.23: LSTM - MAE variation against regressors number.

maximum we are going to test in the next step, so the stopping point is conservative since with a smaller batch size there will be more iterations until that point.

Batch and learning rate Progressively, we started optimizing the batch size and the learning rate, testing the values 32, 64, 128 and 256 for the batch size, while 0.01, 0.001, 0.0001 for the learning rate respectively. The results are reported in Figure 4.25, with a red circle that highlight the best parameters value found. We can notice a high variability in the accuracy, with the best values reached with small learning rates combined with small batch sizes.

Neurons optimization Lastly, we evaluate the effect of changing the number of LSTM cell units. In according to the previous characterization settings, we try changing number of units as following: 50, 100, 150, 200, 250, 300. Figure 4.26 shows all the results, which highlighted the value correspondent to the number of



(a) Whole Building

(b) Room 2A

(c) Room 1D

(d) Corridor

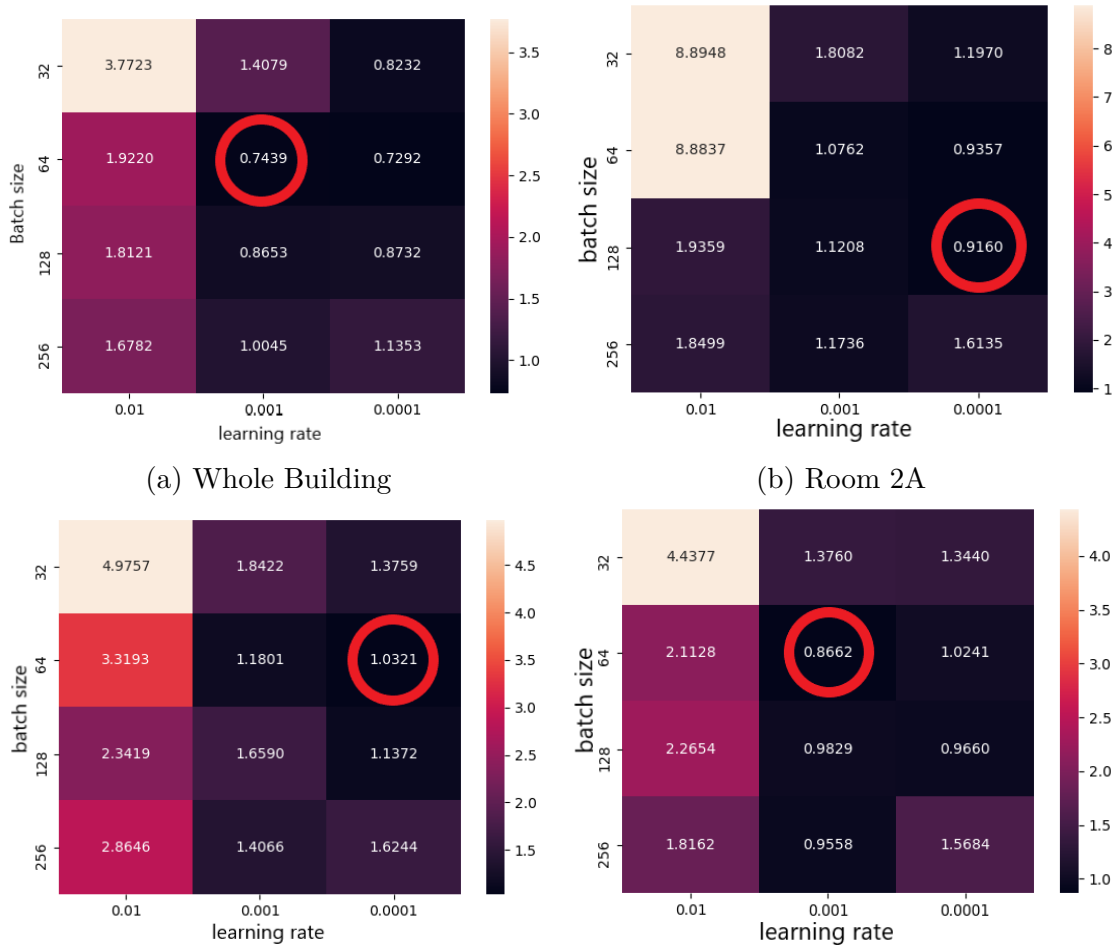
Figure 4.24: LSTM - Epochs loss evaluation.

neurons chosen. We see that increasing the number of LSTM units after a certain value, let the error increase, probably due to overfitting problem.

Table 4.8 show the best configuration of hyper-parameters for all the characterizing rooms and the whole building.

Table 4.8: LSTM hyper-parameters configurations.

Zone	**Regressors**	**Epochs**	**Batch**	**Learning rate**	**Units**
Corridor	3	50	64	0.001	100
Whole Building	3	30	64	0.001	100
Room 2A	5	40	128	0.0001	150
Room 1D	7	40	64	0.0001	150



(a) Whole Building

(b) Room 2A

(c) Room 1D

(d) Corridor

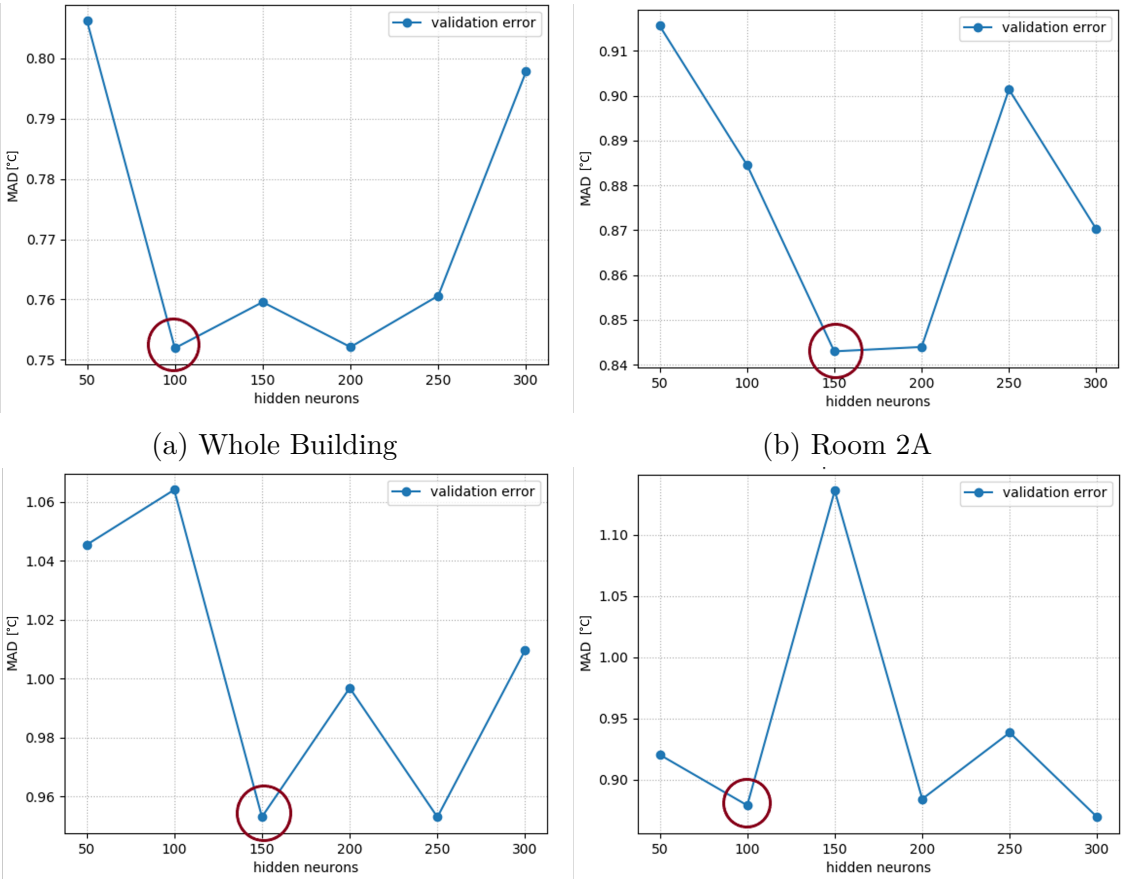
Figure 4.25: LSTM - MAE variation against batch and learning rate.

ESN

Finally, we start implementing and optimizing the last model: the ESN.

Input Optimization As in the previous cases, we start optimizing the number of regressors. Thus, we evaluated the network with 16, 32, 64, 96 and 128 timesteps. The other hyper-parameter are initialized as following: i) reservoir nodes: 200, ii) spectral radius: 4, iii) noise: 0.01 and iv) sparsity: 0.1. Figure 4.27 shows the results. For all the implemented networks, we adopt 96 regressors, which corresponds to a smaller error.

Noise and Spectral radius As the second step, we tried different state noises and spectral radius values. The value tested are respectively: i) the noises: 0.005,



(a) Whole Building

(b) Room 2A

(c) Room 1D

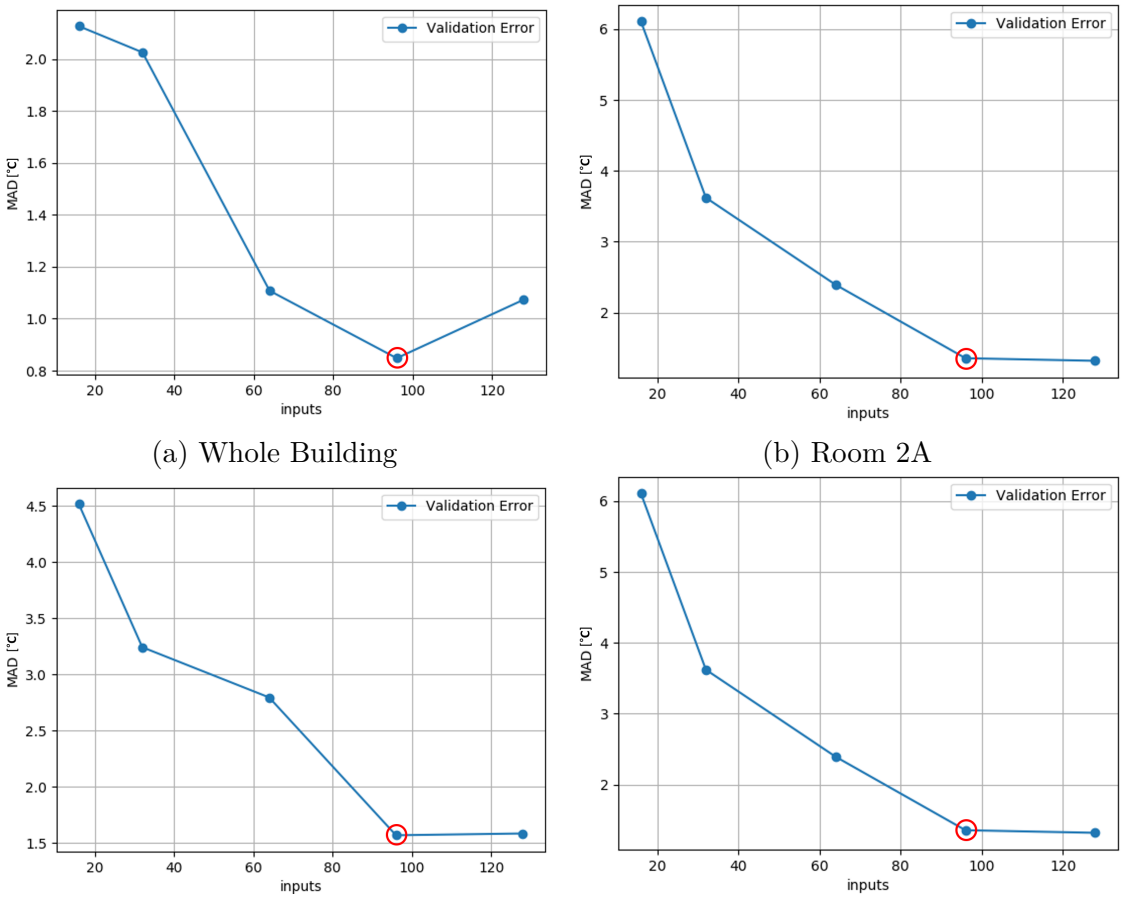
(d) Corridor

Figure 4.26: LSTM - MAE variation against cells number.

0.01, 0.05, 0.1, 0.5, ii) the spectral radius: 0.001, 0.1, 1, 4, 10. Usually, the spectral radius is set around 1, but for long-term prediction, a higher value is desirable [25]. Results are shown in Figure 4.28. In this case, the choice of parameters is a mathematics trick. Indeed, in Figure 4.28, we can see that noise contribution is almost negligible, and the difference, even though really small, in the accuracy is principally due to the variation of the spectral radius.

Reservoirs and sparsity Finally, we investigate the values of the size of the reservoir and the sparsity, respectively. We expect that increasing the reservoir size the accuracy improves. On the other hand, increasing the sparsity, the percentage of connections set to zero should degrade the performance. Figure 4.29 highlights these trends. We can see that the error increases according to the reservoir size, instead, the sparsity does not affect the performance significantly.

Table 4.9 summarizes the best configuration of hyper-parameters for all the



(a) Whole Building

(b) Room 2A

(c) Room 1D

(d) Corridor

Figure 4.27: ESN - MAE variation against regressors number.

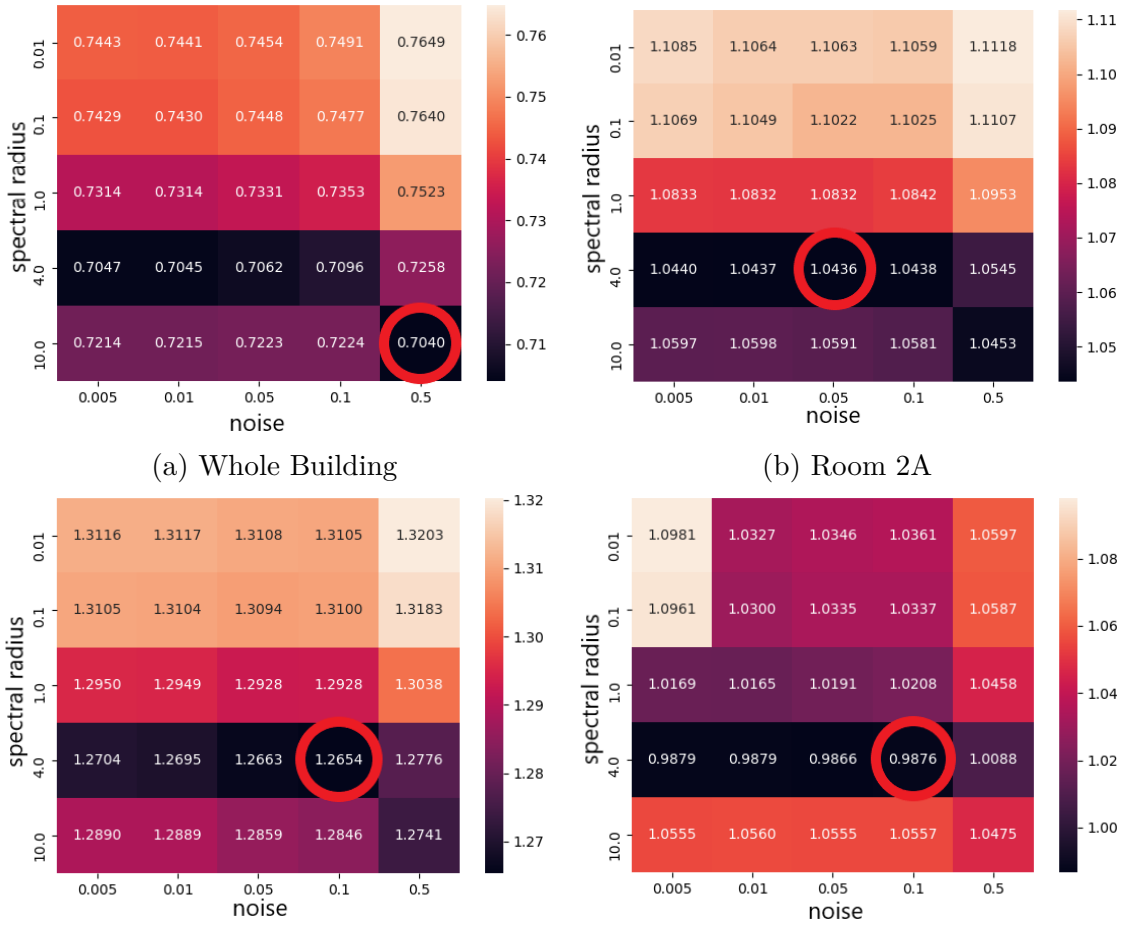
characterizing rooms and the whole building.

Table 4.9: ESN hyper-parameters configurations.

Zone	Regressors	Noise	Radius	Hidden nodes	Sparsity
Corridor	96	0.05	4	200	0.2
Whole Building	96	0.5	10	400	0.1
Room 2A	96	0.05	4	200	0.05
Room 1D	96	0.1	4	100	0.05

4.6.4 Univariate vs Multivariate

In this section, we discuss and compare the prediction results of all the implemented neural networks. Once the sets of the parameter are chosen, we train the



(a) Whole Building

(b) Room 2A

(c) Room 1D

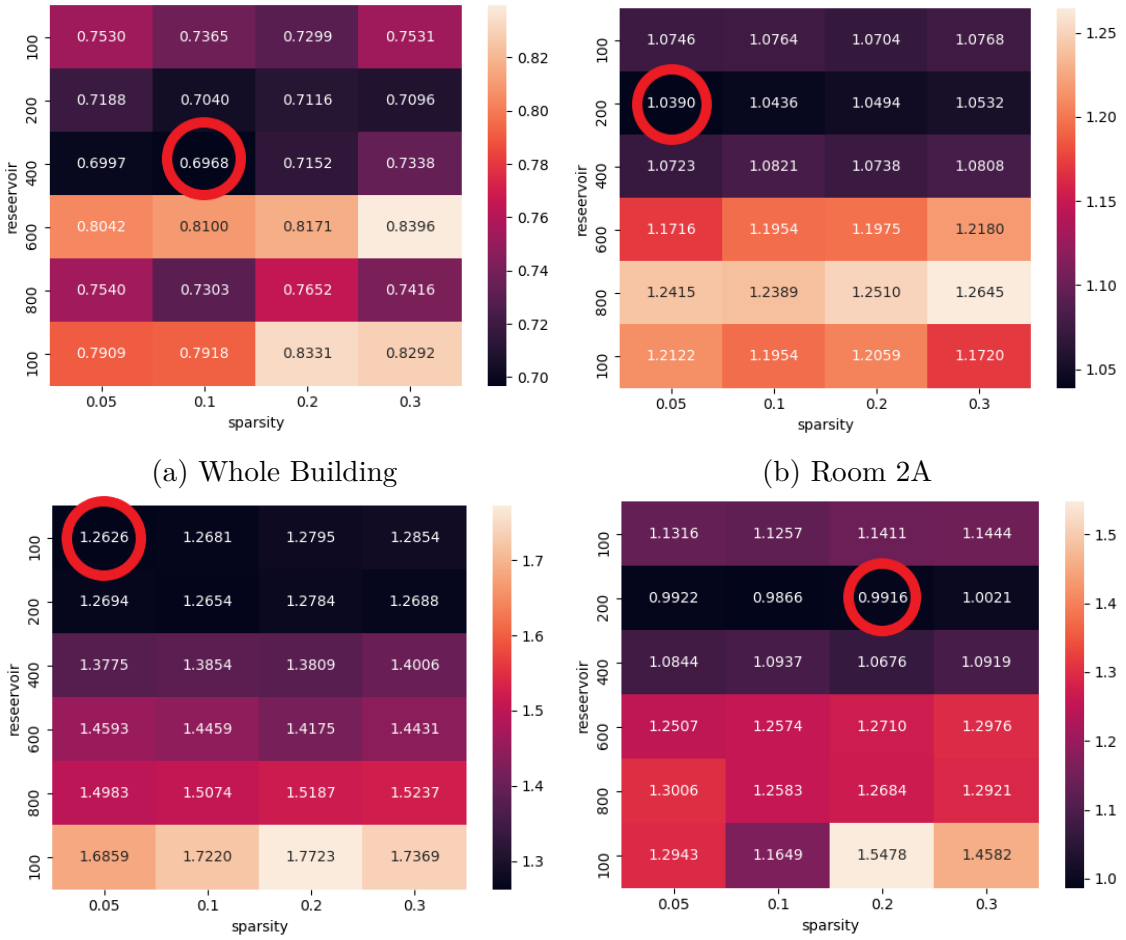
(d) Corridor

Figure 4.28: ESN - MSE for varying noises and spectral radius.

networks for multiple-time horizons, as described in Section 4.4, and we evaluate the analytical and qualitative metrics: the MAD (or MAE), RMSE and PMV/PPD respectively, in according to Section 2.5. As stated in Literature [82, 17], in order to not alter perceived thermal comfort of occupants, the operative temperature should not vary more than 1.1 °C, in the time range of 15 minutes and nor more than 2.2 °C within 1h respectively. According to these directives, we fixed at 2 °C the MAE threshold below which we consider our prediction acceptable.

Univariate

Firstly, we compared all the implemented neural networks in the univariate case. This means that the neural models receive only the indoor air-temperature values of the buildings as input. Figures 4.30 and 4.31 show in detail the results w.r.t. MAE and R^2 index, respectively.



(a) Whole Building

(b) Room 2A

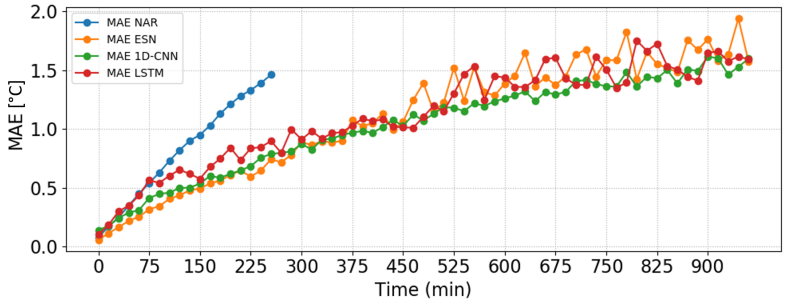
(c) Room 1D

(d) Corridor

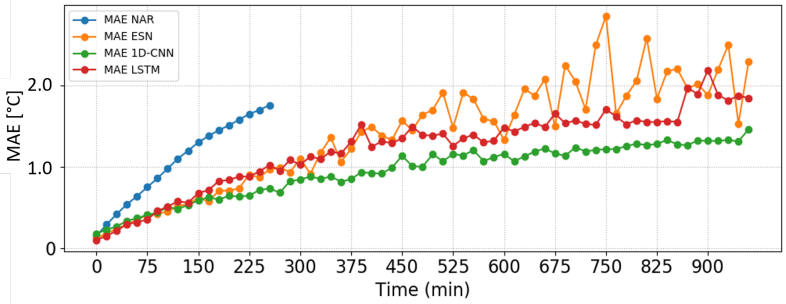
Figure 4.29: ESN - MAE variation against hidden nodes and sparsity.

Figure 4.30 clearly show that, for all the environments, the state-of-art neural networks outperform our benchmark NAR hybrid model, for which we have only predictions up to 270 minutes from Section 4.6.2. In Room 2A and Corridor, 1D-CNN outperforms the other networks for all the prediction horizons, while for Room 1D, for shorter predictions has a slightly worse accuracy but for forecasting of at least 825 minutes, 1D-CNN confirms to be the best architecture. Regarding the whole building, all networks have a similar behaviour of error, even though the 1D-CNN guarantees a more stable prediction. Resuming we are able to predict with a maximum error of 2 °C:

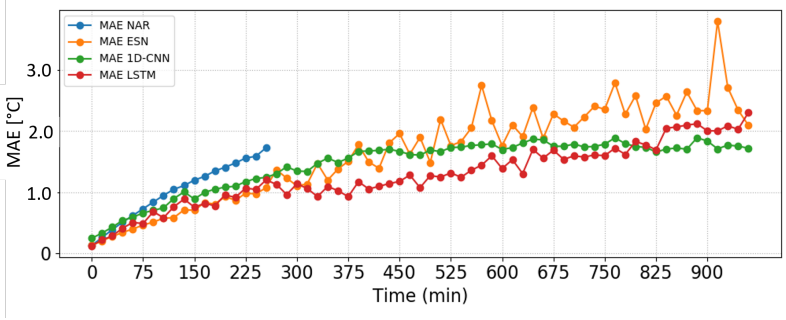
- Whole Building; 133 steps a-head (33,25 hours), with a MAE of 2.00 °C;
- Room 1D; 113 steps a-head (28,75 hours), with a MAE of 2.03 °C;



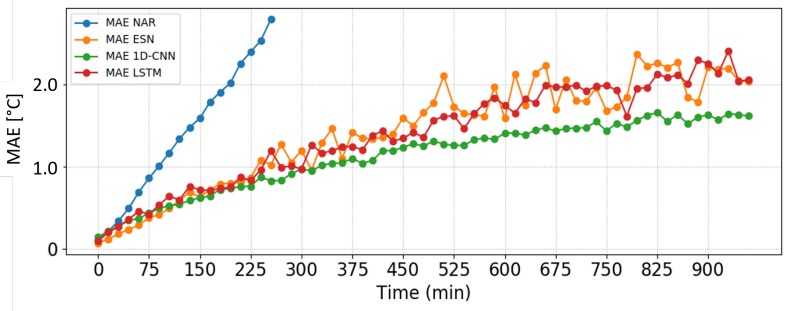
(a) Whole Building



(b) Room 2A

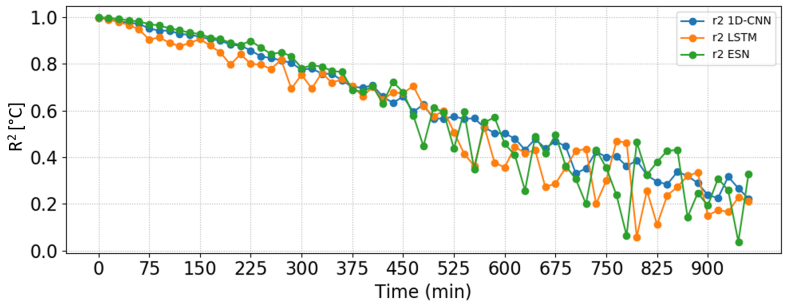


(c) Room 1D

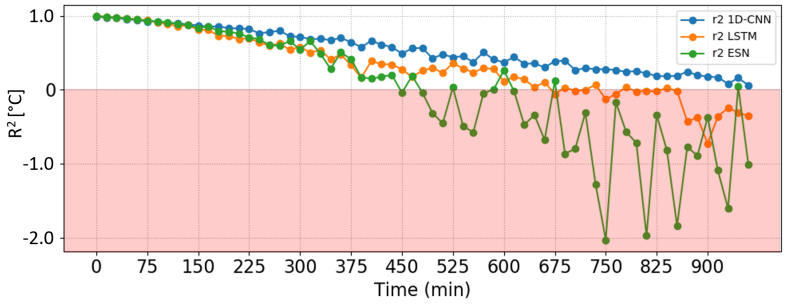


(d) Corridor

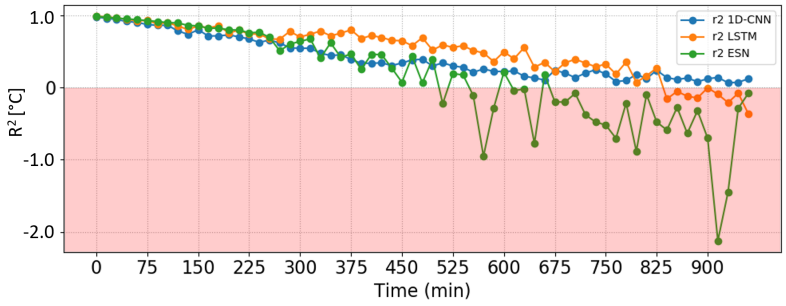
Figure 4.30: Hybrid models - MAE comparison



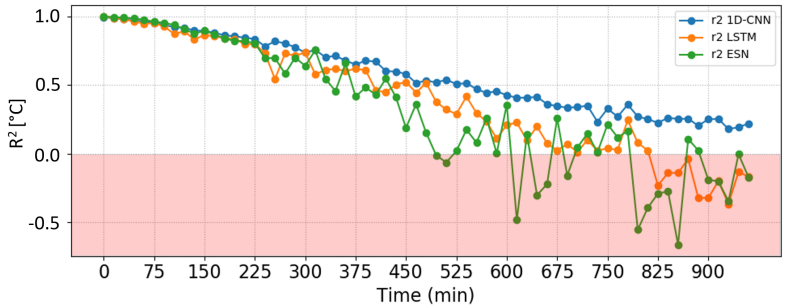
(a) Whole Building



(b) Room 2A



(c) Room 1D



(d) Corridor

Figure 4.31: Hybrid models - R^2 comparison.

- Corridor: 121 steps a-head (30,25 hours), with a MAE of 2.01 °C;
- Room 2A: 140 steps a-head (35,25 hours), with a MAE of 1.91 °C;

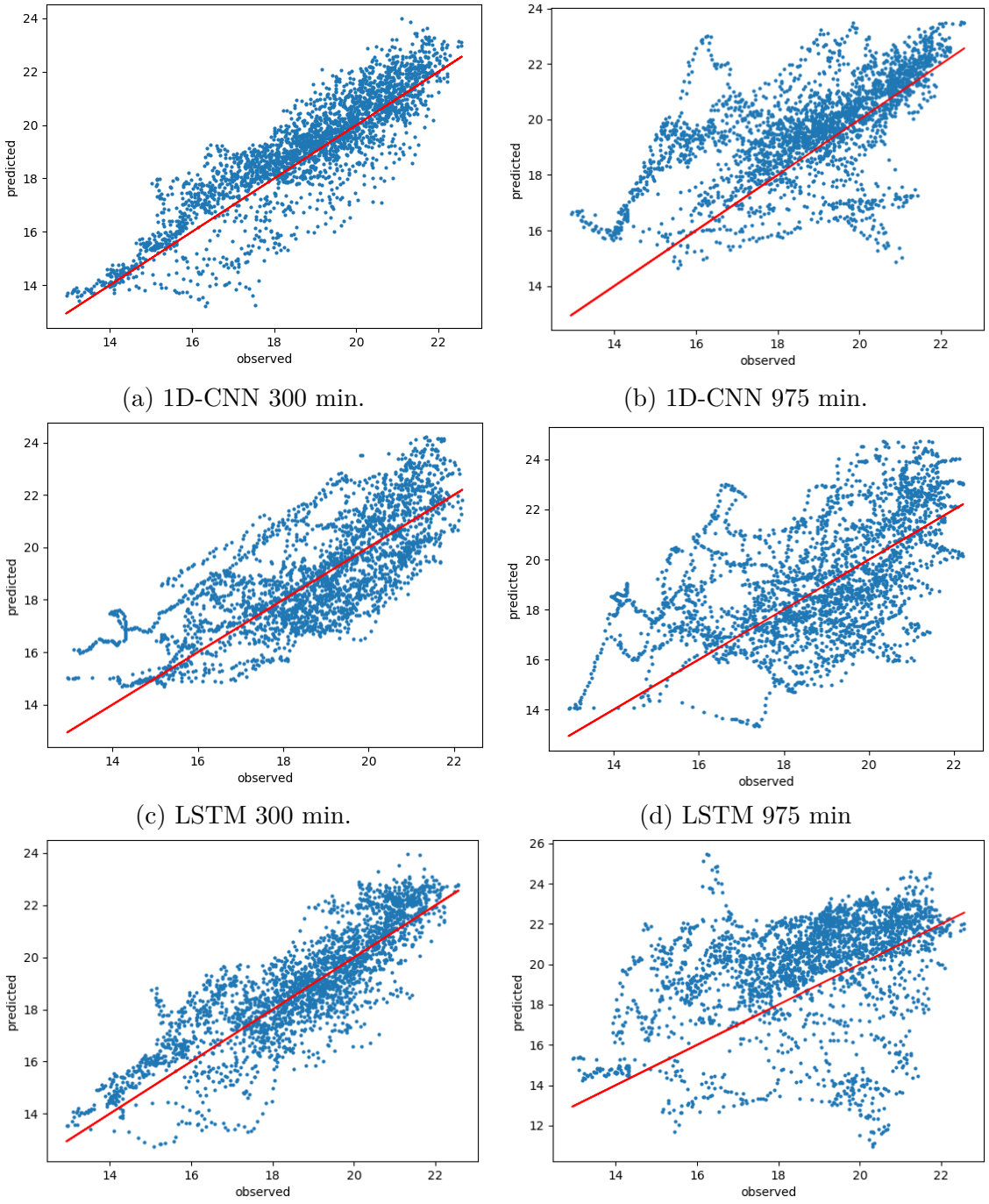
MAE cannot give a complete understanding of the fitting because it is an average of the error, this means that even though we have a MAE of about 2 °C, we could have both values that are perfectly predicted and others with large errors. Consequently, to validate these results, we perform the residual plot of some significant prediction time horizons of the characterizing Room 2A, as depicted in Figure 4.32. These residual plots highlight the distance of the blue points (the predicted values) from the red line that represents the error of the prediction. Analyzing the state-of-art networks for 300 and 975 minutes prediction we can see that in both cases, 1D-CNN is the one with the bigger concentration of points along the red line, which means a lower variance of the error, therefore less amount of "large" errors.

Finally, in order to validate the results of our 1D-CNN, we evaluate occupant's perception of comfort. Therefore, for all prediction horizons (until 149 steps ahead), we computed the Predicted Mean Vote (PMV) value for all the time steps and the Percentage Person Dissatisfied (PPD) (i.e. in the range of ± 0.5, that is our benchmark comfort range, as explained in Section 4.6.2). The complete results for all the rooms are shown in Tables 4.10, 4.11, 4.12 and 4.13. These Tables report the percentage values within the ± 0.5 thermal comfort area into prediction steps (i.e. a single k-step is equal to 15 minutes), for all the environments. The red cell in the tables represents the forecasting horizons for which the MAE is above 2 °C. Therefore, according to our assumptions, the values in the red zone are inaccurate. Moreover, the Figure 4.33 compares all the trends reported in tables. As depicted in Figure 4.33, initially, the trends rapidly decrease as the prediction horizon increases, for all the hybrid models. After a period, the values became almost stationary, with an unstable behaviour for long forecasting horizons. Whole Building is the prediction affected by the major worsening, with the percentage of comfortable temperature reduced up to 50% from the reference value (measured temperature). For all Rooms, instead, we can note that around the day-horizon (about 24 hours), the comfort level starts decreasing faster.

Multivariate

As introduced in Section 4.4.4, we decided to investigate how the neural networks react by inserting some exogenous inputs (i.e. time labels). Thus, we adopted only our hybrid LSTM and 1D-CNN, that are the most promising networks in terms of prediction accuracy, for this multivariate scenario. We used the same set of parameters of finding into the univariate scenario (see Section 4.6.4).

Consequently, as deeply detailed in Section 4.4.4, we added a second variable describing the time period, both for the time slot and for the day of the week.



(a) 1D-CNN 300 min.

(b) 1D-CNN 975 min.

(c) LSTM 300 min.

(d) LSTM 975 min

(e) ESN 300 min.

(f) ESN 975 min.

Figure 4.32: Observed vs predicted value for Room 2A.

As shown in Figure 4.34 and in 4.35 for the 1D-CNN and the LSTM respectively, adding this second variable does not lead to an improvement of the predictions, but

Table 4.10: Whole Building - PMV/PPD evaluation on real data: percentage of predicted values within the ±0.5 thermal comfort area for all prediction horizons (k-step = 15 min.).

k	PPD	k	PPD	k	PPD	k	PPD	k	PPD
0	91.2	30	68.3	60	79.1	90	80.7	120	57.4
1	90.8	31	80.9	61	78.3	91	70.4	121	46.2
2	91.5	32	74.4	62	71.8	92	79.5	122	54.7
3	89.9	33	78.8	63	82.4	93	75.6	123	51.2
4	88.9	34	78.3	64	79.8	94	81.1	124	59.2
5	87.4	35	66.5	65	78.0	95	77.8	125	51.5
6	84.6	36	76.4	66	73.3	96	74.7	126	39.6
7	81.9	37	75.8	67	75.3	97	66.4	127	51.2
8	83.2	38	77.4	68	82.4	98	72.5	128	47.8
9	79.5	39	75.0	69	79.7	99	63.7	129	39.1
10	80.3	40	77.3	70	83.6	100	66.6	130	44.7
11	77.6	41	79.2	71	78.5	101	71.1	131	33.7
12	78.6	42	70.0	72	80.6	102	65.4	132	43.7
13	78.0	43	66.2	73	80.8	103	67.2	133	43.4
14	75.4	44	77.1	74	77.3	104	63.9	134	41.1
15	76.0	45	77.3	75	77.1	105	60.4	135	46.3
16	68.8	46	77.3	76	79.5	106	56.5	136	41.0
17	68.8	47	71.4	77	77.7	107	64.6	137	43.7
18	68.9	48	73.0	78	82.4	108	58.5	138	46.9
19	67.2	49	68.3	79	74.8	109	58.4	139	45.0
20	75.5	50	74.7	80	82.9	110	55.4	140	45.9
21	67.4	51	80.5	81	84.4	111	56.6	141	36.0
22	74.1	52	78.3	82	79.5	112	60.0	142	59.2
23	74.7	53	73.3	83	84.4	113	52.2	143	57.4
24	74.2	54	80.2	84	82.0	114	52.0	144	67.1
25	71.4	55	79.2	85	80.3	115	56.6	145	57.6
26	70.6	56	74.4	86	78.2	116	47.9	146	54.5
27	78.6	57	75.2	87	76.5	117	48.7	147	47.8
28	79.2	58	79.2	88	76.2	118	47.4	148	57.1
29	75.6	59	70.9	89	77.1	119	56.2	149	56.5

even it makes them worse.

By analyzing the results, we can state that adding further complexity to our hybrid model the performance gets worse. We believe that the reason is to be found in the fact that the hybrid models are already able to extrapolate the information of the recurring patterns and seasonality. As a result, they ignore the temporal relationship added by the exogenous time-labels.

Table 4.11: Room 1D - PMV/PPD evaluation on real data: percentage of predicted values within the ±0.5 thermal comfort area for all prediction horizon (k-step = 15 min.).

k	PPD	k	PPD	k	PPD	k	PPD	k	PPD
0	93.7	30	76.7	60	82.9	90	81.1	120	80.5
1	93.5	31	80.2	61	83.0	91	81.8	121	91.4
2	92.8	32	79.7	62	78.0	92	80.2	122	84.8
3	92.5	33	82.3	63	80.5	93	80.8	123	89.5
4	92.2	34	82.0	64	82.3	94	81.2	124	92.7
5	92.2	35	79.9	65	81.7	95	82.4	125	90.2
6	89.5	36	86.5	66	83.3	96	82.3	126	85.3
7	86.9	37	82.4	67	81.7	97	82.7	127	79.8
8	90.2	38	82.2	68	82.0	98	87.7	128	87.0
9	82.7	39	80.1	69	81.4	99	84.8	129	84.4
10	78.2	40	80.7	70	81.4	100	90.9	130	82.2
11	75.6	41	81.1	71	82.7	101	90.3	131	84.2
12	74.0	42	82.7	72	82.6	102	93.5	132	91.4
13	74.6	43	84.9	73	81.7	103	89.8	133	86.9
14	71.7	44	84.4	74	83.5	104	80.2	134	85.6
15	75.9	45	82.8	75	82.1	105	84.8	135	87.2
16	70.4	46	85.4	76	81.5	106	88.9	136	89.6
17	68.5	47	82.9	77	82.7	107	76.5	137	84.9
18	78.6	48	80.3	78	81.1	108	67.9	138	93.2
19	77.3	49	82.1	79	80.8	109	88.0	139	82.6
20	72.0	50	83.4	80	82.1	110	83.2	140	90.3
21	67.2	51	83.2	81	80.5	111	82.4	141	89.5
22	68.5	52	84.4	82	76.4	112	78.9	142	86.1
23	75.5	53	82.0	83	76.5	113	82.1	143	89.2
24	75.8	54	82.8	84	79.2	114	80.3	144	88.2
25	72.9	55	81.3	85	82.4	115	84.8	145	84.3
26	73.7	56	80.8	86	79.1	116	85.8	146	78.3
27	60.3	57	81.5	87	80.3	117	83.8	147	82.8
28	74.5	58	82.6	88	79.7	118	85.3	148	74.6
29	73.6	59	78.8	89	78.9	119	90.2	149	82.9

4.6.5 1D-CNN fine-tuning

As stated in Section 4.6.4, our hybrid 1D-CNN results the most promising solution that performs better both in analytical and qualitative terms. Consequently, we decided to exploit this model for the subsequent fine-tuning experiments. The goal is to specialize our hybrid model on real-world data coming from the IoT devices installed in the demonstrator. In this way, the prediction model will be

Table 4.12: Room 2A - PMV/PPD evaluation on real data: percentage of predicted values within the ±0.5 thermal comfort area for all prediction horizon (k-step = 15 min.).

k	PPD	k	PPD	k	PPD	k	PPD	k	PPD
0	85.8	30	73.2	60	81.8	90	81.8	120	80.3
1	83.8	31	85.9	61	79.2	91	78.3	121	84.6
2	86.8	32	82.0	62	77.1	92	84.2	122	87.4
3	85.4	33	76.7	63	75.8	93	82.3	123	87.6
4	83.7	34	84.2	64	76.1	94	81.4	124	82.5
5	83.3	35	79.5	65	82.4	95	77.2	125	85.6
6	75.9	36	82.7	66	78.9	96	81.6	126	89.0
7	81.9	37	79.4	67	76.4	97	79.0	127	86.1
8	81.1	38	83.0	68	77.9	98	79.7	128	91.9
9	79.0	39	74.8	69	75.0	99	82.8	129	75.3
10	76.3	40	80.6	70	81.2	100	84.1	130	84.5
11	78.1	41	80.3	71	83.2	101	86.0	131	83.3
12	76.9	42	78.8	72	76.4	102	81.0	132	89.6
13	75.0	43	69.7	73	78.5	103	84.4	133	72.6
14	67.4	44	77.6	74	79.7	104	78.1	134	86.8
15	75.0	45	79.2	75	83.3	105	79.7	135	84.5
16	69.6	46	80.3	76	83.3	106	79.0	136	74.2
17	70.3	47	76.4	77	77.7	107	82.4	137	68.1
18	71.3	48	80.2	78	79.2	108	79.6	138	83.7
19	67.0	49	78.5	79	73.8	109	76.7	139	73.0
20	75.5	50	83.0	80	80.6	110	78.4	140	74.2
21	80.2	51	79.1	81	80.2	111	72.4	141	72.1
22	83.1	52	78.3	82	82.7	112	81.9	142	76.9
23	76.8	53	76.5	83	83.2	113	81.3	143	72.2
24	81.4	54	80.2	84	76.1	114	82.4	144	74.3
25	77.9	55	77.7	85	82.0	115	81.5	145	65.4
26	76.2	56	78.0	86	78.2	116	77.3	146	73.8
27	76.8	57	79.2	87	81.7	117	86.1	147	65.2
28	79.4	58	74.2	88	80.9	118	80.2	148	84.7
29	76.7	59	76.2	89	79.7	119	87.0	149	64.4

increasingly accurate.

Therefore, we apply the Transfer Learning methodology described in Section 4.4.5. Figure 4.36 shows the forecasting performances of our hybrid model in terms of prediction accuracy, exploiting the analytical MAE index, for all the environments. Our hybrid model is on average able to predict more than one day of indoor air-temperature values (i.e. about 24 hours). Room 2A represents an exception: the

Table 4.13: Corridor - PMV/PPD evaluation on real data: percentage of predicted values within the ±0.5 thermal comfort area for all prediction horizon (k-step = 15 min.).

k	PPD	k	PPD	k	PPD	k	PPD	k	PPD
0	94.3	30	79.2	60	75.3	90	76	120	65.7
1	93.9	31	74.4	61	70.2	91	75.5	121	73.9
2	93.9	32	77.3	62	75.3	92	74.4	122	64.9
3	93.1	33	78.2	63	76.8	93	73.9	123	66.3
4	92.1	34	78.3	64	71.8	94	75.9	124	77
5	92.4	35	80.5	65	75.6	95	76.9	125	70
6	91.9	36	79.1	66	76.4	96	78.7	126	70.3
7	90.9	37	82.4	67	74.2	97	75.5	127	62.5
8	90.6	38	73.8	68	73.8	98	76.8	128	59
9	88.8	39	79.7	69	73	99	77.5	129	64.1
10	88.5	40	78.2	70	75.5	100	74.4	130	72.4
11	85.8	41	77.3	71	73.8	101	73.1	131	69.2
12	86.8	42	78.5	72	74.2	102	78.5	132	65.9
13	86.2	43	78	73	74.8	103	75.4	133	66.2
14	86.5	44	75.5	74	71.8	104	78.1	134	68.4
15	86.4	45	76.8	75	72.9	105	71.1	135	64.3
16	87.8	46	80.2	76	74.4	106	73.9	136	65.6
17	86.2	47	80.6	77	72.6	107	66.2	137	60.4
18	84.4	48	73.3	78	74.7	108	78.2	138	62.4
19	86.7	49	76.4	79	74.4	109	70.5	139	62.8
20	84.2	50	77.3	80	74.8	110	74.2	140	63.6
21	79.5	51	80.3	81	75.9	111	77.1	141	56.3
22	83.4	52	78.8	82	75.8	112	85.6	142	62
23	81.3	53	77.7	83	74.1	113	74.4	143	66.2
24	78.1	54	76.7	84	77.1	114	70.6	144	58.2
25	83.7	55	76.2	85	74.2	115	69.7	145	63.2
26	82.9	56	73.9	86	71.8	116	72.4	146	71
27	79.1	57	76.7	87	73.8	117	75.9	147	65.9
28	80.8	58	73.2	88	76.8	118	73.5	148	67.3
29	77.3	59	77.6	89	74.4	119	71.3	149	73.7

model can predict further until 35 hours.
Thus, according to Section 4.4.5, we applied three different transfer learning techniques. We split the real data-set into two parts, 80% is used to retrain the network and the other 20% for testing purpose. Figure 4.37 compares the previous 1D-CNN hybrid model (i.e. called *pre-trained* model) and the three 1D-CNN model with the Transfer Learning techniques (i.e. called *tuned* models).

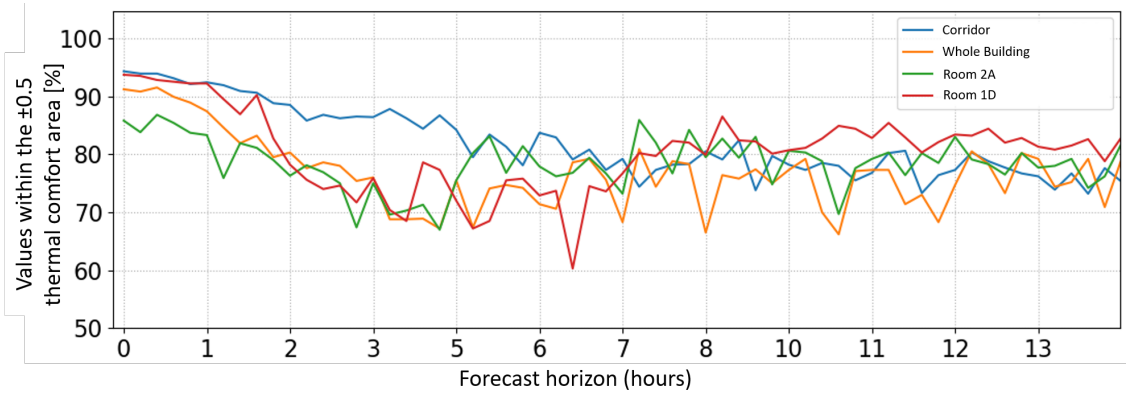
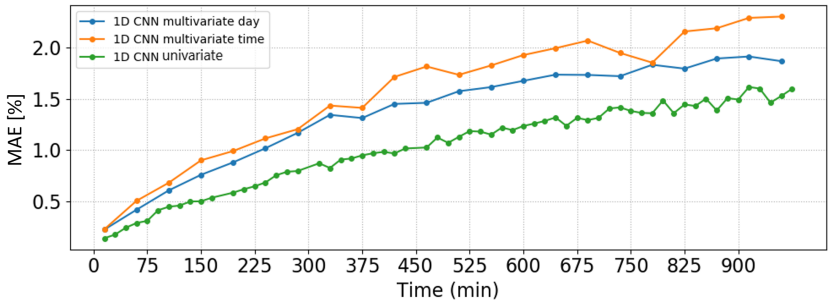


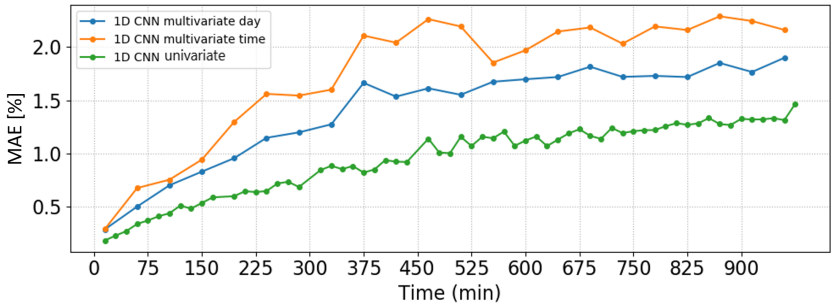
Figure 4.33: 1D-CNN - Values within the ±0.5 thermal comfort area over time.

We can state that, for all the environments, at least one Transfer Learning technique can improve the prediction performance. Furthermore, we notice a different behaviour according to the prediction horizon. Indeed, for shorter times retraining all the network layers is always the best choice, but for longer ones, retraining only the last layer outperforms the other techniques (i.e. Figure 4.37a, Figure 4.37c and Figure 4.37d) with improvements up to 1 °C. Moreover, retraining only the first or the last layers, always led to improve, even though sometimes negligible. In all the cases, the effect of Transfer Learning becomes more relevant as the prediction horizon increase.

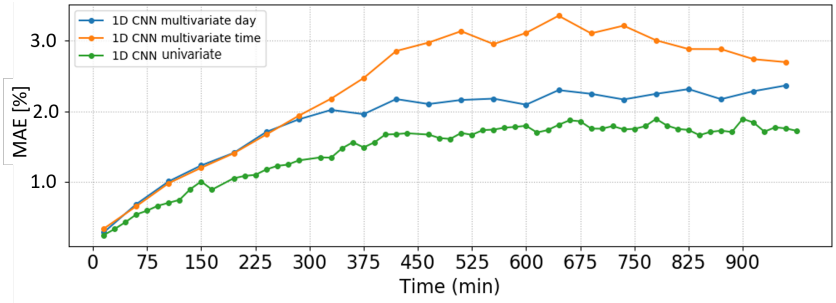
Lastly, we performed the Fanger analysis to obtain a qualitative validation. Therefore, we applied the analysis to our best techniques (i.e. last layer fine-tuned) which is the one with better accuracy for long horizons. Figure 4.38b shows consistent improvements for the whole building and rooms. In detail, we can move the maximum acceptable average prediction limit from 24 hours to 28 hours. Furthermore, in short- and medium-time (i.e. until 13 hours ahead), we can achieve a more comfortable prediction. Indeed, for all the environments, the thermal satisfaction is always above 80%, as depicted in Figure 4.39.



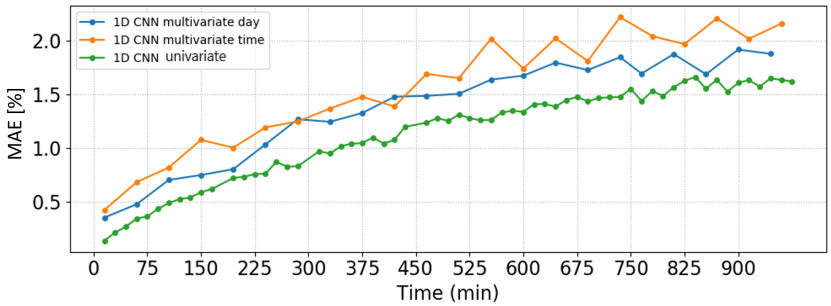
(a) Whole Building



(b) Room 2A

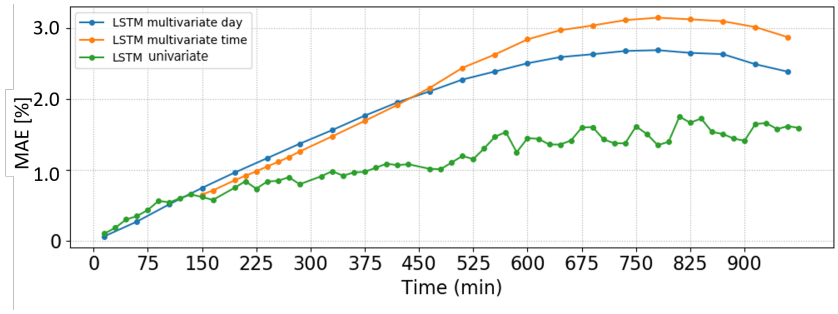


(c) Room 1D

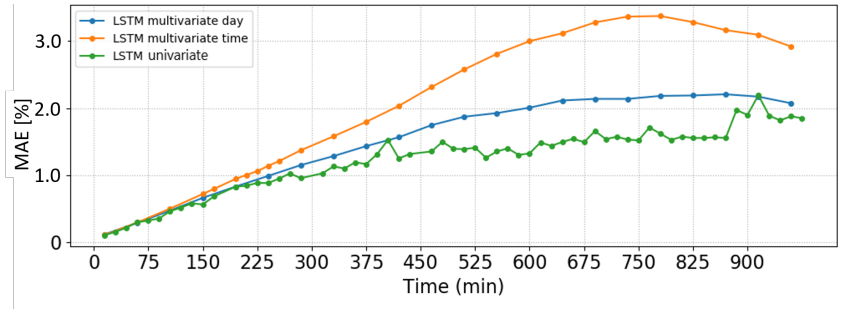


(d) Corridor

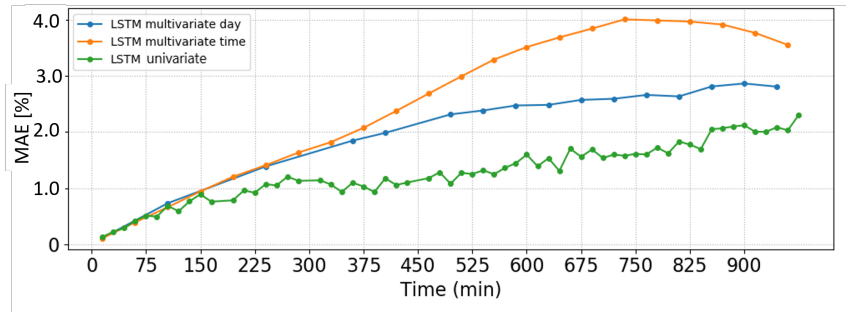
Figure 4.34: 1D CNN - Multivariate MAE comparison.



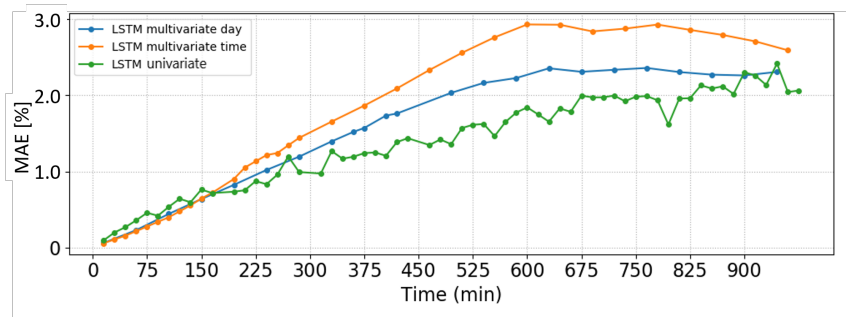
(a) Whole Building



(b) Room 2A

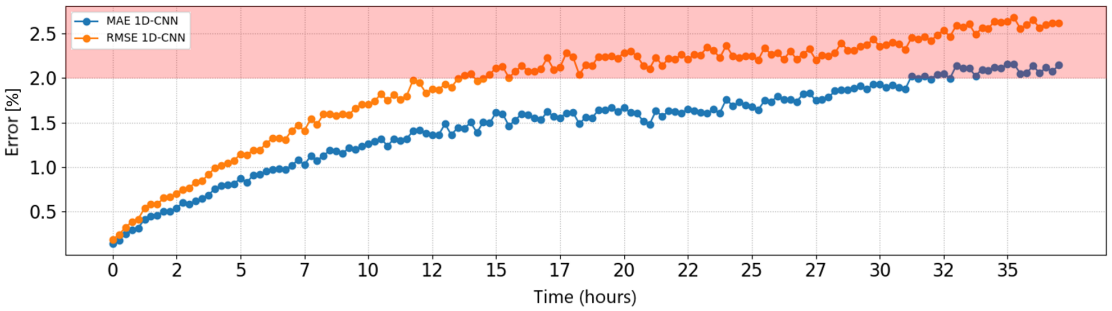


(c) Room 1D

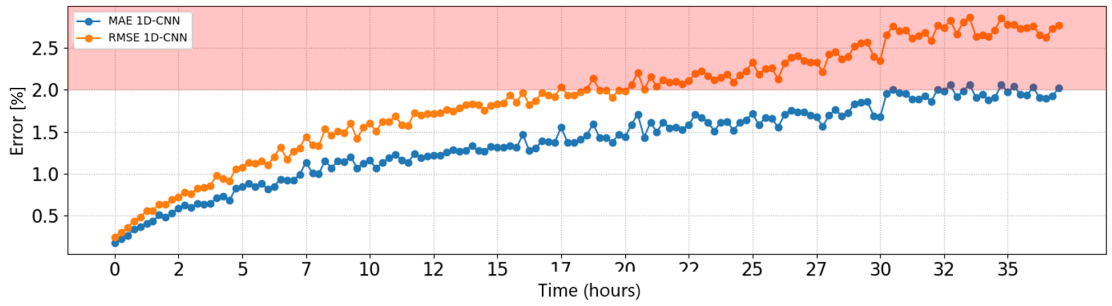


(d) Corridor

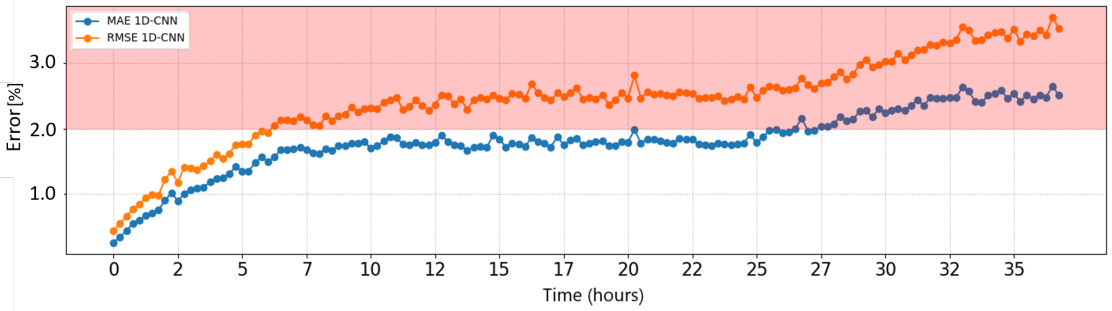
Figure 4.35: LSTM - Multivariate MAE comparison.



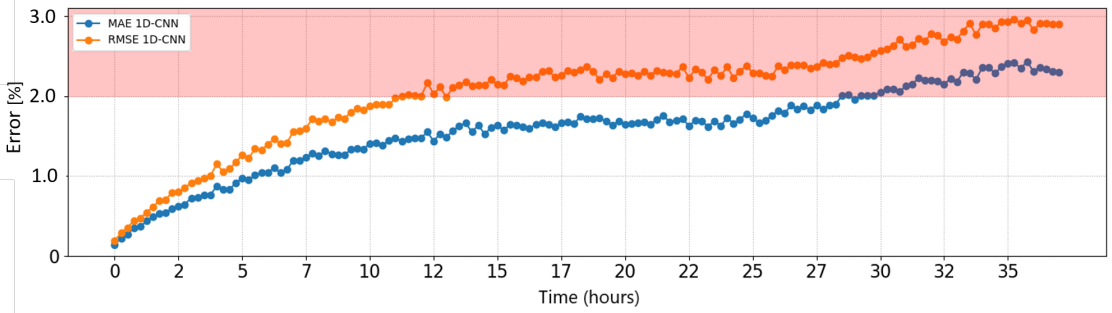
(a) Whole Building



(b) Room 2A

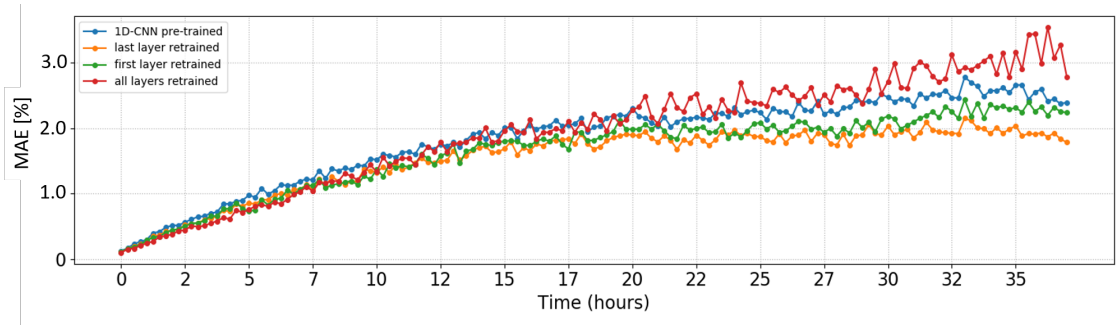


(c) Room 1D

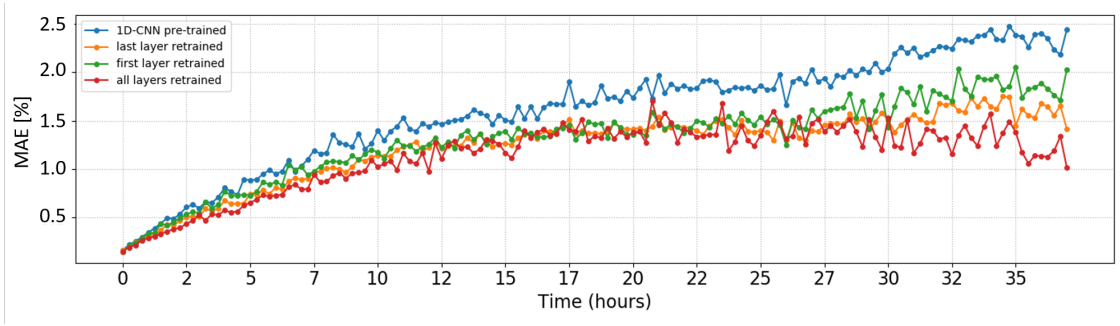


(d) Corridor

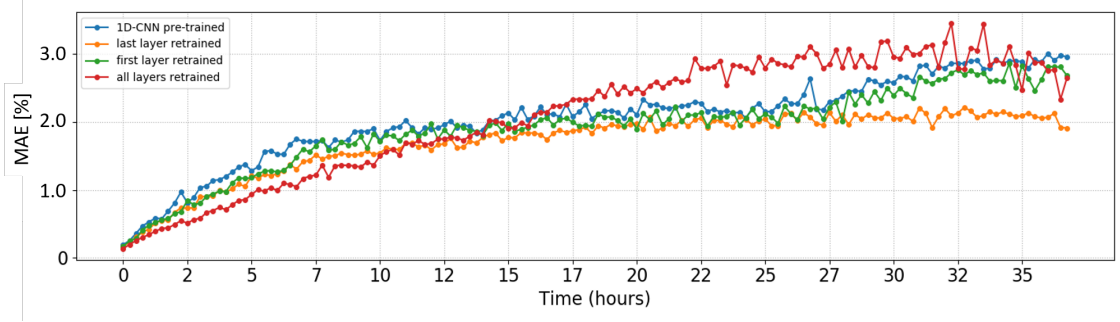
Figure 4.36: 1D-CNN - Forecast performances over time.



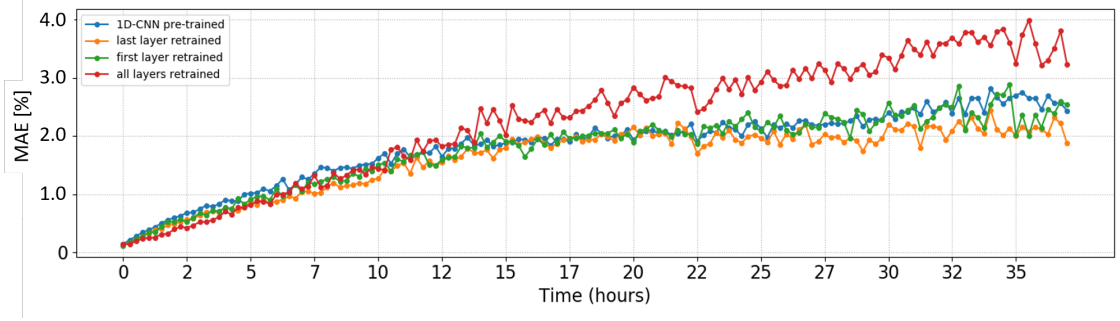
(a) Whole Building



(b) Room 2A

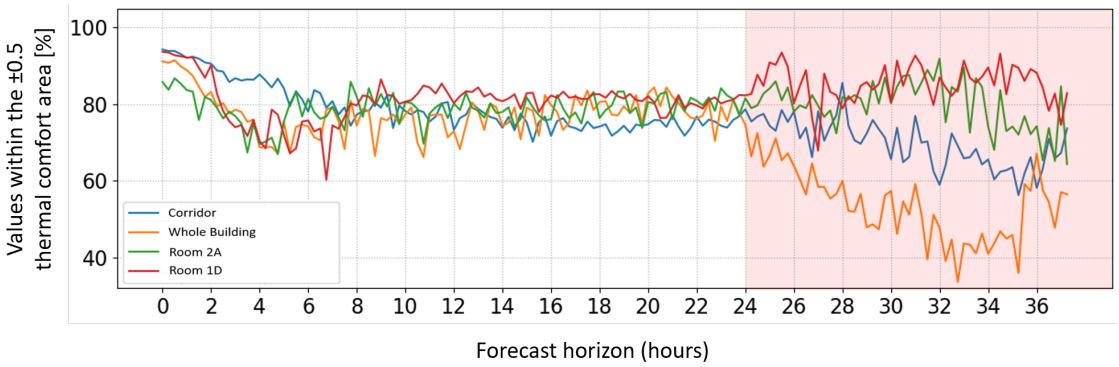


(c) Room 1D

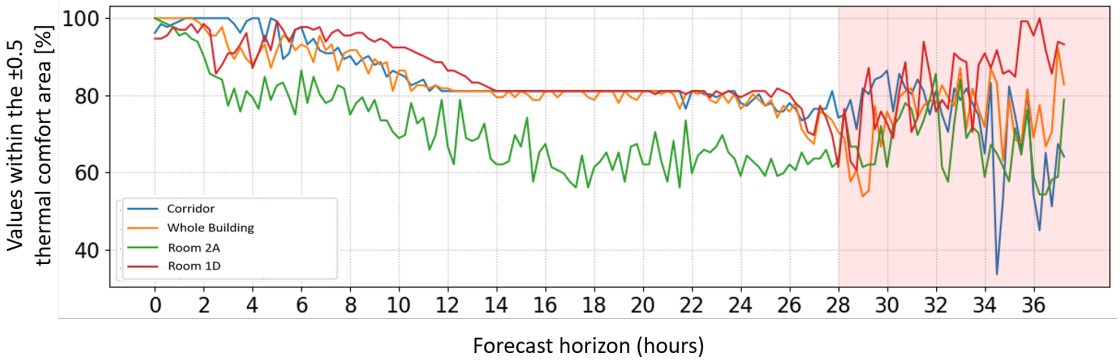


(d) Corridor

Figure 4.37: 1D-CNN - Comparison of Transfer Learning techniques.

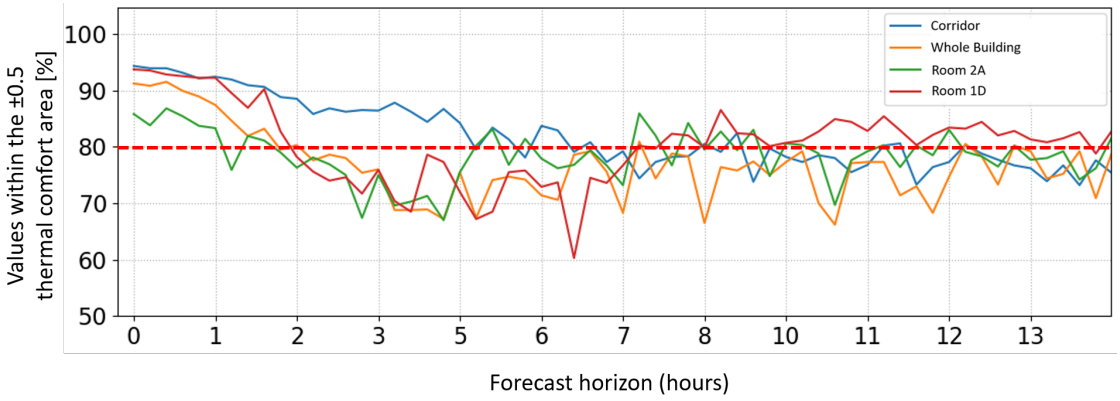


(a) Pre-trained hybrid model

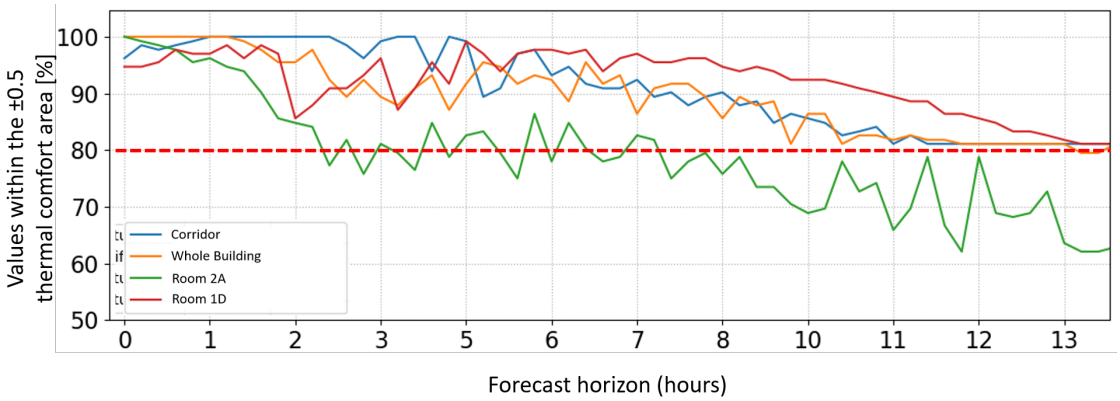


(b) Fine-tuned model (last layer re-trained)

Figure 4.38: 1D-CNN - PMV/PPD evaluation graph over time.



(a) Pre-trained hybrid model



(b) Fine-tuned model (last layer retrained)

Figure 4.39: 1D-CNN - PMV/PPD evaluation graph in short- and medium-term forecasting.

4.7 Discussion and Remarks

In this Chapter, we proposed a methodology to forecast indoor air-temperature of existing or newly constructed Smart Buildings (i.e. buildings without a consistent set of historical data). Indeed, the aim of our methodology is trying to compensate for the lack of real-world data in the context of energy simulations for the energy-efficient management of Smart Buildings. As a matter of fact, buildings are very rarely equipped with suitable temperature sensors and, even in the case the sensors are available, the amount and significance of historical data might not be enough to train a prediction model. Moreover, our solution provides building-, facility- and energy-managers with tools for: i) (near-) real-time visualization of energy consumption information; ii) simulations of temperature trends and energy consumption. Through simulations, the managers can also check and evaluate the efficiency performance of the building, energy behaviours of users and possible refurbishment actions.

In our methodology, BIM and meteorological data are exploited to construct of a realistic and consistent dataset of indoor air-temperature values. The presented experimental results showed that our solution simulates with a good accuracy the heating performance of the case-study building. With respect to literature solutions that consider TMY weather data, our results highlighted that the integration of real weather information into the simulation process strongly increases the accuracy of the simulation itself. Thus, exploiting realistic synthetic data to train prediction models, we are able to implement hybrid model that use real-world data in the inference phase. To validate this, we have specially designed some models (i.e one for each environment of our demonstrator) based on a NAR architecture with a high number of regressors by discussing the prediction accuracy by analyzing the inference results both on synthetic and real data. As demonstrated by our case study, our models provide accurate predictions with time horizons in the order of 3 h for individual rooms and 4 h for the entire building.

At the same time, we tried to add additional information as input to our hybrid models (i.e. time labels). By analyzing the experimental results, we can state that adding further complexity the performance gets worse. The reason is to be found in the fact that the hybrid models are already able to extrapolate the information of the recurring patterns and seasonality. As a result, they ignore the temporal relationship added by the exogenous time-labels.

We then applied Transfer Learning techniques in order to specialize the hybrid models with real data provided by IoT sensors installed in the demonstrator. For this purpose, we have specially designed, implemented and compared some state-of-art neural networks for time-series forecasting. In detail, these are: i) NAR (i.e. the benchmark model), ii) FNN, iii) LSTM and iv) 1D-CNN. For all these models, we initially used the synthetic data for the training phase and the real-world data for the inference phase. Then, we compared all the obtained models in analytical and

qualitative way. The experimental results showed that the most promising hybrid model for this scenario was 1D-CCN. As a result, we have applied 3 fine-tuning techniques for the specialization on real data. We find that by re-training the last layer (i.d. Freezing Layer techniques) we are able to predict until 28 hours head without introducing thermal discomfort for the occupants. However, the real-world dataset available is very limited. Looking at the trend graphs comparing Transfer Learning techniques, we are convinced that by using a more consistent dataset we could achieve further improvements.

Summarizing, the predictions provided by our models can be exploited for the design of control policies for the energy-efficient management of Smart Buildings (e.g., Demand Response, Demand Side Management and peak-shaving, which are all based on thermal behaviours forecasting), especially for those scenarios where real sensors data are unavailable or insufficient.

Chapter 5

Blood glucose forecasting

Continuous Glucose Monitoring Systems (CGMSs) allow measuring the blood glycaemic value of a diabetic patient at a high sampling rate, producing considerable amount of data. This data can be effectively used by machine learning techniques to infer future values of the glycaemic concentration, allowing the early prevention of dangerous hyperglycaemic or hypoglycaemic states and a better optimisation of the diabetic treatment. Most of the approaches in literature learn a prediction model from the past samples of the same patient, which needs extensive calibrations and limits the usability of the system. This Chapter investigates prediction models trained on glucose signals of a large and heterogeneous cohort of patients and then applied to infer future glucose level values on a completely new patient. Next, we study how these neural networks can specialize on the individual patient.

5.1 Introduction

The human body, through the appropriate organs, breaks down carbohydrates into glucose. Insulin, a peptide hormone produced by beta cells of the pancreatic islets, plays a key role in this process, regulating the way the cells absorb glucose and use it as an energy source [222]. Defects in either insulin secretion or insulin sensitivity (or both) lead to diabetes, that is a chronic disease characterised by high glucose levels in the blood (i.e. hyperglycaemia) [18]. This is a severe condition, that is known to at least double a person's risk of early death.

Diabetes can be categorised into three main categories. Type I diabetes, that affects about 10% of the diabetic patients, usually develops since childhood due to a loss of beta cells, that are mistakenly attacked and killed by the immune system [163, 19], resulting in a loss of insulin. Type II diabetes occurs when the body does not make proper use of the released insulin (i.e. insulin insensitivity) or does not produce enough insulin [49]. This is the most common form of diabetes (about 90% of the diabetic population [163]) and usually develops in adulthood.

Gestational diabetes, a temporary condition occurring only during pregnancy, affects between 3 and 20% of mothers-to-be, with increased risks of developing future chronic diabetes for both mother and child [45].

Diabetes in all its forms is among the most widely diffused chronic disorders worldwide, with ever-increasing diffusion trends in both women and men. Hence, there are growing efforts directed towards the development of therapies as well as of better ways of keeping the effects of this disease under control. In this work, we focus especially on Type I diabetes. Nonetheless, our study can be easily generalised to the other forms of pathology. In a normal subject, glycaemia typically oscillates between $70 - 100$ mg/dl in a fasting state and does not exceed 140 mg/dl after a meal. Conversely, diabetic subjects have fasting values higher than 126 mg/dl, and may encounter either *hyperglycaemia* (when the glycaemic value in the blood exceeds $180-200$ mg/dl) or *hypoglycaemia* (when the glycaemic value is much lower than 70 mg/dl), that are both life-endangering situations. Such events need to be timely detected in order to take all possible countermeasures to save the patient's life.

Traditionally, diabetic patients are monitored by measuring their glycaemic value multiple times in a day (four or more, for a Type I diabetes) using a fingerstick blood glucose meter, often accompanied by fixed insulin infusions. This type of monitoring has a significant disadvantage, in that it cannot detect fluctuations of the glycaemia that may be caused by intense physical activity, sudden emotional stress or food assumption. As a result, insulin injections are often over- or under-dosage concerning the actual need [46].

Today, thanks to the widespread use of increasingly intelligent and low-cost technological devices, the medical sector is moving more and more towards the concept of *smart healthcare* [89]. In this scenario, diabetic patients, especially Type I, are subjected to constant monitoring and appropriate and timely insulin treatment, employing Continuous Glucose Monitoring Systems (CGMS) [88]. Using sensors applied to the skin, CGMS can measure the glycaemic value of a subject at a rate of up to one sample per minute. This generates a considerable amount of data that can be either stored or sent to a processing system, and used to infer the future values of glycaemic concentration within different prediction horizons, with a two-fold benefit: i) better prevention of potentially dangerous hyperglycaemic or hypoglycaemic states and ii) optimisation of the insulin dose that needs to be injected [205]. On top of that, the patient can be subjected to continuous remote monitoring by the primary care physicians, triggering automatic alert mechanisms and, whenever needed, faster hospitalisation procedures [140, 199].

Since the introduction of CGMS, literature has proposed several approaches for short-time glucose prediction, that can be broadly categorised into two main groups: i) approaches based on apriori physiological models, that try to reproduce the metabolic response of a patient using equations that mathematically describe glucose kinetics [62, 154]; ii) data-driven approaches [72, 169], that infer the future

values of glucose concentration by applying machine learning techniques trained on real glycaemic data (see [190] for a review of some recent methods). As they do not depend on fixed parameters, machine learning techniques promise higher flexibility and generalisation capability of a fixed physiological model, especially in the presence of unpredictable variability of glucose kinetics due to either internal (e.g. different device calibrations) or external factors (e.g. physical activities, food intake, sudden stress, etc.). Hence, thanks to the higher availability of data, the data-driven approaches are generally preferred in the last few years.

5.2 Related Works

The idea of predicting the future trend of glucose level by using its past values as the only input was first exploited in [37] and then refined in most recent years, taking advantage of more robust and accurate data recording devices and more sophisticated machine learning models. The most common approaches provided by literature are either based on time-series autoregressive models (AR) or artificial neural networks (ANN) [190]. In [231], a first-order AR model is proposed and compared with a first-order polynomial model. In the same years, a method based on Kalman filtering is used to predict hypoglycaemic events [192] based on glucose monitoring signal. Nonetheless, these methods have still significant prediction errors and a minimal forecasting window (maximum 45 and 30 min, respectively). In [194, 195], ANN approaches were used to predict glycaemia for up to 3 hours. Nonetheless, the accuracy of the prediction considerably decreases when the prediction horizon increases. Better accuracy values, albeit on different test sets, were shown by the most recent works, either based on ANN or support vector regression techniques [94, 273, 5, 102]. A common trait of these works is that they are usually calibrated on individual patients (i.e. the prediction for a specific patient is built on top of the past glucose level signal recorded from the same patient). While this approach has the obvious advantage of creating a personalised model that perfectly fits the characteristics of a specific patient and a recording device, it also has multiple disadvantages: i) it limits the usability of the device, in that the system cannot be used on a patient until the calibration is fully performed, ii) it limits the generalisation capabilities of the system and increases the risks of overfitting. Conversely, learning a model from a heterogeneous set of patients and recording devices increases in principle the robustness of the model to unpredictable and unseen changes of the input signal [109]. On top of that, the device can be used on a new patient immediately, without re-training.

Fewer studies in literature explored the idea of creating a generalisable glucose level prediction model from a multi-patient training cohort. In [90, 91] the authors propose an AR model with fixed coefficients (applying data filtering and Tikhonov

regularisation [30]) and compare three different configurations: respectively i) models trained on each individual subject, ii) a model trained on different subjects using the same CGMS device, and iii) a model trained on different subjects using different devices. Their experimental results show comparable prediction errors for the three scenarios on a forecasting horizon of 30 min. Further developments of the same idea are presented by [202], this time using model based on feed-forward ANN, and by [10], using a recurrent neural network (RNN). Nonetheless, the forecasting accuracy obtained by these works is still modest, and the data used for the training is poor both in terms of number and type of patients (e.g. age categories), which intrinsically limits their generalisation capability.

While the most consolidated works are generally based on shallow neural networks, few recent studies started proposing deep learning techniques [169, 276, 146] (e.g. Convolutional Neural Networks). Nonetheless, these methods are typically very demanding, both in terms of training data and computational resources. Hence, the proposed predictors suffer limitations due to the lack of annotated CGM signals used for training the networks, both in terms of number and type of patients considered in the analysis, as well as the type of recording devices.

5.2.1 Contributions

In this work, we continue on the path of learning a model from a multi-patient dataset and using this model to predict the future glucose level values of a new patient. By doing so, we try to improve the previous works in two ways: i) by refining the formulation of the neural network used for the prediction, and ii) by considerably enlarging the dataset used for learning the model. The aim is to improve the prediction accuracy, possibly on a much larger forecasting horizon, and to increase the generalisation and robustness of the model.

As of point i), we designed and compared two different solutions, that were successfully applied to other time-series forecasting problems. The first solution exploits a Non-Linear Autoregressive Neural Network (NAR). This model extends and refines traditional linear AR architectures, in that it is not intrinsically limited by the assumption of linearity, and overcomes the stability problems of past formulations [29]. The second solution exploits a Long Short-Term Memory (LSTM) network, that is generally acknowledged as one of the best for time series prediction, thanks to its versatility and flexibility [98, 230]. This model can overcome the well-known problems of exploding and vanishing gradient that typically affect traditional RNN architectures and to maintain long-term information over time [196, 125]. While, to the best of our knowledge, NAR was never applied before to the problem of glucose profile prediction, few recent works exploited LSTM, eventually embedded into deep learning frameworks [276, 146, 239]. These works show promising results of LSTM model compared to other approaches, but they are still limited by a lack of training data.

As of point ii), we trained and tested our new solutions on an extensive set of CGMS signals, with unprecedented variability both in terms of the type of subjects and recording devices. This choice stems from the consideration that a higher variability of the training set is known to improve the generalisation capabilities of the prediction model and to reduce the risks of overfitting.

To evaluate the forecasting accuracy and robustness of our solutions, we assessed our results in comparison with other multi-patient techniques, exploiting traditional AR, feed-forward ANN and RNN formulations, respectively. For a fair comparison, we re-implemented the architectures proposed by previous works [91, 202, 10] and performed experiments on the same dataset as our proposed solutions, both for training and testing purposes.

Progressively, as second analysis, we focus on patient-specialized blood glucose prediction system exploiting a data-driven approach. Thus, we designed and implemented our solution based on Long Short-Term Memory (LSTM). Consequently, we tested and evaluated our solution capabilities and performances to improve the prediction accuracy, possibly on a much larger forecasting time horizon. For a fair validation of our methodology, we exploited the very same dataset for training and testing the two most significant state-of-art neural networks in blood glucose prediction (i.e. [202] and [10]). Then, to evaluate the performance of the models, we exploited different quantitative and qualitative performance indexes to identify the most promising.

The solution proposed in this work is part of a broader framework introduced previously. That is the realization of a glucose prediction algorithm that, equipped in a modern CGM system, can be able to make direct predictions without a long period of data collection and training on the patient itself. With the first phase of our proposed methodology, we have demonstrated the possibility of designing an effective multi-patient data-driven blood glucose prediction model able to predict the future glucose level values of a new patient. This model allows the realization of a robust and pre-trained system. In this second phase of our methodology, we want to investigate how our LSTM solution works on the individual patient. Proved its potentialities on individual patients, as future works, we plan to apply some techniques, such as Transfer Learning [256], to personalize the multi-patients neural network according to glycaemia behaviours of each patient to monitor.

5.3 Case Study

In the following, this Section describes in details the datasets used to train and test our prediction system for multi-patient and single-patient specialized data-driven methodologies.

5.3.1 Multi-patients dataset

As anticipated in Section 5.1, our first aim is a generalisable CGMS, that learns a prediction model from a fixed set of subjects and then can predict glucose level values on a new patient without needing any re-calibration. To do so, the system needs to be trained on a broad set of CMGS signals, possibly representing a very wide range of possible outcomes. This inherently reduces the risks of learning a model that is too simple or unfit to deal with unseen data. More specifically, the training set should represent wide variations of glucose dynamics and reflect differences due to either different subject categories (e.g. adults and children, female and male, etc.) or different recording systems. On top of that, to ensure a good representation of glucose dynamics, the training samples should be acquired continuously for a sufficiently long monitoring period, with a rate that is high enough to represent glycaemic fluctuations. Supported by past literature, we identified 5 min as the minimum sampling rate for the CGMS (lower frequency is acceptable only during the sleeping time when the signal is less subject to fluctuations), two days as the minimum monitoring time per subject and 30 samples as the minimum length of a monitoring sequence for it to be considered significant [203, 218].

Based on the considerations above, we decided to use the RT_CGM dataset [111], that is freely available for research purposes, in anonymised form in order to protect patients' privacy. This dataset includes glycaemia trends of a heterogeneous population of 451 patients affected by Type I diabetes, already randomised. Patients have different ethnic origins and gender (45% male and 55% female, respectively), and belong to three different age categories (respectively, adults > 25, adolescents and young adults $15 - 24$ and children $8 - 14$). The data consist of glucose level samples acquired every 5 min using three different CGMS devices (provided by Abbott Diabetes [68], DexCom [119] and Medtronic [204], respectively). On top of that, patients take insulin in two different ways, either by injections or using a micropump delivery system.

The original dataset was pre-processed to make it consistent for our analysis. More specifically, we removed sequences with too many gaps as well as sequences with less than 30 consecutive samples, due to device calibration or measurement errors. Then, we randomly split the resulting data into two subsets for training and testing purposes, containing about 70% and 30% of the initial samples, respectively. The full characterisation of these two sets is reported in Table 5.1. The training and test sets are completely independent in terms of patients (i.e. data from a specific patient can be either in training or on the test set).

The training set was used to train the prediction models and optimise their parameters, using a portion of the samples as validation set, whilst the test set was solely used for assessing the final prediction performance.

Table 5.1: Dataset characterisation.

	Age	# of patients	# of samples	# of hyperglyc. samples	# of hypoglyc. samples
Training set	>25	95	304450	142410	24827
	15-24	97			
	8-14	124			
Test set	>25	70	94128	34311	8253
	15-24	46			
	8-14	19			

5.3.2 Single-patient dataset

A second aim is to create patient-specialized predictive models. In this view, to evaluate performance and accuracy, all the patient-specialized models are trained and validated using the same dataset: OhioT1DM [156]. This dataset consists of blood glucose level values sampled every 5 minutes, for about two months of observation. Data refer to six Type-I diabetic patients (i.e. two men and four women), with age ranging between 40 and 60 years. The sensor used to sample the blood glucose level is the Medtronic Enlite CGM, combined with a Medtronic 530G insulin pump. The dataset is divided into two parts: around 80% of the data for the training phase while the remaining 20% for the test and validation phase.

5.4 Multi-patients data-driven methodology

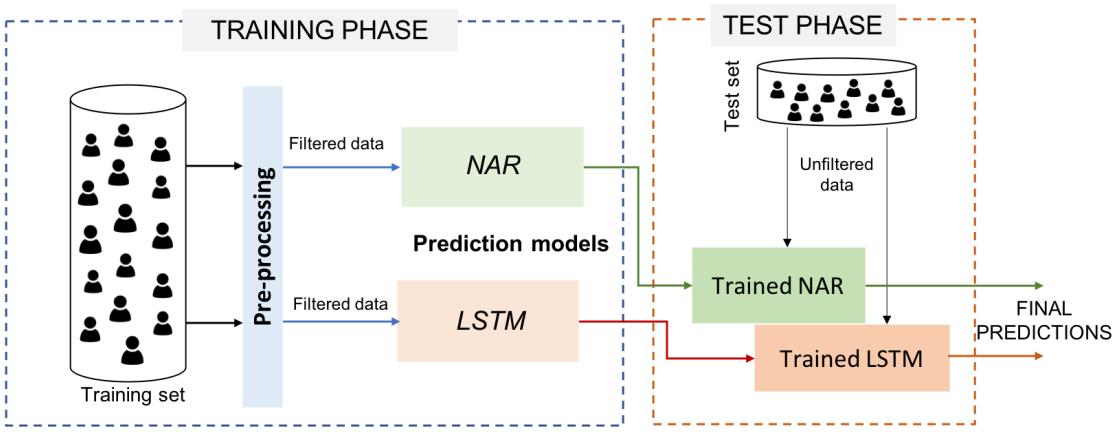


Figure 5.1: Main phases of the study.

Figure 5.1 shows the main phases of the study. In the so-called training phase, training samples first undergo a pre-processing filtering step. The filtered data

are used as input to build a prediction model, either based on NAR or LSTM neural networks. In the test phase, new unseen and unfiltered data are fed into the trained models to obtain the final predictions. All the implementations were done in Matlab (NNSYSID toolbox) and Python, using TensorFlow as back-end and exploiting Scikit-learn, pyrenn and pandas libraries.

5.4.1 Pre-processing

Generally, CGMS sensors introduce some amount of noise during signal sampling [218], [263], that needs to be attenuated or removed before the data can be used either to train a prediction system or to infer new values from unseen data [28].

As it is well known from the literature, over-smoothing the glucose time series data translates into a higher risk of missing out hypoglycaemia and hyperglycaemia events. On the other hand, processing the data completely unfiltered might lead to false alarms caused by a few unexpected sensing errors [28]. Hence, data filtering should be performed, having as objective a reasonable compromise between these two opposite effects.

In according to Section 3.4.2, as denoising pre-processing step, we applied to the training data Tikhonov regularisation [122], that is widely used in time series analysis and glucose level prediction system. As demonstrated by [90, 91], this method allows obtaining a filtered version of the signal without introducing significant delays into the time series.

In Figure 5.2, we show an example of the effect of Tikhonov regularisation applied to our CGMS training data. As it can be easily gathered from the plot, the filter attenuates sudden spikes in the signal, without altering the trend and without introducing delays, as already demonstrated by [90, 91].

5.4.2 Prediction model building

In the following, we present our glucose prediction models based on NAR and LSTM neural networks, respectively. More specifically, for both the solutions we describe i) how we selected and identified the final architecture and related parameters of the prediction models and ii) how we optimised such parameters in order to boost the performance and robustness of the models, preventing the risk of over-fitting.

Non-Linear Autoregressive Neural Network

As anticipated in Chapter 2, NAR extends traditional linear autoregressive model [147], in that it is completely distribution-free. Hence, it can be applied even to time series with intrinsic non-linearities, such as sudden spikes and fleeting transient periods [185].

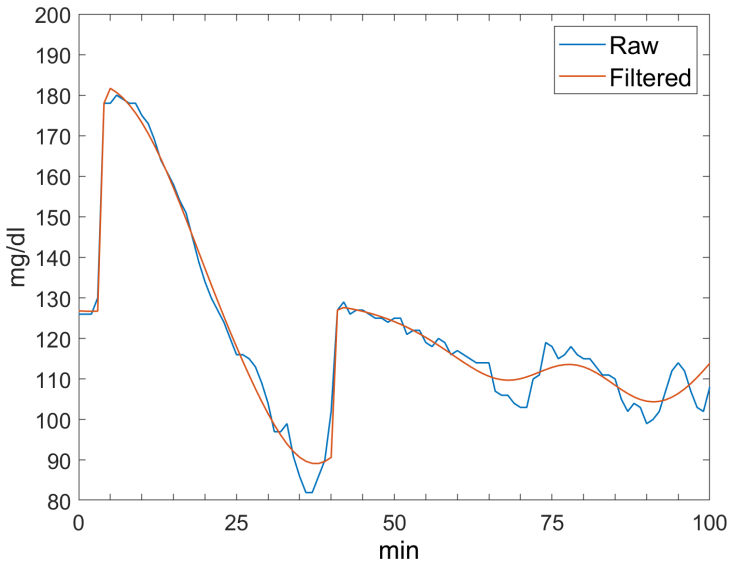


Figure 5.2: Effects of Tikhonov regularisation on glucose level data: example.

A NAR model computes the value of a signal y at time t using n past values of y as regressors (also called feedback delays), as follows:

$$y(t) = f(y(t-1), y(t-2), ..., y(t-n)) + e(t), \tag{5.1}$$

where f is an unknown non-linear function and $e(t)$ is the model approximation error at the time t.

Function $f(\cdot)$ is computed by optimising a multi-layered neural network, whose topology is depicted in Figure 5.3(a).

At the time t, the neural network is fed with the n past values of the signal y. Such inputs are transferred through multiple layers of neurons, where each neuron is a simple computational unit characterised by a set of weights W (one per each input connection j), a bias b and an activation function h. Hence, the output of a neuron i is computed as follows:

$$out_i = h_i \left(\sum_j w_{i,j} \cdot in_{i,j} + b_i \right) \tag{5.2}$$

where the optimal values of $w_{i,j}$ and b_i are computed by back-propagation on the training set [185].

As anticipated in Section 5.3, the minimum number of consecutive samples in our dataset is 30. Keeping this in mind, we initially designed a very simple fully-connected NAR as the one represented in Figure 5.3, with the following topology:

1. one input layer, with 30 units;

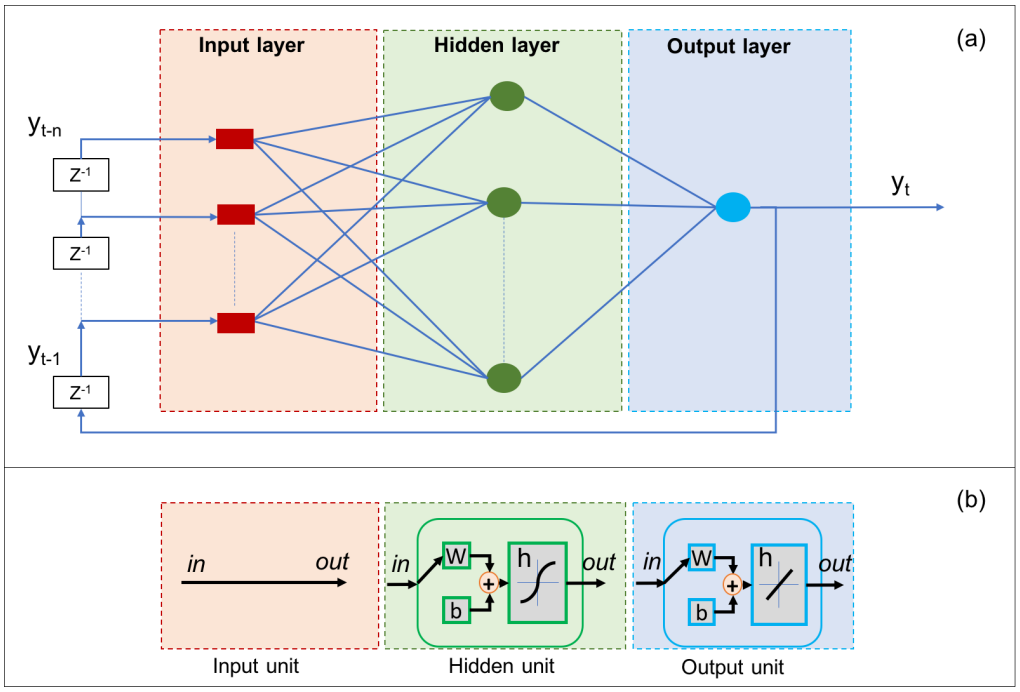


Figure 5.3: Non-linear autoregressive neural network. (a) Network topology. (b) Neuron models of the input, hidden and output layers, respectively.

2. one hidden layer, again with 30 units;

3. one output layer, with one unit.

As reported in Figure 5.3(b), we used hyperbolic tangent activation functions for the hidden units and a linear activation function for the output unit.

As a learning paradigm for the NAR network, we implemented a Levenberg-Marquardt backpropagation procedure (LMBP), which is widely applied to NAR models. This technique approximates second-order derivatives leveraging a *trust region* approach [185], with no need to compute the Hessian matrix. This helps reducing the training speed compared to traditional backpropagation techniques.

As it is widely known, models with too many parameters (i.e. too many neurons and connections) are less fit for hardware implementation and may easily lead to over-fitting. To overcome this problem, we adopted a two-steps design procedure, as follows:

1. we applied an automated optimisation strategy based on Lipschitz methodology [211] to possibly reduce the number of regressors of our model (i.e. the number of past glucose values to be given as input to the system).

2. we performed an automated pruning of the initial fully-connected structure, based on Optimal Brain Surgeon (OBS) method [105].

This two-steps procedure allowed us to design a more compact version of the network compared to the one of Figure 5.3(a), obtaining a non-redundant NAR model with optimal number of inputs and computational units.

As of point 1), we applied Lipschitz methodology for determining the optimum *lag-space*, that in our case is the number of delayed glucose signals to be used as regressors [211]. This method is successfully used for the analysis of Input-Output Models orders in Non-linear Dynamic Systems in many applications. As initially proposed by He and Asada [110], a reliable decision on the optimal order n of a non-linear model characterised by training input-output pairs $(\boldsymbol{x_i}, y_i)$ can be made based on so-called Lipschitz quotients:

$$q_{i,j}^{(n)} = \frac{|y_i - y_j|}{\|\boldsymbol{x_i} - \boldsymbol{x_j}\|}, \tag{5.3}$$

where in our case $\boldsymbol{x_i}$ is a vector of inputs and y_i is the corresponding output of the system, with $i \neq j$ and $i, j = 1..N$ where N is the number of samples in the training set. Hence, the superscript n stands here for the number of regressors of the system.

Lipschitz order index $L^{(n)}$ is defined as the geometric mean of the m largest Lipschitz quotients, as follows:

$$L^{(n)} = \left[\prod_{k=1}^{m} \sqrt{n} \cdot l^{(n)}(k)\right]^{\frac{1}{m}}, \tag{5.4}$$

where m is a positive number recommended to be $\sim 0.01N$ and $l^{(n)}(k)$ is the k-th largest Lipschitz quotient among all $q_{i,j}^{(n)}$.

Finally, as demonstrated by [110], the optimal number of regressors can be found by plotting Lipschitz order index at increasing values of n, in a forward sequential way, and selecting the knee-points of the obtained curve. As reported in Figure 5.4, by applying this procedure to our training data, we found $n = 8$ and $n = 17$ as the best candidates for the number of regressors of our system.

Based on Lipschitz results, we implemented three fully-connected NAR models, respectively with 30 (that is the minimum number of consecutive samples of our dataset, and hence the upper-bound of the input size), 17 and 8 regressors. These three models were trained on the training set described in Section 5.3, using LMBP as the learning algorithm. To assess the goodness of the training and the generalisation capabilities of the three models after this first design phase, in the second column of Table 5.2 we report the normalised sum of squared errors (nSSE) obtained after training the 30, 17 and 8 regressors models, respectively. These values were computed on a portion of the training set solely used for validation and optimisation purposes. (i.e. *validation set*). As it can be easily gathered from the table, the model with 8 regressors is the one obtaining the best performance.

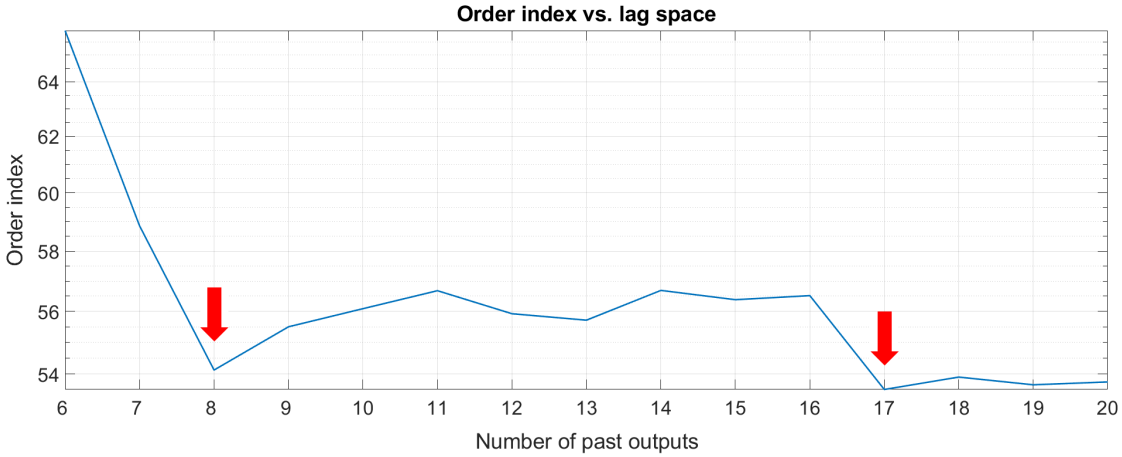


Figure 5.4: Evaluation of Order Index criterion for different lag-space.

As of point 2) of our two-steps design procedure, each of the fully-connected models obtained after Lipschitz underwent automated pruning. The objective of this second design phase is to eliminate redundant connections between the neurons, and hence obtain more efficient and compact models than the initial ones, possibly improving or at least not impacting on their prediction capability. For this purpose, we implemented an Optimal Brain Surgeon (OBS) methodology, as first introduced by [105]. The main idea behind this procedure is to estimate the increase in the training error when deleting weights, leveraging information in the second-order derivatives of the error surface. More specifically, the strategy works towards the minimisation of the error variation, leveraging a recursive calculation of the inverse Hessian matrix from the training data to obtain better approximations of the error function (see [105] for details). This strategy has been demonstrated to eliminate more redundant neuron connections than other pruning techniques, yielding to models with improved generalisation capabilities [70].

The three pruned models (respectively with 30, 17 and 8 regressors) were trained again with a Levenberg-Marquardt backpropagation procedure (LMBP). The nSSE values obtained after this second round of training are reported in the third column of Table 5.2.

Table 5.2: Validation error (nSSE) before and after OBS pruning.

NAR Network Type	**nSSE before pruning**	**nSSE after pruning**
30 regressors	30.49	25.83
17 regressors	28.60	27.05
8 regressors	26.24	25.89

From the analysis of this table, we can see that OBS pruning further reduced the validation error of the three models. As it was reasonable to expect, the model that benefited the most from the pruning is the 30 regressors one. On the other hand, the 8 regressors NAR is the one that had the best compromise between generalisation error (that is comparable with the one obtained by the 30 regressors after pruning) and simplicity of the model, which guarantees the lowest risks of over-fitting. Based on these considerations, we decided to select the pruned network with 8 regressors as the final NAR model for glucose level predictions. The architecture of this model is schematically represented in Figure 5.5.

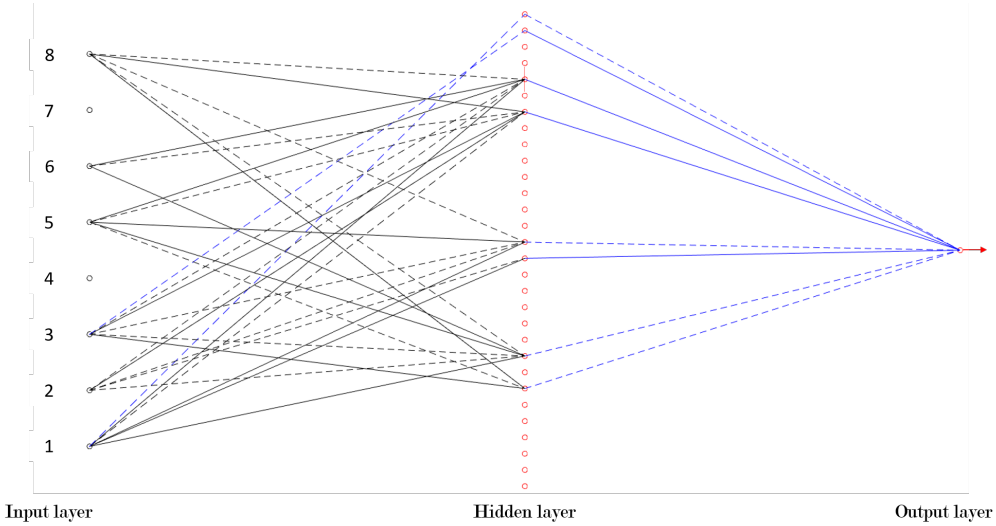


Figure 5.5: NAR final architecture

Long Short-Term Memory Neural Network

As anticipated in Chapter 2, LSTM represents an evolution of the classic Recurrent Neural Networks improving its limitations.

For our specific problem of glucose level prediction, we designed a LSTM network consisting of a layer of 30 LSTM units and a single output layer (dense), with a number of units equal to the future glucose samples that need to be predicted (i.e. 18, corresponding to a 90 min prediction horizon at a 5 min sampling rate). The architecture of the cells is the same that is depicted in Figure 2.8, consisting of input, forget and output gates with sigmoidal gating functions.

As we did for the NAR module, before deciding the final architecture of the LSTM model, we investigated the possibility of reducing the number of cells of the network as well as the number of past glucose levels to be used as regressors for the prediction. We run experiments with respectively 30 (our upper bound), 17 and 8 regressors, that represent the knee-points detected applying Lipschitz order

index method to our training data (see Figure 5.4). In Figure 5.6, we show values of the validation error, in the form of Root Mean Square Error on the validation set, obtained by the 30, 17 and 8 regressors models, at increasing time prediction windows (respectively from a minimum of 30 up to a maximum of 90 min). As it is clear from the plot, the 30 regressors model is the one with the best performance. As expected, for all the models the prediction error increases exponentially with the prediction window. The 8 regressors model is consistently the one with the lowest prediction accuracy.

In general, the 30 regressors model is the one that provides the best performance for either short-time and long-time glucose level predictions.

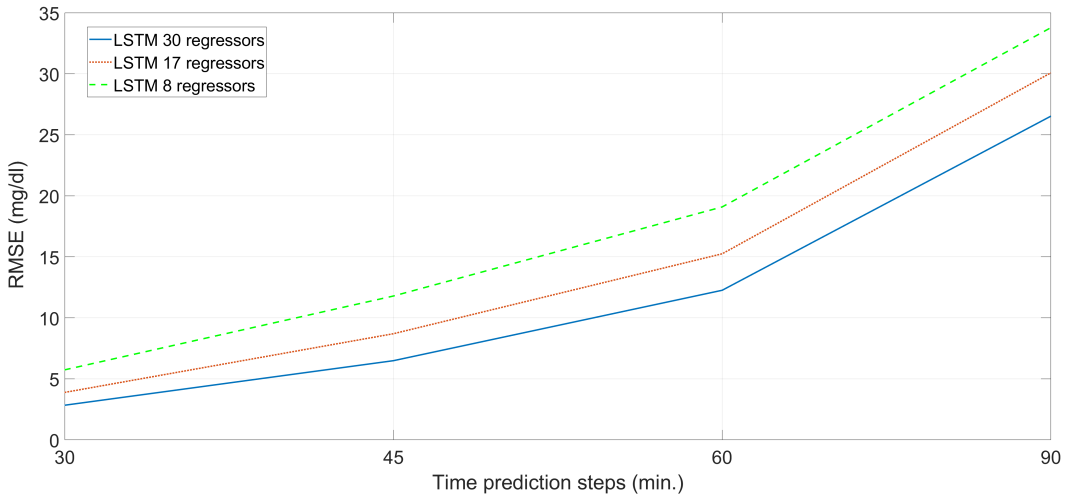


Figure 5.6: Validation error (RMSE) of LSTM models.

To optimise the hyper-parameters of our network and prevent either under-fitting or over-fitting problems, we implemented a training procedure imposing an initial learning rate equal to 0.001, with dropout. More specifically, at each training stage individual nodes and corresponding links are randomly dropped out of the model, leaving a reduced network [168]. This regularisation procedure helps reducing co-dependency of parameters, and hence prevents over-fitting. Then, we chose *Adam* (Adaptive moment estimation) as the optimizer. This algorithm leverages adaptive learning rates methods to set individual learning rates per each parameter, combining this feature with the advantages of classical optimisation techniques such as stochastic gradient descent and root mean square propagation. This technique was demonstrated to be particularly suitable for non-stationary problems with lot of noise [131]. The training steps for the learning procedure were set to 50000, with a batch size of 100. In order to keep the computational costs of the training under control without having to reduce the size of the training set, we implemented a *mini-batch* training paradigm. More specifically, the training set was split into

a number of small sub-sets of size 100, that were used to compute the model error. Each mini-batch was computed in parallel during each iteration. Then, the average error over the mini-batches was used to update the model parameters at each iteration. This method, besides considerably reducing computational time and memory requirements, has been demonstrated to generally improve the model performance [131]. To find a good compromise between a learning rate too low (which may make the training too slow) or too high (which may lead to sub-optimal solutions), we imposed a step decay schedule, that dropped the initial learning rate by a 0.9 factor every 1000 iterations.

After running the learning algorithm on the training set, the so-obtained LSTM model was used as-is to predict glucose level values on an independent test set, with prediction horizons spanning from 30 up to 90 min.

5.5 Single-patient data-driven methodology

Progressively, we investigate a data-driven patient-specialized solution based on LSTM neural network to forecast blood glucose level in short- and medium-term. In Figure 5.7, we show the main phases of our study. In the so-called training phase, training samples first undergo a pre-processing step. The unfiltered data are used as input to build a prediction model. In the test phase, new unseen and unfiltered data are fed into the trained models to obtain the final predictions.

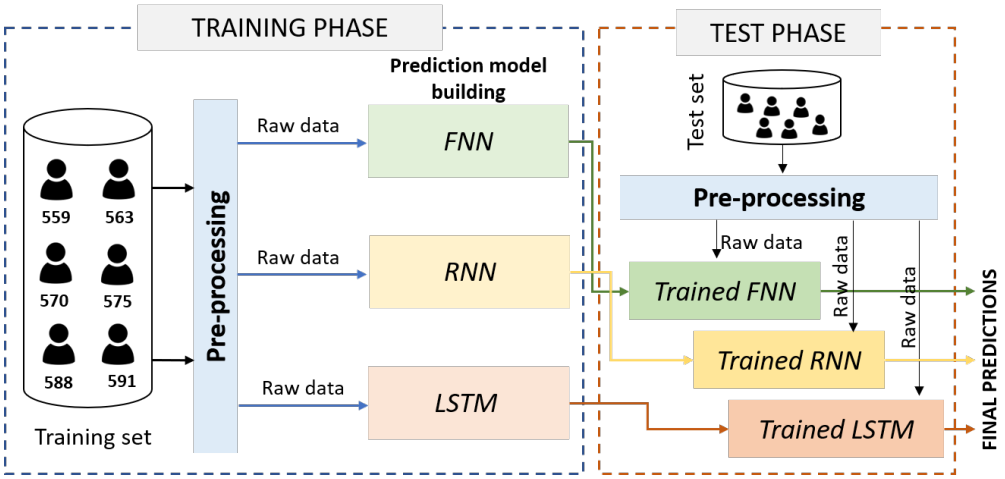


Figure 5.7: Methodology - Main phases of the study.

5.5.1 Data pre-processing

To evaluate performance and accuracy, all the models are trained and validated using the OhioT1DM dataset, as described in Section 5.3.2.

Following a preliminary phase check of the dataset, we noticed that patient measurements present some gaps. These faults may be due to temporary malfunctions or to the maintenance activities of the sensors. Such temporal lacks can lead to a wrong evaluation by the algorithms with a consequent loss of accuracy of the prediction. Consequently, we performed a pre-processing of the data by interpolating and re-sampling linearly.

Finally, we organize the data as follows:

$$train_X = \begin{bmatrix} G_0 & G_1 & G_2 & ... & G_{27} & G_{28} & G_{29} \\ G_{30} & G_{31} & G_{32} & ... & G_{57} & G_{58} & G_{59} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \end{bmatrix} \tag{5.5}$$

$$train_Y = \begin{bmatrix} G_3 & G_4 & G_5 & ... & G_{30} & G_{31} & G_{32} \\ G_{33} & G_{34} & G_{35} & ... & G_{60} & G_{61} & G_{62} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \end{bmatrix} \tag{5.6}$$

where G represents a value of the dataset. The $train_X$ matrix contains the input values of the neural network. It is a matrix composed of 30 columns, i.e. the current measurement plus the 29 previous measurements. Instead, the $train_Y$ matrix contains the output values. The two matrices have the same size, however, $train_Y$ is shifted by as many values as the prediction time horizon of interest. In the given example, the $train_Y$ is shifted by a number of samples equal to the forecast horizon, i.e. 15 minutes, then 3 samples, one every 5 minutes (sampling rate). Furthermore, to avoid over-training, samples already used by the training network are no longer used, which is why the measurements go from G_0 to G_{30} (see $train_X$). Test matrices have the same structure as those for training. To avoid erroneous results, the values within these files are never used for the training phase, but are given as input to the network only when the performance of the final architecture is to be evaluated.

5.5.2 Prediction models

In the following, we introduce our LSTM solution. Moreover, we present the identified and re-implemented state-of-art models, a FNN [202] and a RNN [10], respectively, as reported by authors. Indeed, starting from the original structure, we try to optimize these models by modifying some hyperparameters, to obtain even more robust and high-performance models according to our dataset. The original FNN was composed of 30 inputs, two hidden layers of 10 and 5 units and an output layer [202]. The structure of the RNN is similar but with two hidden layers of 20 and 13 neurons, respectively [10]. For all the models, we selected the best architecture with the best configuration. Thus, starting from the literature,

we evaluated different architectures' setups in terms of number of inputs, number of units in the hidden layer and iterations through a trial-and-error approach.

As far as FNN and RNN are concerned, our re-implemented solutions always consist of a single hidden layer. Indeed, we have managed to simplify the networks structures from a computational point of view, achieving similar or better performance.

Long Short-Term Memory Neural Network

LSTM represents the powerful evolution of the classic RNN. This neural model, particularly suitable for time series, is designed to overcome the limitations of the RNN. Indeed, RNN architectures suffer the instability of long-term predictions due to either vanishing or exploding gradient problems [95]. Often these problems arise during the training of deep structures, when the error gradients are propagated back in time to the initial layer, going through continuous matrix multiplications. To overcome this limitation, the LSTM is designed as *cells* where specific gating functions manage the memory, maintaining or erasing information at each time step. The ability to remove or add information to the cell state is regulated by gates (i.e. input, output, and forget) consisting of sigmoidal activation functions coupled with point-wise multipliers. Each gate modulates how much of the corresponding signal should be let through. In this way, LSTMs are able to remember values that are passed through gates all in 1 state, regardless of how deep the network is.

To find the best neural structure, we investigated different structural combinations with a trial-and-error approach. We found that the best structure consists of 30 inputs, a layer composed of 50 cells and an output layer. The number of optimal iterations is 2000, and the learning rate is 0.001, with the Adaptive Moment Estimation (Adam) as optimization algorithm [131].

Feed-forward Neural Network

This type of models represents the simplest type of ANN in which information propagation is mono-directional. In other words, the information moves from the input nodes to the output nodes, through the hidden layers [42]. Generally, this model is characterized by a fully-connected structure. It is also famous for its low computational costs [42]. Then, starting from the model architecture presented in [202], we performed different structure combinations. Then, we found the best structure that is composed of 30 inputs, 100 neurons in the hidden layer and only one neuron for the output layer. We exploited the sigmoid as activation function and the back-propagation as optimization, as generally recommended in literature [202]. Furthermore, we set the learning rate equal to 0.001, with an optimal number of iterations of 1500.

Recurrent Neural Network

The RNNs propagate the information in a bidirectional way. Indeed, they are composed of recurrent units sharing the same parameters, with loops allowing to propagate the information back to the same computational units. In this way, each computational step takes into account not only the current input but also the previous ones. As in the FNN case, starting from the model presented in [10], we re-implemented the structure by investigating different configurations. The best structure is configured as a follow: i) 30 input units, ii) 50 neurons in the hidden layer and iii) a single output. We exploited the sigmoid as the activation function. As the optimization method, we used Adam with a learning rate of 0.001 and a number of iterations set to 3000.

5.6 Results

In the following, we present and discuss the experiments that were performed to assess the prediction models described in Section 5.4. To obtain a thorough evaluation of our methods, our analysis was divided into two main parts, according to Section 2.5:

1. *Analytical assessment.* In this part, we assess the validity of the predictions from a regression analysis point of view, by computing a set of metrics that are widely used to quantify the similarity of a discrete time-series with a reference ground truth;

2. *Clinical assessment.* In this part, we assess the validity of the predictions from a clinical point of view. To do so, we use metrics that are specifically designed to validate the clinical outcome of blood glucose measurements.

Both the assessments were performed on a test set (fully characterised in Section 5.3) that is completely independent from the one that was used to design, train and optimise the prediction models.

Solutions were compared with the results obtained by literature techniques. As already anticipated in Section 5.1, the most prominent multi-patient data-driven prediction techniques are based either on Autoregressive models (AR), dense Feed-forward Neural Networks (FNN) or standard Recurrent Neural Networks (RNN). Even though most of the works provide a quantitative assessment of these methods, a fair comparison requires that all the predictors are trained and tested on the same data. Hence, we re-implemented three states of the art glucose predictors based respectively on AR, FNN and RNN, strictly following the design reported in the respective publications [91],[202] and [10]. Then, we trained, optimised and tested the three models on our data, with the same procedure that was applied to our proposed solutions.

The experiments were run on a Linux computer equipped with an Intel®Core™ i7-8750H CPU with 2.20 GHz, 6 cores, 12 logical processors and 16,0 GB of installed Physical Memory.

5.6.1 Analytical assessment on multi-patient models

To evaluate the prediction accuracy of the models, we exploited several metrics that are widely used in descriptive statistics and in regression analysis to quantify the similarity between predicted and observed time-series. More specifically we focused on a list of metrics that are more often used by blood glucose level predictions literature [99]: i) *RMSE*, ii) R^2, *Time lag*, *MAD* and *FIT* as deeply described in Section 2.5. Before assessing the prediction accuracy of the models in comparison with literature approaches, we run few preliminary experiments to demonstrate the goodness of the pre-processing stage based on Tikhonov regularisation applied to the training data (see Section 5.4.1).

In the plot of Figure 5.8, we report the results of these experiments. More specifically, we plot the RMSE values obtained by both the NAR and LSTM prediction models on the test dataset, with different prediction horizons (from 30 up to 90 min with steps of 15). The same experiments were performed training the models first on raw data and then on filtered data. In both cases, the testing was performed on the raw unfiltered data, as specified in the diagram of Figure 5.1.

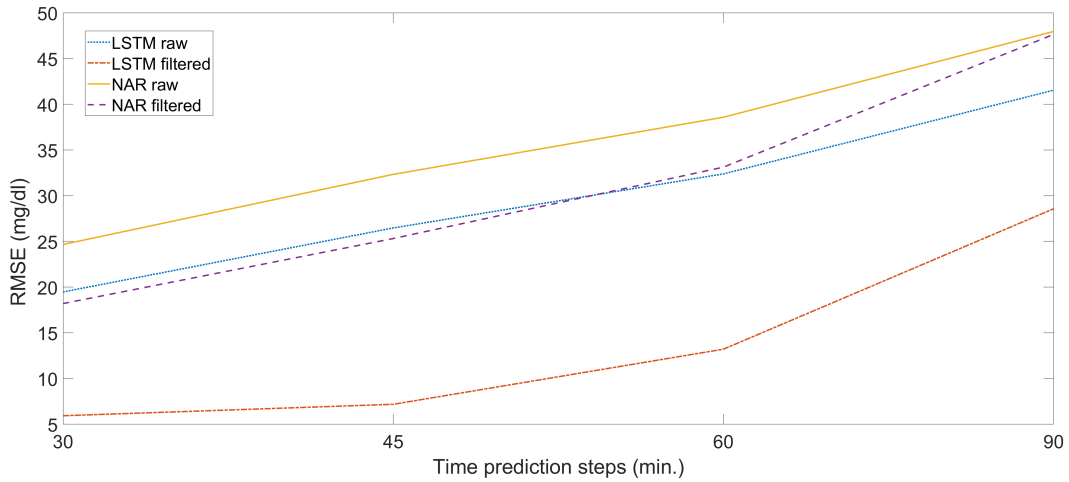


Figure 5.8: Impact of training data filtering on NAR and LSTM models predictions.

As it is visible from the plot, the filtering significantly decreased the prediction error of both the models, with particular benefit for LSTM model. On top of that, we can observe that the error trend against the prediction horizon was more or less the same with or without filtering.

In Table 5.3, we report the values obtained running all the prediction models on the same test set, with our proposed NAR and LSTM networks highlighted in grey and light blue, respectively. Each sub-section of the table shows the values of all the figures of merit defined at the beginning of the section. Different columns show values obtained at different prediction horizons, starting from 30 min (short-term prediction) up to 90 min (very long-term prediction), at steps of 15 min. For completeness, we show on the left the performance of the models trained on raw data and the right the values obtained by the same models trained on filtered data and validated on the unfiltered ones.

Table 5.3: Prediction performance indicators for all the tested models.

		NOT FILTERED TRAINING SET prediction horizon (min)				FILTERED TRAINING SET prediction horizon (min)			
		30	**45**	**60**	**90**	**30**	**45**	**60**	**90**
AR	RMSE (mg/dl)	25.96	34.16	41.02	51.66	17.88	22.6	28.31	41.66
	R^2	0.84	0.72	0.60	0.37	0.92	0.88	0.81	0.59
	Time lag (samples)	5.00	8.00	11.00	17.00	3.00	3.00	4.00	9.00
	MAD (%)	9.80	13.90	17.40	22.97	3.64	5.19	8.57	15.64
	FIT (%)	60.05	47.43	36.88	20.52	72.48	65.22	56.44	35.9
FNN	RMSE (mg/dl)	26.75	35.18	42.39	53.95	21.1	29.6	38.51	54.95
	R^2	0.83	0.71	0.57	0.31	0.89	0.79	0.65	0.28
	Time lag (samples)	5.00	8.00	11.00	17.00	4.00	6.00	8.00	14.00
	MAD (%)	10.60	14.98	18.83	25.28	7.1	11.62	16.47	25.51
	FIT (%)	58.84	45.86	34.78	16.99	67.53	54.45	40.74	15.45
RNN	RMSE (mg/dl)	26.16	37.99	48.28	57.07	18.22	22.51	26.78	38.08
	R^2	0.84	0.66	0.45	0.23	0.92	0.88	0.83	0.66
	Time lag (samples)	5.00	8.00	11.00	17.00	3.00	3.00	3.00	3.00
	MAD (%)	9.41	13.45	16.88	22.10	4.02	4.60	5.80	13.06
	FIT (%)	59.75	41.55	25.72	12.19	71.97	65.36	58.80	41.41
NAR	RMSE (mg/dl)	24.66	32.33	38.58	47.96	18.2	25.31	33.12	47.64
	R^2	0.86	0.76	0.66	0.47	0.92	0.85	0.75	0.48
	Time lag (samples)	5.00	8.00	11.00	17.00	3.00	4.00	6.00	10.00
	MAD (%)	10.10	14.21	17.60	22.70	5.96	9.96	14.41	22.57
	FIT (%)	62.66	51.04	41.59	27.37	72.44	61.67	49.84	27.86
LSTM	RMSE (mg/dl)	19.47	26.47	32.38	41.54	5.93	7.18	13.21	28.57
	R^2	0.91	0.83	0.74	0.58	$\simeq$1.00	0.99	0.96	0.80
	Time lag (samples)	3.00	6.00	9.00	15.00	0.00	0.00	1.00	6.00
	MAD (%)	8.71	12.29	15.36	20.21	2.59	3.25	6.13	13.68
	FIT (%)	69.63	58.59	49.43	34.32	90.75	88.79	79.37	55.25

From the values of Table 5.3, we can draw the following considerations:

- All the models benefited from pre-processing the training data with Tikhonov. Hence, from now on, we will mainly focus on the analysis of this set of experiments.
- the LSTM model is by far the one that obtained the best prediction performance. This is consistently confirmed by all the figures of merit, for both

short-term and long-term predictions. Remarkably, the time-lag observed for this model was zero until a prediction horizon of 60 min. On the contrary, all the other models had a time-lag of 3 samples (15 min) at best for the short-term horizon.

- FNN is the model with the worst performance. On top of that, it is the one presenting the most rapid decrease of prediction accuracy when increasing the prediction horizon. The time lag is especially high, reaching 14 samples for long-term predictions.

- NAR, AR and RNN methods seem to have a comparable behaviour, especially for short-term predictions.

- When comparing NAR with AR, the latter obtained slightly better results (e.g. RMSE was 17.88 against 18.2 mg/dl, at 30 min horizon). This difference is even more remarkable at 45 min horizons. On the other hand, if we observe the values obtained with unfiltered training data, we can see the opposite (e.g. 25.96 mg/dl against 24.66 mg/dl obtained by NAR at 30 min). Most reasonably, NAR is more robust to non-linearities in the training data when compared to AR. Nonetheless, if we consider the overall figures of merit, the performance of the two models in this specific application is almost equivalent.

To perform a more in-depth analysis of the results, in Figure 5.9, we show a plot of the RSME values obtained by all the tested models, this time in %, reporting the prediction horizon in the x-axis. To provide a better interpretation of the obtained results, we highlighted in green the area of the plot within a 20% RMSE range. In the absence of clear indications by past literature, we used this value as an indicative threshold of acceptability for RMSE, consistent with the numerical criteria for accuracy described by International Organisation for Standardisation (ISO) [78, 235].

As it can be gathered from the plot, RSME consistently increases with the prediction time, more or less with the same slope for all models. The plot confirms the undisputed superiority of LSTM, which maintained RSME values at least 10% better that all the other models, throughout all the tested prediction horizons. On top of that, LSTM is the only one that provided RMSE always within the 20% range, and below 15% up to a 60 min prediction horizon. The behaviour of NAR, AR and RNN models do not have significant differences for short-term predictions up to 30 min. The performance of NAR gets worse after that, still maintaining just within the acceptability range up to 60 min prediction horizon. AR and RNN have a similar behaviour up until the longest prediction window, with slightly better performance for the AR model.

As a qualitative confirmation of the good prediction accuracy of LSTM, in the following we show an example of glucose predictions performed by this model,

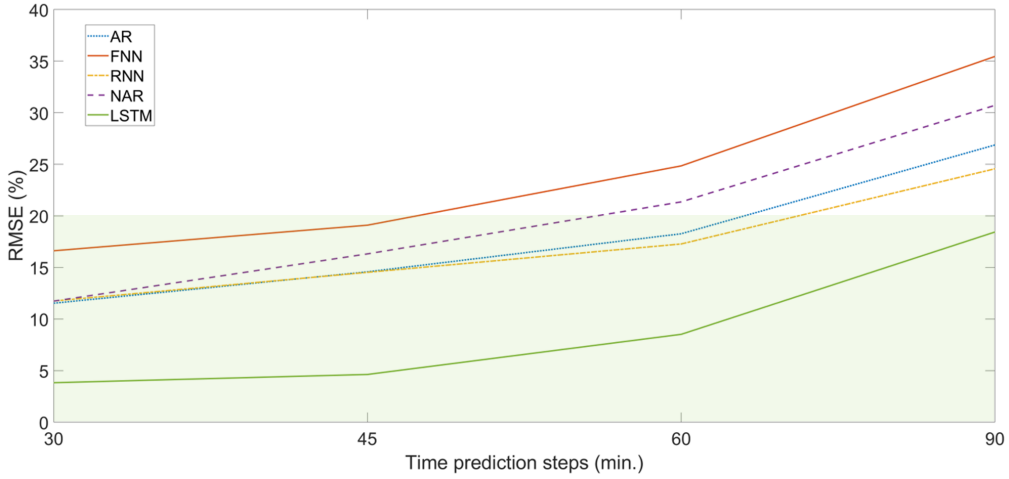


Figure 5.9: Overall RMSE comparison.

respectively short-term (30 min) in Figure 5.10a and long-term (60 min) in Figure 5.10b. In both the plots, the predicted signal is shown in blue, and the corresponding measured signal in red.

5.6.2 Clinical assessment on multi-patients models

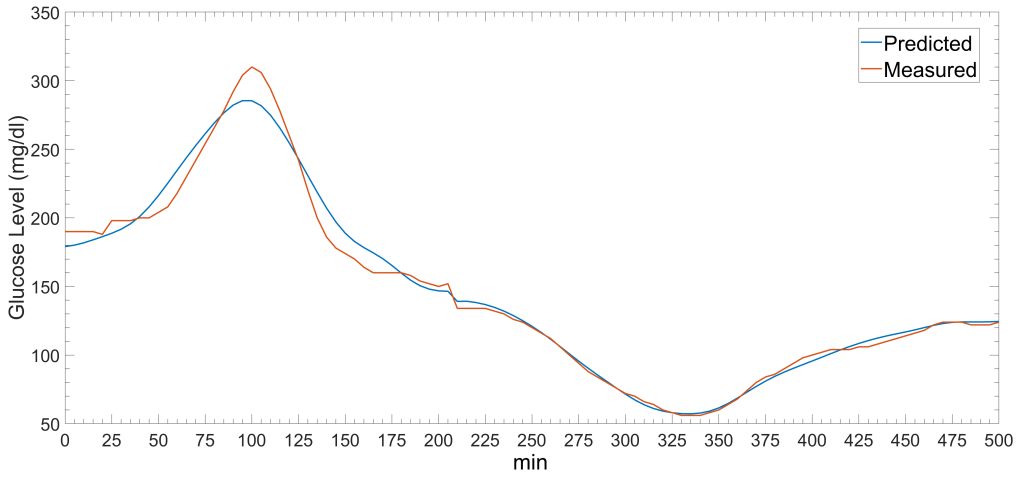
Even though the metrics identified in Section 5.6.1 are essential to understand the performance and prediction accuracy of the different models from a regression analysis point of view, they are not able to identify the most significant outliers, and they do not provide any information about the clinical impact of the prediction errors and of their consequences on medical treatment decisions. Then, to provide a more thorough picture of the models performance, we integrated our assessment with Clarke Error Grid analysis (EGA) [53].

EGA is a semi-quantitative methodology introduced in 1987 that is nowadays the most widely accepted tool for the analysis of clinical accuracy of blood glucose estimations. It provides a clinical interpretation of the mapping between predicted and measured blood glucose levels, that can be represented in a scatterplot with five main regions (see Figure 5.11):

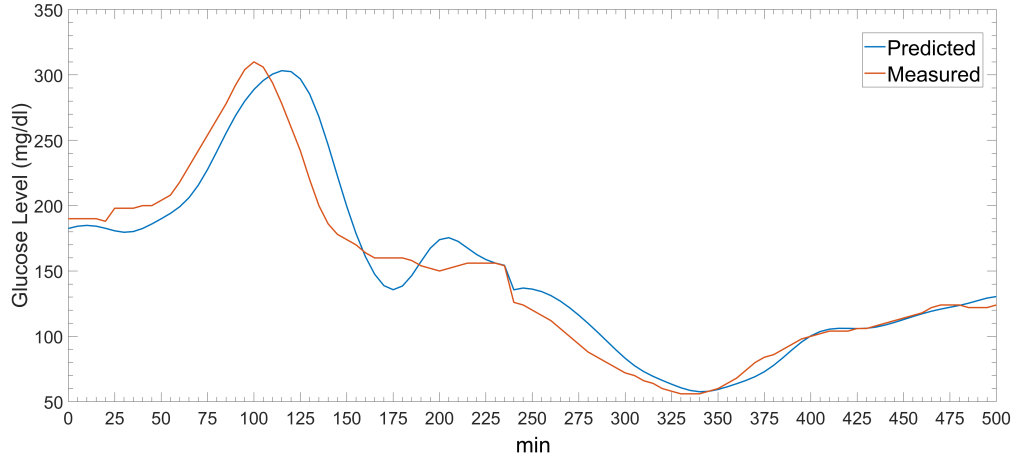
A: values within 20% of the reference;

B: values that, in spite of being outside 20% of the reference, do not lead to inappropriate treatment of the patient;

C: values leading to inappropriate treatment, but without dangerous consequences for the patient;



(a) LSTM prediction (30 min horizon).



(b) LSTM prediction (60 min horizon).

Figure 5.10: LSTM predictions at different forecasting horizons.

D: values leading to potentially dangerous failure to detect hypoglycaemic or hyperglycaemic events;

E: values leading to treat hypoglycaemia instead of hyperglycaemia and vice-versa.

Hence, zones A and B are the ones with full clinical acceptability. D and E, on the other hand, are the zones where prediction errors are most dangerous for a patient [59].

The overall results of our Clarke Error Grid analysis are reported in Table 5.4. More specifically, in each sub-section of this table we report the EGA results of a tested model, with our designed NAR and LSTM highlighted in grey and light blue, respectively. As for the analytical assessment, different columns of the table refer to different prediction horizons (30 to 90 min, with steps of 15). Values in

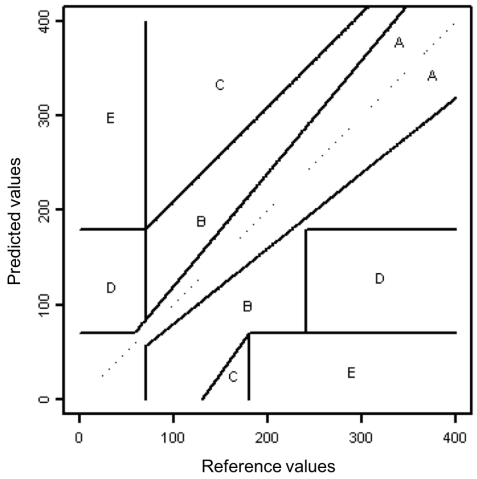


Figure 5.11: Clarke Error Grid Analysis: reference regions mapping.

different rows represent the percentage of values that fall within a specific zone of the EGA map (A to E, respectively). Again, we show for completeness the values obtained training the models on raw and on filtered data, respectively.

Table 5.4: Overall Clarke Error Grid results.

		NOT FILTERED TRAINING SET prediction horizon (min)				FILTERED TRAINING SET prediction horizon (min)			
		30	**45**	**60**	**90**	**30**	**45**	**60**	**90**
AR	A	87.01	76.99	68.55	56.30	97.74	96.33	91.8	71.68
	B	11.4	20.18	27.35	37.1	1.49	2.47	6.4	24.37
	C	0.29	0.50	0.78	1.38	0.25	0.4	0.55	0.99
	D	1.15	2.09	2.99	4.67	0.38	0.59	0.94	2.47
	E	0.15	0.24	0.33	0.57	0.14	0.22	0.3	0.49
FNN	A	85.05	74.14	64.62	50.66	94.41	83.4	70.82	51.62
	B	13.41	23.25	31.47	42.71	4.68	14.94	26.5	42.41
	C	0.27	0.51	0.96	2.19	0.26	0.42	0.72	2.4
	D	1.11	1.84	2.51	3.62	0.5	0.98	1.58	2.69
	E	0.16	0.27	0.44	0.82	0.14	0.25	0.38	0.88
RNN	A	88.06	77.96	69.44	57.67	97.51	96.43	95.14	82.04
	B	10.02	18.41	25.29	34.6	1.65	2.33	3.15	14.3
	C	0.25	0.38	0.57	0.99	0.26	0.44	0.62	0.99
	D	1.54	3.05	4.44	6.39	0.44	0.57	0.76	2.19
	E	0.13	0.21	0.26	0.34	0.14	0.23	0.32	0.48
NAR	A	84.86	72.96	64.26	52.92	95.03	84.44	73.06	54.36
	B	11.41	20.61	28.52	38.51	3.07	9.97	20.22	37.34
	C	0.18	0.32	0.46	0.72	0.16	0.28	0.44	1.02
	D	3.49	6.04	6.66	7.64	1.64	5.52	6.17	7.04
	E	0.07	0.07	0.11	0.21	0.09	0.09	0.11	0.24
LSTM	A	88.55	78.06	68.91	56.43	99.73	99.56	95.7	73.53
	B	9.88	18.73	26.28	36.73	0.21	0.36	3.76	22.7
	C	0.04	0.1	0.24	0.55	0.00	0.00	0.00	0.10
	D	1.52	3.1	4.55	6.2	0.06	0.09	0.54	3.66
	E	<0.01	0.01	0.02	0.09	0.00	0.00	0.00	0.01

From the analysis of Table 5.4, we can draw the following considerations:

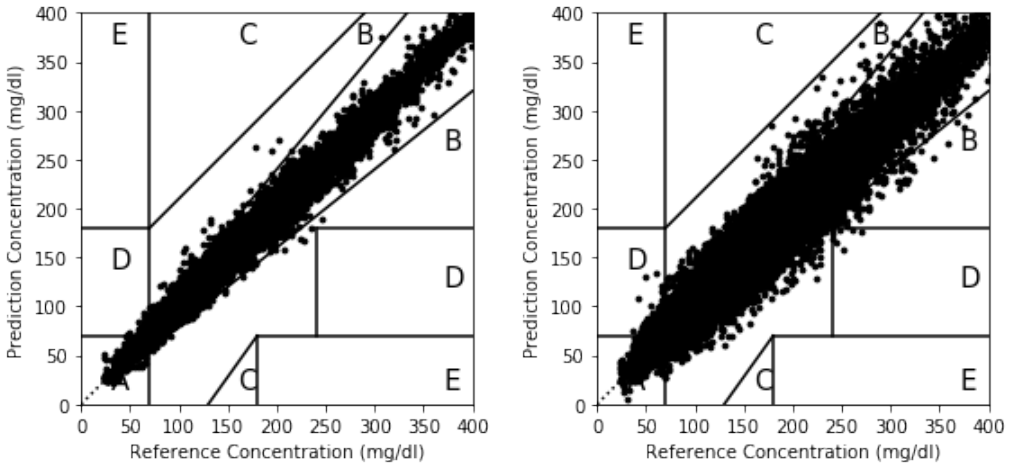
- The benefit of pre-processing the training data is again confirmed by Clarke EGA, for all the tested models. Hence, we will focus on these results.
- All the tested models had a satisfactory performance in short-term predictions. More than 94% of the data in the 30 min prediction horizon lay in the zone with best clinical outcome A. When considering a medium-term prediction horizon of 45 min, about 95% of the predictions fall within borders of the clinically acceptable zones A and B. Consistently with analytical assessment, the performance got worse when increasing further the prediction horizon.
- Partially contradicting the outcome of the analytical assessment, when considering the clinical outcome, NAR model was outperformed by the other methods. A possible interpretation of the discrepancy between the analytical and the clinical assessments is that the latter is much more affected by the presence of measurement spikes and outliers. Hence, while other prediction methods (and especially FNN) provide predictions that are on average not well-correlated with the observed measurements, they probably provided a lower number of outliers in comparison with NAR. Nonetheless, as previously observed, the short-term prediction performance of all the methods is quite comparable even from a clinical point of view.
- The undisputed superiority of LSTM model, both for short-term and long-term predictions, was confirmed even by EGA results. This model maintained 99% of the predicted values within the clinically acceptable zone up to 60 min horizon window and more than 99% in zone A up to 45 min forecasting. No other model showed comparable performance.

To have a better view of LSTM performance, in Figure 5.12, we show a graphical representation of LSTM EGA, respectively for the short-term (30 min) predictions and the long-term (60 min) predictions.

As it can be easily gathered from the plots, the data in Figure 5.12a distribute along the bisector, which is the region of highest correlation possible with the reference values. As expected, the data in Figure 5.12b have much higher dispersion. Nonetheless, we can observe that, even though the forecasting window was in this case extremely large (60 min), the wide majority of the values were still within the borders of best clinical acceptability.

5.6.3 Analytical assessment on single-patient models

As for Section 5.6.1, to evaluate the prediction accuracy of the single-patient models, we exploited a set of metrics that are often used by blood glucose level predictions literature. These are: *RMSE* and R^2, in according to Section 2.5.



(a) LSTM 30 min. (b) LSTM 60 min.

Figure 5.12: LSTM Clarke Error Grid analysis.

Table 5.5: Prediction performance indicators for FNN.

		Index	Patient ID					
			559	563	570	575	588	591
FNN	30 min	RMSE (mg/dl)	22.22	20.49	18.53	25.13	20.52	23.79
		r^2	0.88	0.79	0.92	0.80	0.80	0.79
	45 min	RMSE (mg/dl)	28.08	31.49	24.74	31.68	27.00	30.29
		r^2	0.81	0.55	0.86	0.72	0.65	0.65
	60 min	RMSE (mg/dl)	33.14	34.59	31.81	40.57	33.70	35.63
		r^2	0.73	0.47	0.77	0.55	0.46	0.49
	90 min	RMSE (mg/dl)	43.43	40.25	44.16	51.59	41.71	42.85
		R^2	0.51	0.27	0.58	0.36	0.21	0.29

Tables 5.5, 5.6 and 5.7 report all the experimental results obtained by performing the re-implemented FNN and RNN architectures and our proposed LSTM solution. In all three scenarios, we used raw data in training, validation and inference phase for individual patients (marked with an identification code 559, 563, 570 575, 588, 591 respectively). Moreover, the forecast time horizons of interest are 30, 45, 60 and 90 minutes, respectively. For each prediction time horizon, we report values of $RMSE$ and R^2. As these networks are patient-specialised, consequently, we carried out the analysis of the results per patient.

Analyzing the analytical indices for all the patients, we found that state-of-art RNN is more performant than the FNN in terms of prediction accuracy. This

Table 5.6: Prediction performance indicators for RNN.

		Index	Patient ID					
			559	563	570	575	588	591
RNN	30 min	RMSE (mg/dl)	18.26	21.00	17.53	22.53	19.62	23.46
		r^2	0.92	0.78	0.93	0.84	0.80	0.79
	45 min	RMSE (mg/dl)	24.00	32.06	23.71	30.26	26.24	30.12
		r^2	0.86	0.54	0.87	0.74	0.65	0.65
	60 min	RMSE (mg/dl)	30.05	35.48	30.87	37.15	33.04	35.61
		r^2	0.77	0.45	0.78	0.62	0.46	0.49
	90 min	RMSE (mg/dl)	40.07	41.47	42.31	47.44	42.07	43.46
		r^2	0.56	0.23	0.61	0.45	0.20	0.27

Table 5.7: Prediction performance indicators for our LSTM solution.

		Index	Patient ID					
			559	563	570	575	588	591
LSTM Proposed	30 min	RMSE (mg/dl)	11.52	10.84	10.20	9.29	11.55	10.56
		r^2	0.93	0.95	0.97	0.97	0.92	0.95
	45 min	RMSE (mg/dl)	19.58	12.05	16.14	18.32	19.86	16.91
		r^2	0.90	0.94	0.94	0.90	0.79	0.87
	60 min	RMSE (mg/dl)	27.67	20.88	23.02	29.07	25.00	25.46
		r^2	0.80	0.85	0.88	0.76	0.68	0.71
	90 min	RMSE (mg/dl)	43.99	36.03	34.24	49.89	30.95	41.44
		r^2	0.52	0.63	0.74	0.36	0.54	0.29

translates into smaller values of $RMSE$ and R^2, as described in Table 5.5 and Table 5.6 respectively. However, this happens for all time horizons except for Patient 563. In this specific case, the indices for each prediction time horizon are slightly higher. This phenomenon occurs also for our proposed LSTM solution (see Table 5.7). Also, for the patient 591, we noted that, except for a few specific time horizons (i.e. 60 an 90 minutes), our LSTM solution achieve slightly worse results. In our opinion, this is because the dataset at our disposal is minimal for adequate training, especially for recurrent networks, which generally require a large enough amount of data better to understand their trends and non-linear relationships [50]. However, even with this limitation due to the dataset, our LSTM achieves better performance. For patients 559 and 588, the results are much better for each time

horizon analyzed (i.e. from 30 min. to 90 min.). Instead, Patient 570 is the best case found. In fact, for each time horizon, we lower the RMSE value from 6 mg/dl to 8 mg/dl compared to RNN, which in turn performs better than FNN. Generally, we can attest that our solution based on a neural network type LSTM results the most promising architecture. In terms of prediction accuracy, compared to all the investigated time horizons, we found improvements at every time horizon under analysis.

Referring to the most recent literature, the maximum acceptable RMSE value ranges from 19.32 mg/dl to 24.83 mg/dl in 30 min. predictions [160]. Using these state-of-art limits as a benchmark, we can achieve good prediction accuracy up to 60 min. in the future for Patients 563 and 570. This limit can also be considered suitable for Patients 588 and 591, as the maximum deviation is about 0.6 mg/dl.

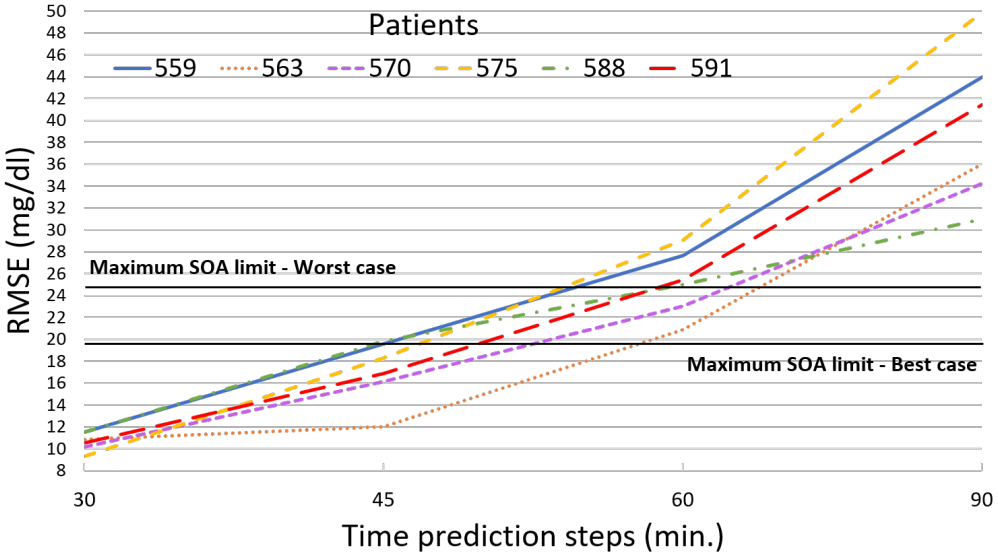


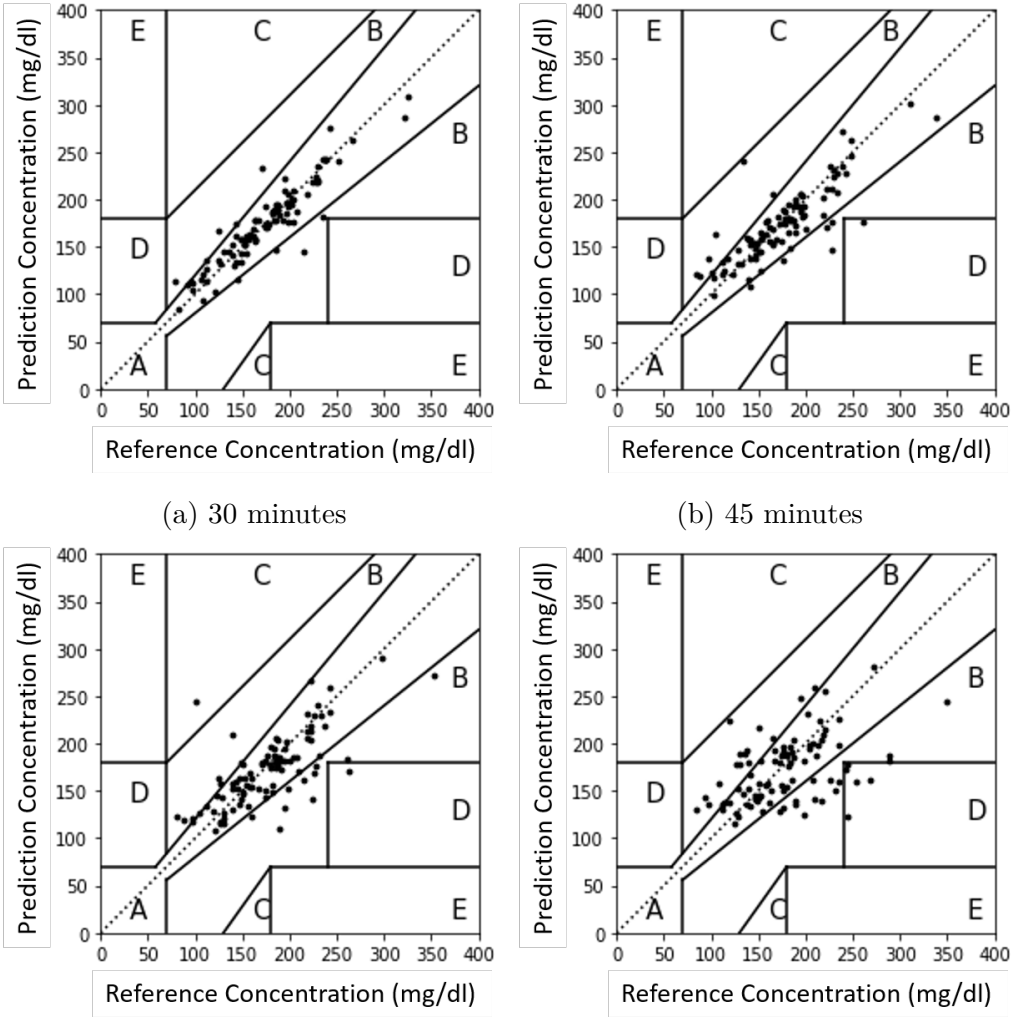
Figure 5.13: LSTM predictions analysis.

Figure 5.13 shows a plot of the RSME values obtained by all patients for our proposed LSTM solution. The horizontal continuous black lines represent the best and the worst limit for 30 min. predictions, as described in [160]. Taking as reference the most conservative limit (i.e. the lowest black line), we can affirm that our model allows us to achieve good results up to 45 min. for all patients and up to 60 min. ahead for Patients 588 and 591.

5.6.4 Clinical assessment on single-patient models

As introduced in Section 5.6.2, the identified analytical metrics are suitable to understand the performances and the prediction accuracy of models from a regression analysis point of view. However, these metrics are not appropriate to

identify the most significant outliers, and they do not provide any information about the clinical impact of the prediction errors and their consequences on medical treatment decisions. Therefore, to give a more thorough picture of the models' performance, we integrated our assessment with Clarke Error Grid analysis, as detailed in Section 5.6.2.



(a) 30 minutes

(b) 45 minutes

(c) 60 minutes

(d) 90 minutes

Figure 5.14: Clarke Error Grids for specialized LSTM network for Patient 575.

Figure 5.14 shows the EGA analysis performed for all time horizons (i.e. from 30 to 90 min.) for Patient 575. This represents the worst case we found, in terms of prediction accuracy, especially for the 60 and 90 min. prediction horizons. The EGA analysis confirms what was found analytically in the Section 5.6.3. Up to 45 minutes in the future, all values are concentrated in the zones fully clinical

acceptable (i.e. A and B). Differently, for longer time horizons some values are placed in zone D, therefore already potentially dangerous for the patient.

5.7 Discussion and Remarks

In this Chapter, we addressed the problem of automated glucose level prediction leveraging multi-patient CGMS data. Our specific aim is to learn a generalisable glucose level prediction model from a multi-patient training set, using this model to predict the future glucose values of a new patient. This allows improving the usability of models that are solely based on the past recordings of the same patient.

The main contribution of our work compared to past literature is two-fold: i) we learn the prediction models using a broad set of CMGS data from a very heterogeneous set of diabetic patients. This possibly increases the generalisation capability of the model and minimises the risks of overfitting; ii) we design and compare different types of prediction models, analysing the prediction outcome both from the analytical and from the clinical point of view. To address the limitations of literature approaches, we explored two types of models. The first solution exploits a Non-Linear Autoregressive Neural Network (NAR) that is supposed to extend the assumptions of linearity and overcome stability problems of traditional AR. The second solution exploits Long Short-Term Memory (LSTM) that addresses the exploding and vanishing gradient problems of classic RNN networks.

According to our experiments, the NAR network obtained satisfactory results only for short-term predictions, within 30 min. Nonetheless, if we take into account the model's simplicity (the NAR is based on just 8 regressors against 30 of all the other approaches), which makes it very convenient for hardware implementation, we can still consider it a good solution for systems not requiring a very large forecasting window. Finally, as confirmed by both the analytical and clinical assessment, our LSTM network overcame by far the prediction accuracy of all the other models, for both short-term and long-term predictions. Hence, we can conclude that LSTM is the preferable approach for systems requiring a very long-term forecasting window.

In our future work, we will extend our multi-patient data-driven system by integrating real-time information. More specifically, we plan to perform a real-time fine-tuning of the model, leveraging the glucose level measurements of the patient that is currently using the system. Indeed, in this view, in the second part of our methodology, we addressed the problem of automated glucose level prediction leveraging patient-specialized CGMs data. In detail, we have designed, implemented and compared different specialized state-of-art models based on neural networks. Our specific aim is to find the best neural solution that best fits the specialization on the individual patient. Thus, starting with a deeply specialized dataset and some neural solution in literature, we identify the best-optimized structure, with the best prediction performance in terms of forecast accuracy.

Chapter 6

Conclusion

In recent years, the research about energy waste and pollution reduction has gained a strong momentum, also pushed by European and national funding initiatives. The primary purpose of this large effort is to reduce the effects of greenhouse emission, climate change to head for a sustainable society. In this scenario, Information and Communication Technologies (ICT) play a key role in reducing energy consumption and to move forward to a more sustainable and smart society.

Consequently, moving towards smart and sustainable energy use, during my research activities, I focused on the design and development of neural models for the forecasting of time-series in Smart City scenario. Time-series represent the heart of the "smart" philosophy, and therefore researchers and specialists have to know how to manage and predict them to implement increasingly effective control policies. In detail, with the support of my supervisors and the EDA research team, I had the opportunity to address the issues of thermal modelling of Smart Buildings, the prediction of GHI which is the energy component necessary for the development of photovoltaic energy and, finally, I moved toward the person, by addressing the theme of the prediction of blood glucose level for Type I diabetic patients. All the studies were conducted following a bottom-up approach: starting from the analysis and appropriate pre-processing of IoT data, to design neural models suitable for the type of dataset and its characteristics and, finally, by comparing these new models with the literature methodologies.

In the context of renewable, I developed a methodology that can make photovoltaic predictions starting from the physical phenomenon of GHI. I am also investigating how to properly exploit exogenous inputs, related to the physical phenomenon of solar radiation, to obtain increasingly accurate predictions. In detail, the first part of the developed methodology focus on a newel solution to forecast GHI in short-term period by implementing two different non-linear autoregressive neural networks, NAR and NARMA, respectively. Both neural networks have been implemented, trained and validated exploiting a dataset consisting of four years of solar radiation values collected by a real weather station. Then, the results of these

ANNs have been given as input to a Photovoltaic simulator (PVsim) [35] to estimate the energy production of photovoltaic systems. The accuracy of these results is also acceptable, especially for time horizon from future 15 to 45 minutes. The second part of the methodology, instead, focus on the development of state-of-art artificial neural networks specifically on short- and mid-term GHI forecast. The main goal was to identify the most performing solution that allows more accurate GHI predictions. In this view, I designed and optimized four ANN architectures based on i) NAR, ii) FFN, iii) LSTM and iv) ESN. Contextually, using these architectures, three different approaches were evaluated: i) the raw GHI was directly used to train the networks and to make predictions; ii) the training set of the GHI series was filtered with Tikhonov regularisation before performing the training procedure; iii) the GHI time-series was transformed into the clear-sky index series, which was then used to train the networks and make predictions. The obtained results suggest that using a multi-output approach the accuracy significantly improves, specifically when more than 60 min in the future is required. In our case, this means that for forecast horizons of 45 or 60 min and longer this approach is to be preferred over a single-output model used iteratively. Considering that a single-output model is simpler, it may be better to use the iterative approach when a long-term prediction is not needed for the application. Futhermore, the ESN has given very good results compared to other models, mainly when directly predicting GHI. Moreover, compared to the other models, ESN needs a smaller number of regressors to give very accurate results. This can be important in terms of availability. Given the lack of research in the literature on GHI forecasting with ESN, these findings are genuinely new results worthy of further studies. Another major achievement concerns the use of Tikhonov regularisation. In detail, the results have shown that the proposed model that uses Tikhonov regularisation to filter the training data and uses the unfiltered GHI for the testing part, which is used successfully in other fields involving time-series forecasting, like blood glucose predictions, does not appear suitable for GHI forecasting. Therefore, filtered data was used in input for the testing part, too. This is not ideal since this method would require, when the system is used for actual predictions "in the field", to filter the data every time a new forecast is requested, which might limit the applicability of the method. Moreover, the Tikhonov filter was applied to the whole testing set, divided into long segments, but in a real application, using real-time data, this is not possible, because new data would have to be filtered when it becomes available (e.g. in our scenarios every 15 min). The algorithm would have to be modified accordingly, and the way it might affect the results needs to be studied more in detail. However, with this approach (i.e. exploiting filtered data in the entire process, from training to inference) the results were more interesting, showing that potentially, filtered data allows maintaining a better accuracy for short- and mid-term forecast. Last but not least, the obtained results have shown that the clear-sky index K_c significantly improves prediction accuracy when predicting many steps ahead, particularly for

NAR, FFNN and LSTM networks. As already stated, the improvement for ESN is small. However, using K_c allowed the ESN to give better results with a smaller reservoir, which is important in terms of memory usage. Finally, I investigated the effectiveness of using exogenous inputs for short-term solar radiation forecasting. In detail, with my research group, we identified a subset of relevant input variables for predicting GHI by applying different feature selection techniques to a broader set of variables. The results of feature selection revealed that the most significant input variables for predicting solar radiation are: i) UV index, ii) cloud cover, iii) temperature, iv) humidity, v) dew point, vi) wind bearing, vii) sunshine duration and viii) hour of the day. To assess the usefulness of the selected features, we evaluated and compared the prediction performance of five different machine learning models, namely i) a FNN), ii) an ESN), iii) a 1D-CNN, iv) LSTM and v) a Random Forest. Overall, the LSTM demonstrated the best prediction performance among the five models, producing acceptable forecasting errors up to 4 hours ahead. The FNN and the 1D-CNN also demonstrated excellent prediction performance, comparable to those of the LSTM for prediction horizons shorter than 2 hours. The ESN presented the highest forecasting errors, revealing poor prediction performance in modelling multivariate time series. The RF performed slightly better than the ESN, showing promising results. Finally, in order to demonstrate the effectiveness of using exogenous inputs for short-term solar radiation forecasting, we compared the multivariate models with their univariate counterparts. The results showed that the adoption of exogenous inputs can significantly improve the forecasting performance for prediction horizons greater than 15 min, while for shorter prediction horizons the performance improvement due to exogenous inputs can be considered as negligible. Overall, the results demonstrated the effectiveness of using exogenous inputs for short-term solar radiation forecasting.

In the context of Smart Buildings, I developed a comprehensive methodology that allows thermal modelling in both new generation and historic buildings. This by exploiting the possibility of creating a very reliable synthetic dataset based on BIM technology and real weather data. The achievement of hybrid models (i.e. based on synthetic data for the opening and real data for the inference) allows the thermal modelling of all those buildings with a lack of historical data. Consequently, I investigated how to specialize hybrid models on real data (i.e., provided by IoT devices embedded in a real demonstrator). For this purpose, I applied different techniques of Transfer Leaning, in search of the best model for the case study. In the proposed methodology, BIM and meteorological data are exploited to construct of a realistic and consistent dataset of indoor air-temperature values. The presented experimental results showed that the proposed solution simulates with a good accuracy the heating performance of the case-study building. With respect to literature solutions that consider TMY weather data, our results highlighted that the integration of real weather information into the simulation process strongly increases the accuracy of the simulation itself. Thus, exploiting realistic synthetic

data to train prediction models, we are able to implement hybrid model that use real-world data in the inference phase. To validate this, I have specially designed some models (i.e one for each environment of demonstrator) based on a NAR architecture with a high number of regressors by discussing the prediction accuracy by analyzing the inference results both on synthetic and real data. As demonstrated by the case study, the models provide accurate predictions with time horizons in the order of 3 h for individual rooms and 4 h for the entire building. At the same time, in collaboration with the EDA group, we tried to add additional information as input to our hybrid models (i.e. time labels). By analyzing the experimental results, we can state that adding further complexity the performance gets worse. The reason is to be found in the fact that the hybrid models are already able to extrapolate the information of the recurring patterns and seasonality. As a result, they ignore the temporal relationship added by the exogenous time-labels. Finally, I tried to optimize the models by applying some Transfer Learning techniques in order to specialize the hybrid models with real data provided by IoT sensors installed in the demonstrator. For this purpose, I have specially designed, implemented and compared some state-of-art neural networks for time-series forecasting. In detail, these are: i) NAR (i.e. the benchmark model), ii) FNN, iii) LSTM and iv) 1D-CNN. For all these models, I initially used the synthetic data for the training phase and the real-world data for the inference phase. Then, I compared all the obtained models in analytical and qualitative way. The experimental results showed that the most promising hybrid model for this scenario was 1D-CCN. As a result, I have applied 3 fine-tuning techniques for the specialization on real data. We find that by re-training the last layer (i.d. Freezing Layer techniques) the models are able to predict until 28 hours head without introducing thermal discomfort for the occupants.

Finally, I had the opportunity to address the problem of automated glucose level prediction averaging multi-patient CGMS data. The aim is to learn a generalizable glucose level prediction model from a multi-patient training set, using this model to predict the future glucose values of a new patient. In practice, the objective is to create a device that can be purchased and is ready to use, without the need for initial tuning. Besides, I started to evaluate techniques to specialize this methodology by integrating real-time information to specialize the automated glucose predictor specifically on a single end-user. The main contribution, compared to past literature, is two-fold: i) I learned the prediction models using a broad set of CMGS data from a very heterogeneous set of diabetic patients. This possibly increases the generalisation capability of the model and minimises the risks of overfitting; ii) I designed and compared different types of prediction models, analysing the prediction outcome both from the analytical and from the clinical point of view. To address the limitations of literature approaches, I explored two types of models. The first solution exploits a Non-Linear Autoregressive Neural Network that is supposed to extend the assumptions of linearity and overcome stability problems

of traditional AR. The second solution exploits Long Short-Term Memory that addresses the exploding and vanishing gradient problems of classic RNN networks. According to the experiments, the NAR network obtained satisfactory results only for short-term predictions, within 30 min. Nonetheless, if we take into account the model simplicity (the NAR is based on just 8 regressors against 30 of all the other approaches), which makes it very convenient for hardware implementation, we can still consider it a good solution for systems not requiring a very large forecasting window. Finally, as confirmed by both the analytical and clinical assessment, the proposed LSTM network overcame by far the prediction accuracy of all the other models, for both short-term and long-term predictions. Hence, we can conclude that LSTM is the preferable approach for systems requiring a very long-term forecasting window. In future, I would like to extend the proposed multi-patient data-driven system by integrating real-time information. More specifically, I plan to perform a real-time fine-tuning of the model, leveraging the glucose level measurements of the patient that is currently using the system. Indeed, in this view, in the second part of our methodology, I addressed the problem of automated glucose level prediction leveraging patient-specialized CGMs data. In detail, I designed, implemented and compared different specialized state-of-art models based on neural networks. The specific aim is to find the best neural solution that best fits the specialization on the individual patient. Thus, starting with a deeply specialized dataset and some neural solution in literature, I identify the best-optimized structure, with the best prediction performance in terms of forecast accuracy. In detail, I designed a specialized performing solution based on Long Short-Term Memory neural network. The proposed solution was then experimentally compared with two literature approaches, respectively based on Feed-Forward and Recurrent neural networks. The experimental results have highlighted that the LSTM obtained good performance both for short- and long-term glucose level inference (60 min.), overcoming the other methods both in terms of correlation between measured and predicted glucose signal and in terms of clinical outcome.

Bibliography

[1] Mohamed Abdel-Nasser and Karar Mahmoud. "Accurate photovoltaic power forecasting models using deep LSTM-RNN". In: *Neural Computing and Applications* (2017), pp. 1–14.

[2] Andrea Acquaviva et al. "Forecasting Heating Consumption in Buildings: A Scalable Full-Stack Distributed Engine". In: *Electronics* 8.5 (2019), p. 491.

[3] Allam Ahmed. "Digital revolution, smart cities and performance improvement towards a sustainable knowledge-based inclusive development". In: (2017).

[4] Hannele Ahvenniemi et al. "What are the differences between sustainable and smart cities?" In: *Cities* 60 (2017), pp. 234–245.

[5] Jaouher Ben Ali et al. "Continuous blood glucose level prediction of Type 1 Diabetes based on Artificial Neural Network". In: *Biocybernetics and Biomedical Engineering* 38.4 (2018), pp. 828–840.

[6] Alessandro Aliberti et al. "A Multi-Patient Data Driven Approach to Blood Glucose Prediction". In: *IEEE Access* (2019).

[7] Alessandro Aliberti et al. "A Non-Linear Autoregressive Model for Indoor Air-Temperature Predictions in Smart Buildings". In: *Electronics* 8 (Sept. 2019), p. 979. DOI: 10.3390/electronics8090979.

[8] Alessandro Aliberti et al. "Forecasting Short-term Solar Radiation for Photovoltaic Energy Predictions". In: *Proceedings of the 7th International Conference on Smart Cities and Green ICT Systems - Volume 1: SMARTGREENS,* INSTICC. SciTePress, 2018, pp. 44–53. ISBN: 978-989-758-292-9. DOI: 10.5220/0006683600440053.

[9] Alessandro Aliberti et al. "Indoor Air-Temperature Forecast for Energy-Efficient Management in Smart Buildings". In: *2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe).* IEEE. 2018, pp. 1–6.

[10] Fayrouz Allam et al. "A recurrent neural network approach for predicting glucose concentration in type-1 diabetic patients". In: *Engineering Applications of Neural Networks*. Springer, 2011, pp. 254–259.

[11] A Allouhi et al. "Energy consumption and efficiency in buildings: current status and future trends". In: *Journal of Cleaner production* 109 (2015), pp. 118–130.

[12] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[13] Ahmad Alzahrani et al. "Solar irradiance forecasting using deep neural networks". In: *Procedia Computer Science* 114 (2017), pp. 304–313.

[14] Omid Ameri Sianaki et al. "Machine Learning Applications: The Past and Current Research Trend in Diverse Industries". In: *Inventions* 4.1 (2019), p. 8.

[15] An, EEI-AEIC-UTC. "A Discussion of Smart Meters And RF Exposure Issues". In: (2011).

[16] Ricardo Cesar de Andrade and Chigueru Tiba. "Extreme global solar irradiance due to cloud enhancement in northeastern Brazil". In: *Renewable energy* 86 (2016), pp. 1433–1441.

[17] ANSI/ASHRAE Standard 55 (2013). "Thermal Environmental Conditions for Human Occupancy". In: (2013).

[18] American Diabetes Association et al. "Diagnosis and classification of diabetes mellitus". In: *Diabetes care* 37.Supplement 1 (2014), S81–S90.

[19] Mark A Atkinson, George S Eisenbarth, and Aaron W Michels. "Type 1 diabetes". In: *The Lancet* 383.9911 (2014), pp. 69–82.

[20] Nivine Attoue, Isam Shahrour, and Rafic Younes. "Smart building: Use of the artificial neural network approach for indoor temperature forecasting". In: *Energies* 11.2 (2018), p. 395.

[21] Autodesk. *Revit*. Available at: http://www.autodesk.com/products/revit-family/overview/. 2020.

[22] Danielly B Avancini et al. "Energy meters evolution in smart grids: A review". In: *Journal of cleaner production* 217 (2019), pp. 702–715.

[23] Viorel Badescu. *Modeling solar radiation at the earth's surface*. Springer, 2014.

[24] Alireza Bahmanyar et al. "Emerging smart meters in electrical distribution systems: Opportunities and challenges". In: *2016 24th Iranian Conference on Electrical Engineering (ICEE)*. IEEE. 2016, pp. 1082–1087.

[25] Sebastián Basterrech. "Empirical Analysis of the Necessary and Sufficient Conditions of the Echo State Property". In: (Mar. 2017).

[26] Souhaib Ben Taieb, Antti Sorjamaa, and Gianluca Bontempi. "Multiple-output Modeling for Multi-step-ahead Time Series Forecasting". In: *Neurocomput.* 73.10-12 (June 2010), pp. 1950–1957. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2009.11.030. URL: http://dx.doi.org/10.1016/j.neucom.2009.11.030.

[27] Clara Benevolo, Renata Paola Dameri, and Beatrice D'Auria. "Smart mobility in smart city". In: *Empowering Organizations*. Springer, 2016, pp. 13–28.

[28] B Wayne Bequette. "Continuous glucose monitoring: real-time algorithms for calibration, filtering, and alarms". In: *Journal of diabetes science and technology* 4.2 (2010), pp. 404–418.

[29] Stephen A Billings. *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[30] Chris M Bishop. "Training with noise is equivalent to Tikhonov regularization". In: *Neural computation* 7.1 (1995), pp. 108–116.

[31] A Blanter and M Holman. *Internet of Things 2020: a glimpse into the future (2020)*.

[32] A Bonati et al. "The integration of exergy criterion in energy planning analysis for 100% renewable system". In: *Energy* 174 (2019), pp. 749–767.

[33] Lorenzo Bottaccioli et al. "A novel integrated real-time simulation platform for assessing photovoltaic penetration impacts in smart grids". In: *Energy Procedia* 111 (2017), pp. 780–789.

[34] Lorenzo Bottaccioli et al. "Building energy modelling and monitoring by integration of IoT devices and Building Information Models". In: *Computer Software and Applications Conference (COMPSAC), 2017 IEEE 41st Annual*. Vol. 1. IEEE. 2017, pp. 914–922.

[35] Lorenzo Bottaccioli et al. "GIS-based Software Infrastructure to Model PV Generation in Fine-grained Spatio-temporal Domain". In: *IEEE Systems Journal* (2017).

[36] George EP Box et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[37] Troy Bremer and David A Gough. "Is blood glucose predictable from previous values? A solicitation for data." In: *Diabetes* 48.3 (1999), pp. 445–451.

[38] Isabel Sofia Brito et al. "Leveraging IoT Framework to Enhance Smart Mobility: The U-Bike IPBeja Project". In: *Smart Systems Design, Applications, and Challenges*. IGI Global, 2020, pp. 166–185.

[39] Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting*. springer, 2016.

[40] Francesco Gavino Brundu et al. "IoT Software Infrastructure for Energy Management and Simulation in Smart Cities". In: *IEEE Transactions on Industrial Informatics* 13.2 (2017), pp. 832–840.

[41] buildingSMART. *IFC Overview summary*. Available at: http://www.buildingsmart-tech.org/specifications/ifc-overview/. 2020.

[42] Weipeng Cao et al. "A review on neural networks with random weights". In: *Neurocomputing* 275 (2018), pp. 278–287.

[43] A. Caragliu, C. Del Bo, and P. Nijkamp. *Smart cities in Europe*. Serie Research Memoranda 0048. VU University Amsterdam, Faculty of Economics, Business Administration and Econometrics, 2009.

[44] Filipe Gabriel Carloto et al. "The Role of a Smart Street Lighting into a Smart Grid Environment". In: *2019 IEEE PES Innovative Smart Grid Technologies Conference-Latin America (ISGT Latin America)*. IEEE. 2019, pp. 1–6.

[45] Marshall W Carpenter and Donald R Coustan. "Criteria for screening tests for gestational diabetes". In: *American Journal of Obstetrics & Gynecology* 144.7 (1982), pp. 768–773.

[46] Michael Caswell et al. "Accuracy and user performance evaluation of a blood glucose monitoring system". In: *Diabetes technology & therapeutics* 17.3 (2015), pp. 152–158.

[47] José Marìa Cavanillas, Edward Curry, and Wolfgang Wahlster. "The big data value opportunity". In: *New horizons for a data-driven economy*. Springer, Cham, 2016, pp. 3–11.

[48] Girish Chandrashekar and Ferat Sahin. "A survey on feature selection methods". In: *Computers & Electrical Engineering* 40.1 (2014), pp. 16–28.

[49] Sudesna Chatterjee, Kamlesh Khunti, and Melanie J Davies. "Type 2 diabetes". In: *The Lancet* 389.10085 (2017), pp. 2239–2251.

[50] Zhengping Che et al. "Recurrent neural networks for multivariate time series with missing values". In: *Scientific reports* 8.1 (2018), pp. 1–12.

[51] Francois Chollet et al. *Keras*. https://keras.io. 2015.

[52] Joseph Andrew Clarke and JLM Hensen. "Integrated building performance simulation: Progress, prospects and requirements". In: *Building and Environment* 91 (2015), pp. 294–306.

[53] William L Clarke. "The original Clarke error grid analysis (EGA)". In: *Diabetes technology & therapeutics* 7.5 (2005), pp. 776–779.

[54] David A Cohn. "Neural network exploration using optimal experiment design". In: *Advances in neural information processing systems*. 1994, pp. 679–686.

[55] Wesley J Cole et al. "Reduced-order residential home modeling for model predictive control". In: *Energy and Buildings* 74 (2014), pp. 69–77.

[56] William Colglazier. "Sustainable development agenda: 2030". In: *Science* 349.6252 (2015), pp. 1048–1050.

[57] Communication from the commission to the European Parliament, the Council, the European Economic and Social committee and the committee of the Regions. *Addressing the challenge of energy efficiency through Information and Communication Technologies*. 2008.

[58] Jerome Connor, Les E Atlas, and Douglas R Martin. "Recurrent networks and NARMA modeling". In: *Advances in Neural Information Processing Systems*. 1992, pp. 301–308.

[59] Daniel J Cox et al. *Understanding error grid analysis*. 1997.

[60] Drury B Crawley et al. "EnergyPlus: creating a new-generation building energy simulation program". In: *Energy and buildings* 33.4 (2001), pp. 319–331.

[61] Jochen L. Cremer et al. "Optimal Scheduling of Heat Pumps for Power Peak Shaving and Customers Thermal Comfort". In: *SMARTGREENS*. 2017, pp. 23–34. ISBN: 978-989-758-241-7. DOI: 10.5220/0006305800230034.

[62] Chiara Dalla Man, Michael Camilleri, and Claudio Cobelli. "A system model of oral glucose absorption: validation on gold standard data". In: *IEEE Transactions on Biomedical Engineering* 53.12 (2006), pp. 2472–2478.

[63] Mark Deakin and Husam Al Waer. "From intelligent to smart cities". In: *Intelligent Buildings International* 3.3 (2011), pp. 140–152.

[64] Ali Deihimi and Hemen Showkati. "Application of echo state networks in short-term electric load forecasting". In: *Energy* 39.1 (2012), pp. 327–340.

[65] Howard B Demuth et al. *Neural network design*. Martin Hagan, 2014.

[66] Kun Deng et al. "Building thermal model reduction via aggregation of states". In: *American Control Conference (ACC), 2010*. IEEE. 2010, pp. 5118–5123.

[67] Maimouna Diagne et al. "Review of solar irradiance forecasting methods and a proposition for small-scale insular grids". In: *Renewable and Sustainable Energy Reviews* 27 (2013), pp. 65–76.

[68] Abbott Diabetes Care Division. *WELCOME TO THE FOREFRONT OF DIABETES CARE*. 2020. URL: http://www.diabetescare.abbott/ (visited on 12/12/2018).

[69] Ömer Altan Dombaycı and Mustafa Gölcü. "Daily means ambient temperature prediction using artificial neural network method: A case study of Turkey". In: *Renewable Energy* 34.4 (2009), pp. 1158–1161.

[70] Xin Dong, Shangyu Chen, and Sinno Pan. "Learning to prune deep neural networks via layer-wise optimal brain surgeon". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4857–4867.

[71] Georg Dorffner. "Neural Networks for Time Series Processing". In: *Neural Network World* 6 (1996), pp. 447–468.

[72] Souradeep Dutta, Taisa Kushner, and Sriram Sankaranarayanan. "Robust Data-Driven Control of Artificial Pancreas Systems Using Neural Networks". In: *International Conference on Computational Methods in Systems Biology*. Springer. 2018, pp. 183–202.

[73] David Duvenaud, Dougal Maclaurin, and Ryan Adams. "Early stopping as nonparametric variational inference". In: *Artificial Intelligence and Statistics*. 2016, pp. 1070–1077.

[74] EarthSense Systems Limited. *Earthsense*. Available: https://www.earthsense.co.uk/. 2020.

[75] Charles M Eastman et al. *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors*. John Wiley & Sons, 2011.

[76] Thom J Eguia et al. "General parameterized thermal modeling for high-performance microprocessor design". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 20.2 (2012), pp. 211–224.

[77] Bryan Eisenhower and Igor Mezić. "Extracting dynamic information from whole-building energy models". In: *ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers. 2012, pp. 3–10.

[78] Laya Ekhlaspour et al. "Comparative accuracy of 17 point-of-care glucose meters". In: *Journal of diabetes science and technology* 11.3 (2017), pp. 558–566.

[79] European Parliament. *Directive 2010/31/EU of the European Parliament and of the Council of 19 May 2010 on the energy performance of buildings*. 2010.

[80] F.M. Ugliotti, M. Dellosta, A. Osello. "BIM-based energy analysis using Edilclima EC770 plugin. Case study: Archimede Library, EEB Project". In: *World Multidisciplinary Civil Engineering-Architecture-Urban Planning Symposium 2016* 161 (2016), pp. 3–8.

[81] Etimad Fadel et al. "A survey on wireless sensor networks for smart grid". In: *Computer Communications* 71 (2015), pp. 22–33.

[82] P.O. Fanger. *Thermal comfort: analysis and applications in environmental engineering*. McGraw-Hill, 1970. ISBN: 9780070199156. URL: https://books.google.it/books?id=mUFSAAAAMAAJ.

[83] Poul O Fanger. *Thermal comfort. Analysis and applications in environmental engineering*. Copenhagen: Danish Technical Press., 1970.

[84] Ricardo Faria et al. "Smart mobility: A survey". In: *2017 International Conference on Internet of Things for the Global Community (IoTGC)*. IEEE. 2017, pp. 1–8.

[85] BWAC Farley and W Clark. "Simulation of self-organizing systems by digital computer". In: *Transactions of the IRE Professional Group on Information Theory* 4.4 (1954), pp. 76–84.

[86] Muhammad Fayaz and DoHyeun Kim. "A Prediction Methodology of Energy Consumption Based on Deep Extreme Learning Machine and Comparative Analysis in Residential Buildings". In: *Electronics* 7.10 (2018), p. 222.

[87] Pieter J Fourie et al. "Modeling and design of smart mobility systems". In: *Urban Systems Design*. Elsevier, 2020, pp. 163–197.

[88] Guido Freckmann et al. "System accuracy evaluation of 27 blood glucose monitoring systems according to DIN EN ISO 15197". In: *Diabetes technology & therapeutics* 12.3 (2010), pp. 221–231.

[89] 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.

[90] Adiwinata Gani et al. "Predicting subcutaneous glucose concentration in humans: data-driven glucose modeling". In: *IEEE Transactions on Biomedical Engineering* 56.2 (2009), p. 246.

[91] Adiwinata Gani et al. "Universal glucose models for predicting subcutaneous glucose concentration in humans". In: *IEEE Transactions on Information Technology in Biomedicine* 14.1 (2010), pp. 157–165.

[92] Hao Gao, Christian Koch, and Yupeng Wu. "Building information modelling based building energy modelling: A review". In: *Applied Energy* 238 (2019), pp. 320–343.

[93] Clark W Gellings and Kelly E Parmenter. "Demand-side management". In: *Energy Management and Conservation Handbook*. CRC Press, 2016, pp. 399–420.

[94] E. I. Georga et al. "Multivariate Prediction of Subcutaneous Glucose Concentration in Type 1 Diabetes Patients Based on Support Vector Regression". In: *IEEE Journal of Biomedical and Health Informatics* 17.1 (2013), pp. 71–81. ISSN: 2168-2194. DOI: 10.1109/TITB.2012.2219876.

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

[96] Maedeh Ghorbanian, Sarineh Hacopian Dolatabadi, and Pierluigi Siano. "Big data issues in smart grids: A survey". In: *IEEE Systems Journal* 13.4 (2019), pp. 4158–4168.

[97] Green Building XML (gbXML) Schema. *gbXML*. Available at: http://http://www.gbxml.org/. 2020.

[98] Klaus Greff et al. "LSTM: A search space odyssey". In: *IEEE transactions on neural networks and learning systems* 28.10 (2017), pp. 2222–2232.

[99] Christian A Gueymard. "A review of validation methodologies and statistical performance indicators for modeled solar radiation data: Towards a better bankability of solar projects". In: *Renewable and Sustainable Energy Reviews* 39 (2014), pp. 1024–1034.

[100] Anushri Gupta, Panos Panagiotopoulos, and Frances Bowen. "Leveraging Capabilities in Smart City Initiatives. The Case of London City Data". In: *EGOV-CeDEM-ePart 2018* (2018), p. 115.

[101] Isabelle Guyon and André Elisseeff. "An introduction to variable and feature selection". In: *Journal of machine learning research* 3.Mar (2003), pp. 1157–1182.

[102] Takoua Hamdi et al. "Accurate prediction of continuous blood glucose based on support vector regression and differential evolution algorithm". In: *Biocybernetics and Biomedical Engineering* 38.2 (2018), pp. 362–372.

[103] James Douglas Hamilton. *Time series analysis*. Vol. 2. Princeton university press Princeton, 1994.

[104] Song Han et al. "Learning both weights and connections for efficient neural network". In: *Advances in Neural Information Processing Systems*. 2015, pp. 1135–1143.

[105] Lars Kai Hansen and Morten With Pedersen. "Controlled growth of cascade correlation nets". In: *ICANN'94*. Springer, 1994, pp. 797–800.

[106] Per Christian Hansen. *Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion*. Vol. 4. Siam, 2005.

[107] Peter Harrop and Raghu Das. "Wireless sensor networks 2014-2024". In: *IDTechEx* (2014).

[108] Babak Hassibi and David G Stork. "Second order derivatives for network pruning: Optimal brain surgeon". In: *Advances in neural information processing systems*. 1993, pp. 164–171.

[109] Douglas M Hawkins. "The problem of overfitting". In: *Journal of chemical information and computer sciences* 44.1 (2004), pp. 1–12.

[110] Xiangdong He and Haruhiko Asada. "A new method for identifying orders of input-output models for nonlinear dynamic systems". In: *American Control Conference, 1993*. IEEE. 1993, pp. 2520–2523.

[111] Jaeb Center for Health Research (JCHR). *Diabetes Research Studies*. http://http://diabetes.jaeb.org/. [Online; accessed October-2020].

[112] Donald O Hebb. "The organization of behavior; a neuropsycholocigal theory." In: *A Wiley Book in Clinical Psychology*. (1949), pp. 62–78.

[113] Jan LM Hensen and Roberto Lamberts. *Building performance simulation for design and operation*. Routledge, 2012.

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

[115] Philip N Howard. "How big is the Internet of Things and how big will it get?" In: *The Brookings Institution* (2015).

[116] David H Hubel and Torsten N Wiesel. "Receptive fields and functional architecture of monkey striate cortex". In: *The Journal of physiology* 195.1 (1968), pp. 215–243.

[117] Rob J Hyndman. "Yule-Walker estimates for continuous-time autoregressive models". In: *Journal of Time Series Analysis* 14.3 (1993), pp. 281–296.

[118] IBM. *Green Horizons*. Available: https://www.research.ibm.com/green-horizons/interactive/. 2020.

[119] Dexcom Inc. *DEXCOM CONTINUOUS GLUCOSE MONITORING*. 2020. URL: http://www.dexcom.com/ (visited on 12/12/2018).

[120] Pierre Ineichen and Richard Perez. "A new airmass independent formulation for the Linke turbidity coefficient". In: *Solar Energy* 73 (Sept. 2002), pp. 151–157. DOI: 10.1016/S0038-092X(02)00045-2.

[121] Standard ISO 7730:1997(E). International Organization for Standardization, 1997.

[122] Kazufumi Ito and Bangti Jin. *Inverse problems: Tikhonov theory and algorithms*. World Scientific, 2015.

[123] Herbert Jaeger. "Echo state network". In: *Scholarpedia* 2.9 (2007), p. 2330.

[124] Marco Jahn, Edoardo Patti, and Andrea Acquaviva. "Smart Energy Efficient Buildings-A Living Lab Approach." In: *International Conference on Smart Grids and Green IT Systems (SMARTGREENS)*. 2013, pp. 171–176.

[125] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. "An empirical exploration of recurrent network architectures". In: *International Conference on Machine Learning*. 2015, pp. 2342–2350.

[126] Herman Kahn. *World economic development: 1979 and beyond*. Routledge, 2019.

[127] S Kaplanis and E Kaplani. "Stochastic prediction of hourly global solar radiation profiles". In: (2016).

[128] Bekir Karlik and A Vehbi Olgac. "Performance analysis of various activation functions in generalized MLP architectures of neural networks". In: *International Journal of Artificial Intelligence and Expert Systems* 1.4 (2011), pp. 111–122.

[129] Alboukadel Kassambara. *Practical guide to cluster analysis in R: Unsupervised machine learning*. Vol. 1. STHDA, 2017.

[130] Donghun Kim and James E Braun. "Reduced-order building modeling for application to model-based predictive control". In: *Proceedings of SimBuild* 5.1 (2012), pp. 554–561.

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

[132] Serkan Kiranyaz et al. "1D convolutional neural networks and applications: A survey". In: *arXiv preprint arXiv:1905.03554* (2019).

[133] Douglas Kline. "Methods for Multi-Step Time Series Forecasting with Neural Networks". In: Jan. 2004, pp. 226–250. DOI: 10.4018/978-1-59140-176-6.ch012.

[134] Douglas M Kline. "Methods for multi-step time series forecasting neural networks". In: *Neural networks in business forecasting*. IGI Global, 2004, pp. 226–250.

[135] Tibor Kmet and Maria Kmetova. "A 24h forecast of solar irradiance using echo state neural networks". In: *Proceedings of the 16th International Conference on Engineering Applications of Neural Networks (INNS)*. ACM. 2015, p. 6.

[136] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (May 2017), pp. 84–90. ISSN: 00010782. DOI: 10.1145/3065386. URL: http://dl.acm.org/citation.cfm?doid=3098997.3065386.

[137] Miroslav Kubat. "Artificial neural networks". In: *An Introduction to Machine Learning*. Springer, 2017, pp. 91–111.

[138] Ravi Kumar and Bharti Nagpal. "Analysis and prediction of crime patterns using big data". In: *International Journal of Information Technology* 11.4 (2019), pp. 799–805.

[139] USA NOAA Earth System Research Laboratory. *Meteorological Assimilation Data Ingest System*. https://madis.ncep.noaa.gov/. Accessed on: 2019-06-01.

[140] Giordano Lanzola et al. "Remote blood glucose monitoring in mHealth scenarios: a review". In: *Sensors* 16.12 (2016), p. 1983.

[141] Philippe Lauret et al. "A benchmarking of machine learning techniques for solar radiation forecasting in an insular context". In: *Solar Energy* 112 (2015), pp. 446–457.

[142] David R Legates and Gregory J McCabe. "A refined index of model performance: a rejoinder". In: *International Journal of Climatology* 33.4 (2013), pp. 1053–1056.

[143] Kenneth Levenberg. "A method for the solution of certain non-linear problems in least squares". In: *Quarterly of applied mathematics* 2.2 (1944), pp. 164–168.

[144] Duo Li et al. "Parameterized architecture-level dynamic thermal models for multicore microprocessors". In: *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 15.2 (2010), p. 16.

[145] Gang Li et al. "Echo state network with bayesian regularization for forecasting short-term power production of small hydropower plants". In: *Energies* 8.10 (2015), pp. 12228–12241.

[146] Kezhi Li et al. "Convolutional Recurrent Neural Networks for Blood Glucose Prediction". In: *CoRR* abs/1807.03043 (2018). arXiv: 1807.03043. URL: http://arxiv.org/abs/1807.03043.

[147] Lennart Ljung. "System identification". In: *Signal analysis and prediction*. Springer, 1998, pp. 163–173.

[148] The Dark Sky Company LLC. *Dark Sky API*. https://darksky.net/dev. Accessed on: 2019-06-01.

[149] Mantas Lukoševičius. "A practical guide to applying echo state networks". In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 659–686.

[150] M Lydia et al. "Linear and non-linear autoregressive models for short-term wind speed forecasting". In: *Energy Conversion and Management* 112 (2016), pp. 115–124.

[151] Somayya Madakam and Rajesh M Holmukhe. "Songdo Smart City: An Aerotropolis and a Ubiquitous City". In: *Big Data Analytics for Smart and Connected Cities*. IGI Global, 2019, pp. 278–298.

[152] Ata Madanchi et al. "Strong short-term non-linearity of solar irradiance fluctuations". In: *Solar Energy* 144 (2017), pp. 1–9.

[153] Spyros Makridakis and Steven C Wheelwright. "Adaptive filtering: An integrated autoregressive/moving average filter for time series forecasting". In: *Journal of the Operational Research Society* 28.2 (1977), pp. 425–437.

[154] Chiara Dalla Man et al. "The UVA/PADOVA type 1 diabetes simulator: new features". In: *Journal of diabetes science and technology* 8.1 (2014), pp. 26–34.

[155] Danilo P Mandic, Jonathon A Chambers, et al. *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. Wiley Online Library, 2001.

[156] Cindy Marling and Razvan Bunescu. "The OhioT1DM dataset for blood glucose level prediction". In: *The 3rd International Workshop on Knowledge Discovery in Healthcare Data, Stockholm, Sweden*. 2018.

[157] Donald W Marquardt. "An algorithm for least-squares estimation of nonlinear parameters". In: *Journal of the society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441.

[158] Gonçalo Marques and Rui Pitarma. "A Cost-Effective Air Quality Supervision Solution for Enhanced Living Environments through the Internet of Things". In: *Electronics* 8.2 (2019), p. 170.

[159] Luis Martín et al. "Prediction of global solar irradiance based on time series analysis: Application to solar thermal power plants energy production planning". In: *Solar Energy* 84.10 (2010), pp. 1772–1781.

[160] John Martinsson et al. "Blood Glucose Prediction with Variance Estimation Using Recurrent Neural Networks". In: *Journal of Healthcare Informatics Research* (2019), pp. 1–18.

[161] Marco Massano et al. "A Grey-box Model Based on Unscented Kalman Filter to Estimate Thermal Dynamics in Buildings". In: *IEEE 19th International Conference on Environment and Electrical Engineering (EEEIC 19)*. IEEE. 2019.

[162] Valerie Masson-Delmotte et al. "IPCC, 2018: Summary for Policymakers". In: *Global Warming of* $1.5°C$. *An IPCC Special Report on the impacts of global warming of* $1.5°C$ *above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty* 1 (2018).

[163] Elizabeth J Mayer-Davis et al. "Incidence trends of type 1 and type 2 diabetes among youths, 2002–2012". In: *New England Journal of Medicine* 376.15 (2017), pp. 1419–1429.

[164] Leopold Mba, Pierre Meukam, and Alexis Kemajou. "Application of artificial neural network for predicting hourly indoor air temperature and relative humidity in modern building in humid region". In: *Energy and Buildings* 121 (2016), pp. 32–42.

[165] TC McCandless, SE Haupt, and GS Young. "A regime-dependent artificial neural network technique for short-range solar irradiance forecasting". In: *Renewable Energy* 89 (2016), pp. 351–359.

[166] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[167] Siamak Mehrkanoon. "Deep shared representation learning for weather elements forecasting". In: *Knowledge-Based Systems* 179 (June 2019), pp. 120–128. DOI: 10.1016/j.knosys.2019.05.009.

[168] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. "Regularizing and optimizing LSTM language models". In: *arXiv preprint arXiv:1708.02182* (2017).

[169] Hrushikesh N Mhaskar, Sergei V Pereverzyev, and Maria D van der Walt. "A deep learning approach to diabetic blood glucose prediction". In: *Frontiers in Applied Mathematics and Statistics* 3 (2017), p. 14.

[170] Tomas Mikolov et al. "Learning longer memory in recurrent neural networks". In: *arXiv preprint arXiv:1412.7753* (2014).

[171] Geoffrey F Miller, Peter M Todd, and Shailesh U Hegde. "Designing Neural Networks using Genetic Algorithms." In: *ICGA*. Vol. 89. 1989, pp. 379–384.

[172] Marvin Minsky and Seymour Papert. "An introduction to computational geometry". In: *Cambridge tiass., HIT* (1969).

[173] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[174] Stéphanie Monjoly et al. "Hourly forecasting of global solar radiation based on multiscale decomposition methods: A hybrid approach". In: *Energy* 119 (2017), pp. 288–298.

[175] Pedro Monteiro et al. "Indoor Temperature Prediction in an IoT Scenario". In: *Sensors* 18.11 (2018), p. 3610.

[176] Douglas C Montgomery, Cheryl L Jennings, and Murat Kulahci. *Introduction to time series analysis and forecasting*. John Wiley & Sons, 2015.

[177] Giorgio Mustafaraj, Gordon Lowry, and Jie Chen. "Prediction of room temperature and relative humidity by autoregressive linear and nonlinear neural network models for an open office". In: *Energy and Buildings* 43.6 (2011), pp. 1452–1460.

[178] H Nazaripouya et al. "Univariate time series prediction of solar power using a hybrid wavelet-ARMA-NARX prediction method". In: *Transmission and Distribution Conference and Exposition (T&D), 2016 IEEE/PES*. IEEE. 2016, pp. 1–5.

[179] USA NCEP. *Canadian Meteorological Center ensemble model*. https://nomads.ncep.noaa.gov/txt_descriptions/CMCENS_doc.shtml. Accessed on: 2019-06-01.

[180] John Nikoletatos and Stathis Tselepis. "Renewable Energy Integration in Power Grids". In: *Technology Brief. IEA-ETSAP and IRENA* (2015).

[181] Nils J. Nilsson. "Artificial intelligence: A modern approach". eng. In: *Artificial Intelligence* 82.1-2 (1996), pp. 369–380. ISSN: 00043702.

[182] USA NOAA. *Global Forecast System*. https://www.emc.ncep.noaa.gov/index.php?branch=GFS. Accessed on: 2019-06-01.

[183] USA NOAA. *Integrated Surface Database*. https://www.ncdc.noaa.gov/isd. Accessed on: 2019-06-01.

[184] Magnus Norgaard, Ole Ravn, and Niels Kjo lstad Poulsen. "NNSYSID-toolbox for system identification with neural networks". In: *Mathematical and computer modelling of dynamical systems* 8.1 (2002), pp. 1–20.

[185] Peter Magnus Norgaard et al. "Neural Networks for Modelling and Control of Dynamic Systems-A Practitioner's Handbook". In: (2000).

[186] Bogdan Oancea and Ştefan Cristian Ciucu. "Time series forecasting using neural networks". In: *arXiv preprint arXiv:1401.1333* (2014).

[187] German Meteorological Office. *Icosahedral Nonhydrostatic model*. https://www.dwd.de/EN/research/weatherforecasting/num_modelling/01_num_weather_prediction_modells/icon_description.html. Accessed on: 2019-06-01.

[188] Christopher Olah. "Understanding lstm networks". In: (2015).

[189] World Health Organization. *How air pollution is destroying our health*. 2018. URL: https://www.who.int/air-pollution/news-and-events/how-air-pollution-is-destroying-our-health (visited on 06/10/2019).

[190] Silvia Oviedo et al. "A review of personalized blood glucose prediction strategies for T1DM patients". In: *International journal for numerical methods in biomedical engineering* 33.6 (2017), e2833.

[191] F Pacheco-Torgal. "Eco-efficient construction and building materials research under the EU Framework Programme Horizon 2020". In: *Construction and building materials* 51 (2014), pp. 151–162.

[192] Cesar C Palerm et al. "Hypoglycemia prediction and detection using optimal estimation". In: *Diabetes technology & therapeutics* 7.1 (2005), pp. 3–14.

[193] Song Pan et al. "Energy waste in buildings due to occupant behaviour". In: *Energy Procedia* 105 (2017), pp. 2233–2238.

[194] Scott M Pappada, Brent D Cameron, and Paul M Rosman. "Development of a neural network for prediction of glucose concentration in type 1 diabetes patients". In: *Journal of diabetes science and technology* 2.5 (2008), pp. 792–801.

[195] Scott M Pappada et al. "Neural network-based real-time prediction of glucose in patients with insulin-dependent diabetes". In: *Diabetes technology & therapeutics* 13.2 (2011), pp. 135–141.

[196] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks". In: *International Conference on Machine Learning*. 2013, pp. 1310–1318.

[197] Edoardo Patti, Andrea Acquaviva, and Enrico Macii. "Enable sensor networks interoperability in smart public spaces through a service oriented approach". In: *Advances in Sensors and Interfaces (IWASI), 2013 5th IEEE International Workshop on*. IEEE. 2013, pp. 2–7.

[198] Edoardo Patti et al. "Event-driven user-centric middleware for energy-efficient buildings and public spaces". In: *IEEE Systems Journal* 10.3 (2016), pp. 1137–1146.

[199] Edoardo Patti et al. "IoT Software Infrastructure for Remote Monitoring of Patients with Chronic Metabolic Disorders". In: *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE. 2018, pp. 311–317.

[200] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[201] Hugo TC Pedro and Carlos FM Coimbra. "Assessment of forecasting techniques for solar power production with no exogenous inputs". In: *Solar Energy* 86.7 (2012), pp. 2017–2028.

[202] Carmen Pérez-Gandía et al. "Artificial neural network algorithm for online glucose prediction from continuous glucose monitoring". In: *Diabetes technology & therapeutics* 12.1 (2010), pp. 81–88.

[203] John C Pickup, Melissa Ford Holloway, and Kritika Samsi. "Real-time continuous glucose monitoring in type 1 diabetes: a qualitative framework analysis of patient narratives". In: *Diabetes care* 38.4 (2015), pp. 544–550.

[204] Medtronic plc. *Medtronic*. 2020. URL: https://www.medtronic.com/us-en/index.html (visited on 12/12/2018).

[205] Kevin Plis et al. "A machine learning approach to predicting blood glucose levels for diabetes management." In: *AAAI Workshop: Modern Artificial Intelligence for Health Analytics*. 31. 2014, pp. 35–39.

[206] Lutz Prechelt. "Early stopping-but when?" In: *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.

[207] Project TABULA. Available at: http://episcope.eu/index.php?id=97/. 2020.

[208] Atika Qazi et al. "The artificial neural network for solar radiation prediction and designing solar systems: a systematic literature review". In: *Journal of Cleaner Production* 104 (2015), pp. 1–12.

[209] Habiballah Rahimi-Eichi et al. "Battery management system: An overview of its application in the smart grid and electric vehicles". In: *IEEE Industrial Electronics Magazine* 7.2 (2013), pp. 4–16.

[210] Sumedha Rajakaruna, Farhad Shahnia, and Arindam Ghosh. *Plug in electric vehicles in smart grids*. Springer, 2016.

[211] Rajesh Rajamani. "Observers for Lipschitz nonlinear systems". In: *IEEE transactions on Automatic Control* 43.3 (1998), pp. 397–401.

[212] Mashud Rana, Irena Koprinska, and Vassilios G Agelidis. "Univariate and multivariate methods for very short-term solar photovoltaic power forecasting". In: *Energy Conversion and Management* 121 (2016), pp. 380–390.

[213] M Mazhar Rathore et al. "Urban planning and building smart cities based on the internet of things using big data analytics". In: *Computer Networks* 101 (2016), pp. 63–80.

[214] Priyanka Rawat et al. "Wireless sensor networks: a survey on recent developments and potential synergies". In: *The Journal of supercomputing* 68.1 (2014), pp. 1–48.

[215] Gordon Reikard. "Predicting solar radiation at high resolutions: A comparison of time series forecasts". In: *Solar Energy* 83.3 (2009), pp. 342–349.

[216] Christoph F Reinhart and Carlos Cerezo Davila. "Urban building energy modeling–A review of a nascent field". In: *Building and Environment* 97 (2016), pp. 196–202.

[217] Claude Rochet. *Smart Cities: Reality or Fiction*. John Wiley & Sons, 2018.

[218] David Rodbard. "Continuous glucose monitoring: a review of successes, challenges, and opportunities". In: *Diabetes technology & therapeutics* 18.S2 (2016), S2–3.

[219] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[220] Antonio E Ruano et al. "Prediction of building's temperature using neural networks models". In: *Energy and Buildings* 38.6 (2006), pp. 682–694.

[221] Stephen M Ruffing and Ganesh K Venayagamoorthy. "Short to medium range time series prediction of solar irradiance using an echo state network". In: *Intelligent System Applications to Power Systems, 2009. ISAP'09. 15th International Conference on*. IEEE. 2009, pp. 1–6.

[222] Alan R Saltiel and C Ronald Kahn. "Insulin signalling and the regulation of glucose and lipid metabolism". In: *Nature* 414.6865 (2001), p. 799.

[223] Benjamin Schrauwen, David Verstraeten, and Jan Van Campenhout. "An overview of reservoir computing: theory, applications and implementations". In: *Proceedings of the 15th european symposium on artificial neural networks. p. 471-482 2007*. 2007, pp. 471–482.

[224] A Sfetsos and AH Coonick. "Univariate and multivariate forecasting of hourly solar radiation with artificial intelligence techniques". In: *Solar Energy* 68.2 (2000), pp. 169–178.

[225] Aaron Shapiro. "Predictive Policing for Reform? Indeterminacy and Intervention in Big Data Policing". In: *Surveillance & Society* 17.3/4 (2019), pp. 456–472.

[226] Pierluigi Siano. "Demand response and smart grids—A survey". In: *Renewable and Sustainable Energy Reviews* 30 (2014), pp. 461–478.

[227] Hava T Siegelmann, Bill G Horne, and C Lee Giles. "Computational capabilities of recurrent NARX neural networks". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 27.2 (1997), pp. 208–215.

[228] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. "A review of supervised machine learning algorithms". In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. Ieee. 2016, pp. 1310–1315.

[229] Christopher J Smith, Jamie M Bright, and Rolf Crook. "Cloud cover effect of clear-sky index distributions and differences between human and automatic cloud observations". In: *Solar Energy* 144 (2017), pp. 10–21.

[230] Antti Sorjamaa et al. “Methodology for long-term prediction of time series”. In: *Neurocomputing* 70.16-18 (2007), pp. 2861–2869.

[231] G. Sparacino et al. “Glucose Concentration can be Predicted Ahead in Time From Continuous Glucose Monitoring Sensor Time-Series”. In: *IEEE Transactions on Biomedical Engineering* 54.5 (2007), pp. 931–937. ISSN: 0018-9294. DOI: 10.1109/TBME.2006.889774.

[232] Donald F Specht et al. “A general regression neural network”. In: *IEEE transactions on neural networks* 2.6 (1991), pp. 568–576.

[233] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[234] Shikhar Srivastava and Stefan Lessmann. “A comparative study of LSTM neural networks in forecasting day-ahead global horizontal irradiance with satellite data”. In: *Solar Energy* 162 (2018), pp. 232–247.

[235] International Organization for Standardization. *ISO 15197: 2013. In vitro diagnostic test systems–Requirements for blood-glucose monitoring systems for self-testing in managing diabetes mellitus*. 2013.

[236] Ralf Steuer et al. “The mutual information: detecting and evaluating dependencies between variables”. In: *Bioinformatics* 18.suppl_2 (2002), S231–S240.

[237] STMicroelectronics. *SPIRIT1 - Low data rate, low power sub-1GHz transceiver*. Available: https://www.st.com/resource/en/datasheet/spirit1.pdf. 2020.

[238] STMicroelectronics. *STM32 Nucleo-64 boards*. Available: https://www.st.com/resource/en/data_brief/nucleo-f401re.pdf. 2020.

[239] Qingnan Sun et al. “Predicting Blood Glucose with an LSTM and Bi-LSTM Based Deep Neural Network”. In: *2018 14th Symposium on Neural Networks and Applications (NEUREL)*. IEEE. 2018, pp. 1–5.

[240] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[241] Ahmed Tealab, Hesham Hefny, and Amr Badr. “Forecasting of nonlinear time series using artificial neural network”. In: *Future Computing and Informatics Journal* (2017).

[242] Georg Thimm and Emile Fiesler. *Pruning of neural networks*. Tech. rep. IDIAP, 1997.

[243] A Trnsys. “Transient System Simulation Program”. In: *University of Wisconsin* (2000).

[244] John Twidell and Tony Weir. *Renewable energy resources*. Routledge, 2015.

[245] U.S. Department of Energy's (DOE) Building Technologies Office (BTO). *EnergyPlus*. Available at: https://energyplus.net/. 2020.

[246] Naciye Güliz Uğur and Aykut Hamit Turan. "Understanding Big Data". In: *Big Data Analytics for Sustainable Computing*. IGI Global, 2020, pp. 1–29.

[247] United Nations, FCCC. *Adoption of the Paris Agreement. Proposal by the President*. Available: http://unfccc.int/resource/docs/2015/cop21/eng/l09r01.pdf. 2015.

[248] United States General Services Administration, GSA Building Information Modeling Guide Series 05 – Energy Performance. Version 2.0. Available at: www.gsa.gov/bim/. 2020.

[249] Vittorio Verda et al. "Thermal peak load shaving through users request variations". In: *International Journal of Thermodynamics* 19.3 (2016), pp. 168–176.

[250] Michal Vesely and Wim Zeiler. "Personalized conditioning and its impact on thermal comfort and energy performance–A review". In: *Renewable and Sustainable Energy Reviews* 34 (2014), pp. 401–408.

[251] Cyril Voyant et al. "Bayesian rules and stochastic models for high accuracy prediction of solar radiation". In: *Applied Energy* 114 (2014), pp. 218–226.

[252] Cyril Voyant et al. "Machine learning methods for solar radiation forecasting: A review". In: *Renewable Energy* 105 (2017), pp. 569–582.

[253] Cyril Voyant et al. "Optimization of an artificial neural network dedicated to the multivariate forecasting of daily global radiation". In: *Energy* 36.1 (2011), pp. 348–359.

[254] Sam Weckx and Johan Driesen. "Load balancing with EV chargers and PV inverters in unbalanced distribution grids". In: *IEEE Transactions on Sustainable Energy* 6.2 (2015), pp. 635–643.

[255] Andreas S Weigend. *Time series prediction: forecasting the future and understanding the past*. Routledge, 2018.

[256] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. "A survey of transfer learning". In: *Journal of Big data* 3.1 (2016), p. 9.

[257] Paul Werbos. "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences". In: *Ph. D. dissertation, Harvard University* (1974).

[258] Jurgen Willems, Joachim Van den Bergh, and Stijn Viaene. "Smart city projects and citizen participation: The case of London". In: *Public sector management in a globalized world*. Springer, 2017, pp. 249–266.

[259] Cort J Willmott, Scott M Robeson, and Kenji Matsuura. "A refined index of model performance". In: *International Journal of Climatology* 32.13 (2012), pp. 2088–2094.

[260] Ian H Witten et al. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[261] Jonathan Woetzel et al. "Smart cities: Digital solutions for a more livable future". In: *Mc Kinsey Global Institute* (2018).

[262] LT Wong and WK Chow. "Solar radiation model". In: *Applied Energy* 69.3 (2001), pp. 191–224.

[263] Xi Xie et al. "Reduction of measurement noise in a continuous glucose monitor by coating the sensor with a zwitterionic polymer". In: *Nature Biomedical Engineering* (2018), p. 1.

[264] Hao Xing et al. "Decentralized optimal scheduling for charging and discharging of plug-in electric vehicles in smart grids". In: *IEEE Transactions on Power Systems* 31.5 (2016), pp. 4118–4127.

[265] Chengliang Xu et al. "Improving prediction performance for indoor temperature in public buildings based on a novel deep learning method". In: *Building and Environment* 148 (2019), pp. 128–135.

[266] Donna Xu et al. "Survey on Multi-Output Learning". In: *IEEE transactions on neural networks and learning systems* (2019).

[267] Amit Kumar Yadav and SS Chandel. "Solar radiation prediction using Artificial Neural Network techniques: A review". In: *Renewable and Sustainable Energy Reviews* 33 (2014), pp. 772–781.

[268] Nitish Yadav et al. "City crime mapping using machine learning techniques". In: *International Conference on Advanced Machine Learning Technologies and Applications*. Springer. 2019, pp. 656–668.

[269] Izzet B Yildiz, Herbert Jaeger, and Stefan J Kiebel. "Re-visiting the echo state property". In: *Neural networks* 35 (2012), pp. 1–9.

[270] Danilo Yu et al. "Predicting indoor temperature from smart thermostat and weather forecast data". In: *Proceedings of the Communications and Networking Symposium*. Society for Computer Simulation International. 2018, p. 9.

[271] Hao Yu and Bogdan M Wilamowski. "Levenberg-marquardt training". In: *Industrial electronics handbook* 5.12 (2011), p. 1.

[272] Lingxi Zhang, Nicholas Good, and Pierluigi Mancarella. "Building-to-grid flexibility: Modelling and assessment metrics for residential demand response from heat pump aggregations". In: *Applied energy* 233 (2019), pp. 709–723.

[273] Chunhui Zhao et al. “Online prediction of subcutaneous glucose concentration for type 1 diabetes using empirical models and frequency-band separation”. In: *AIChE Journal* 60.2 (2014), pp. 574–584.

[274] Hengyang Zhao et al. “Learning based compact thermal modeling for energy-efficient smart building management”. In: *Computer-Aided Design (ICCAD), 2015 IEEE/ACM International Conference on*. IEEE. 2015, pp. 450–456.

[275] J. Zheng, D. W. Gao, and L. Lin. “Smart Meters in Smart Grid: An Overview”. In: *2013 IEEE Green Technologies Conference (GreenTech)*. 2013, pp. 57–64. DOI: 10.1109/GreenTech.2013.17.

[276] Taiyu Zhu et al. “A Deep Learning Algorithm For Personalized Blood Glucose Prediction”. In: *Proc. of the International Workshop on Knowledge Discovery in Healthcare Data*. Vol. 2148. 2018, pp. 1–5.

[277] Roland Zimmermann. *easyesn*. https://github.com/kalekiu/easyesn. 2017.

[278] Ahmed Zoha et al. “Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey”. In: *Sensors* 12.12 (2012), pp. 16838–16866.

Author's Publication List

Below is the complete list of research papers published during the PhD. The list is divided by categories and in chronological descending order. Each document provides a brief description of the work carried out, the novelty introduced and the author's contributions.

Journals

[1] *A Multi-Patient Data Driven Approach to Blood Glucose Prediction* / **Aliberti, Alessandro**; Pupillo, Irene; Terna, Stefano; Macii, Enrico; Di Cataldo, Santa; Patti, Edoardo; Acquaviva, Andrea. - In: IEEE ACCESS. - ISSN 2169-3536. - 7:1(2019), pp. 69311-69325.

In this paper, I dealt with the design and the implementation of an universal prediction model trained on glucose signals of a large and heterogeneous cohort of patients and then applied to infer future glucose level values on a completely new patient. Therefore, I designed a non-linear autoregressive (NAR) neural network and on long short-term memory (LSTM). Then, I compare these solutions with three literature approaches, respectively, based on feed-forward neural network (FNNs), autoregressive (AR) model, and recurrent neural network (RNN). While the NAR obtained good prediction accuracy only for short-term predictions (i.e., with prediction horizon within 30 min), the LSTM obtained extremely good performance both for short- and long-term glucose-level inference (60 min and more), overcoming all the other methods in terms of correlation between the measured and the predicted glucose signal and in terms of clinical outcome.

[2] *A Non-Linear Autoregressive Model for Indoor Air-Temperature Predictions in Smart Buildings* / **Aliberti, Alessandro**; Bottaccioli, Lorenzo; Macii, Enrico; Di Cataldo, Santa; Acquaviva, Andrea; Patti, Edoardo. - In: ELECTRONICS. - ISSN 2079-9292. - 8:9(2019), pp. 979-995.

In this paper, I designed an innovative methodology based on Internet-of-Things (IoT) technology for smart building indoor air-temperature forecasting. In detail, I exploited a specialized non-linear autoregressive neural network for short- and medium-term predictions, envisioning two different exploitation: i) on realistic artificial data and ii) on real data collected by IoT devices deployed in the building. Therefore, I designed and optimized four neural models, focusing respectively on three characterizing rooms and on the whole building. Experimental results on both a simulated and a real sensors data-set demonstrate the prediction accuracy and robustness of proposed models, by providing accurate predictions with time horizons in the order of 3 hours for individual rooms and 4 hours for the entire building.

Chapters Book

[1] *Non-linear Autoregressive Neural Networks to Forecast Short-Term Solar Radiation for Photovoltaic Energy Predictions* / **Aliberti, Alessandro**; Bottaccioli, Lorenzo; Cirrincione, Giansalvo; Macii, Enrico; Acquaviva, Andrea; Patti, Edoardo. - Communications in Computer and Information Science, 992(2019), pp. 3-22.

In this work, I propose an innovative methodology for implementing two different non-linear autoregressive neural networks to forecast Global Horizontal Solar Irradiance (GHI) in short-term time periods (i.e. from future 15 to 120 min). Both neural networks have been implemented, trained and validated exploiting a dataset consisting of four years of solar radiation values collected by a real weather station. We also present the experimental results discussing and comparing the accuracy of both neural networks. Then, the resulting GHI forecast is given as input to a Photovoltaic simulator to predict energy production in short-term time periods.

Proceedings in International Conferences

[1] *Data Driven Patient-Specialized Neural Networks for Blood Glucose Prediction* / **Aliberti, Alessandro**; Bagatin, Andrea; Acquaviva, Andrea; Macii, Enrico; Patti, Edoardo. - (2020), pp. 1-6. - 2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW 2020) - London, United Kingdom, United Kingdom, 6-10 July 2020.

In this work, I dealt with the design and the implementation of a patient-specialized prediction model for future values of the glycaemic concentration. Thus, I designed a specialized solution based on Long Short-Term Memory (LSTM) neural network. Then, I compared LSTM solution with two literature approaches, respectively based on Feed-Forward (FNN) and Recurrent (RNN) neural networks. The experimental results have highlighted that our LSTM solution obtained good performance both for short- and long-term glucose level inference (60 min.), overcoming the other methods both in terms of correlation between measured and predicted glucose signal and in terms of clinical outcome.

[2] *Indoor Air-Temperature Forecast for Energy-Efficient Management in Smart Buildings* / **Aliberti, Alessandro**; Ugliotti, Francesca Maria; Bottaccioli, Lorenzo; Cirrincione, Giansalvo; Osello, Anna; Macii, Enrico; Patti, Edoardo; Acquaviva, Andrea. - 18th IEEE International Conference on Environment and Electrical Engineering (EEEIC) - Palermo, Italy, 12-15 June 2018.

In this paper, I designed and implemented an innovative methodology for smart building indoor air-temperature forecasting. This methodology is based on a non-linear autoregressive neural network. This neural network has been trained and validated with a dataset consisting of six years indoor air-temperature values of a

building demonstrator. In detail, I have studied three characterizing rooms and the whole building. The analysis of performance indicators highlighted an overall good performance in predicting temperature values up to two hours.

[3] *Forecasting short-term solar radiation for photovoltaic energy predictions* / **Aliberti, Alessandro**; Bottaccioli, Lorenzo; Cirrincione, Giansalvo; Macii, Enrico; Acquaviva, Andrea; Patti, Edoardo. - (2018), pp. 44-53. - 7th Conference on Smart Cities and Green ICT Systems (SMARTGREENS 2018) - Funchal, Madeira, Portugal nel 16 - 18 March 2018.

In this paper, I designed and implemented an innovative methodology for short-term (e.g. 15 minutes) forecasting of Global Horizontal Solar Irradiance (GHI). The proposed methodology is based on a non-linear autoregressive neural network. This neural network has been trained and validated with a dataset consisting of solar radiation samples collected for four years by a real weather station. Then GHI forecast, the output of the neural network, is given as input to a Photovoltaic simulator (previously developed by my research group) to predict energy production in short-term time periods. The analysis of performance indicators highlighted an overall good performance in predicting the solar radiation, especially for the next 15, 30 and 45 minutes

[4] *Building energy modelling and monitoring by integration of IoT devices and Building Information Models* / Bottaccioli, Lorenzo; **Aliberti, Alessandro**; Ugliotti, Francesca Maria; Osello, Anna; Macii, Enrico; Patti, Edoardo; Acquaviva, Andrea. - 01(2017), pp. 914-922. - 41st IEEE Annual Computer Software and Applications Conference (COMPSAC 2017) - Torino, Italy, 4-8 July 2017.

In this paper, together with my colleagues, we present a software architecture for management and simulation of energy behaviours in buildings that integrates heterogeneous data such as BIM, IoT, GIS (Geographical Information System) and meteorological services. This integration allows: *i)* (near-) real-time visualisation of energy consumption information in the building context and *ii)* building performance evaluation through energy modelling and simulation exploiting data from the field and real weather conditions. With respect to literature solutions that consider TMY weather data, our results highlighted that the integration of real weather information into the simulation process strongly increases the accuracy of the simulation itself.

[5] *A Participatory Design Approach for Energy-Aware Mobile App for Smart Home Monitoring* / **Aliberti, Alessandro**; Camarda, Christian; Ferro, Valeria; Acquaviva, Andrea; Patti, Edoardo. - (2017), pp. 158-165. - 6th Conference on Smart Cities and Green ICT Systems (SMARTGREENS 2017) - Porto, Portugal, 22-24 April, 2017.

In this paper, I presented the participatory design approach that I followed to design and develop an energy-aware mobile application for user-awareness on energy consumption for Smart Home monitoring. To engage end-users from the early design stages, in an European project, I conduct two on-line surveys and a focus group involving about 630 people. Results allowed on identifying functional requirements and guidelines for mobile app design. The purpose of this research is to increase user-awareness on energy consumption using tools and methods required by users themselves.

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was pdfLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.