SAPIENZA
UNIVERSITÀ DI ROMA

Machine Learning for the Sustainable Energy Transition: a Data-Driven Perspective along the Value Chain from Manufacturing to Energy Conversion

Department of Astronautical, Electrical and Energy Engineering
PhD in Energy and Environment (XXXV cycle)

Eric Stefan Miele
ID number 1643696

Advisor
Prof. Alessandro Corsini

Co-Advisor
Dr. Fabrizio Bonacina

Academic Year 2022/2023

Thesis defended on 25 May 2023
in front of a Board of Examiners composed by:

Lucia Fontana (chairman)
Fulvio Palmieri
Marco Sabatini

Machine Learning for the Sustainable Energy Transition: a Data-Driven Perspective along the Value Chain from Manufacturing to Energy Conversion
PhD thesis. Sapienza University of Rome

© 2023 Eric Stefan Miele. All rights reserved

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: ericstefan.miele@uniroma1.it

This thesis is dedicated to friendship. It might sound like a broad and abstract concept but it has never been so meaningful and concrete to me. The past years have been a complete rollercoaster. Physically, mentally, and emotionally. Nevertheless, good people supported me, making this thesis and this journey possible. Old immovable pillars were there to give structure during deep quakes. First, I want to thank my brothers Mastro and Jules, who heard me out countless times and have been soul companions while facing the strongest waves. Words are useless to describe what you mean to me. It is inevitable to mention mom and dad as the foundations of everything. Always there, no matter what, no matter when, no matter where. From the highest peak to the lowest chasm, they were always there. I am so grateful for my family. I want to thank Gioia and Christian, who have always been ready to share my enthusiasm for everything and fuelled my creativity. Books, movies, games, and art have been so important to me during this journey and having someone with whom I could share all of this has been a blessing. I want to thank my childhood friend Kyle with whom I recently reconnected after many years through a falafel at Kichererbse or a hike into the misty mountains. May the future hold more adventures for us together. I am grateful for newly blossomed friendships. Florian, Jakob, Ronja, Nina, Jannik, Gwen, Sara, Ludwig. From concerts to musicals, from ramen to homemade Käsespätzle, from dragons to vampires, I treasure these memories as a pirate would do with his shiniest gems. All of you made this journey something truly special, something that changed my perspective, way of living, and way of being forever. It is unavoidable to thank Victor and Rebecca, adventurers like me in a foreign country. With Victor, we shared our first experience living abroad. We explored with curiosity, from Leipzig to forest walks, and shared the enthusiasm for a new chapter in our lives. Rebecca has been, even if for just a glimpse, part of what I and Victor call the owl's nest. Lovecraftian and gloomy themes, curious and artsy, nerdy and enthusiastic, we could be the same person in parallel universes, as you once said. I want to deeply thank those who became a new family and made me feel at home at any time. Julian, Elias, Rene, Viki, Paula, and Katha. Living together with you all has been just amazing. From Hellsing to David Lynch, from jam sessions to tea tasting sessions, from music to politics. These moments are chiseled in my memory like the stars are chiseled in the dark sky during a summer night along the Ammer. Finally, I want to give special thanks to Alessandro, Nicole, and Fabrizio, who made this journey possible. Alessandro opened the doors to the realm of energy systems and impactful applications and provided me with the tools to read the world through different lenses. During this journey, he became a reference point and has been the biggest promoter of my technical and, more importantly, personal growth. As Alessandro opened the doors in Rome, Nicole opened the doors in Tübingen, allowing me to dive into a new dimension. She warmly welcomed me into her research group and always supported my proposals with constructive feedback. Fabrizio has been more than a colleague during this journey. He has been a reference from day zero all the way until the last day. We worked on almost every project together and exchanged ideas on a daily basis. We grew together and learned a lot from each other. But, most importantly, I remember the laughs, stories we told each other, and deep talks about life. I treasure the knowledge of this journey with the hope of making this world a better place, even if just by a tiny bit. But above everything else, I treasure the journey itself.

Abstract

According to the special report *Global Warming of* $1.5°C$ of the IPCC, climate action is not only necessary but more than ever urgent. The world is witnessing rising sea levels, heat waves, events of flooding, droughts and desertification resulting in the loss of lives and damage to livelihoods, especially in countries of the Global South. To mitigate climate change and commit to the Paris agreement, it is of the uttermost importance to reduce GreenHouse Gas emissions coming from the most emitting sector, namely the energy sector. To this end, large-scale penetration of Renewable Energy Systems into the energy market is crucial for the energy transition toward a sustainable future by replacing fossil fuels and improving access to energy with socio-economic benefits. With the advent of Industry 4.0, Internet of Things technologies have been increasingly applied to the energy sector introducing the concept of smart grid or, more in general, Internet of Energy. These technologies are steering the energy sector towards more efficient, reliable, flexible, resilient, safe, and sustainable solutions with huge environmental and potential social benefits. To realize these solutions, new information technologies are required and among the most promising possibilities are Artificial Intelligence and Machine Learning which in many countries have already revolutionized the energy industry. This thesis presents different Machine Learning algorithms and methods for the implementation of new strategies to make Renewable Energy Systems more efficient and reliable. It presents various learning algorithms, highlighting their advantages and limits, and evaluates their application for different tasks in the energy context. In addition, different techniques are presented for the preprocessing and cleaning of time series data, nowadays collected by sensor networks mounted on every Renewable Energy System. With the possibility to install large numbers of sensors that collect vast amounts of data, it is vital to detect and remove irrelevant, redundant, or noisy features, and alleviate the curse of dimensionality, thus improving the interpretability of predictive models, speeding up their learning process, and enhancing their generalization properties. Therefore, this thesis discusses the importance of dimensionality reduction in sensor networks mounted on Renewable Energy Systems and, to this end, presents two novel unsupervised algorithms. The first approach maps time series data in the network domain through visibility graphs and uses a community detection algorithm to identify clusters of similar variables and select representative parameters. This method can group both homogeneous and heterogeneous physical parameters, even when related to different functional areas of a system. The second approach proposes the Combine Predictive Power Score, a method for feature selection with a multivariate formulation that explores multiple sub-sets of expanding variables and identifies the combination of features with the highest predictive power over specified target variables. This method proposes a selection algorithm for the optimal combination of variables that converges to the smallest set of predictors with the highest predictive power. Once the combination of variables is identified, the most relevant parameters in a sensor network can be selected to perform dimensionality reduction. Data-driven methods open the possibility of supporting strategic decision-making, resulting in a reduction of Operation and Maintenance costs, machine faults, repair stops, and spare parts inventory size. Therefore, this thesis presents two approaches

in the context of predictive maintenance to improve the lifetime and efficiency of the equipment based on anomaly detection algorithms. The first approach proposes an anomaly detection model based on Principal Component Analysis that is robust to false alarms, can isolate anomalous conditions, and can anticipate equipment failures. The second approach has at its core a neural architecture, namely a Graph Convolutional Autoencoder, which models the sensor network as a dynamical functional graph by simultaneously considering the information content of individual sensor measurements (graph node features) and the nonlinear correlations existing between all pairs of sensors (graph edges). The proposed neural architecture can capture hidden anomalies in wind farms even when the turbines continue to deliver the power requested by the grid and can anticipate equipment failures. Since the model is unsupervised and completely data-driven, this approach can be applied to any wind turbine equipped with a SCADA system. When it comes to Renewable Energys, the unschedulable uncertainty due to their intermittent nature represents an obstacle to the reliability and stability of energy grids, especially when dealing with large-scale integration. Nevertheless, these challenges can be alleviated if the natural sources or the power output of Renewable Energy Systems can be forecasted accurately, allowing power system operators to plan optimal power management strategies to balance the dispatch between intermittent power generation and load demand. To this end, this thesis proposes a multi-modal spatio-temporal neural network for multi-horizon wind power forecasting. In particular, the model combines high-resolution Numerical Weather Prediction forecast maps with turbine-level SCADA data and explores how meteorological variables on different spatial scales, together with the turbines' internal operating conditions impact wind power forecasts. The world is undergoing a third energy transition with the main goal of tackling global climate change through decarbonization of the energy supply and consumption patterns. This is not only possible thanks to global cooperation and agreements between parties, power generation systems advancements, and Internet of Things and Artificial Intelligence technologies but also necessary to prevent the severe and irreversible consequences of climate change that are threatening life on the planet as we know it. This thesis is intended as a reference for researchers that want to contribute to the sustainable energy transition and are approaching the field of Artificial Intelligence in the context of Renewable Energy Systems.

Contents

Chapter 1

Introduction

1.1 Sustainable Energy Transition

According to the special report *Global Warming of* $1.5°C$ of the Intergovernmental Panel on Climate Change (IPCC), climate change impacts are worse than expected, and climate action is not only necessary but also urgent [1]. More recently, the IPCC report of 2021 indicated that several climate changes are already irreversible and mostly caused by anthropogenic activities [2].
Modern society is heavily based on the combustion of fossil fuels for electricity and heat generation in the energy and related sectors, manufacturing activities, industrial operations, transportation, agriculture, forestry, and the building industry [3]. This leads, inevitably, to the production and emission of GreenHouse Gases (GHGs) like Carbon Dioxide (CO2), Methane (CH4), Sulphurdioxide (SO2), Nitrous Oxides (NOx), which are the main causes of the global warming that is threatening the very existence of humanity. It is well known that climate is important to mankind and other living organisms on planet Earth, yet there is overwhelming evidence of changing climatic conditions due to the greenhouse effect as demonstrated by the increase in global average surface temperature, rising sea levels, heat waves, events of flooding, droughts and desertification [4].
The greenhouse effect is a natural process that has taken place over millions of years and consists in the long-term increase of the planet's temperature resulting from the interaction between solar energy and GHGs accumulated in the atmosphere [5]. These gasses can absorb and radiate thermal energy and, therefore, are necessary to keep the global temperature above the freezing point, warm the planet, and maintain life [6]. Nevertheless, global GHG concentrations have been growing drastically during the last century due to anthropogenic activities, threatening life itself.
The overabundance of GHGs in the atmosphere is responsible for re-absorbing the solar energy radiating from the earth's surface back out to space. The trapped heat energy and radiations are then radiated back to Earth, heating both the lower atmosphere and the planet's surface. This has led to serious long-term disruptions to global weather and climate systems which are all interconnected and finely balanced. As a result, extreme weather events and disasters have become more and more frequent. Moreover, climate changes have created severe problems for plant and animal species around the globe, some of which have not been able to adapt fast

enough and are facing the risk of extinction [7].
The greenhouse effect was discovered by Jean Baptiste-Joseph de Fourier in 1827, demonstrated experimentally by John Tyndall in 1861, and quantified by Svante Arrhenius in 1896. Then, in 1957, Roger Revelle and Hans Suess remarked that the accumulation of CO2 in the atmosphere should be monitored and controlled because it represents a large-scale geophysical experiment whose consequences were unknown [8]. Therefore, in 1958 Charles Keeling started the ongoing program of continuous measurements of atmospheric CO2 levels at Mauna Loa (Hawaii) in the United States, which demonstrated the steady rising of CO2 levels during the second half of the last century [4]. Although CO2 is less potent than CH4 or NOx, it is more abundant, lingers longer in the atmosphere, and, therefore, is responsible for about two-thirds of the total temperature rise. Additionally, when dissolved in ocean water, CO2 reacts with water molecules and produces carbonic acid which lowers the PH of the body of water. This has already caused a 30% increase in the acidity of the ocean since the beginning of the industrial revolution, leading to biological effects that interfere with marine life [6].
Figure 1.1 shows the global CO2 emissions and concentrations from 1750 to 2022. It is important to note that global emissions remained constant between 1750 and 1840, after which they started to increase rapidly. The atmospheric concentration slightly increased between 1750 and 1960, after which they strongly grew because of the industrial activities powered by fossil fuels. It is evident that the industrialization process triggered a rapid increase in CO2 levels, leading to a stronger greenhouse effect, global warming, and, ultimately, climate change of anthropogenic nature [7]. Human activities have led to about $1°C$ rise in average global temperature above pre-industrial levels which are further projected to reach $1.5°C$ between the year 2030 and 2052 if current GHG emission rates remain unaltered [9].

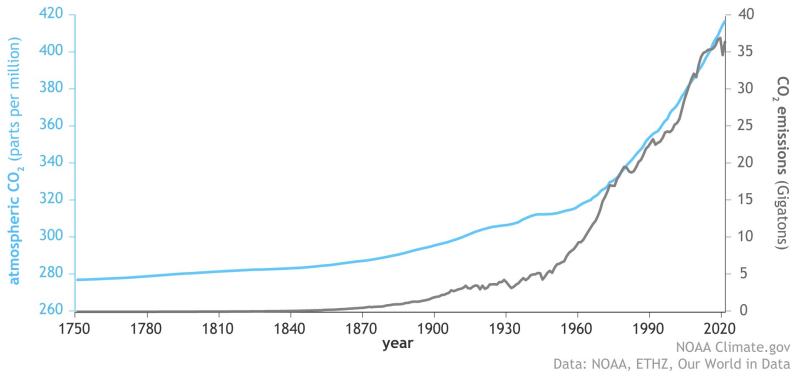


Figure 1.1. Concentration of carbon dioxide emissions between the year 1750 and 2020 [6].

The *United Nations Conference on the Environment* held in Stockholm, Sweden, in 1972 was the first world conference to address the environment as a major issue. Participants adopted several principles for sound environmental management including the *Stockholm Declaration* and the *Action Plan for the Human Environment*. The *Stockholm Declaration* placed environmental issues as a major international concern and laid the foundation of dialogue between industrialized and developing

countries on the connection between economic growth, pollution, and human well-being. One of the most important outcomes of the Stockholm conference was the creation of the United Nations Environment Programme (UNEP) [10].
Then, in 1992, *Agenda 21*, the *Rio Declaration on Environment and Development*, and the *Statement of Principles for the Sustainable Management of Forests* were adopted by more than 178 Governments at the United Nations Conference on Environment and Development (UNCED), also known as Earth Summit, held in Rio de Janeiro, Brazil. *Agenda 21* is a comprehensive plan of global, national, and regional actions to be taken by the United Nations (UN) system, governments and major groups in all areas of human impact on the environment. The Commission on Sustainable Development (CSD) was created in December 1992 to ensure effective follow-up of UNCED, to monitor and report on the implementation of the agreements [11]. The World Summit on Sustainable Development (WSSD) held in Johannesburg, South Africa, in 2002 strongly reaffirmed its commitment to the full implementation of *Agenda 21*, the *Program for Further Implementation of Agenda 21*, and the *Rio Principles* [12]. During the Earth Summit, the United Nations Framework Convention on Climate Change (UNFCCC) was signed by 154 states to combat dangerous human interference with the climate system.
In 1997, 192 parties adopted the *Kyoto Protocol*, an international treaty that extends the UNFCCC by committing countries to limit and reduce GHG emissions under agreed individual targets. The *Kyoto Protocol* was based on the principle of common but differentiated responsibilities, acknowledging that individual countries have different capabilities in fighting climate change due to economic developments. Therefore, it placed a heavier burden on developed countries to reduce emissions since historically responsible for the current levels of GHGs in the atmosphere [13]. In December 2012, the *Doha Amendment* to the *Kyoto Protocol* was adopted in Doha, Qatar, for a second commitment period, starting in 2013 and lasting until 2020 [14].
In 2015, the *Paris Agreement* was negotiated and adopted by 196 parties at the 21st United Nations Climate Change Conference of Parties (COP21) near Paris, France. The *Paris Agreement* is a milestone in the multilateral climate change process because, for the first time, a binding agreement led all nations to a common purpose to make ambitious efforts to combat climate change and adapt to its impacts. The ambition in the agreement is to maintain the increase in global average temperature to well below $2°C$ above pre-industrial levels and pursue efforts to limit the temperature to $1.5°C$. The commitment to aim for $1.5°C$ degrees is important because every fraction of a degree of warming results in the loss of lives and damage to livelihoods [15]. The Paris Agreement works on a 5-year cycle of increasingly ambitious climate action carried out by countries. Delayed for a year due to the COVID-19 pandemic, in 2021 countries updated their plans for reducing emissions at the 26st United Nations Climate Change Conference of Parties (COP26), where it became clear that commitments laid out in Paris 6 years earlier did not come close to limiting global warming to $1.5°C$ degrees, and that the time window for achieving it is closing [16].
During the same year of the *Paris Agreement*, the *2030 Agenda for Sustainable Development* was adopted by all UN Member States, as a universal call to action to end poverty, protect the planet, and ensure that by 2030 humankind as a whole can

enjoy peace and prosperity. At its core are the 17 Sustainable Development Goals (SDGs), which are an urgent call for action by all countries in a global partnership. SDGs recognize that ending poverty and other deprivations must advance together with strategies that improve health and education, and reduce inequality, all while tackling climate change and working to preserve oceans, forests, and biodiversity [17].

To reduce GHG levels most effectively and understand which emissions can and cannot be eliminated with current technology, it is essential to first understand the different sources of emission. The diagram in Figure 1.2 shows the global GHG emissions by sector for the year 2016, where global emissions were 49.4 billion tonnes CO_2eq. About one-fifth comes from agriculture and land and 8% from industry and waste. The remaining three-quarters of global emissions are, instead, produced by the energy sector from a wide variety of sub-sectors, from power generation to industrial manufacturing [18].

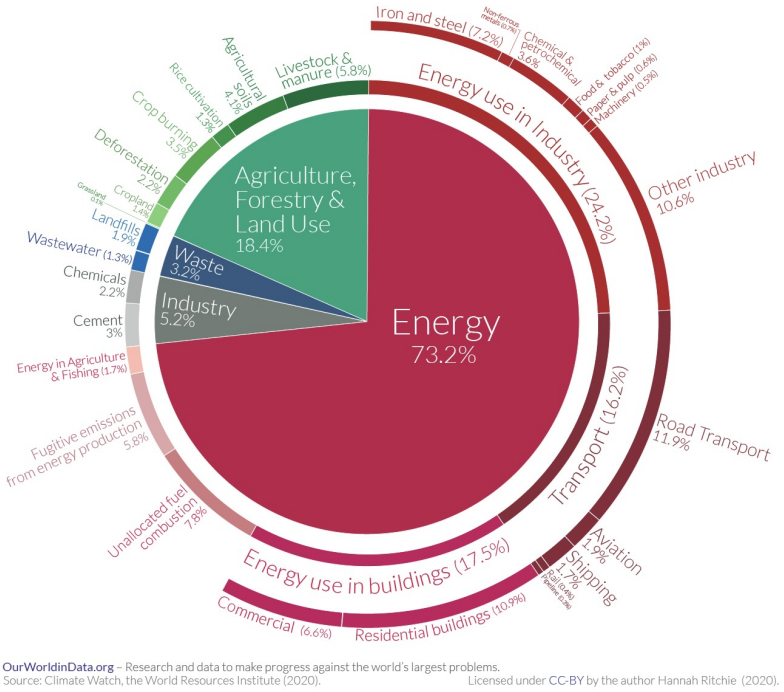


Figure 1.2. Global GHG emissions by sector, shown for the year 2016 where global GHG emissions were 49.4 billion tonnes CO_2eq [18].

Electricity is essential in modern society as it accounts for an increasing share of energy production and consumption in all countries, and plays a vital role in all important human activities and operations, either directly or indirectly [19]. Electricity demand has been growing over the past decades and is expected to continue increasing due to rising household incomes, electrification of transport, and increasing thermal energy applications, as well as continued growth in digitally connected appliances and air conditioning [20]. In response, also the global energy supply has been increasing steadily over the past years, as shown in Figure 1.3 where the world total energy supply is plotted over time by source, from 1971 to 2019. More specifically, the share of the world's total energy supply more than doubled

during the past 50 years, as reported in Figure 1.4, and the share of energy produced by fossil fuels accounted for more than 80% of the total supply in 2019, making the energy sector a leading source of GHG emissions causing global warming [21].

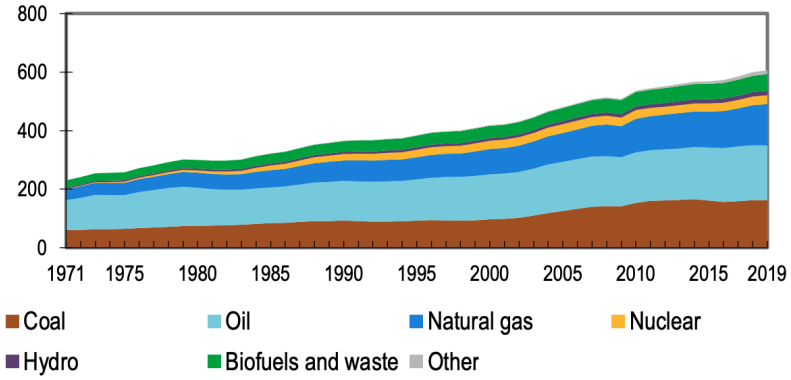


Figure 1.3. World total energy supply by source, 1971-2019 (EJ) [21].

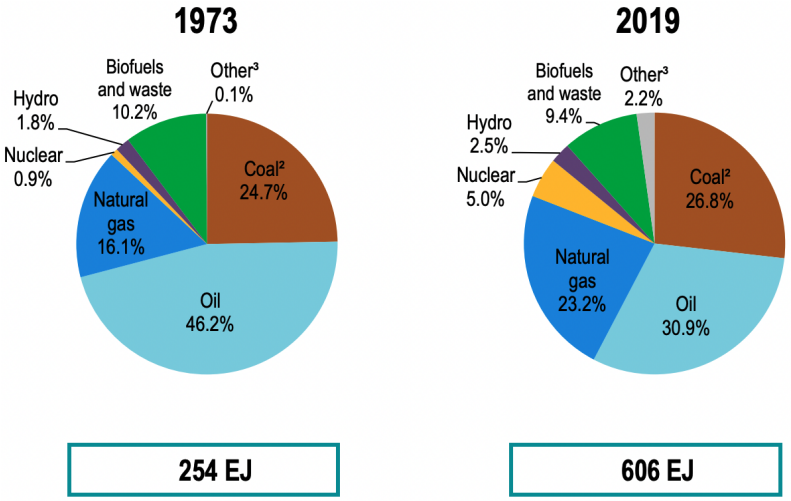


Figure 1.4. Share of world total energy supply by source, 1973 and 2019 [21]. Notably, it includes international aviation and international marine bunkers.

To mitigate climate change and protect the planet from irreversible consequences, it is evident that part of the solution is to reduce emissions coming from the energy sector. This can be achieved by deploying a large-scale Renewable Energy (RE) supply in energy systems and corresponds to the SDG no. 7 accounting for affordable and clean energy to substantially increase the share of RE by 2030. Moreover, achieving 100% Renewable Energy System (RES) would also contribute to the fulfillment of SDG no. 6 (clean water and sanitation), no. 9 (industry, innovation, and infrastructure), no. 11 (sustainable cities and communities), no. 12 (responsible production and consumption) and no. 13 (climate action) [22].
RE sources are derived from naturally occurring sources that replenish themselves through natural forces. As an inexhaustible source of clean energy, RE sources play a key role in the energy transition towards a sustainable future [23]. Not only

RE sources can help mitigate GHG emissions, climate change, and environmental pollution, but can also improve access to energy for a large part of the population since locally available, and contribute to local socio-economic benefits through job creation and improved local economies [24, 25].
With the increasing presence of distributed generation systems that are mainly based on variable RE sources, centralized power transmission and distribution networks that were initially conceived and designed to distribute electricity from central power stations to consumers are no longer valid. Economies of scale are starting to adapt to new distributed generation technologies that have increased the viability of small-scale energy systems, and the use of information technology has created new opportunities for a less hierarchical and more flexible energy and infrastructure management model [26]. Decentralized small-scale energy systems like Energy Community (EC) are becoming more and more important as a resilient answer to the global energy crisis characterizing the current historical period, which is witnessing the COVID-19 pandemic emergency and the war in Ukraine [27, 28].
The technical feasibility of 100% RE systems has been extensively scrutinized by more than 180 publications and is gaining momentum among various stakeholders [22]. Examples are Sweden, where the goal is to achieve net zero GHG emissions by 2040, and Denmark with the target of net zero emissions by 2050 [29, 30]. In addition, many countries, such as Bangladesh, Barbados, Cambodia, Colombia, Ethiopia, Ghana, Mongolia, Vietnam, Hawaii, or California, are targeting 100% RE by 2045 or 2050 [31]. Already today, some countries, such as Norway and Costa Rica, derive their electricity almost exclusively from RE sources such as hydropower [31]. Similarly, several cities have committed to using 100% RE for all energy consumption by 2050. These cities include Copenhagen (2050), Denmark, Frankfurt and Hamburg (2050), Germany, Malmö and Växjö (2030), Sweden, Oxford Country (2050), Australia, Vancouver (2050), Canada, and The Hague (2040), Netherlands [31].
The world has experienced two major energy transitions in the past. The first replaced wood with coal, and, during the second, oil and gas replaced coal as the main energy source. Today, the world is undergoing a third energy transition with the main goal to tackle global climate change through decarbonization of the energy supply and consumption patterns [32]. For most of history, the idea of a broad-scale energy transition was unthinkable as decisions were made at the local, regional or individual level, with limited or no coordination [33]. Today, it is not only possible thanks to global cooperation and agreements between parties but also necessary to prevent the severe and irreversible consequences of climate change that are threatening life on the planet as we know it.

1.2 Renewable Energy Systems

In 2020, RE use increased by 3% as demand for all non-renewable sources decreased by more than 3%. The primary driver was an almost 7% growth in electricity generation from renewable sources, together with an overall decline in global electricity demand caused by the COVID-19 emergency. Specifically, the share of renewables in global electricity jumped from 27% in 2019 to 29% in 2020, led by wind and solar PhotoVoltaic (PV), which grew by 12% and 23%, respectively [34]. These two sources contributed to two-thirds of renewables expansion, reaching 10.7% of the global power mix in 2021, and are witnessing the fastest growth since the 1970s, as shown in Figure 1.5. Therefore, wind and solar PV will be discussed next as the main energy sources instrumental in the sustainable energy transition.

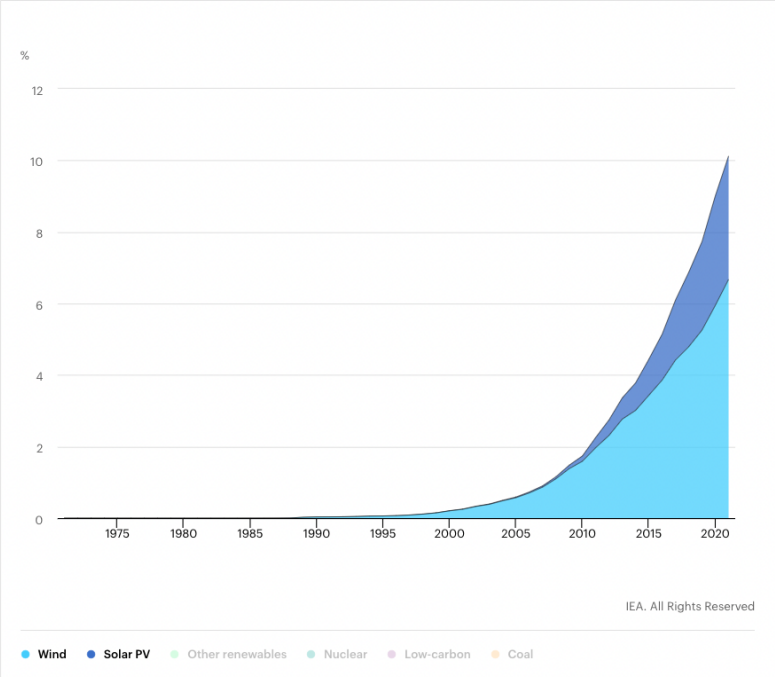


Figure 1.5. Share of wind and solar PV in world electricity generation, 1971-2021 [34].

Wind Energy

In 2021, wind energy contributed to 6.6% of the global electricity generation and is expected to continue growing with the possibility of becoming the main energy resource in the near future. Moreover, offshore wind capacity additions more than tripled in 2021 to nearly 56 GW [35].
In 1887, Professor James Bryce invented the world's first Wind Turbine (WT) to generate electricity from wind. Several months later, Charles F. Brish constructed the first Horizontal-Axis Wind Turbine (HAWT) which has become the most common type nowadays, making him the inventor of the modern WT industry. Then, in 1956, the first three-bladed turbine was built by Johannes Jul, namely, the Gedser WT, which greatly inspired the development of later technologies. Moreover, Jul

also designed a novel emergency aerodynamic tip break to slow the rotation of the rotor when the blade is subjected to an excessive amount of centrifugal force, laying the foundation for modern safety measures in WTs [36].
The development of WTs is progressing rapidly, and, in the past few decades, their capacity, rotor diameter and height have been steadily increasing, as shown in Figure 1.6. Currently, the biggest WT in the world is the offshore *MySE 16.0-242* turbine, produced by MingYang Smart Energy, with a capacity of 16 MW, a 242-meter diameter rotor, 118 m long blades, and a staggering 46000 m^2 swept area equivalent of more than six soccer fields [37]. Moreover, WTs are expected to grow even further in the near future, as depicted in Figure 1.7, with an estimate of median turbine capacity reaching 5.5 MW and 17 MW for onshore and offshore WTs, respectively, in 2035 [38].

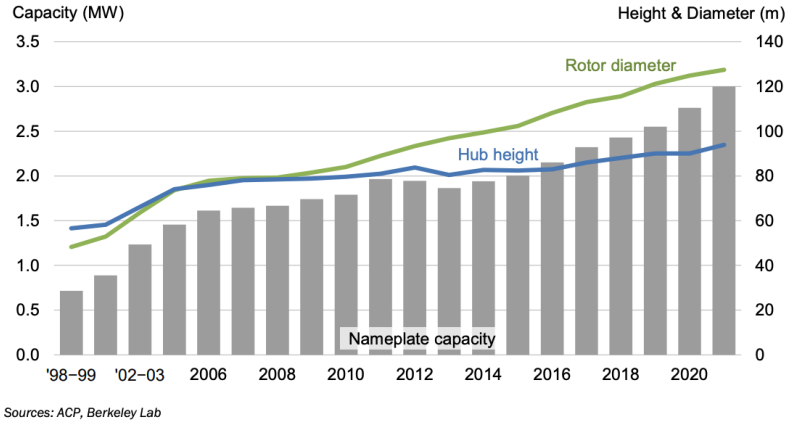


Figure 1.6. Average turbine nameplate capacity, hub height, and rotor diameter for land-based wind projects in the US [39].

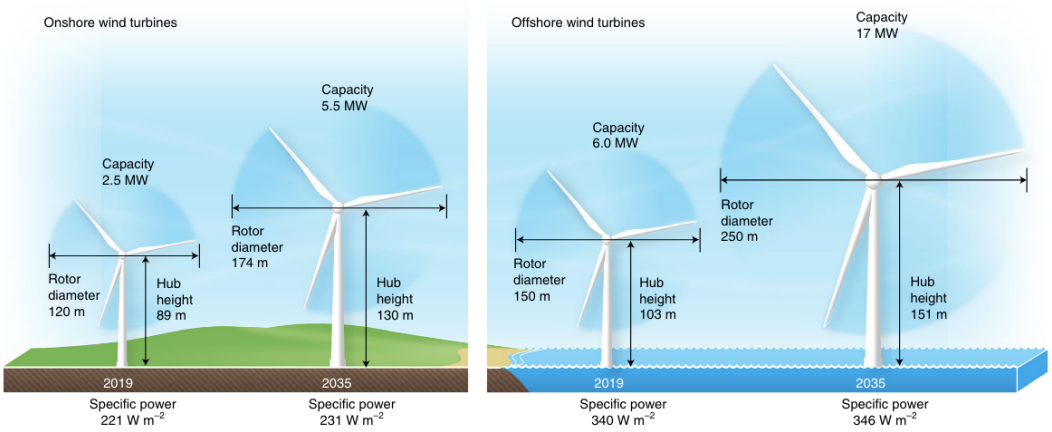


Figure 1.7. Expected turbine size in 2035 for onshore and offshore wind, compared with 2019 medians [38]. For offshore wind, a global 2019 median is provided, while for onshore wind, a US median is shown given the lack of data to estimate a global median.

When the wind blows through a WT, the blades absorb some of the kinetic energy and convert it into aerodynamic lift, creating a rotational momentum that

enables the turbine to spin. The curved design of the blades also referred to as airfoil shaped, deflects the airflow downwards, and, according to Newton's Third Law of Motion, creates an opposite force having two components, namely lift and drag, as shown in Figure 1.8. The lift force is responsible for propelling the blade, allowing it to rotate around the center of the turbine, called the hub. Notably, the force response of the blades is governed by the geometry of the flow, better known as the angle of attack α, and the relative speed of the wind. The rotating hub is connected to the main shaft inside the nacelle which is linked to an alternator for electricity generation.

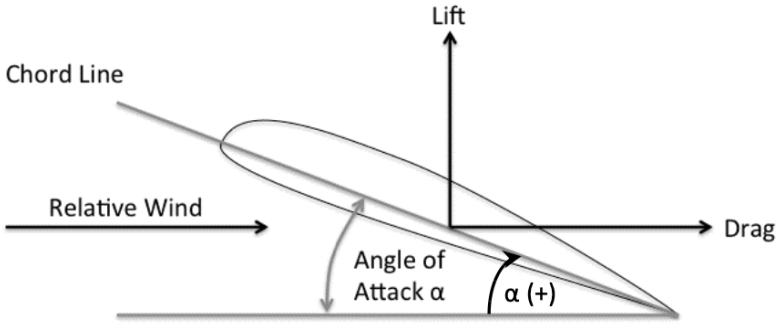


Figure 1.8. Lift and drag force of a WT airfoil [40].

More specifically, the main components of a HAWT, shown in Figure 1.9, are listed below.

1. *Rotor Blades*: the rotor blades are installed to provide angular momentum that enables the turbine to spin, and, ultimately, to generate electricity.

2. *Hub*: the hub connects the rotor blades to the main shaft and is a rotating component.

3. *Control Monitoring System*: also referred to as CMS, this system allows the operators to monitor the overall performance of the turbine, together with the status of all its components.

4. *Main Shaft*: the main shaft is composed of two components, namely the low-speed shaft and the high-speed shaft. The low-speed shaft is connected to the hub of the turbine and rotates at a lower speed compared to the high-speed shaft. The high-speed shaft is connected to the generator. The higher speed is required to increase the rotational speed of the magnetic field within the generator, which allows for higher speed power generation.

5. *Gearbox*: this component connects the low-speed and high-speed shafts and consists of multiple gears with different ratios that allow the rotating speed to increase.

6. *Disk Brake*: the braking system is used to stop the WT for routine operations or during emergencies. When strong winds blow, the turbine could spin too fast or even out of control, causing an overload to the generator which could

lead to an onboard fire. The disk brake is usually mounted on the high-speed shaft.

7. *Generator Coupling*: this component is used to transmit the rotational power from the high-speed shaft to the generator.

8. *Cooling Radiator*: the cooling radiator cools down the turbine to prevent overheating inside the nacelle.

9. *Wind Measuring System*: this system, usually composed of an anemometer and an air velocity meter, measures the wind speed and direction, allowing the WT to align into the wind to maximize the absorbed kinetic energy. Modern designs use digital measurements instead of conventional designs like, for example, the Doppler LiDAR system [41].

10. *Generator*: the generator is typically an alternator that converts mechanical energy to electrical energy in the form of alternating current. This is achieved by a rotating magnetic field produced by a rotor with a fixed armature coil. Specifically, the main shaft has multiple poles with permanent magnets and is surrounded by a static armature coil fitted outside the rotor. When spinning, the magnetic field intersects the armature coil and generates an altering current.

11. *Yaw Drive*: the yaw drive is responsible for the orientation of the rotor towards the wind.

12. *Hydraulic System*: the hydraulic system is used for multiple functions in a WT, which include blade pitch adjustment, yaw and rotor braking, cooling, and lubrication.

13. *Yaw Bearing*: the yaw bearing is one of the most crucial components in a WT since it has to support enormous static and dynamic loads during the operation of the turbine, providing smooth rotation when orienting the nacelle under all weather conditions.

14. *Tower*: the tower allows the blades to be elevated from ground level to an altitude where winds are usually faster and smoother. The tower is also used to support the whole system including the nacelle and all the other components.

The power that can be captured from the wind through a WT is equal to

$$P = \frac{1}{2}\rho_{air} C_p \pi r^2 v^3, \tag{1.1}$$

where ρ_{air} is the air mass density, C_p the power coefficient, r the rotor radius (or the blade length), and v the wind speed. Notably, the power coefficient depends on the specific design of the blade, the blade pitch angle θ, and the tip speed ratio (blade tip speed divided by wind speed). The maximum possible value for the power coefficient is $Cp = 0.593$, meaning that the efficiency of a WT can never exceed 59.3%. This is also referred to as the Betz Limit [43]. However, in practical designs, due to various forms of energy loss, the Betz Limit is never reached, resulting in efficiencies of around 30-40%.

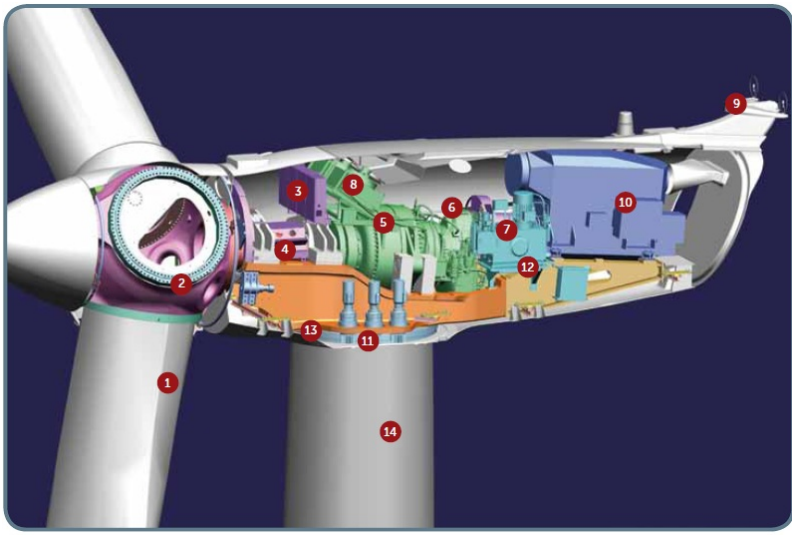


Figure 1.9. A cross-section of a HAWT showing its main internal components [42]. 1. Rotor Blades, 2. Hub, 3. Control System Monitoring, 4. Main Shaft, 5. Gearbox, 6. Disk Brake, 7. Generator Coupling, 8. Cooling Radiator, 9. Wind Measuring System, 10. Generator, 11. Yaw Drive, 12. Hydraulic System, 13. Yaw Bearing, 14. Tower.

Overall, WTs have little environmental impact compared to other traditional forms of energy. When operating, they emit no waste that pollutes the air or water and do not require water to cool the generator. Therefore, WTs play a key role in reducing the use of fossil fuels in power generation, thereby tackling pollution and mitigating GHG emissions.

Solar PV Energy

In 2021, solar energy was the fastest-growing RE contributing to 3.6% of the global electricity generation and is expected to continue growing in the future [35].
The PV effect was observed for the first time in 1839 by Edmond Becquerel through an electrode in a conductive solution exposed to light [44] and, in 1873, Willoughby Smith discovered the photoconductive potential of selenium. Three years later, in 1876, William Grylls Adams and Richard Evans Day observed that selenium creates electricity when exposed to sunlight [45, 46]. In 1883, the first solar cell made from selenium wafers was invented by Charles Fritts, attracting the attention of the scientific community and sparking further research and development, and, in 1904, Wilhelm Hallwachs made the first semiconductor-junction solar cell with copper and copper oxide [47, 48].
However, these discoveries were not based on a solid understanding of the science behind the operation of the first PV devices, and a theoretical foundation was formulated from 1905 to 1950. Key events were Einstein's photon theory in 1905, for which he later won a Nobel prize in physics, and, in 1918, Jan Czochralski's method to grow single crystals of metal, adapted decades later by Gordon Teal and John Little to produce single-crystalline germanium and, later, silicon [49, 50].

The solar era began in the 1950s when Bell Laboratory scientists focused on PV developments and began utilizing silicon to produce solar cells. This breakthrough is credited to Daryl Chapin, Calvin Fuller, and Gerald Pearson who created the first silicon PV cell in 1954 [51]. This led the US government to invest in solar cell technology from 1960 to 1980, first for applications on space satellites and then for initial terrestrial applications. Another driver for PV investments was the oil embargo in 1973 which drove the US to seek energy independence [52].
From the 1980s, solar technology developed also worldwide, and many companies around the globe, especially in East Asian countries, started producing utility-scale solar technology, leading to cost reduction, and efficiency improvements [48]. Nevertheless, solar energy generation became mainstream only in the mid-2010s, when costs declined by about 90% over the decade, as shown in Figure 1.10, allowing PV technology to become a deployable energy solution, also in the residential market [53].

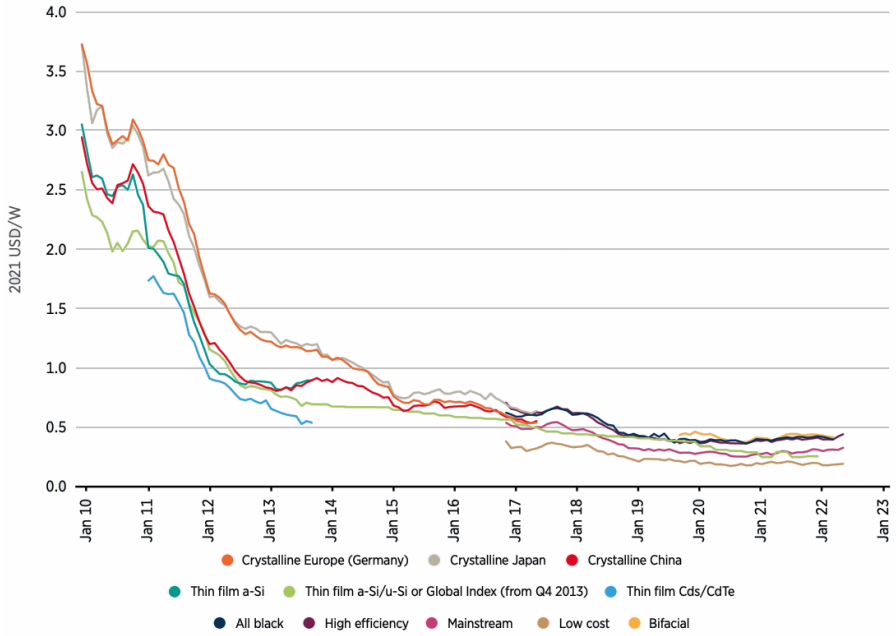


Figure 1.10. Average monthly solar PV module prices by technology and manufacturing country sold in Europe, 2010 to 2021 [53].

The typical solar cell is composed of two types of semiconductor layers called *p-type* and *n-type* silicon layers, obtained by introducing impurities into silicon crystals. The p-type silicon layer is made by doping silicon with atoms having one less electron in the outer energy level than silicon, like boron. By having one less electron than needed to form a bond with the surrounding silicon atoms, an electron vacancy or *hole* is created. The n-type silicon layer, instead, is made by doping silicon with atoms that have one more electron in the outer energy level than silicon, such as phosphorus. This last, for example, has five electrons in its outer energy levels and bonds with neighboring silicon atoms. However, during the bonding process, one electron does not participate and remains free to move within the silicon structure.

When stacked on top of each other, as shown in Figure 1.11, the excess of electrons in the n-type layer and the lack of electrons (holes) in the p-type layer result in electrons moving into holes near the *junction* of the two layers, also called *depletion zone*. When all the holes in the depletion zone are filled in with electrons, the p-type side of the depletion zone, originally having holes, will contain negatively charged atoms (ions), and the n-type side, originally having an excess of electrons, will contain positively charged ions. This results in an internal electric field that prevents electrons in the n-type layer to fill in holes in the p-type layer.
When rays of photons strike the surface of the PV cell, electrons are liberated from the neighboring silicon atoms and become free to move around within the semiconductor. The electron excitation causes them to become released from the previously formed bonds, allowing them to move through the junction, creating and filling in holes in the cell. As there are billions of photons striking the cell every second, plenty of electrons are knocked loose. When connecting the n-type and p-type layers with a metallic wire, the electrons can travel from the n-type layer to the p-type layer by crossing the depletion zone due to the collision with photons and, then, go back to the n-type layer through the external wire, creating a flow of electricity.

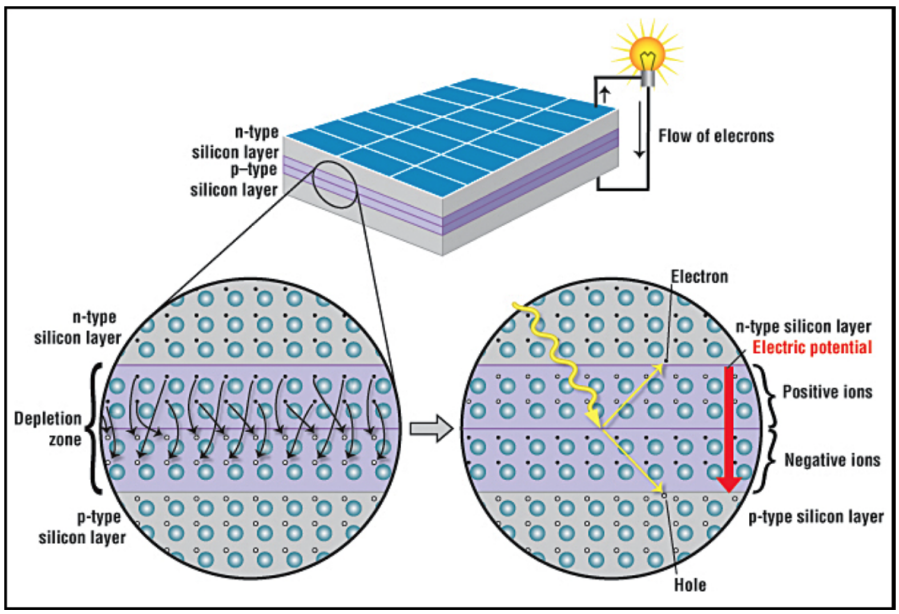


Figure 1.11. Schematic representation of a solar cell, showing the n-type and p-type layers, with a close-up view of the depletion zone around the junction between the n-type and p-type layers [54].

When assembling PV modules as a pre-wired, field-installable unit, it forms a *solar panel*, with a standard output voltage of $12V$ or $24V$. The area of modern solar panels ranges from $1.7m^2$ to $3.12m^2$, with power outputs from $300W$ to more than $680W$, as shown in Figure 1.12 [55]. Multiple solar panels form a complete power-generating unit named *solar array* which can be connected in series for a higher voltage and in parallel for higher currents to achieve the desired power output.

The efficiency of a solar PV panel is measured under Standard Test Conditions

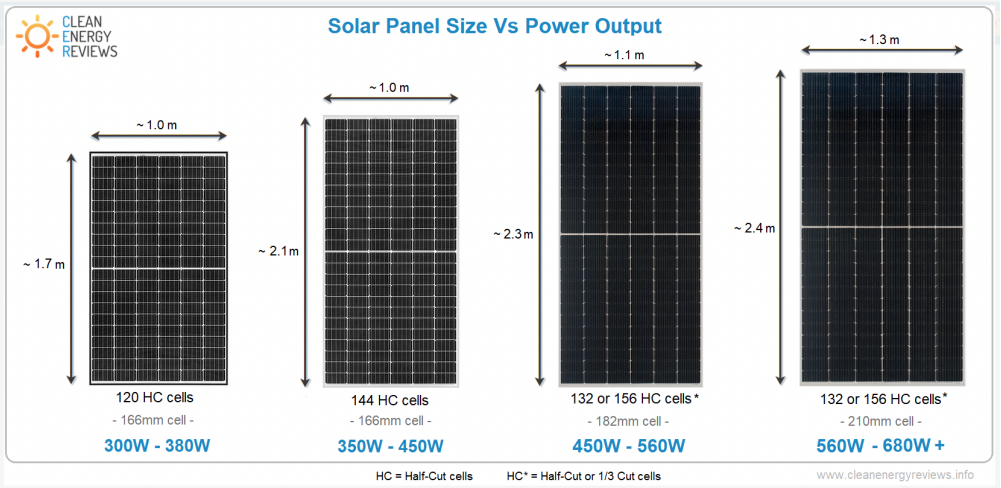


Figure 1.12. Modern solar panel sizes and power [55].

(STC), considering a cell temperature of $25°C$, solar irradiance of $1000\frac{W}{m^2}$ and air mass of 1.5, and represents the amount of sunlight the panel can turn into usable energy. Under STC, a $150W$ panel generates $150W$ of electricity when solar irradiance is $1000\frac{W}{m^2}$, and the panel is at $25°C$. Specifically, the efficiency of a panel is calculated as

$$\eta = \frac{P_{max}}{A \cdot E}, \tag{1.2}$$

where P_{max} is the maximum power rating at STC, A the area of the solar panel, and E the solar irradiance at STC, namely $1000\frac{W}{m^2}$. It is important to note the importance of the area of the solar panel. For example, a 10% efficient $100W$ panel and a 20% efficient $100W$ panel will both produce the same amount of power, even though the 20% efficient panel is half the total size of the 10% efficient panel. Overall, the efficiency of a solar panel is influenced by multiple factors like temperature, irradiance level, cell type and design, and interconnection between cells.

The very first solar cell built by Bell Laboratory scientists in 1954 had an efficiency of 6%. Then, between 1957 and 1960, several breakthroughs were made by Hoffman Electronics, increasing the efficiency from 8% to 14% [48], and, in 1985, the University of South Wales achieved 20% efficiency for silicon cells [56]. In 1999, Spectrolab, Inc. and the National Renewable Energy Laboratory developed a PV solar cell having 32.3% efficiency, and, later in 2016, broke that record again reaching an efficiency of 34.5% [57, 58]. The latest advancement in solar PV technology was achieved by NREL researchers who developed a six-junction III-V solar cell with an efficiency of 47.1% [59]. These efficiencies are based on measurements taken in clean laboratory environments under STC and do often not reflect what solar PV panels produce when installed on the field. Many studies demonstrated the actual outputs could be reduced by as much as 60% in a dusty or polluted climate without regular cleaning [60].

As for wind energy, solar PV technology provides clean and renewable energy with little environmental impact. When operating, PV systems do not pollute or emit GHGs, providing distributed power generation. Differently from wind energy

technologies, Operation and Maintenance (O&M) costs for solar PV are lower, having almost no mechanically moving parts, and a very modular design. Moreover, solar panels have enormous potential in residential areas since they are easy to install on rooftops without any interference with people's lifestyles. Other land-use efficient solutions have been proposed like agrovoltaic systems, where shaded space underneath solar panels is used to grow crops, making solar farms and agriculture share ground, rather than making them compete against one another [61, 62]. As for wind energy, also solar PV plays a key role in the ongoing energy transition towards a sustainable future.

1.3 From Internet of Things to Internet of Energy

In recent years, Internet of Things (IoT) has evolved into one of the most important technologies of the last century. Everyday objects such as cars, kitchen appliances, or thermostats, can now connect to the Internet through embedded devices, enabling seamless communication between people, processes, and, in general, *things*. Figure 1.13 shows the number of IoT devices connected over the last decade, with 50.1 billion devices in 2020 [63].

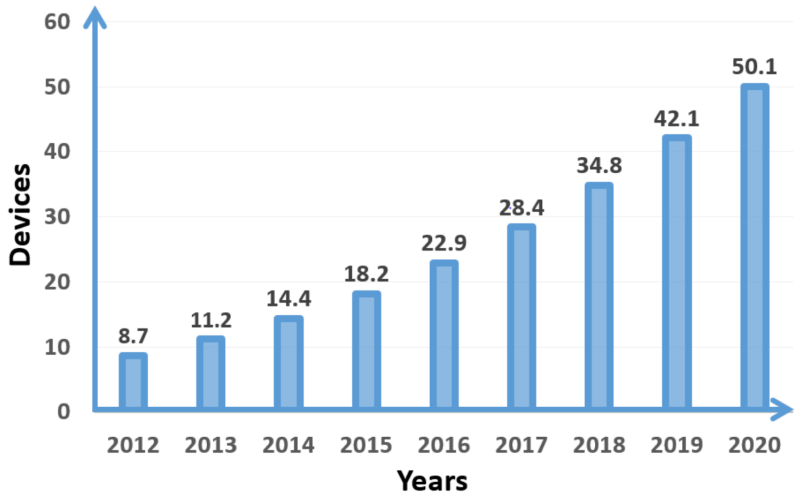


Figure 1.13. Number of connected IoT devices (billions) from 2012 to 2020 [63].

In such a highly connected world, digital systems record, monitor, and coordinate interactions between connected things making the physical and digital worlds meet and collaborate. Moreover, low-cost computing, big data, the cloud, analytics, and mobile technology, have allowed physical things to share and collect data with almost no human intervention. In general, the IoT refers to the network of physical objects (things) that are embedded with sensors, software, and other technologies to connect and exchange data with other devices over the Internet [64].
During the last decades, smart sensors have been increasingly integrated into manufacturing machinery, energy systems and infrastructure, enabling industries to boost their efficiency, productivity, and safety. This is referred to as Industrial Internet of Things (IIoT), namely the application of IoT technology in industrial settings, providing machinery control and optimization via communicating sensors. Moreover, industries have recently adopted Machine to Machine (M2M) communications for wireless control, reaching new levels of automation without any human intervention. For these reasons, IIoT is often referred to as the fourth wave of the industrial revolution (see Figure 1.14), namely Industry 4.0 (I4.0) [65].

The first industrial revolution began in the 18th century through the use of steam power and the mechanization of production. The most iconic and significant invention of this revolution was the power loom (or mechanical loom), developed by Edmund Cartwright in 1784 and used to weave threads together to produce fabric,

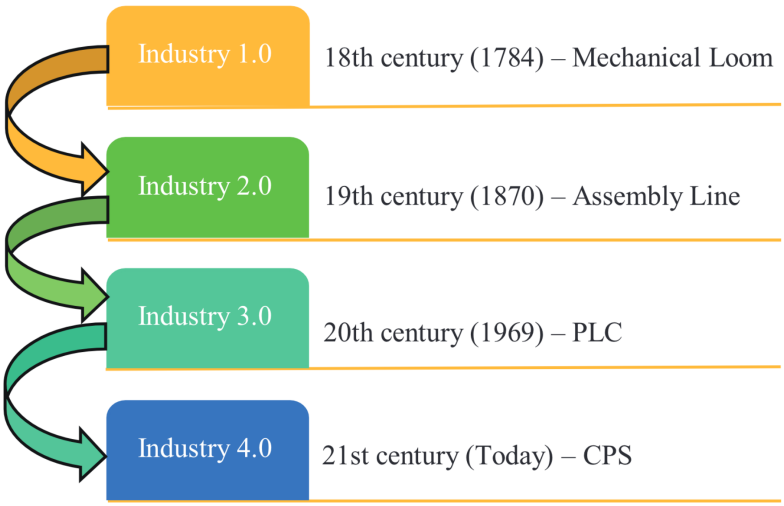


Figure 1.14. History of I4.0 [66].

which allowed to automatize a great share of the weaving process [67]. The second revolution took place during the 19th century through the discovery of electricity and the concept of assembly line production, introduced by Henry Ford during the 1870s in the automobile industry, leading to significantly faster manufacturing times and lower production costs [68]. The third industrial revolution began in the 20th century when informatics were incorporated into industrial processes laying the foundations for modern automation. One of the most important inventions of this revolution is the Programmable Logic Controller (PLC), proposed by Dick Morley in 1968 and first deployed in the automotive sector in 1969. PLCs allow controlling the functions of a system by using the internal logic programmed into them and are nowadays considered the standard automated control system in manufacturing industries [69].

Currently, we are witnessing the fourth industrial revolution characterized by the application of information and communication technologies to industry. It extends the developments of the previous industrial revolution by expanding network connections among systems, introducing the concept of Cyber-Physical System (CPS) where communication between the physical and virtual worlds is made possible through sensors or actuators. The networking of all systems is leading to the concept of smart industry driven by IIoT technologies, in which systems, components and people communicate seamlessly and processes are nearly autonomous [65].

IIoT has become increasingly relevant for factory applications where sensors allow monitoring the manufacturing process, analyzing variations, and acting accordingly [70]. Some applications include automation to enable machines to perform complicated tasks autonomously, logistics management for storage and distribution optimization, predictive maintenance to monitor the status of components and predict equipment failure, and safety to ensure security for industrial workers [71, 72, 73, 74]. IIoT finds other applications in smart transportation, where distributed network technologies allow to control, track, and increase awareness of traffic patterns of vehicles, pedestrians, metro, and train stations, contributing to

the general concept of smart cities [75]. Another example of IIoT use case is smart farming which aims at making farming organized, self-learned, and self-aware, with improved productivity. This is made possible by integrated IoT devices that sense light, temperature, humidity, soil, and moisture by fulfilling the energy requirements [76].
Nevertheless, one of the most important applications of IIoT is the energy sector where a variety of solutions are adopted to increase flexibility, reliability, efficiency, and security, giving birth to the concept of smart grids. The *smart grid*, regarded as the next-generation power grid, is an intelligent electricity distribution infrastructure that uses two-way flows of electricity and information to create a widely distributed automated energy delivery network [77]. In 2008, Jeremy Rifkin extended the concept of smart grids by coining the term Internet of Energy (IoE) which refers to an internet-style solution for electricity based on bidirectional information and power flow [78]. In general, IoE refers to a new paradigm for the operation of many power system elements such as distributed RE sources, plug-in Eletric Vehicles (EVs), Factory Energy Management (FEM) systems, Buildings Energy Management (BEM) systems, Home Energy Management (HEM) systems, and energy storage technologies, which can be monitored or optimized through the Internet [79]. A comprehensive architecture of the IoE communication network is presented in Figure 1.15.

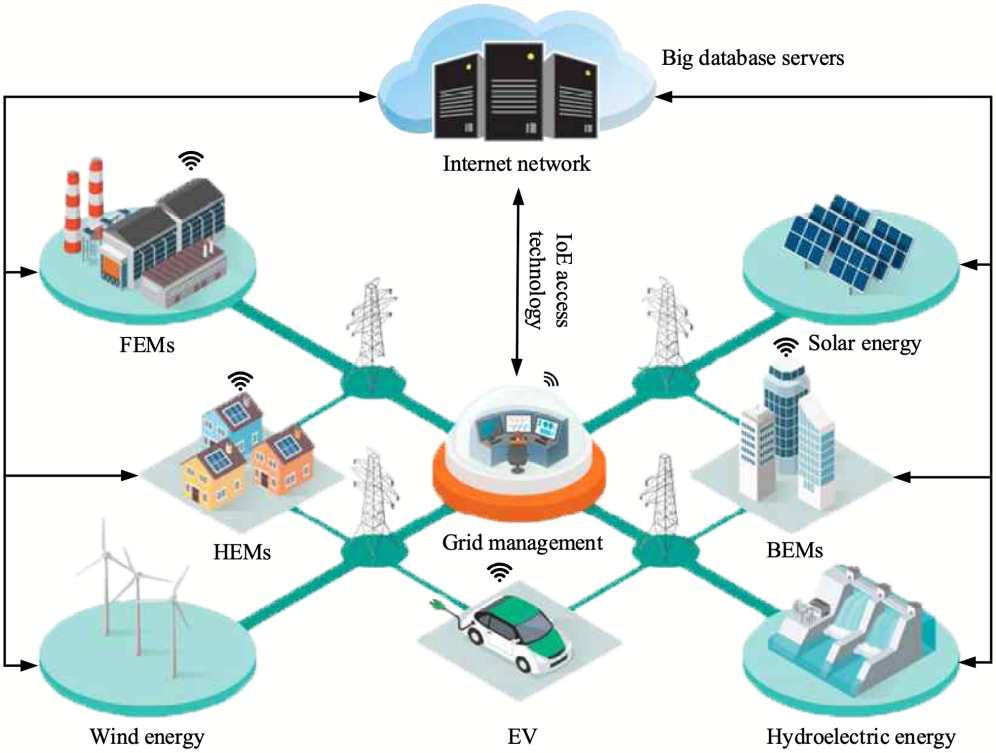


Figure 1.15. Overview architecture of the IoE communication network [80].

The general idea is that energy goes from source to load, with a direct analogy to the routing of information on the Internet, as shown in Figure 1.16. Different

detailed architectures have been proposed to implement the concept of IoE and the underlying key technologies include energy routers, storage devices, distributed RE sources, plug-and-play interfaces, and EVs [81].

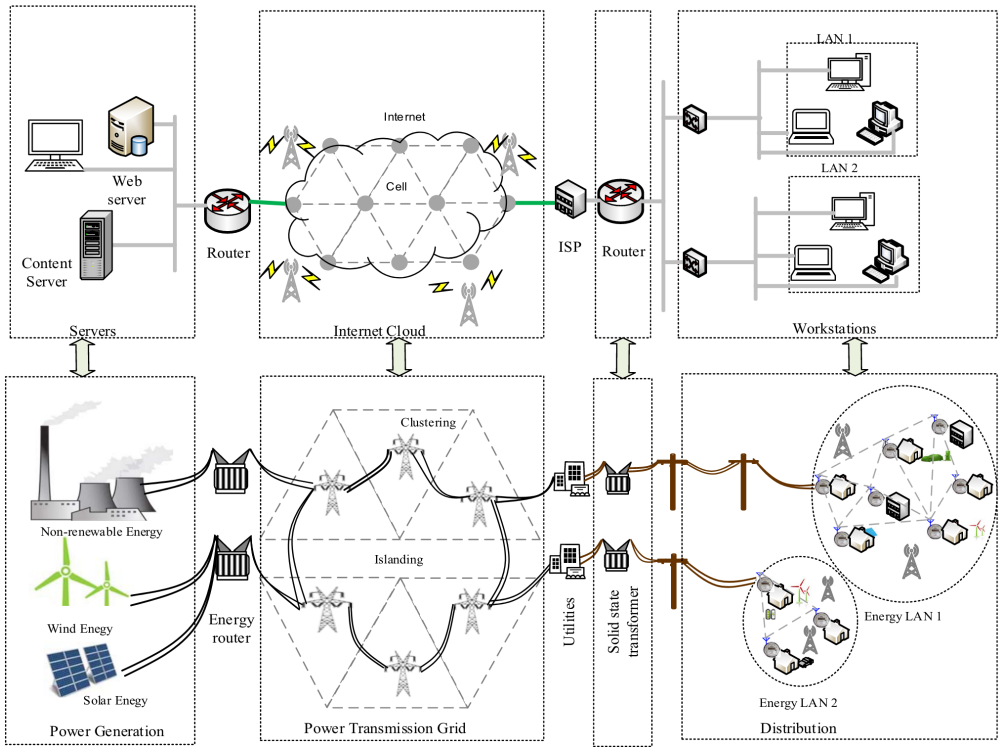


Figure 1.16. Analogy between the IoE and the Internet [81].

Energy routers are essential for an IoE network to dynamically and efficiently manage the energy supply and demand. These devices have the role to improve the reliability, efficiency, and safety of the power grid, and optimizing energy usage by processing information, dispatching electricity, and converting voltage. Moreover, energy routers are designed to communicate with each other to balance energy supply and demand on a wider scale and sell surplus energy to the grid when the local supply exceeds the local demand. This is made possible by their ability to receive, process, and transmit grid status information regarding current generation capacity, and current demand. More specifically, energy routers consist of three modules, namely a solid-state transformer for voltage conversion, a distributed grid intelligent control unit to optimize energy generation and distribution, and a communication unit for communicating with other energy routers [82].
Storage systems are fundamental to increasing grid robustness by improving the stability of grid operations and by providing a reliable energy supply. These systems are particularly useful when the grid is under stress during peak demand periods and is not able to effectively balance supply and demand. In such a scenario, storage systems can support power generation systems to meet the electricity demand, thereby preventing possible blackouts by smoothing the energy supply. This becomes crucial when power generation systems are distributed and based on RE sources, due to their intermittent nature. When the sun does not shine or the wind is too

weak, and, therefore, distributed RE systems do not deliver power, storage devices can provide previously-stored excess power to the grid. Some of the most common storage devices include batteries, supercapacitors, fuel cells, flywheels, compressed air, and pumped hydro [83].
Plug-and-play interfaces facilitate the connection of distributed REs, storage devices, and loads to the IoE network. Just like regular plug-and-play devices, such as USB flash drives, which are automatically recognized by the computer when plugged in, IoE plug-and-play devices are automatically recognized when connected to the energy network. These interfaces have different interface types, namely Alternating Current (AC) or Direct Current (DC), to allow a wide variety of components to seamlessly connect to the network. This is made possible by open-standard communication protocols which allow plugged-in devices to be immediately recognized. When, for example, a RE system is plugged into the network to provide power, the plug-and-play interface sends an energy generation request to the closest energy router which checks the local power demand and decides whether granting access to the system. Similarly, when RE systems do not deliver power, the plug-and-play interface sends a stop signal to the energy router, thus disconnecting the generation system from the grid [84].
EVs can be both a source or a load on the grid, and therefore have the potential to either support or burden the electricity grid depending on their state of charge and usage patterns. These devices can support the grid by providing power supply during peak demand periods, acting like batteries that increase the stability of the network. Since the number of EVs has been steadily growing over the past years, intelligent EV energy management systems are crucial to properly integrate these devices into the IoE network and benefit from their potential [85].
The concept of IoE has been attracting attention among academics, companies, and governments. Examples are the ARTEMIS IoE project where 38 partner companies from 10 EU countries developed IoE solutions for electric mobility infrastructure and smart grid communications, and the Future Renewable Electric Energy Delivery (FREEDM) Center funded by the US National Science Foundation where an electric power distribution system suitable for plug-and-play of distributed RE and distributed energy storage devices was created [86, 87]. In 2008, the E-Energy project was launched in Germany to motivate the development of efficient energy systems based on information and communication technology and, in 2015, China funded the Global Energy Internet (GEI) project to extend the Chinese smart grid and integrate RE using the Internet [88, 89]. Another example is Japan where, in 2011, the Digital Grid Consortium (DGC) proposed the concept of a digital grid composed of asynchronous and addressable digital grid routers capable of sending discrete energy packets over existing transmission lines to any location using IP addresses [90].
IoT, IIoT, and especially IoE are steering the industry towards more efficient, reliable, safe, and sustainable solutions with huge environmental and social potential benefits. The future of smart power grids depends more and more on the Internet for monitoring and control of distributed RE systems but can only be realized when substantial changes to electricity infrastructure, business models, and regulations are implemented. These are not only technical challenges and, therefore, require a collective and political effort to transform the current energy market into a sustainable

alternative.

1.4 The Role of Artificial Intelligence

Humankind continues to struggle when it comes to transforming human activities into a sustainable long-term alternative that ensures existential survival. Climate change is presenting complex challenges threatening life on the planet and all the available technological expertise is required to develop long-term solutions immediately, especially for energy systems which are the backbone of modern society [91, 92]. Among the most promising possibilities is Artificial Intelligence (AI) which in many countries has already revolutionized the energy industry by incorporating learning algorithms to perform different types of tasks, such as controlling, forecasting, and operating energy systems [93].
AI is often related to the concept of Machine Learning (ML), a sub-field that allows computer systems to automatically build mathematical models based on sample data, dating back to the late 1950s when many algorithms were proposed for classification and regression problems. Before the 1980s, almost all learning methods were based on linear models, like Linear Regression (LR) or logistic regression and, during the 1990s, researchers started to propose efficient learning algorithms for nonlinear functions, based on computational learning, mathematical and probabilistic theories. One example is Support Vector Machines (SVM)s, a discriminative two-class classifier that outputs a hyperplane as an optimal decision boundary to separate different classes, maximizing the distance between data points and the hyperplane itself [94]. Another example are ensemble methods, meta-algorithms that combine several ML techniques to improve predictions [95].
In 1999, when computers started becoming faster at processing data and Graphics Processing Unitss (GPUs) were developed, computational speed increased by many orders of magnitude. This allowed Artificial Neural Networks (ANNs) to compete with classical ML algorithms, leading to a new sub-field of AI, namely Deep Learning (DL). The theory was already developed in the past years but was never successfully used in real applications because of its computational requirements. ANNs were already proposed in 1943 and the *backpropagation* algorithm required to train them in 1974 [96, 97] . An ANN is a biologically-inspired model based on a collection of connected basic units called *neurons* or *perceptrons*. Neurons are grouped into layers and multiple layers form a Deep Neural Network (DNN). In general, an ANN is considered a *Universal Function Approximator* since it can approximate any function, even if highly nonlinear, given enough neurons and layers, and represented a breakthrough for AI [98].
Then, in 1979, a hierarchical and multilayered network named *Neocognitron* was introduced to deal with the processing of images and recognize visual patterns [99]. This network can be considered the first Convolutional Neural Network (CNN) that laid the foundations of DL in the field of Computer Vision [100]. Many more CNNs were developed during the years reaching state-of-the-art performances in tasks like object recognition. One example is AlexNet, a neural architecture that won several international competitions during 2011 and 2012 [101].
Since standard ANNs are not able to take into account the order and the temporal correlation between successive inputs, in 1991, Recurrent Neural Networks (RNNs) were introduced, a variant of ANNs with a memory cell able to capture temporal dependencies [102]. With RNNs, temporal sequences of data are considered as inputs

for the model, making the neural architecture suitable in the context of time series forecasting or sequence learning.
To increase the flexibility of the predictive model, Sequence-to-Sequence (S2S) architectures were proposed, allowing to generate predictions for multiple time steps [103]. The architecture is composed of two components named *encoder* and *decoder* that allow predicting multiple time steps given the same input sequence. Since RNNs are often unreliable due to optimization problems, the Long Short-Term Memory (LSTM) cell was proposed as a replacement for the standard memory cell [104].
RNNs dealing with temporal sequences often suffer from the incapability to process long sequences, forgetting or ignoring useful information. *Attention mechanisms* tackle this problem by allowing the model to look at the entire input sequence and assign more importance (or attention) to the most relevant elements [105]. One of the most effective and successful attention mechanisms is *Self-Attention*, heavily used in neural architectures like the *Transformer* which reached state-of-the-art performances in various sequence learning tasks [106].
In 1997, the Bidirectional Recurrent Neural Network (BRNN) was proposed, a neural architecture able to output predictions taking into account separately both future and past information in the sequence [107]. One of the most recent and promising neural architectures was proposed in 2016, namely Graph Convolutional Networks (GCNs), which leverages and exploits the topological structure of knowledge graphs [108].
AI experienced a continuous expansion in a wide variety of applications and contributed to the automation of important but repetitive and time-consuming tasks, allowing humans to focus on higher-value work. Moreover, learning algorithms proved their ability to reveal insights hidden in massive quantities of unstructured data that would require an amount of human management and analysis that is not only unavailable but also unfeasible. In fact, AI can integrate thousands of machines and other resources to solve complex problems almost without any human intervention [91].
As discussed in the previous chapters, RE represents one of the most promising and immediate ways to tackle climate change by reducing the impact of the energy sector on the global emission rates. Therefore, the potential and capabilities of AI become essential to allow RE systems to penetrate the energy market by mitigating the variability of RE sources, thus balancing the offset between supply and demand in the energy grid. Moreover, AI technologies support the RE industry by providing better O&M decisions, monitoring the power infrastructure, increasing the security of system operations, and new market designs [109].
One of the main applications of AI in the context of REs is predictive maintenance which is becoming an essential component in modern I4.0 environments and applications [110, 111]. Compared to classical retrospective and reactive approaches of quality control and condition monitoring that recognize problems after their occurrence, predictive maintenance aims at isolating untypical system behaviors and undesired patterns at an early stage [112]. In this way, when extreme or catastrophic system failures occur, any severe damage to the infrastructure, the machines, or the whole system can be avoided, together with any severe risks for operators working with the system. Moreover, predictive maintenance is important for guaranteeing a higher quality of production items, thus reducing repair and production costs, waste

and pollution of the environment [113].
Manual supervision is usually unrealistic to be conducted in a reasonable amount of time with reasonable efforts and costs for companies and is, therefore, considered a bottleneck. Moreover, it is affected by human inconsistencies caused by fatigue, uncertainty, boredom, or different cognitive abilities during different workloads or daytimes [114]. To increase the level of automatization and consistency of predictive maintenance, AI models based on ML algorithms are employed since capable of automatically and permanently producing forecasts to isolate arising problems and faults at an early stage or conducting a diagnosis about a predicted upcoming abnormal behavior. Even though data-driven models generate consistent outputs over time and are not affected by human errors, they may still suffer from low experience when designed for only very particular system operating conditions or when trained on insufficient or outdated historical data. Therefore, these models must be regularly re-calibrated or have the ability to self-adapt over time using routinely collected data [115].
Predictive maintenance is particularly relevant for wind energy, where difficult access to wind farms and their remote location, especially when offshore or in mountainous areas, the considerable height of WTs, and the size of the required equipment, give rise to high O&M costs. Many works broadly agree that O&M costs of wind farms account for approximately 25% to 35% of the total cost of power generation, and could increase by recurring failures of different WT components [116, 117, 118, 119, 120, 121]. Therefore, early detection of potential failures and appropriate CBM strategies are vital for operators and manufacturers for better dispatch, maintenance planning, and determination of required operating equipment. ML algorithms can contribute to reducing O&M costs with predictive models able to monitor and detect scenarios that might trigger maintenance, when WTs behave differently from the expected normal operating conditions. Moreover, the use of Supervisory Control And Data Acquisition (SCADA) systems for historical data generation has led to a variety of approaches based on data-driven ML techniques for effective CBM strategies [122, 123]. A review of ML methods for WT CBM was written by Stetco et. al in which the authors cover different works dealing with various tasks, for example, blade fault detection, generator temperature monitoring, or power curve monitoring [124]. They state that most models in the literature use SCADA or simulated data to train classification and regression models. Specifically, NNs, SVM, and Decision Tree (DT) are the most commonly used methods. Moreover, the authors highlight the importance of DNNs since capable of learning complex non-linear functions, thus achieving better performance than more traditional models as data volume grows [124].
Even though the O&M costs of solar PV are lower compared to wind power, they can still be significant enough to consider and develop reliable anomaly detection and maintenance strategies. Many believe that the PV systems require almost no maintenance and regular monitoring due to many research and reliability studies around PV panels and cells but this is not necessarily the case for PV systems as a whole [125]. In fact, many factors can cause PV systems' performance to deviate from the expected one. For example, the design or installation of the system could be flawed, leading to overall system performance degradation. Moreover, the power output and efficiency of the entire system can be already reduced when only an

individual cell of a single panel is compromised. Another problem is the long-term accumulation of dirt and debris on the panels which can significantly reduce the power output, especially when not mounted at an angle that allows adequate water runoff or in drier climates [126, 127]. This problem can greatly impact maintenance costs when the panels are mounted in remote locations making routine cleaning more difficult. Another critical component is the inverter which can lead to lower conversion efficiency or even fail when incorrectly sized [128].
If failures and performance degradation are not properly monitored and handled, it can lead to loss of energy generation and the failure of components, thus increasing O&M costs. For this reason, many predictive maintenance and anomaly detection strategies based on ML algorithms have been recently developed. For example, Platon et al. developed a fault-detection approach developed and validated using data measured from a real PV system. The proposed model aims at identifying values not representative of normal PV system operation by comparing the measured AC to the model's prediction and achieves a fault detection rate greater than 90% for different irradiance intervals [129]. Another example is the work by De Benedetti et al. in which the authors developed a model to predict the AC power production using an ANN taking in input solar irradiance and panel temperature measurements. They compared the model's prediction to the live trend data collected from the PV system and analyzed the resulting residual vector to detect anomalies and produce daily predictive maintenance alerts [130].
Another important application of AI for WTs is wind power forecasting since it can increase wind reliability by dealing with its intermittent nature. In fact, when accurate wind power forecasts are available, the power output of WTs can be scheduled accordingly and have a significant economic and technical impact on the energy grid, where a balance is highly desired between the power supply and demand [131]. A critical review of wind power forecasting methods is provided in the work of Hanifi et al. in which the authors discuss physical methods, ML models, and hybrid approaches for very short- (few minutes to 30 seconds), short- (30 minutes to 6 hours), medium- (6 hours to 1 day), or long-term (1 day to a month) forecasting [132]. The most frequently used ML models for wind power forecasting are NNs and SVMs, due to the ability to adapt to many specific cases. These two families of models can be computationally demanding, therefore also the K-Nearest Neighbors (KNN) and the Random Forest (RF) algorithms have been extensively used to forecast wind with lower computational time [133].
When it comes to solar PV systems, their power output is directly affected by the variability of meteorological parameters like air temperature, solar irradiance, humidity, or cloudiness, and the current state of the systems which can be, for example, dusty, inefficient, or damaged. Therefore, it becomes essential to design accurate and reliable models for PV systems, usually described by non-linear equations that are computationally demanding to solve. Recently, data-driven models have been gaining momentum since able to provide satisfactory results comparable to the physical models without requiring the same amount of computational power [134]. For example, Yaïci et. al used an ANN to predict the performance parameters of a thermal collector used for domestic hot water and space heating, and Ahmad et al. proposed a data-driven model to predict the hourly useful solar thermal energy in a thermal system using Support Vector Regression (SVR) and tree-based algorithms

[135, 136].
As for wind power, also in the case of solar PV, it is important to accurately forecast the renewable source, namely solar radiation, to balance the energy demand and supply. A review of ML methods for solar radiation forecasting was written by Voyant et al. in which the authors discuss many types of available algorithms [137]. Specifically, the most commonly used methods to estimate solar radiation are ANNs, Autoregressive Integrated Moving Average (ARIMA), SVM, SVR, k-mean, and tree-based algorithms.
Even though these are the main applications in the context of RE systems, AI has potentially limitless other applications in the energy sector, especially with the concept of smart grids and IoE, as discussed in the previous section. For example, since batteries do not have enough capacity to store large amounts of energy yet, load forecasting is crucial to ensure an adequate energy supply. ML techniques have been extensively implemented for load forecasting since providing satisfactory results while dealing with high volatility, complexity, and irregularity. Many approaches have been proposed for both residential buildings and industrial-scale electric loads, employing a wide variety of ML algorithms including ANNs, DNNs, CNNs, SVMs, and LSTMs [138, 138, 139]. Smart grids are usually coupled with the idea of a decentralized power grid and, therefore, optimization techniques in the context of both large-scale electrical grids and microgrids are required to balance supply and demand. Many ML algorithms have been proposed as distributed grid controllers, from supervised learning methods to Reinforcement Learning (RL), a sub-field of AI dealing with learning agents [140, 141, 142, 143]. Other RL algorithms have been employed to set prices in multi-microgrid settings for power grid balancing purposes, and ML models have been developed to enhance current structural designs and materials of batteries, PV cells, or solar thermal systems [141, 144, 145, 146]. Figure 1.17 presents an overview of many other sustainable energy systems applications where ML has been applied.

The next chapters discuss and propose data-driven approaches based on AI in the context of RE systems, and more in general, for the energy sector. Specifically, Chapter 2 deals with the most important preprocessing techniques used for time series collected from RE systems and covers the most commonly used ML algorithms and techniques. Then, Chapeter 3 presents novel algorithms together with their application in energy systems. In particular, Chapter 3.1 deals with the topic of dimensionality reduction for sensor networks, crucial in modern energy systems where hundreds of quantities are continuously monitored. Chapter 3.2 discusses predictive maintenance approaches in the context of RE systems, and Chapter 3.3 deals with energy forecasting as an important aspect for current smart energy grids.

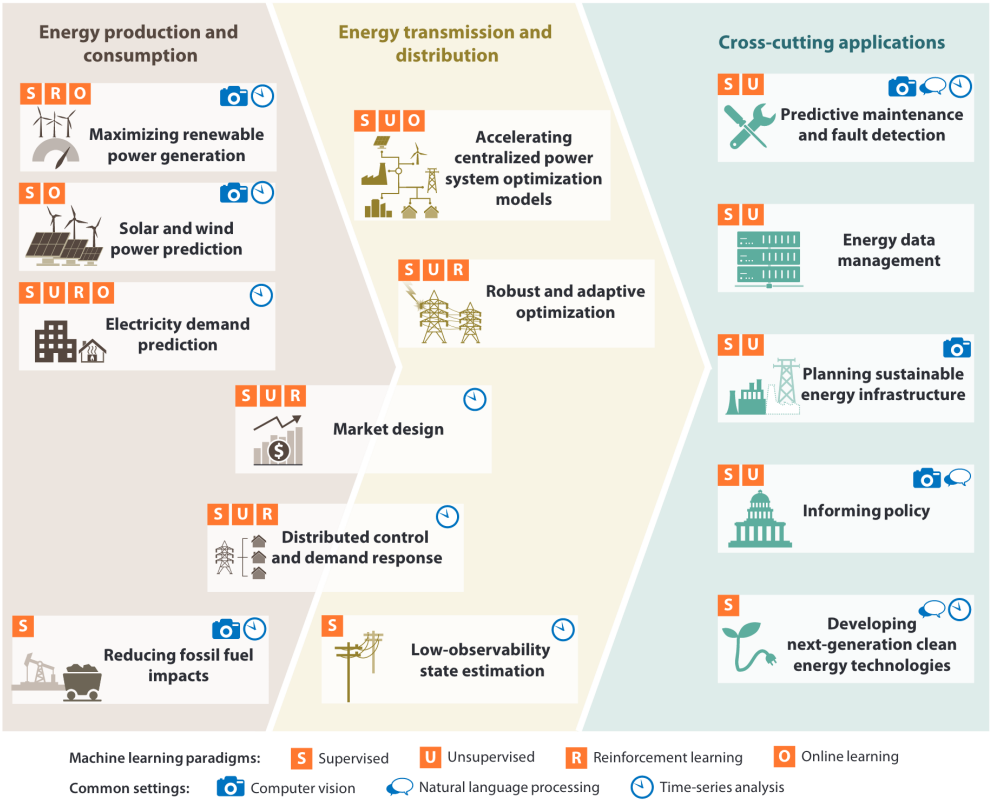


Figure 1.17. An overview of sustainable energy systems applications where ML has been applied, alongside common ML paradigms used within each setting [92].

Chapter 2

Methodology

2.1 Data Preprocessing Methods for Time Series

A large amount of data collected from sensor networks in energy systems is stored in the form of time series. However, time series can have un-ordered and missing timestamps, missing values, noise, or outliers, and, therefore, preprocessing becomes an essential part of time series analysis for cleaning real data. Moreover, different signals can be difficult to compare when having different distributions or scales. Preprocessing is essential not only to improve the quality of data but also to increase the performance of predictive models trained on time series [147]. This chapter covers the most commonly used techniques for handling missing values, outlier detection, denoising, transformations, and sliding windows.

2.1.1 Handling Missing Values

Since time series models work with complete data, missing values must be replaced with meaningful ones before the actual analysis. Broadly speaking, this problem is handled in two ways, namely by *deleting* or *replacing* missing values, or by erasing the entire time series. However, removing missing values can be a poor solution due to the temporal ordering of the data and the correlation between observations in successive time steps. Estimating missing values without distorting the components of the series is desirable. Figure 2.1 presents an example of time series having missing values that will be used to demonstrate different methods to handle missing values.

Mean/Median/Mode Imputation Mean imputation replaces missing values in a time series by estimating the mean of the available values in the sequence as shown in Figure 2.2 [148]. Median and mode imputation replace missing values with the median and mode of the time series, respectively. All three methods are fast and simple to implement but can introduce distortions to the signal.

Last Observation Carried Forward The Last Observation Carried Forward (LOCF) method imputes a missing value by replacing it with the first available value previous to the current missing one [149]. This imputation method can also introduce distortions as demonstrated by Figure 2.3.

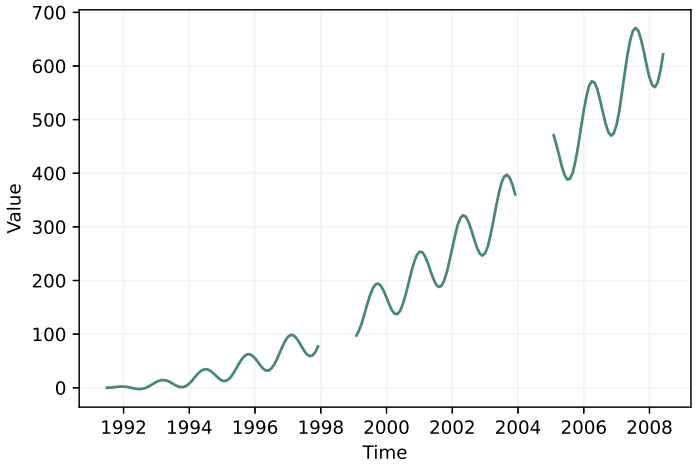


Figure 2.1. An example of time series having missing values, namely from 1998 to 1999 and from 2004 to 2005.

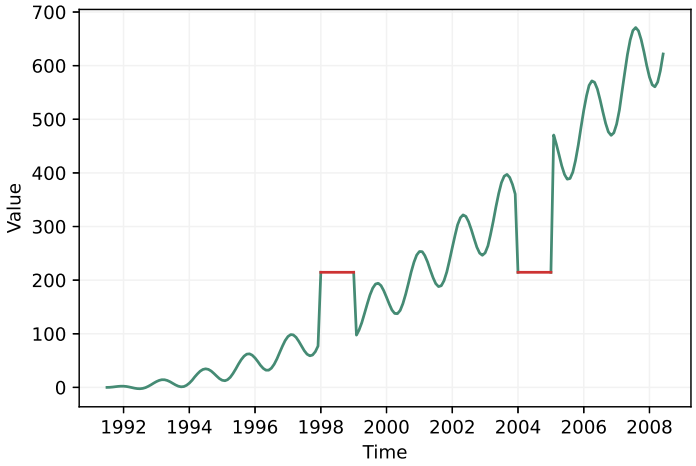


Figure 2.2. An example of mean imputation.

Next Observation Carried Backward The Next Observation Carried Backward (NOCB) method imputes a missing value by replacing it with the first available value next to the current missing one [150] and can introduce distortions as shown in Figure 2.4.

Linear Interpolation Linear interpolation imputes a missing value by looking at both the first past and the first future available values in the time series [151]. In particular, this method connects these values with a straight line to impute the missing value in between. If too many subsequent values are missing and linearly interpolated, a strong distortion can be introduced in the time series as shown in Figure 2.5.

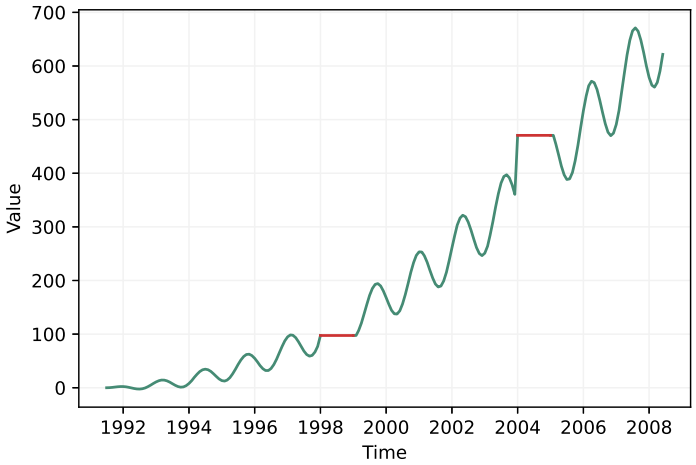


Figure 2.3. An example of LOCF imputation.

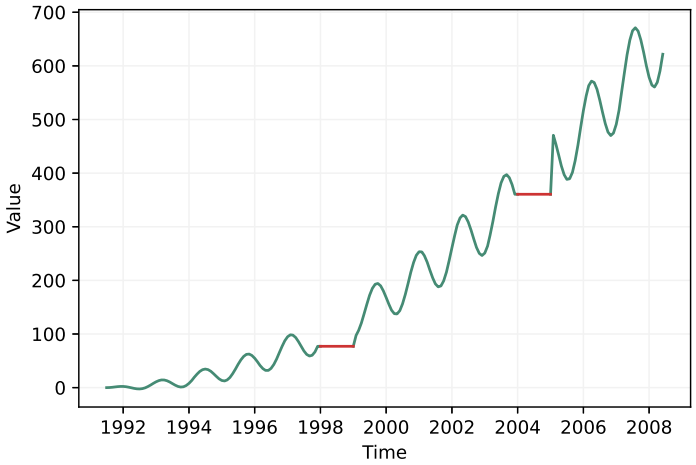


Figure 2.4. An example of NOCB imputation.

Spline Interpolation Spline interpolation fits a piecewise cubic polynomial which is twice continuously differentiable to the available points around the missing value [151]. This method can be very slow but can also provide smooth and meaningful missing value replacements. An example of spline interpolation is provided in Figure 2.6.

Feature Removal When too many values are missing, it can become impossible to replace them with meaningful values without introducing distortion or inconsistencies. In this case, a possibility is to erase the entire time series from the dataset. This operation has to be performed carefully and could be adopted when there is a large pool of available signals that contain similar information with respect to the involved one. When the signal is irrelevant, redundant, or duplicate, it can be removed with

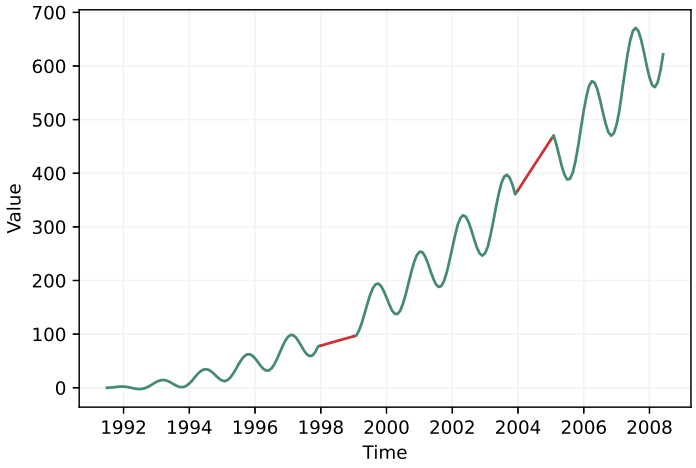


Figure 2.5. An example of linear interpolation for missing values imputation.

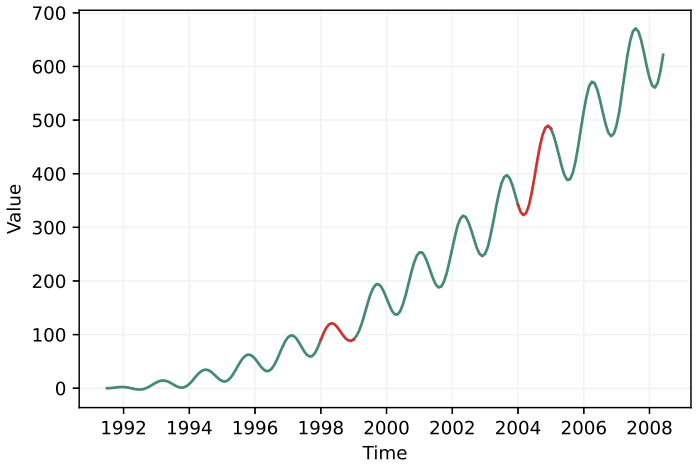


Figure 2.6. An example of spline interpolation for missing values imputation.

no further risks.

2.1.2 Outlier Detection

When performing outlier detection in time series data, the goal is to identify unexpected or rare instances which deviate excessively from the distribution of samples. Outliers can be caused by sensor measurement errors, noise, data acquisition problems, or anomalies in the system generating the time series. For this reason, depending on the application, it is important to handle outliers properly. When performing a forecasting task, for example, outliers can be deleted or corrected to clean the dataset before training a learner. When, instead, the aim is to detect anomalous patterns in the context of predictive maintenance, outliers have to be

isolated and analyzed carefully on their own.
Outliers can be divided into three main categories, namely *global* or point-wise outliers, *contextual* or conditional outliers, and *collective* outliers. Global outliers are data points having a value far outside the entirety of the dataset. Contextual outliers are samples that significantly deviate from the rest of the dataset when considering the same context. In this case, the same value can be classified as an outlier in one specific context and a normal sample in a different one. In time series, the context usually depends on time and a good example is when having seasonalities in the data. A collective outlier is a subset of samples that deviates significantly from the dataset, where the value of individual observations is neither a global nor a contextual outlier. This kind of outlier usually involves multiple time series which together are classified as outliers as a result of their unusual combination.

Sigma Rule A common method to isolate global outliers is the so-called sigma rule of thumb [152]. This is a conventional heuristic that assumes that the time series has a normal distribution and expresses that nearly all values lie within k standard deviations σ of the mean μ. When $k = 3$, a $\mu \pm 3\sigma$ range is defined as containing 99.73% of the data. Points that fall outside this range are classified as anomalies. Figure 2.7 shows the percentage of samples included by the k-sigma rule for different values of k.

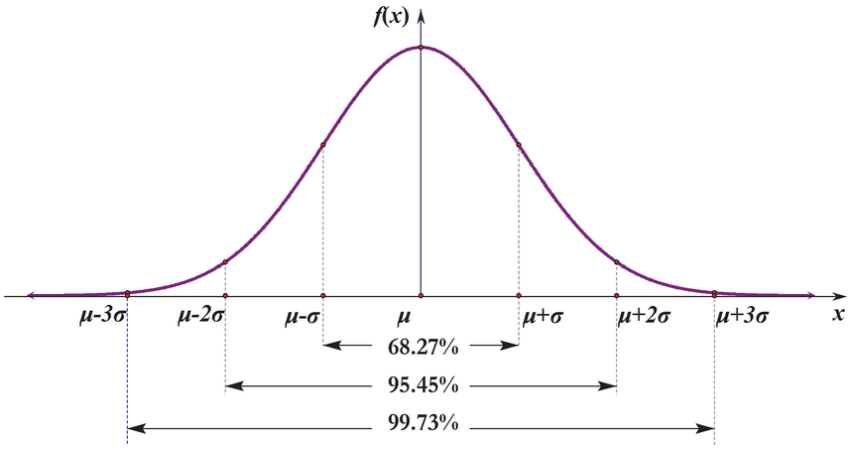


Figure 2.7. Percentage of samples included by the k-sigma rule for different values of k [153].

Rolling Sigma Rule The sigma rule isolates outliers by computing a static interval for the whole time series. For this reason, a rolling sigma rule can be adopted by defining a $\mu \pm k\sigma$ range on a rolling basis, where the lower and upper limits are computed for every subsequent subset of observations. In this way, the rolling sigma rule is applied locally by considering the surrounding measurements of every sample, making this method useful to detect contextual anomalies.

Boxplot A boxplot is a method for graphically analyzing the locality, spread and skewness of data through their quartiles [154]. Specifically, a boxplot displays a

five-number summary composed of the following:

- median (Q2/50th Percentile): the middle value.
- first quartile (Q1/25th Percentile): the middle observation between the lowest value and the median.
- third quartile (Q3/75th Percentile): the middle observation between the median and the highest value.
- upper limit: $\text{Q3} + \lambda \cdot \text{IQR}$
- lower limit: $\text{Q1} - \lambda \cdot \text{IQR}$

IQR represents the InterQuartile Range computed as the distance between the third and first quartiles ($\text{Q3} - \text{Q1}$). A boxplot can be used to detect global outliers by considering values smaller than the lower limit or greater than the upper limit. These limits depend on the parameter λ which is usually set to 1.5. An example of a boxplot is provided in Figure 2.8.

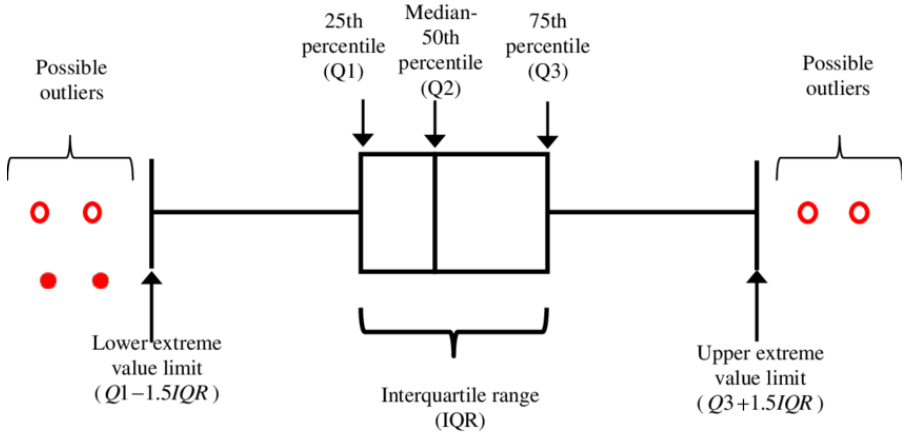


Figure 2.8. Example of a boxplot [155].

Isolation Forest Isolation Forests (IFs) are based on Decision Tree (DTs) (see Section 2.2.1.2) that are created in an unsupervised manner since there are no labels indicating which samples are outliers and which are not [156]. The idea behind this method is that anomalies are few and isolated data points deviating from the vast majority of the observations.
The IF algorithm constructs an ensemble of DTs by selecting a random subsample of the data for each tree. As for standard DTs, each tree structure breaks down the subsample into incrementally smaller subsets that contain instances with similar values. Specifically, each tree splits observations by randomly selecting a feature and, then, randomly selecting a split value between the maximum and minimum values of the selected feature. The number of splittings required to reach leaf nodes is equivalent to the path length from the root node to the leaf node itself. Since random partitioning produces noticeably shorter paths for anomalies, the average

path length over the ensemble is a measure of how likely is it that a sample is anomalous. Samples ending in short branches are considered anomalies because the DT could easily separate them from other observations. On the other hand, subsamples that require many branches in the tree to become homogeneous are unlikely to be anomalous. A graphical representation of an IF is presented in Figure 2.9.

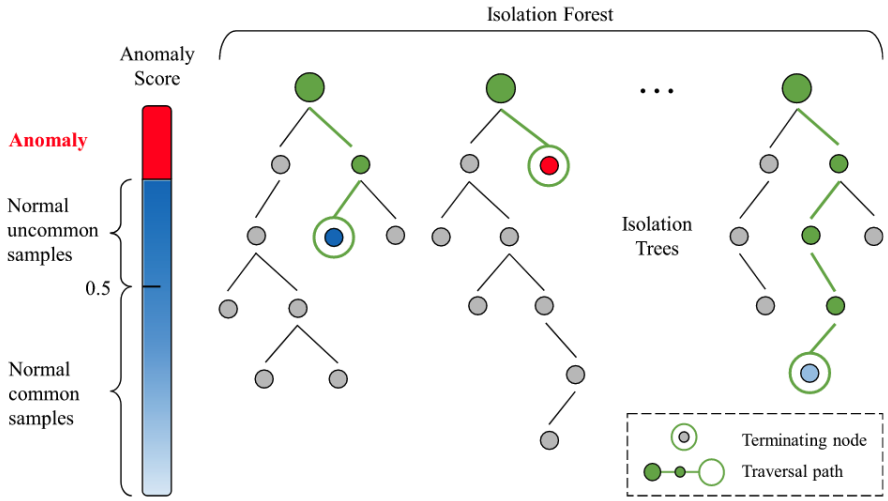


Figure 2.9. A graphical representation of an IF used to perform anomaly detection for WTs [157].

Density-based Clustering The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm (see Section 2.2.2.1) can be employed to detect collective outliers involving different time series. This can be achieved by classifying as outliers all the data points that do not belong to dense clusters at the end of the algorithm.
Another option is to consider as outliers all the points that do not belong to the densest cluster, depending on the application. An example is provided in Figure 2.10, where the power curve of a wind turbine is cleaned from outliers by filtering the data points that do not belong to the densest cluster, namely the main power curve.

2.1.3 Smoothing

Time series smoothing is essential when dealing with signals that contain noise or short-term fluctuations. The goal of the smoothing procedure is to filter out the irregular roughness of the time series and make the signal clearer to analyze. Specifically, smoothing attempts to remove the higher frequency components of a signal so that the lower frequency patterns can emerge.

Simple Moving Average When using a simple moving average, each point in the time series is smoothed by computing a weighted average of the values that

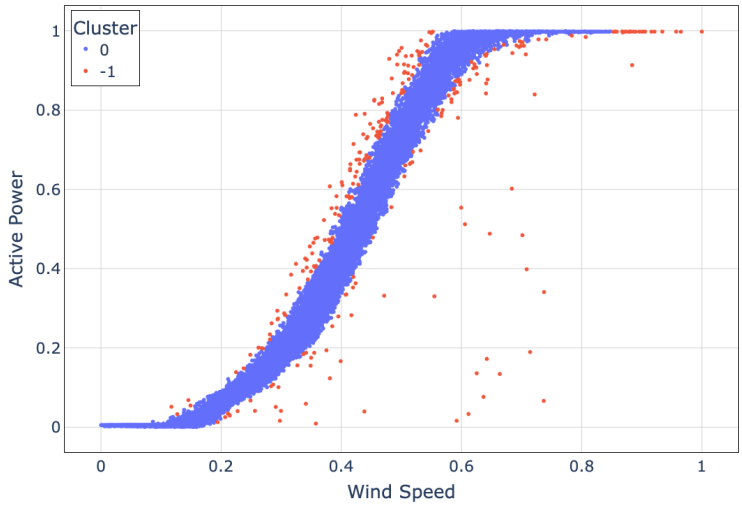


Figure 2.10. An example of outlier detection using the DBSCAN algorithm on the power curve of a WT. Points that do not belong to the densest cluster, namely the main power curve, are classified as outliers.

surround it [158]. In particular, given a time series $x_1, \cdots, x_T$, the simple moving average computes the smoothed value of each point x_t as

$$\hat{x}_t = \frac{1}{p} \sum_{k=i-l}^{i-l+p} x_k, \tag{2.1}$$

where p is the window (or filter) size defining how many samples are averaged. The parameter l controls the alignment of the moving average and different values lead to different smoothing methods, specifically:

- $l = p$: the smoothing procedure is referred to as *backward moving average*. Each point is computed by averaging all previous values within the time window. This method is suitable for real-time data since newly recorded values can immediately be smoothed using previous data in the time series. In this way, however, all information only comes from the left side of the point and can lead to poor results if the data has different trends on the left and right sides of the smoothed point.

- $l = 0$: the smoothing procedure is referred to as *forward moving average*. The smoothed value is computed by averaging all subsequent values within the time window. Similarly to the *backward moving average*, all information only comes from one side of the smoothed value. Moreover, this method is not suitable for real-time applications since the time window is not available for newly recorded values.

- $l = [p/2]$: the smoothing procedure is referred to as *centered moving average*, where $[p/2]$ is an integer division. Each point is computed by averaging all values within the time window, where the value being smoothed is at the

center of the window. In this way, half of the window contains previous values and half of the window subsequent values. Differently from the other two methods, information comes from both sides of the smoothed point, making this procedure more stable and with a smaller bias. As for the *forward moving average*, this method is not suitable for real-time applications since the right side of the window is not available for newly recorded values.

Figure 2.11 provides a visual representation of the *backward moving average*, *forward moving average*, and *centered moving average*.

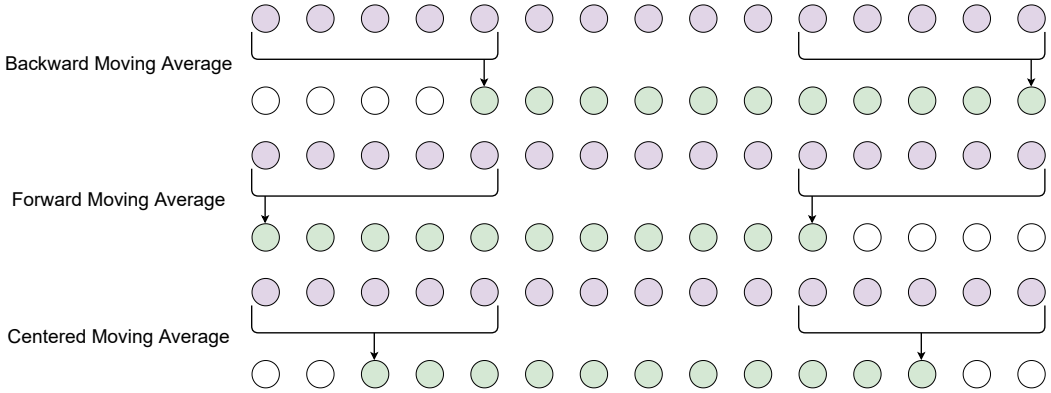


Figure 2.11. A visual representation of the *backward moving average*, *forward moving average*, and *centered moving average*.

Savitzky-Golay Filter The Savitzky-Golay filter is a method used to remove higher-frequency components of time series and has the characteristics of a low-pass filter [159]. As for simple moving averages, this filter smooths each data point by considering values in a time window p. For every window, a polynomial of order k is fitted by minimizing the distance between the values in the window and the polynomial. Then, the smoothed value is computed by evaluating the polynomial at the point of interest. An example of smoothing using the Savitzky-Golay filter considering different polynomial orders is provided in Figure 2.12.

2.1.4 Transformations

It is often the case that time series require some prior transformation before being modeled by predictive algorithms. For example, many ML models require signals to have a Gaussian distribution to be more robust and reliable. The goal of time series transformations is to simplify the time series by removing known sources of variation or by making the patterns more consistent across time. It is important to remember that when transforming an input signal, an inverse transformation is required for the model predictions to have a meaningful output.

Difference Transform The difference transform can be used to remove the trend or seasonal patterns from a signal to simplify the time series [161]. A first-order differencing can be used to remove the trend by subtracting the previous value from

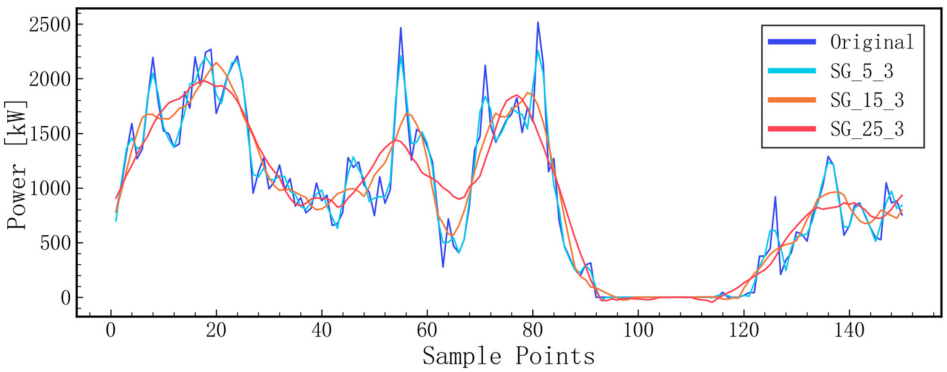
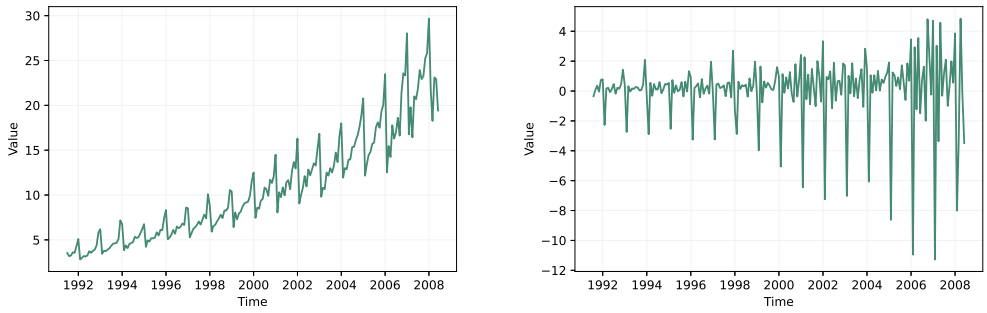


Figure 2.12. An example of the Savitzky-Golay filter used to smooth the active power signal of a wind turbine considering different window lengths k with the same polynomial order p [160].

each value in the series. This procedure can be repeated on the obtained differenced series to remove second-order trends, and so on. Similarly, seasonal patterns can be removed by subtracting the value from the prior season from each value in the series. Given a time series $x_1, \cdots, x_T$, the difference transformation replaces each point x_t with

$$x_t = x_t - x_{t-k}, \tag{2.2}$$

where k defines the interval of differencing. When $k = 1$, it performs a first-order differencing and, when dealing with seasonalities, k should coincide with the length of the seasonal cycle. An example of first-order differencing applied to a time series with a trend is provided in Figure 2.13.



(a) Time series before first-order differencing. **(b)** Time series after first-order differencing.

Figure 2.13. An example of first-order differencing applied to a time series with a trend.

Power Transform The power transformation is a family of methods to stabilize the variance of a signal across time that removes a shift from the data distribution to make it more normal-like [162]. The power transformation replaces each data point by calculating its log, square root, or inverse to remove the skewness in the data. A generalized version of the power transformation finds a parameter λ that

best transforms a variable to a Gaussian-like probability distribution as

$$\hat{x}_t = x_t^{\lambda}, \tag{2.3}$$

where x_t is the original value and $\hat{x}_t$ is its transformed version. An example of a power transformation applied to time series with a skewed distribution is shown in Figure 2.14.

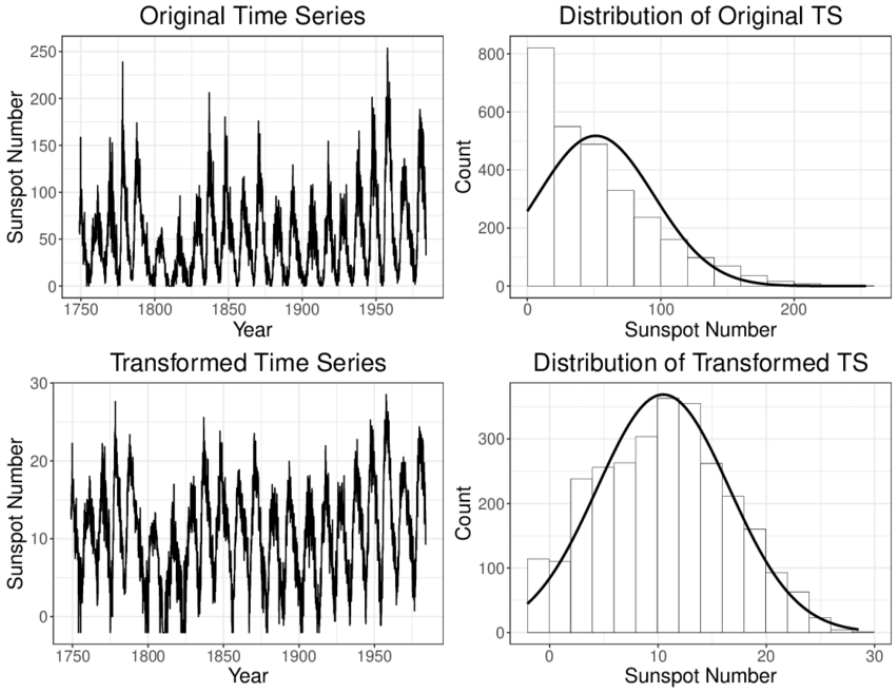


Figure 2.14. Example of a power transformation applied to a time series having a skewed distribution [163].

Box-Cox Transform The Box-Cox transformation [164] belongs to the familiy of power transformations and replaces each data point x_t with

$$\hat{x}_t(\lambda) = \begin{cases} \frac{x_t^{\lambda}-1}{\lambda}, & \lambda \neq 0. \\ log(x_t), & \lambda = 0. \end{cases} \tag{2.4}$$

The translation by -1 and scaling by λ for $\lambda \neq 0$ is performed so that the transformation is continuous, making it easier for theoretical analysis. When $\lambda = 0$, the natural logarithm is applied.

The Box-Cox transformation can be used only when $x_t > 0$. For this reason, a second parameter λ_2 can be introduced to allow this transformation to work with both negative and positive values. The Box-Cox transformation can be rewritten accordingly as

$$\hat{x}_t(\lambda) = \begin{cases} \frac{(x_t+\lambda_2)^{\lambda_1}-1}{\lambda_1}, & \lambda \neq 0. \\ log(x_t + \lambda_2), & \lambda = 0. \end{cases} \tag{2.5}$$

where λ_1 represents the λ parameter in the previous Equation 2.4. The parameters λ_1 and λ_2 can be estimated using the *profile likelihood* statistical method [165].

Yeo-Johnson Transform The Yeo-Johnson transformation [166] is another power transformation method that replaces each data point x_t with

$$\hat{x}_t(\lambda) = \begin{cases} \frac{(x_t+1)^\lambda - 1}{\lambda}, & x_t \geq 0, \lambda \neq 0. \\ log(x_t + 1), & x_t \geq 0, \lambda = 0. \\ -\frac{(-x_t+1)^{2-\lambda}-1}{2-\lambda}, & x_t < 0, \lambda \neq 2. \\ -log(-x_t + 1), & x_t < 0, \lambda = 2. \end{cases} \tag{2.6}$$

As for the Box-Cox transformation expressed in Equation 2.5, the Yeo-Johnson transformation also allows negative values of the input. Moreover, when $\lambda = 1$ it produces the identity transformation. If x_t is strictly positive, then the Yeo-Johnson transformation coincides with the Box-Cox transformation of $(x_t + 1)$ expressed in Equation 2.4. When strictly negative, it is a Box-Cox transformation of $(-x_t + 1)$, but with power $2 - \lambda$. For all other values, the Yeo-Johnson transformation results in a mixture of the previous two transformations.

2.1.5 Scaling

Comparing and combining different time series is a reoccurring challenge when dealing with multi-variate datasets, and many ML algorithms perform better when the input variables have a consistent scale or distribution. For this reason, scaling time series is an essential preprocessing step and is critical in guaranteeing that each variable does not bias predictions. Figure 2.15 presents an example of raw time series that will be used to demonstrate different scaling methods.

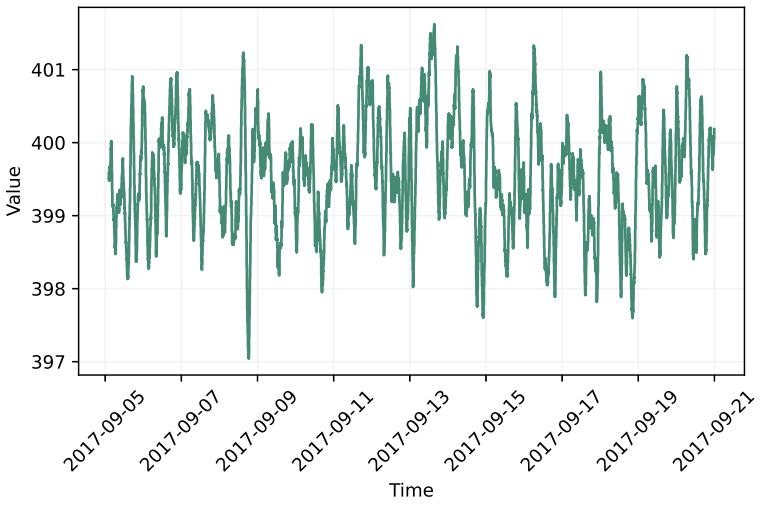


Figure 2.15. An example of raw time series before being scaled.

Min-Max Scaling The Min-Max scaling is a method that rescales a time series from its original range to a new range between 0 and 1 [167]. Specifically, each value x_t in the time series is rescaled as

$$\hat{x}_t = \frac{x_t - x_{\min}}{x_{\max} - x_{\min}}. \tag{2.7}$$

Notably, the Min-Max scaler requires an accurate estimate of the minimum $x_{\min}$ and maximum $x_{\max}$ values in the time series which can be obtained from the available data. It is important to notice that, when dealing with outliers, these values could impact the estimate of both minimum and maximum, and result in a poor normalization as shown in Figure 2.16. Moreover, when dealing with trends, minimum and maximum could be difficult to estimate. This scaling method is suitable when the range of original values is known a priori. This is the case with most measurements collected from sensor networks in energy systems, where the minimum and maximum values of a variable are bound by physical constraints.
The Min-Max scaler can be generalized to map the original range to a new range between any arbitrary values a and b as

$$\hat{x}_t = a + \frac{(x_t - x_{\min})(b - a)}{x_{\max} - x_{\min}}. \tag{2.8}$$

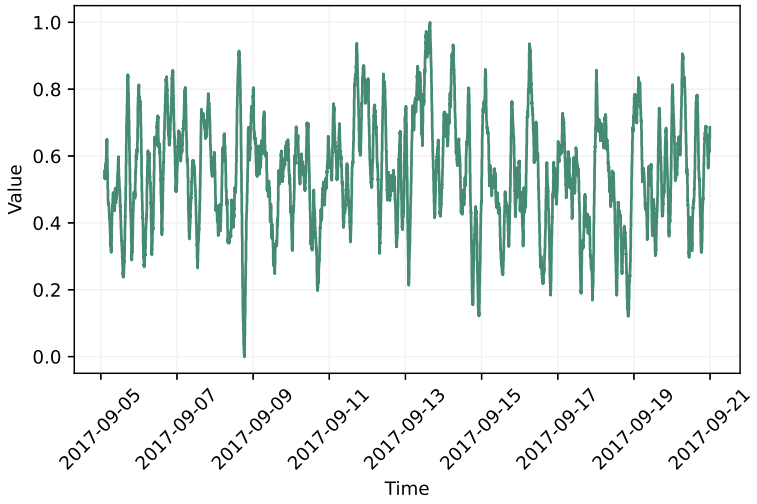


Figure 2.16. An example of time series scaled using the Min-Max scaler.

Standardization Standardization rescales the data by subtracting the mean and dividing the result by the standard deviation of the data sample [167]. In this way, the time data has a zero mean (centered), with a standard deviation of 1. The resulting distribution is called a standard Gaussian distribution, hence the name of the scaling method. Specifically, each sample x_t is scaled as

$$\hat{x}_t = \frac{x_t - \mu}{\sigma}, \tag{2.9}$$

where μ is the mean and σ the standard deviation of the data sample.
Standardization assumes that the data has a Gaussian distribution and works best for samples having such a distribution. This scaling method can still be applied to time series not having a Gaussian distribution but results might be unreliable or poor. Differently from the Min-Max scaler which considers the minimum and maximum estimates, the mean and standard deviation estimates are more robust

towards outliers, making the Standardization method more reliable. Figure 2.17 presents an example of time series scaled using the Standardization method.

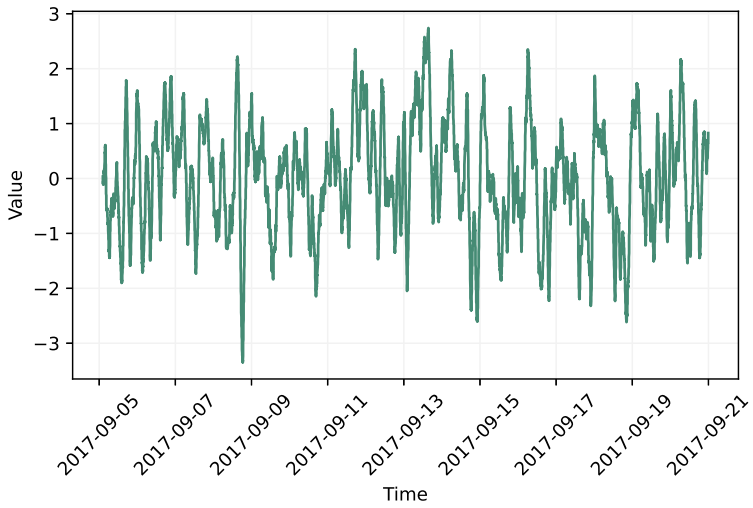


Figure 2.17. An example of time series scaled using the Standardization method.

Robust Scaling Even when using the Standardization scaling method, outliers can often influence the estimate of the sample mean and variance. For this reason, the Robust scaling method scales each sample x_t by using statistics that are robust to outliers, namely the median x_{median} and the IQR, as

$$\hat{x}_t = \frac{x_t - x_{\text{median}}}{\text{IQR}}. \tag{2.10}$$

An example is provided in Figure 2.18 where it can be seen that the scaled time series approximately lies in the range $(-1, 1)$.

2.1.6 Sliding Windows

A simple ML algorithm could take in input the current value x_t at time t and predict the next value x_{t+1} at $t+1$. Nevertheless, more complex models like NNs can take in input a window of past measurements from time $t-a$ to t and as target sequence, a window of future samples from time $t+1$ to $t+b$, depending on the task [168]. When extracting windows of inputs and outputs from a time series $x_1, \ldots, x_T$ for each time step t, it is possible to define an input sliding window

$$s_{t,i} = (x_{t-a}, \ldots, x_t), \tag{2.11}$$

and an output sliding window

$$s_{t,o} = (x_{t+1}, \ldots, x_{t+b}), \tag{2.12}$$

where $a \geq 0$, $b \geq 0$, and $t = 1, 1+p, 1+2p, \ldots, T$.
The length of the input sliding window $l_i = a + 1$ (the $+1$ is due to the inclusion

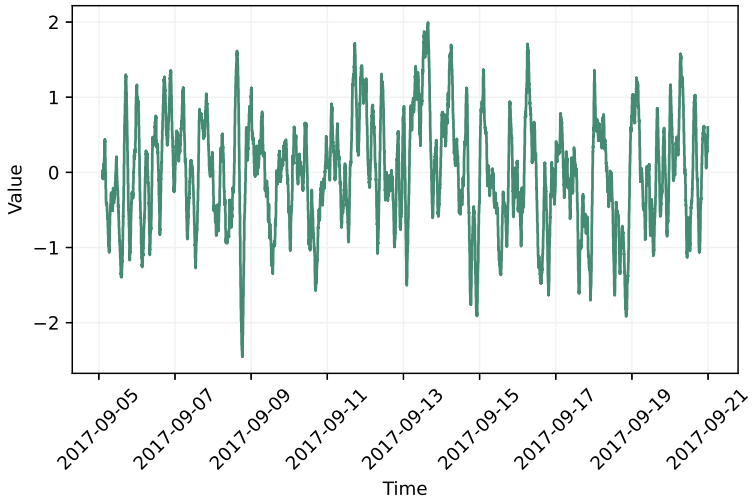


Figure 2.18. An example of time series scaled using the Robust scaler.

of the current measurement at time t) and the output sliding window $l_o = b$ highly depends on the application. For example, a model that forecasts the next 10 time steps ($l_o = 10$) of a time series given the past 19 observations and the current observation ($l_i = 20$) considers sliding windows with $a = 19$ and $b = 10$.
The other important parameter is the stride p which determines the number of time steps between extracted windows. Depending on the size of the window and the stride parameter, subsequent input windows and subsequent output windows can have an overlap. When setting $p = 1$, there is maximum overlap and this can be seen as a form of data augmentation. A visual explanation of the sliding window approach with overlap is shown in Figure 2.19.

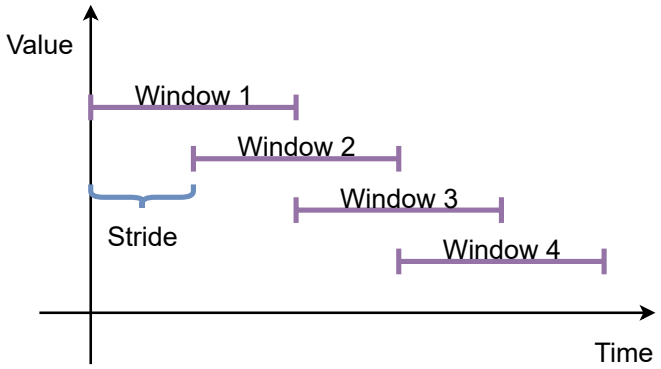


Figure 2.19. A visual explanation of the sliding window approach with overlap.

2.2 Machine Learning Algorithms

Machine Learning (ML) is a subfield of Artificial Intelligence that uses algorithms to learn and extract patterns from data. ML methods are usually divided into

broad categories based on how feedback is provided to the model during the training process. The most common classification includes *supervised learning* when dealing with labeled datasets and *unsupervised learning* when analyzing unlabeled datasets. Deep Learning (DL), instead, is a subfield of ML that uses biologically-inspired artificial NNs to automate as much as possible the feature extraction process, thus eliminating the need for manual human intervention and enabling the use of larger datasets.
This chapter provides a description of the ML and DL algorithms employed in the applications described in Chapter 3.1, 3.2 and 3.3 for dimensionality reduction in sensor networks, predictive maintenance in energy systems and power forecasting.

2.2.1 Supervised Learning

Supervised learning algorithms try to learn a function mapping input features to known targets exploiting a labeled dataset. Both classification and regression tasks fall in this category [169].

2.2.1.1 Classification

ML algorithms trained for a classification task learn how to assign a categorical label to examples from the problem domain $\mathbb{D}$, defined as

$$\mathbb{D} = \{(\mathbf{x_i}, y_i)\}_{i=1}^{n}, \tag{2.13}$$

where $\mathbf{x_i} \in \mathbb{R}^d$, $y_i \in \{1, 2, \ldots, k\}$, n is the number of samples, d the number of input features, and k the number of classes. A classification algorithm tries to approximate a function $f : \mathbf{x} \rightarrow y$ mapping the input feature vector $\mathbf{x}$ to its discrete output class y. The model is trained on a training set $\mathbb{S} \in \mathbb{D}$ and can successively be used to perform predictions over unseen data $(\mathbf{x_i}, y_i) \notin \mathbb{S}$ as

$$\hat{y}_i = f(\mathbf{x_i}), \tag{2.14}$$

where $\hat{y}_i$ is the prediction of the model.
Classification is a common task in ML and many algorithms have been proposed, like Logistic Regression, Decision Trees, Random Forests, Support Vector Machines or K-Nearest Neighbors [170, 171, 172, 94, 173].

2.2.1.2 Regression

Regression models are used to predict a continuous value, approximating a function mapping input features to real-valued outputs. The domain formulation is the same as in Equation 2.13, with the difference that $y_i \in \mathbb{R}$. A regression algorithm tries to approximate the function $y = \hat{f}(\mathbf{x}) + \epsilon$ that generated the observations (ϵ indicates a random error) by learning a function $f : \mathbf{x} \rightarrow y$ that maps the input feature vector $\mathbf{x}$ to its real-valued output y. Also in this case, predictions over unseen data $(\mathbf{x_i}, y_i) \notin \mathbb{S}$ can be computed using Equation 2.14, where $\hat{y}_i \in \mathbb{R}$.
A wide variety of algorithms belong to this category, like Linear Regression, Lasso Regression, Ridge Regression, Polynomial Regression, Decision Trees, Random Forests, Support Vector Regression or K-Nearest Neighbors [174, 175, 176, 177, 171, 172, 178, 173].

Linear Regression Linear Regression (LR) is a linear model that assumes a linear relationship between the input variables and the single output variable [174]. It approximates the real function producing the observations by using a polynomial and calculates the output y as a linear combination of the input variables $\mathbf{x}$. Given a dataset as expressed in Equation 2.13 with $y_i \in \mathbb{R}$, LR can be formulated as

$$y = \beta_0 + \mathbf{x}^T \boldsymbol{\beta}, \tag{2.15}$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_d)$ are the regression coefficients and β_0 the intercept (the predicted value of y when $\mathbf{x}$ is a zero vector).
The most common approach for fitting LR is the least-squares method, which computes the best-fitting coefficients for the observed data by minimizing the sum of the squared deviations of the observations from the fitted polynomial as

$$\beta_0, \boldsymbol{\beta} = \arg\min_{\beta_0, \boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \beta_0 + \mathbf{x_i}^T \boldsymbol{\beta})^2. \tag{2.16}$$

An example of LR between density-corrected monthly average wind speed and 30-day normalized available energy from a wind plant is shown in Figure 2.15.

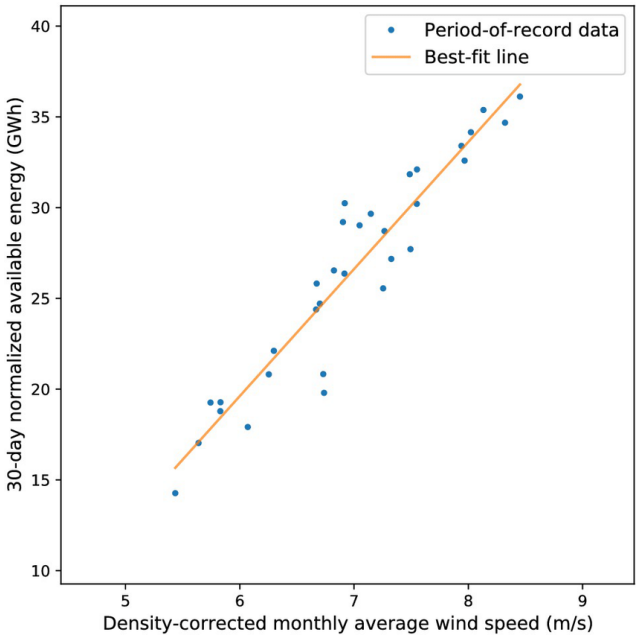


Figure 2.20. Example of LR between density-corrected monthly average wind speed and 30-day normalized available energy from a wind plant [179].

Decision Trees A Decision Tree (DT) is a model that can be used for solving both classification and regression tasks [171]. As for LR, DTs predict a target variable based on the value of the input variables. As the name suggests, this model has a tree structure that breaks down the dataset into incrementally smaller subsets that contain instances with similar values, while associating decisions to each branch. The final result is a tree with decision nodes and leaf nodes, which represent the possible

predictions of the model. DT estimates the output by asking a series of boolean questions about the data, each question narrowing the possible outcomes until the model is confident enough to make a prediction. The answer to each question (true or false) generates a new branch in the tree.
The algorithm starts from the root node representing the whole sample space and selects the feature that partitions the data into subsets having the highest homogeneity by minimizing their standard deviation (also known as variance reduction criteria). If a numerical sample is completely homogeneous its standard deviation is zero. The partitioning of the sample space is performed by the condition $f < v$, where f is the selected feature and v is the mean value of f, thus generating a decision node having two new branches with subsets of data. For each subset, the same process is repeated recursively until reaching a stopping condition, namely a maximum tree depth, a minimum number of nodes in a subset or a minimum value of standard deviation. When reaching a leaf node, the algorithm predicts the mean value of the target associated with the subset of samples in that node.
An example of fault diagnosis of Wind Turbine (WT) structures is provided in Figure 2.21. It is evident that the tree structure has a natural visualization, with its nodes and edges, and is very suitable for interpretability since it allows to analyze "what if" scenarios.

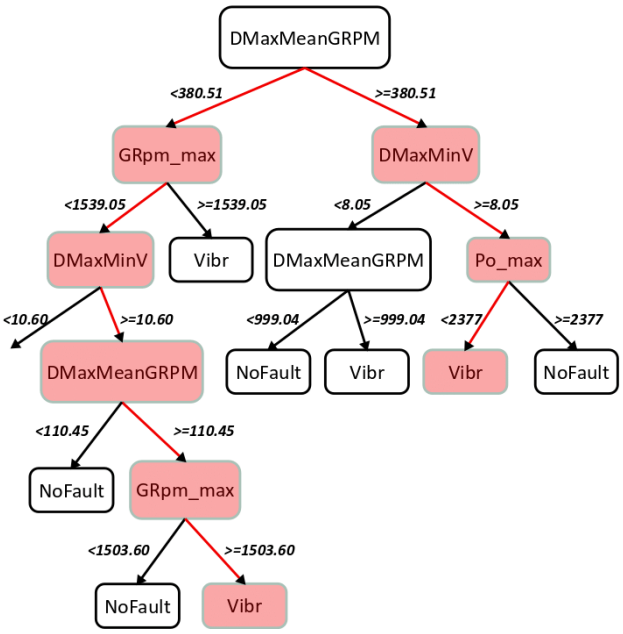


Figure 2.21. Example of a DT used for fault diagnosis of WT structures [180]. Notably, the predictions of a DT can be discrete (classification) as in this case, or continuous (regression).

2.2.1.3 Ensemble Methods

It is important to mention ensemble methods, meta-algorithms that combine several ML techniques into one predictive model to improve predictions [95]. There are

various ensemble families, but the two most widely used are *bagging* and *boosting*. BAGGing, or Bootstrap AGGregating, combines bootstrapping, namely a resampling technique used to estimate statistics on a population by sampling a dataset with replacement (bootstrap sample), and an aggregation algorithm to form the final ensemble model. Sampling with replacement ensures each bootstrap sample is independent of the others, as it does not depend on previously chosen samples. In particular, multiple independent models (weak learners) are trained on different bootstrap samples, and their predictions are combined by an aggregation algorithm, for example by averaging their outputs. The prediction $F(\mathbf{x})$ produced by M weak learners can be formulated as

$$F(\mathbf{x}) = \sum_{i=1}^{M} f_m(\mathbf{x}), \tag{2.17}$$

where $f_m(\mathbf{x})$ is the prediction of the m-th model. The aggregation of many weak learners usually outperforms a single learner over the entire dataset. A popular example of bagging is the Random Forest algorithm, which trains multiple DTs and aggregates their output to perform predictions [172].
In contrast to bagging, where multiple models are trained in parallel, boosting methods train them sequentially, with each new model trying to correct its predecessor. The final ensemble model is composed of several weak learners having different accuracies, which together can provide better overall performance. The prediction $F(\mathbf{x})$ of the ensemble model is a weighted sum of M weak learners and can be formulated as

$$F(\mathbf{x}) = \sum_{i=1}^{M} \alpha_m f_m(\mathbf{x}) \tag{2.18}$$

where $f_m(\mathbf{x})$ is the prediction of the m-th learner and α_m its weight coefficient. Finding all together the best coefficients and weak learners is a difficult optimization problem since each model depends on the previous one. Therefore, an iterative optimization process is adopted, where the best coefficient α_m and weak learner f_m are computed for each iteration m and incrementally added to the current ensemble model as

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \alpha_m f_m(\mathbf{x}), \tag{2.19}$$

where $F_{m-1}(\mathbf{x})$ is the prediction of the ensemble model without the m-th weak learner. At each iteration, α_m and f_m are computed as

$$\alpha_m, f_m = \arg\min_{\alpha_m, f_m} \sum_{i=1}^{n} L(y_i, F_{m-1}(\mathbf{x_i}) + \alpha_m f_m(\mathbf{x_i})), \tag{2.20}$$

where

$$L(y_i, F_m(\mathbf{x_i})) = \frac{1}{2}(y_i - F_m(\mathbf{x_i}))^2 \tag{2.21}$$

when considering the commonly used squared error as loss function L for the optimization process.
One of the most popular boosting algorithms is Gradient Boosting (GB) [181]. When training a new weak learner, it is useless to uncover the same pattern in the data as the previous model. For this reason, GB trains the next learner to fit the gradient of

the error with respect to the predictions of its predecessors. When using the Mean Squared Error (MSE), the gradient is proportional to the residuals of the previous learner. In particular, GB applies a gradient descent step to the minimization problem expressed in Equation 2.20 to find a local minimum of the loss function L defined in Equation 2.21 with respect to the predicted value $F_m(\mathbf{x_i})$. The local maximum-descent direction of the loss function is the negative gradient and can be written as

$$-\frac{\partial L(y_i, F_m(\mathbf{x_i}))}{\partial F_m(\mathbf{x_i}))} = -\frac{\partial \frac{1}{2}(y_i - F_m(\mathbf{x_i}))^2}{\partial F_m(\mathbf{x_i}))} = y_i - F_m(\mathbf{x_i}) = y_i - F_{m-1}(\mathbf{x_i}) - \alpha_m f_m(\mathbf{x_i}). \tag{2.22}$$

In order to minimize the loss function, it is necessary to set the gradient to zero as

$$y_i - F_{m-1}(\mathbf{x_i}) - \alpha_m f_m(\mathbf{x_i}) = 0, \tag{2.23}$$

which leads to the equation

$$\alpha_m f_m(\mathbf{x_i}) = y_i - F_{m-1}(\mathbf{x_i}), \tag{2.24}$$

meaning that the prediction $\alpha_m f_m(\mathbf{x_i})$ of the m-th weak learner is, ideally, equal to the residual

$$r_{im} = y_i - F_{m-1}(\mathbf{x_i}) \tag{2.25}$$

of the previous learner when considering the MSE as loss function. Except for the first learner, which predicts the mean value of y as

$$F_0(\mathbf{x}) = \frac{1}{2} \arg\min_{\hat{y}} \sum_{i=1}^{n} L(y_i, \hat{y}), \tag{2.26}$$

the next weak learners are, therefore, trained to predict the residuals r_{im} of their predecessors. GB reformulates the Equation 2.19 as

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \alpha_m f_m(\mathbf{x}), \tag{2.27}$$

where $\nu \in (0, 1]$, the learning rate, should be sufficiently small to guarantee the validity of the linear approximation adopted for the gradient descent optimization step towards the local minimum of the loss function. Finally, figure 2.22 illustrates the difference between a single learner, bagging and boosting methods.

XGBoost Regressor XGBoost, which stands for Extreme Gradient Boosting, is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable [183]. It implements ML algorithms under the GB framework and uses DTs as weak learners. In particular, XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solves many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and is the leading ML library for regression, classification, and ranking problems.

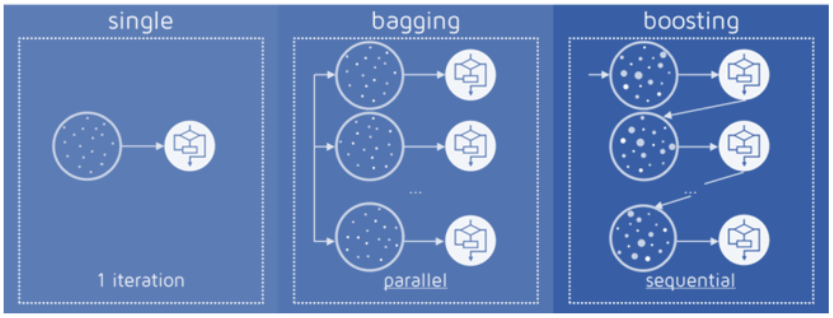


Figure 2.22. Difference between a single learner, bagging and boosting methods [182].

2.2.2 Unsupervised Learning

Unsupervised learning employs algorithms to analyze, cluster, and discover hidden patterns in unlabeled datasets [184]. Since data labeling requires manual intervention, it is often the case that the data is unlabelled, especially when dealing with large data streams coming from sensor networks. Unsupervised learning includes clustering and dimensionality reduction. Moreover, this chapter introduces concepts of Complex Network Analysis used to perform unsupervised dimensionality reduction in sensor networks.

2.2.2.1 Clustering

In clustering, the goal is to discover hidden structures within the data by grouping similar samples together. Specifically, clustering tries to divide the population into multiple groups so that data points in the same group are more similar to each other than points in other groups. *Partitional* and *density-based* methods are two of the most widely used families of algorithms when dealing with clustering.

DBSCAN The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm groups data points together that are spatially close and have many nearby neighbors based on the assumption that clusters are dense regions in the multidimensional space separated by lower density regions [185]. DBSCAN has two main hyperparameters, namely ϵ and m, and classifies data points into *core points*, *border points*, and *noise points* as shown in Figure 2.23.
For each data point, the algorithm creates a hypersphere with ϵ as the radius (neighborhood) and checks whether at least m points fall inside it using the Euclidean distance. If the condition is satisfied, the point is classified as a *core point*, if the number of points is less than m, it is considered a *border point* and if no points fall inside the hypersphere it is classified as a *noise point*. When a *core point* is identified, a new cluster is created composed of all points within its hypersphere. If another *core point* is included, its neighborhood is added also to the cluster. During the algorithm, a *noise point* can be revisited multiple times while exploring different neighborhoods and eventually assigned to a cluster.

DBSCAN is very sensitive to the hyperparameter choice and slight variations can

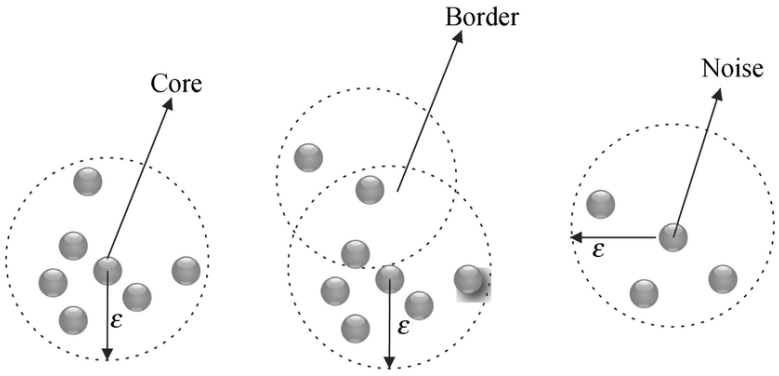


Figure 2.23. 2D example of the DBSCAN algorithm presenting *core points*, *border points* and *noise points* [186].

significantly change the results produced by the algorithm. The minimum number of neighbors m should be at least one greater than the number of dimensions of the dataset and is usually set as twice the number of dimensions, even though it can strongly depend on the domain application. ϵ, instead, can be chosen looking at the k-distance graph, generated by plotting the average distance between each point and its $k = m - 1$ closest neighbors in ascending order [187]. As shown in Figure 2.24, the point of maximum curvature (knee) corresponds to a sharp change in the gradient of the k-distance curve and can be considered as the optimal value of ϵ. If the chosen value is too small, a higher number of clusters is created, and too many points are classified as noise. If instead, ϵ becomes too large, smaller clusters are merged into bigger ones until, eventually, all points belong to the same cluster, thus losing detail about previous groupings and outliers. The value of ϵ associated with the point of maximum curvature of the k-distance point represents the optimal tradeoff between the number of clusters, their density, and outliers.

Unlike most partitional algorithms as K-means [189], DBSCAN is not bound to a fixed number of centroids but computes the number of clusters based on the shape of data which can be arbitrary. For this reason, the algorithm is very flexible and can also be used to detect outliers by considering *noise point* that are not included in any cluster at the end of the algorithm as shown in Figure 2.25.

Time Series Clustering Conventional clustering algorithms are unable to capture temporal dynamics and sequential relationships among data [190]. For these reasons, tremendous research efforts have been devoted to identifying time series-specific algorithms. The most common approaches involve modifying the conventional clustering algorithms to adapt them to deal with raw time series or to convert time series into static data (feature vectors or model parameters) so that conventional clustering algorithms can be directly applied. The former class of approaches includes direct methods, also called *raw data-based* [191, 192, 193, 194], while the latter refers to indirect methods that can be distinguished between *model-based* [195, 196, 197] and *feature-based* [198, 199, 200].
In addition, according to the way clustering is performed, the algorithms can be

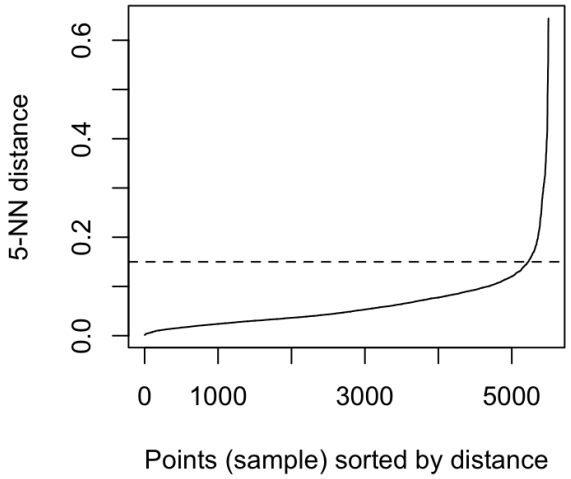


Figure 2.24. Example of k-distance plot used to identify the optimal value of ϵ [188].

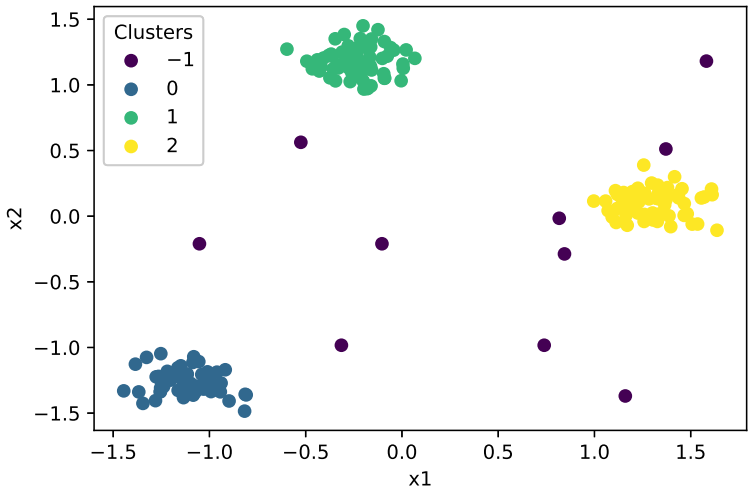


Figure 2.25. Example of the DBSCAN algorithm with a 2D dataset. Points in purple are outliers or *noise points*, while the other points belong to different clusters.

grouped into *whole time-series clustering* and *subsequence clustering*, a valid alternative to reduce the computational costs by working separately on time series segments. Detailed reviews of clustering algorithms for time series can be found in [201, 202, 203].

2.2.2.2 Dimensionality Reduction

Dimensionality reduction is a popular sub-field of unsupervised learning that has the goal to transform data from a high-dimensional space into a low-dimensional space retaining properties of the original data. It is often the case that data collected from control monitoring systems in engineered processes and machines have redundant measurements related to physical quantities when sensors are mounted at different points of the system and monitor the same parameters. For this reason, large data streams collected from sensor networks usually contain an overabundant number of features, some of which can be redundant or even useless. Moreover, high-dimensional data becomes difficult to analyze and visualize since the sample density decreases exponentially with the increase of the dimensionality. It is also important to mention that high-dimensional datasets imply more computational power requirements when training ML algorithms and can impact the learner's performance, especially when dealing with redundancies or noisy data.
All these phenomena are referred to as *curse of dimensionality* [204] and can be tackled following two possible approaches: *feature selection* and *feature extraction* [205].

2.2.2.3 Feature Selection

Feature selection is the process of selecting a subset of relevant features for the problem. In general, there are three groups of feature selection methods: *filters*, *wrappers* and *embedded* methods [206]. While filters do not rely on ML but on features' correlation thresholds, wrappers use ML techniques and the selection process is based on the (out-of-sample) performance of a learning algorithm.
The most commonly used techniques under wrapper methods are the Sequential Selection algorithms, e.g., Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS) [207, 208]. The SFS begins the search with an empty set of features, adding one feature at a time while trying to find the best set of combined selected parameters according to the evaluation criteria. The SBS, instead, refers to a search that begins with the full set of features, including all independent variables, and then removes the unimportant features until achieving the final set of selected significant parameters. Even though this latter approach may capture interacting features more easily, it is not fast nor computationally cost-effective [209]. Finally, embedded methods are a combination of filters and wrappers, where filters are integrated into the learner construction process [208]. This class includes the large family of DT methods. To mention but a few, XGBoost, or RF [183, 172].
Feature selection approaches can be further categorized into *supervised* and *unsupervised*. The former performs feature selection by data class labels, while the latter relies on the intrinsic properties of the data. Recently, unsupervised feature selection methods are attracting an ever-growing interest due to the widespread occurrence of unlabeled datasets in many applications [210, 211].

2.2.2.4 Feature Extraction

Feature extraction methods reduce the dimensionality of data by combining input features into derived values while preserving most of the original information content.

Principal Component Analysis Principal Component Analysis (PCA) is a popular feature extraction method for dimensionality reduction that produces a reduced number of latent variables by linearly combining the input features of the data [212].
In particular, PCA computes a new set of orthonormal (and hence linearly independent) dimensions, ordered in descending order according to the variance of data when projected on them. Therefore, the goal of the algorithm is to find a projection matrix W such that the projected data has the maximum possible variance in the lower dimensional space. Given a dataset

$$X = (\mathbf{x_1}, \ldots, \mathbf{x_i}, \ldots, \mathbf{x_n})^T, \tag{2.28}$$

where $\mathbf{x_i} \in \mathbb{R}^d$ is the i-th multivariate sample, its projection on W can be written as

$$Z = XW. \tag{2.29}$$

In order to compute W such that the variance of Z is maximized, PCA computes the eigenvectors and eigenvalues of the covariance matrix of X which is defined as

$$C = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x_i} - \bar{\mathbf{x}})(\mathbf{x_i} - \bar{\mathbf{x}})^T, \tag{2.30}$$

where $\bar{\mathbf{x}}$ is the mean vector of X, or as

$$C = \frac{1}{n}XX^T, \tag{2.31}$$

when the data is centered in zero ($\bar{\mathbf{x}} = 0$). This can be achieved using the eigendecomposition method and, after sorting the eigenvectors in decreasing order according to the magnitude of their corresponding eigenvalues, the first k eigenvectors (or principal components) are selected as the $k < d$ directions of the new feature space. The eigenvalues, instead, determine the importance of each new dimension by explaining the variance of the data after projection. Another popular method for computing the eigenvectors and eigenvalues of the data matrix is the Singular Value Decomposition (SVD) algorithm [213].

2.2.2.5 Complex Network Analysis

Complex Network Analysis (CNA), even though not classified as unsupervised learning, is discussed in this section since it is employed for unsupervised feature selection and extraction.
CNA studies how to analyze, describe and visualize complex networks that display substantial non-trivial topological features, with patterns of connection between their elements. Many real systems have been modeled as complex networks, like telecommunication networks, social networks, or public transport networks. Recently, they have also been adopted for the study of time series, making them suitable for the analysis of data streams collected from sensor networks [215]. The next paragraphs will describe principles of graph theory to better understand CNA and how it can be used for time series.

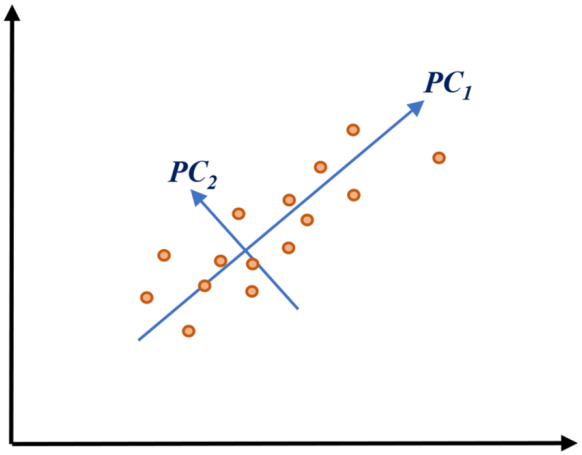
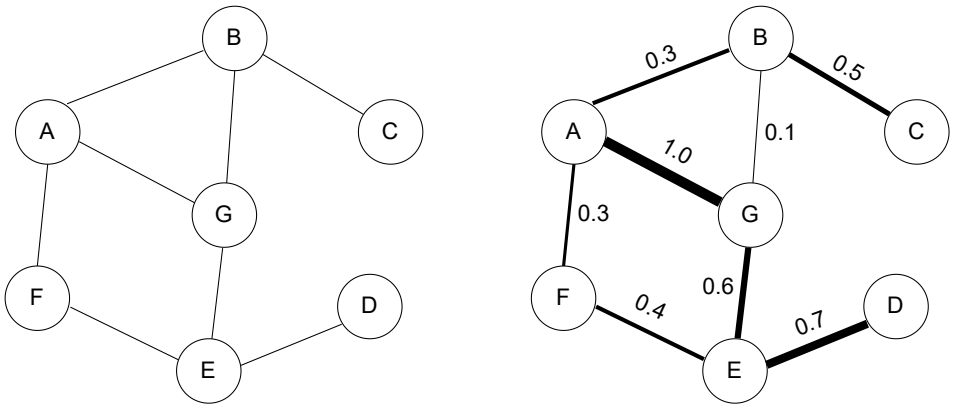


Figure 2.26. 2D example of the PCA algorithm showing the first two principal components maximizing variance [214].

Basic Principles A graph is a pair $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ and $\mathbf{E}$ are the set of nodes and edges, respectively. Nodes represent elements composing the network, while edges are the relationships between pairs of nodes which are typically expressed as an $N \times N$ adjacency matrix A, where N is the number of nodes. The element a_{ij} of the matrix A is associated with the edge that connects nodes i and j, and assumes real values when dealing with *weighted* graphs or binary values (0 or 1) when considering *unweighted* graphs as shown in Figure 2.27. In the first case, weights quantify the relationship between nodes on a continuous scale, while, in the second, they simply indicate if the connection exists or not.

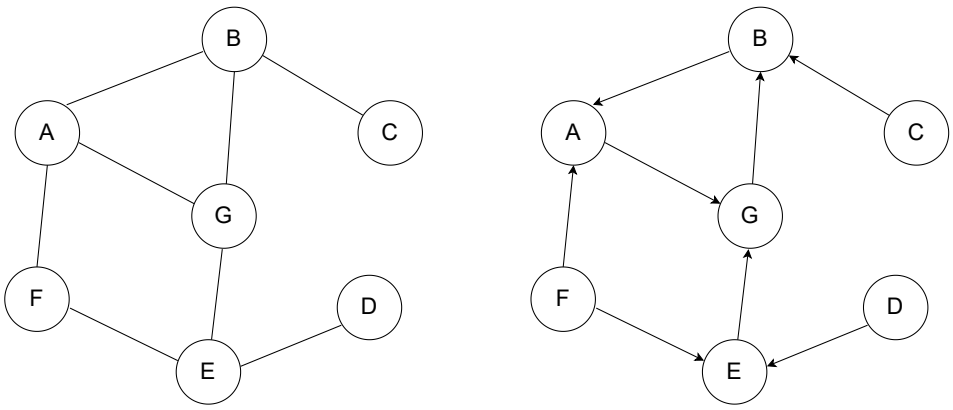


(a) Unweighted graph. **(b)** Weighted graph.

Figure 2.27. Example of an unweighted and weighted graph.

Moreover, when the graph is *unidirected*, the condition $a_{ij} = a_{ji}$ holds, but not necessarily when dealing with *directed* graphs, where connections are not symmetrical. Figure 2.28 shows an example of undirected and directed graphs.

Networks can be composed of many nodes and edges, making the analysis and the unveiling of hidden relationships very challenging. For this reason, global and



(a) Unidirected graph. **(b)** Directed graph.

Figure 2.28. Example of an unidirected and directed graph.

local network measures are used, respectively, to extrapolate synthetic topological information from the whole network and study the role nodes play in its structure.

Local Network Measures Local network measures are used to extract node-specific properties of the graph. The first historically proposed one is the degree centrality which allows detecting the most influential nodes within the network [216]. This measure is based on the simple concept that the centrality of a vertex in a network is closely related to the total number of its connections. In particular, the weighted degree centrality of a node i in a graph reads as

$$k_i = \sum_{j \in L} w_{ij}, \tag{2.32}$$

where L is the number of nodes, w_{ij} is the weight of the edge connecting nodes i and j, and $k = (k_1, k_2, \ldots, k_L)$ is also called the degree sequence of the graph. When dealing with a weighted graph, this local measure is called weighted degree centrality, and an example is provided in Figure 2.29.

Another measure is the eigenvector centrality, which is used for determining elements that are related to the most connected nodes [217]. The betweenness centrality, instead, can highlight which nodes are more likely to be in the network communication paths, and, finally, the closeness centrality measures how quickly information can spread from a given node [218, 218].

Global Network Measures Global network measures are used to extract general characteristics from networks that can be employed to classify a graph, analyze its structure, or compare different networks [219]. These metrics are based on the evaluation of the topology of the entire network and provide information about the node arrangement and the edge connectivities from a global perspective.
Modularity is one of the most important global network measures and evaluates the strength of the division of a network into groups (or communities) [220]. High values of modularity correspond to dense connections between the nodes composing

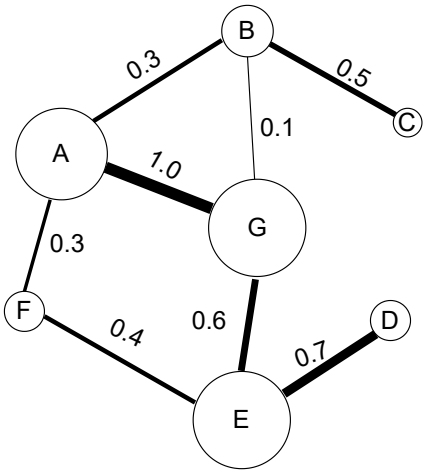


Figure 2.29. Example of weighted degree centrality as local network measure. Notably, the size of the nodes is proportional to their centrality.

communities and sparse connections between nodes in different groups, and an example is shown in Figure 2.30.

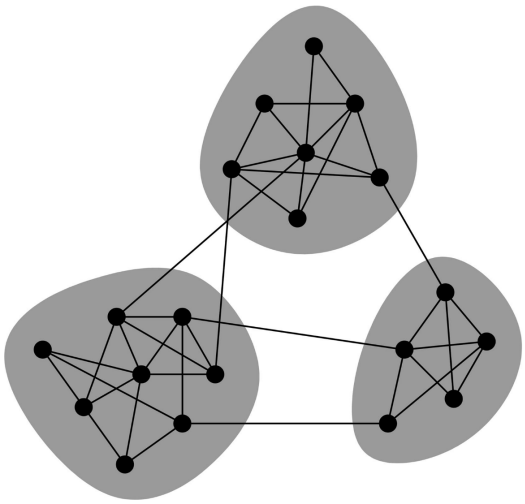


Figure 2.30. Example of network topology characterized by high modularity where three different communities are connected by sparse edges [221].

Modularity can be expressed as

$$Q = \sum_{c=1}^{n} [\frac{L_m}{m} - \gamma(\frac{k_c}{2m})^2], \tag{2.33}$$

where n is the number of communities, m is the number of edges, L_c is the number of intra-community links for community c, k_c is the sum of degrees of the nodes in community c, and γ is the resolution parameter [222]. This last regulates the tradeoff between intra- and inter-group edges and is commonly set to 1. Modularity is often used in optimization methods for detecting community structures in networks as discussed in the next section.

Community Detection Community detection algorithms are used to identify groups of nodes (communities) that are strongly connected by edges (relations), sharing common properties or playing similar roles in the network. In particular, nodes that are central in a community may be strongly influential on the control and stability of the group, while boundary nodes are crucial in terms of mediation and exchanges between different communities as shown in Figure 2.31 [223].

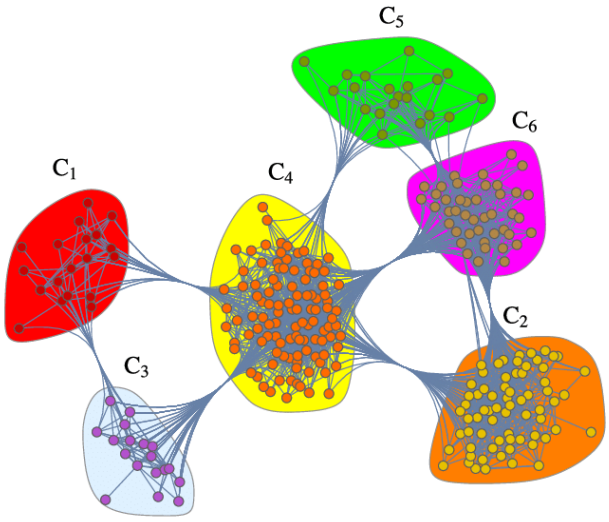


Figure 2.31. Example of community detection where six communities are identified [224].

Many community detection methods have been proposed to date and a possible classification includes traditional, modularity-based, spectral, and statistical inference algorithms.
Traditional methods include graph partitioning, which selects groups of predefined size by minimizing the number of inter-group edges [225, 226]; distance-based methods, where a distance function is minimized starting from local network measures [227]; and hierarchical algorithms that produce multiple levels of groupings evaluating a similarity measure between vertices [228].
Modularity-based methods, instead, try to maximize the modularity measure, which evaluates the strength of division into communities [229, 230, 231, 232]. One of the most popular algorithms is Louvain's method which is based on a bottom-up approach where groups of nodes are iteratively created and aggregated in larger clusters. In particular, nodes are initially considered independent communities and the best cluster partition is identified by moving single nodes to different communities searching for local maxima of the modularity measure. Then, a new network is constructed by modeling clusters as graph vertices and by computing edge weights as the sum of the connection weights between adjacent nodes belonging to different communities. These steps are iteratively repeated until convergence, corresponding to a local maximum in modularity.
Another category of community detection methods are the spectral algorithms [233], which detect communities by using the eigenvectors of matrices such as the Laplacian matrix of the graph. Finally, statistical inference algorithms aim at extracting properties of the graph based on hypotheses involving the connections

between nodes [234, 235].

Visibility Graph The visibility graph algorithm is a method to transform time series into complex network representations. This concept was originally proposed in the field of computational geometry for the study of mutual visibility between sets of points and obstacles, with applications such as robot motion planning [236]. The idea was extended to the analysis of univariate time series, making it possible to map a time series into a network that inherits several properties of the time series itself [237]. Moreover, visibility graphs are able to capture hidden relations, merging complex network theory with nonlinear time series analysis [238].
In particular, the visibility graph algorithm maps a generic time series $s = (s_1, s_2, \ldots, s_L)$ into a graph by considering a node (or vertex) for every observation s_i for $i = 1, \ldots, L$, where L is the length of the time series. The edges of the graph, instead, can be generated using two different algorithmic variants: the *natural* visibility graphs and the *horizontal* visibility graphs.
The natural visibility graph algorithm generates an edge with a unitary weight between two nodes if their corresponding observations in the series are connected by a straight line that is not obstructed by any intermediate observation [237]. Formally, two nodes a and b have visibility if their corresponding observations $s_a = (t_a, v_a)$ and $s_b = (t_b, v_b)$ satisfy the condition

$$v_c < v_b + (v_a - v_b)\frac{t_b - t_c}{t_b - t_a} \tag{2.34}$$

for any intermediate observation $s_c = (t_c, v_c)$ such that $a < c < b$. t_a and t_b represent the timestamps of the two samples, while v_a and v_b are the actual observed values. A computationally more efficient algorithmic variant is the horizontal visibility graph, based on a simplified version of Equation (2.34) [239, 240].
Visibility graphs can be enhanced by considering its weighted version [241], where the weight between any pair of nodes $s_a = (t_a, v_a)$ and $s_b = (t_b, v_b)$ reads as

$$w_{ab} = \sqrt{(t_b - t_a)^2 + (v_b - v_a)^2}. \tag{2.35}$$

A schematic illustration of the weighted visibility graph construction is shown in Figure 2.32, while its graph representation is presented in Figure 2.33.

Visualization Exploratory tools for visualization are essential to study the composition of a network, by revealing structural relationships that may otherwise be missed. As described by Sakkalis, there is a large variety of specialized exploratory network layouts (e.g., force-directed, hierarchical, circular, etc.) based on different criteria [243]. Among them, force-directed layouts are extensively applied to identify communities with denser relationships and capture the modularity of the network. An example of force-directed layout is the Frushterman–Reingold algorithm, shown in Figure 2.34, which considers nodes as mass particles and edges as springs between the particles [244]. The algorithm minimizes the energy of this physical system to find the optimal configuration. This process is only influenced by the connections between nodes, thus, in the final configuration, the position of a node cannot be interpreted on its own but has to be considered related to the others.

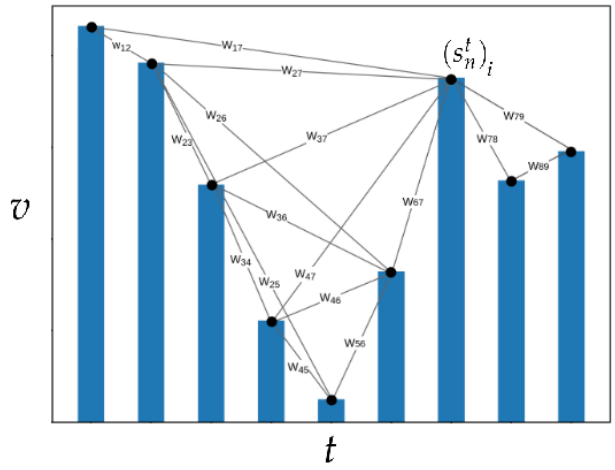


Figure 2.32. A schematic illustration of the algorithm for the construction of the weighted natural visibility graph. Notably, the time series is represented as a bar chart and s_i is the i-th observation [242].

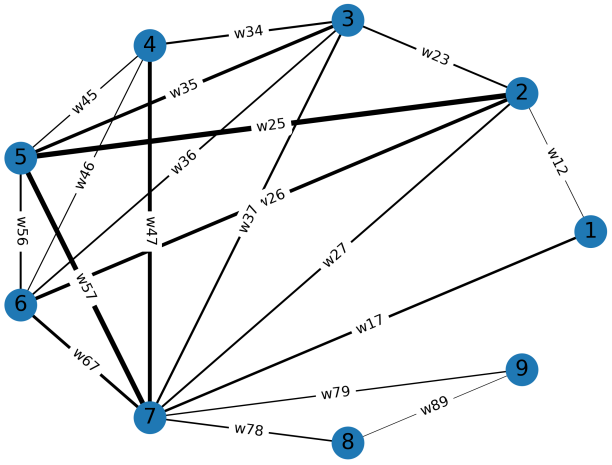


Figure 2.33. The results of the visibility graph algorithm in terms of graph representation [242].

2.2.3 Deep Learning

Deep Learning (DL) is a subset of ML that uses biologically-inspired Artificial Neural Network (ANNs) to extract meaningful patterns from data. Classical ML algorithms leverage structured data to make predictions and usually require some pre-processing steps before they can be trained. DL algorithms, instead, can ingest and process unstructured data, like text and images, thus automating feature extraction and reducing the dependency on human experts.

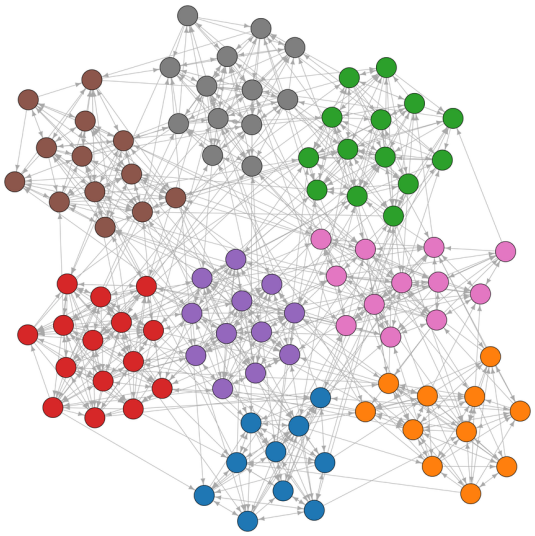


Figure 2.34. An example of force-directed layout based on the Frushterman–Reingold algorithm [245].

2.2.3.1 Artificial Neural Networks

An ANN is a biologically-inspired model based on a collection of connected basic units called *neurons* or *perceptrons* [96, 246]. A *perceptron* is nothing more than a simple linear classifier that takes in input a real-valued vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$, computes a weighted sum of the inputs and produces a real-valued output after applying an *activation function*.
The output of a *neuron* can be formalized as

$$\mathbf{o} = \sigma(\sum_{i=1}^{n} w_i x_i + b) = \sigma(\mathbf{w} \cdot \mathbf{x} + b), \tag{2.36}$$

where $\mathbf{w} = (w_1, w_2, \ldots, w_n)$ is a vector of real-valued weights, and b is an additional bias independent from the inputs used to shift the output. σ is the *activation function* and resembles the firing system of biological neurons. Several activation functions can be chosen based on the requirements, for example, the *step function*, the *sigmoid*, the *hyperbolic tangent*, the *ReLU* and many others. All these activation functions will be discussed in detail in Section 2.2.3.2. The perception is shown graphically in Figure 2.35.

Neurons are grouped into *layers*. Layers that receive input data are called *input layers*, the ones that produce the prediction of the network are called *output layers* and all the layers between the *input* and the *output* layers are called *hidden layers*. Every layer has the role of producing a new representation for the inputs it received. In general, layers that have dense connections between all the neurons are also called Fully-Connected (FC) or *dense* layers. ANNs with exactly one hidden layer are called *shallow* Neural Networks (NNs), while networks having more than one hidden layer are called Deep Neural Network (DNN).
A DNN (shown in Figure 2.36) is a ML model that uses a hierarchical composition

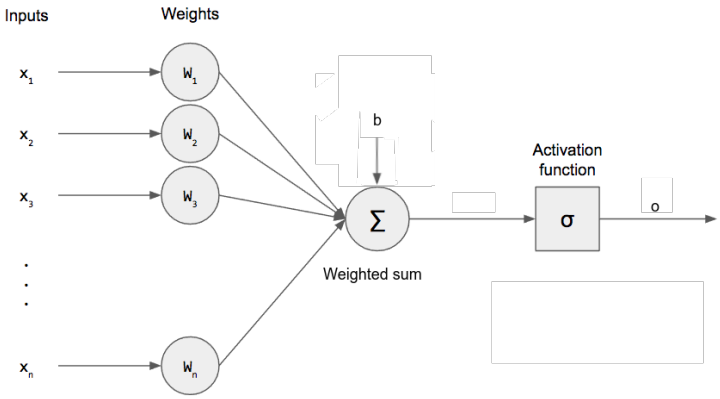


Figure 2.35. Graphical representation of the perceptron.

of n parametric functions to model an input $\mathbf{x}$. Each function f_i $(i = 1, \ldots, k)$ is modeled using a layer of neurons, which are elementary units applying an *activation function* to the previous layer's weighted representation of the input to generate a new representation. Each layer is parameterized by a weight vector $\mathbf{w}_i$ representing the weights applied to the outputs of the previous layer f_{i-1}. In general, the output of a DNN can be written as

$$\mathbf{o} = f_n(\mathbf{w}_k, f_{n-1}(\mathbf{w}_{n-1}, \ldots, f_2(\mathbf{w}_2, f_1(\mathbf{w}_1, \mathbf{x})))). \tag{2.37}$$

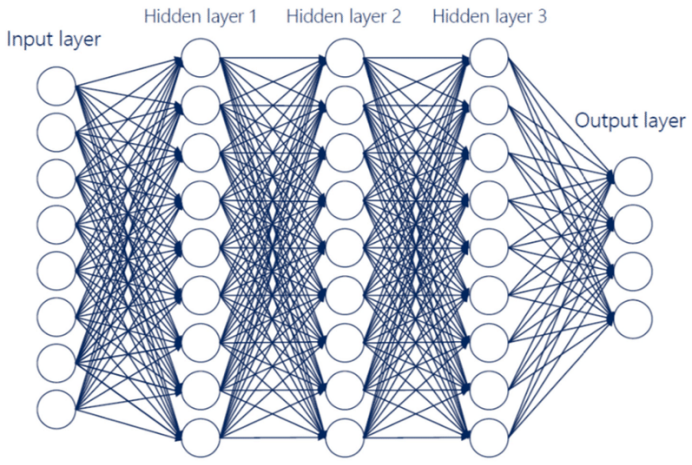


Figure 2.36. Example of DNN with three hidden layers [247].

A NN where the information only flows from the input in the direction of the output is called *feedforward* network. In general, these networks perform a *feedforward step* to obtain a prediction for some input vector $\mathbf{x}$.
The *training phase* of a NN learns values for its parameters $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_k)$. This is done by adjusting the weight parameters to reduce a *cost function* quantifying the error between the prediction of the network (obtained through the *feedforward step*) and the correct output associated with the input vector $\mathbf{x}$. The adjustment is

typically performed using techniques derived from the *backpropagation algorithm* (described in Section 2.2.3.5) which successively propagate error gradients with respect to network parameters from the network's output layer to its input layer [248]. The amount of weights adjustment performed at every training step is defined by the *learning rate* hyperparameter.
During the *evaluation phase* the set of parameters **W** is fixed to make predictions of inputs unseen during training and only the *feedforward* step is performed.

2.2.3.2 Activation Functions

Since activation functions have been mentioned multiple times in the previous section, they will be discussed here. As previously said, activation functions resemble the firing system of biological neurons and are responsible for the flow of information. They decide which information flows to the next layer and how much of it. Given an input space, they can map it into a completely different (usually smaller) output space.
These functions, when *non-linear*, are very important because they allow a network to model an arbitrarily complex function. In general, NNs are considered *Universal Function Approximators* since they can approximate any function, even if extremely non-linear, given enough layers and parameters [98]. If instead only *linear* activation functions are chosen, this property does not hold since the NN would just be a composition of n *linear* functions and so a *linear* function itself, unable to capture non-linearities.
Moreover, these function have to be *differentiable*, so that the *backpropagation algorithm* can be applied through the computation of gradients.
Some of the most common activation functions will be discussed next.

Step Function

The *step function* (shown in Figure 2.37) is a threshold-like function defined as

$$f(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.38}$$

When the input is greater or equal to zero, the neuron fires and outputs the unitary value, otherwise no information passes. This function is avoided in practice since it is not differentiable and is limited to output only two values, namely 0 or 1.

Linear function

The *linear function* (Figure 2.38) outputs a weighted combination of the inputs, based on the slope c, defined as

$$f(x) = c \cdot x. \tag{2.39}$$

It is more expressive than the step function because it can output continuous values. Nevertheless, its derivative is a constant that coincides with the slope c and does not depend on the input. For this reason, this function is not suitable to adjust the weights of the network through the *backpropagation algorithm*. In fact, when all the activation functions are linear, the NN becomes a linear function itself.

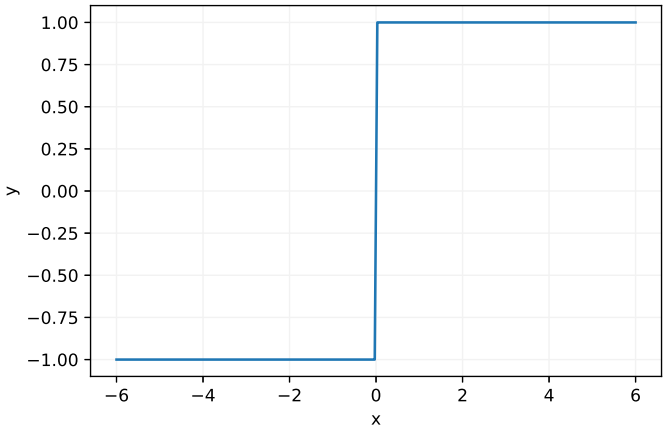


Figure 2.37. Step function.

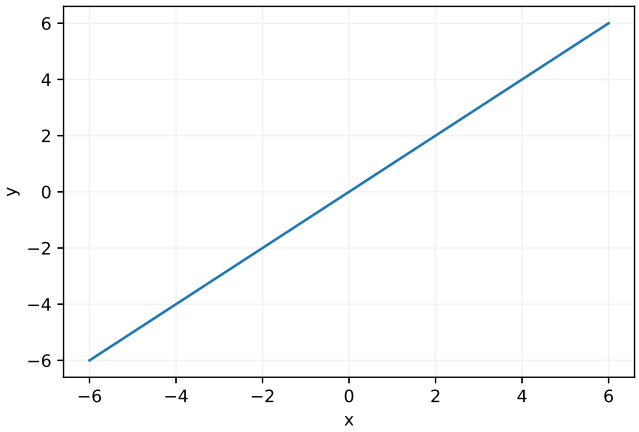


Figure 2.38. Linear function.

Sigmoid

The sigmoid function is defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.40}$$

and outputs a real value between 0 and 1. Differently from the previous functions, this function is non-linear, as evident from Figure 2.39. The sigmoid function allows a NN to model non-linear mappings between inputs and outputs, and is suitable for the *backpropagation algorithm* since always continuously differentiable.
On the other hand, the sigmoid function has small variations in the output independently from the input around its saturation values, i.e. 0 and 1. This is because the value of the exponent $-x$ becomes infinitesimally small around those values, making this function a slow learner when the gradients are small. This problem is also known as the *vanishing gradient problem*.

The sigmoid function is commonly used when the output is required to be between 0 and 1, making it very suitable to capture a probability distribution, for example in a binary classification model.

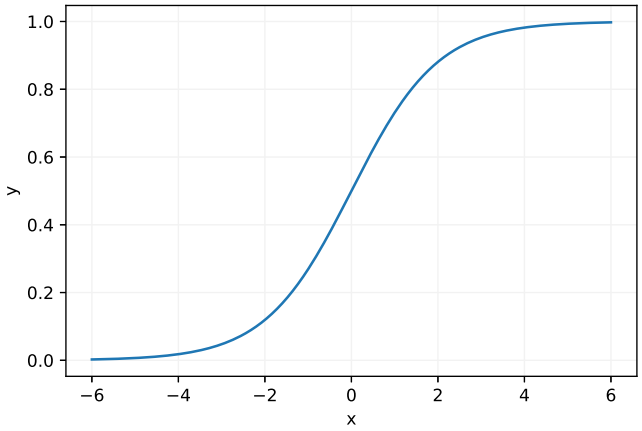


Figure 2.39. Sigmoid function.

Softmax

The softmax function is the generalization of the sigmoid and is defined as

$$f(x,i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}. \tag{2.41}$$

This function can model a probability distribution over N different classes (or more in general outcomes), making the sum of the probabilities of all classes sum up to 1. It provides a probability distribution between 0 and 1 for every class and is commonly used in multi-classification models. In particular, the formula 2.41 expresses the probability relative to the i-th class.

Hyperbolic Tangent

The hyperbolic tangent is a variant of the sigmoid function defined as

$$\tanh(x) = \frac{2}{1+e^{-2x}} - 1 = 2 \cdot \sigma(2x) - 1 \tag{2.42}$$

where the output values are limited between -1 and 1 as shown in Figure 2.40. This function is non-linear and continuously differentiable. Even though the gradient is stronger for the hyperbolic tangent than the sigmoid, it still suffers from the *vanishing gradient problem*. Since the two functions are very similar, the choice between the sigmoid and the hyperbolic tangent highly depends on the application requirements.

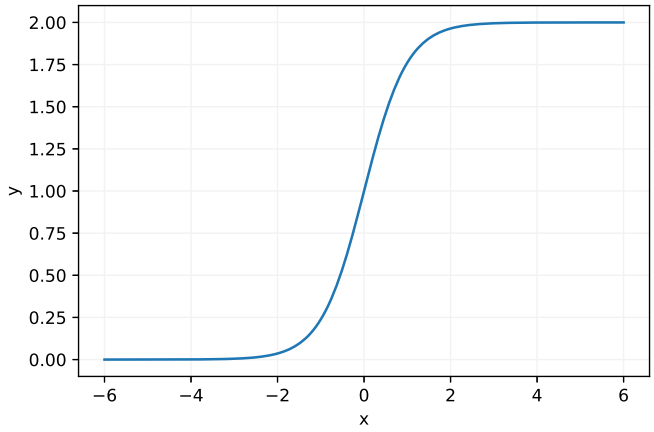


Figure 2.40. Hyperbolic tangent function.

Rectified Linear Unit

The Rectified Linear Unit (ReLU) is defined as

$$ReLU(x) = max(0, x) \tag{2.43}$$

and is shown in Figure 2.41. For negative inputs, it outputs 0, while for positive inputs it becomes the identity function. ReLU is non-linear even though it has a linear component for positive inputs and can be used to learn and model non-linearities. Moreover, since half of the input space maps to zero, this function allows fewer neurons to fire, making the network computationally lighter.
When inputs approach zero or are negative, the gradients become zero, preventing the network to learn through the *backpropagation algorithm*. When this happens, the involved neurons will stop responding to input variations, causing them to "die" and making part of the network passive and unresponsive. This problem is also known as the *dying ReLU problem*.

Leaky ReLU

To overcome the *dying ReLU problem*, other activation functions were proposed like the Leaky ReLU (shown in Figure 2.42) which is defined as

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{if } x < 0 \end{cases} \tag{2.44}$$

For non-negative input values, it becomes the identity function (same as ReLU), while, in the negative region, it outputs a scaled version of the negative inputs based on the slope α (usually in the order of 0.01) of the function. In this way, the weights of the network can be adjusted even when the inputs approach zero or are negative, preventing neurons from dying.

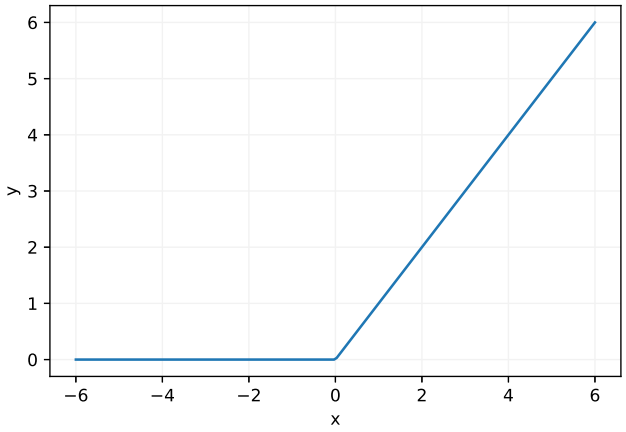


Figure 2.41. ReLU function.

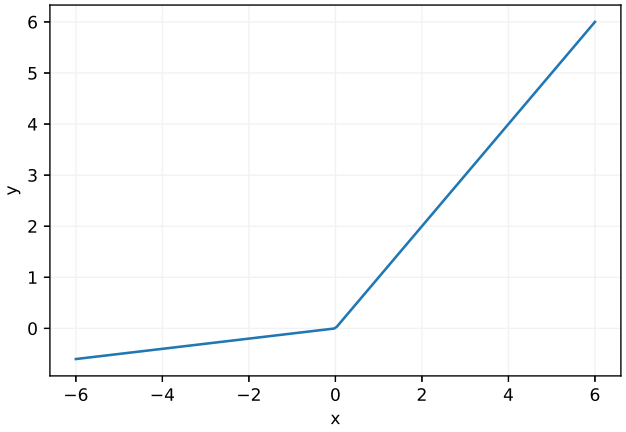


Figure 2.42. Leaky ReLU function.

Exponential Linear Unit

The Exponential Linear Unit (ELU), shown in Figure 2.43), is defined as

$$ELU(x) = \begin{cases} x, \text{ if } x \geq 0 \\ \alpha(e^x - 1), \text{ if } x < 0 \end{cases} \tag{2.45}$$

This function lies in between the ReLU function and the Leaky ReLU function. It has a positive linear component and for negative inputs, it produces negative outputs. Differently from the Leaky ReLU, ELU, after a slow smoothing produced by its exponential component, will converge to $-\alpha$ and the gradients will become zero like ReLU.

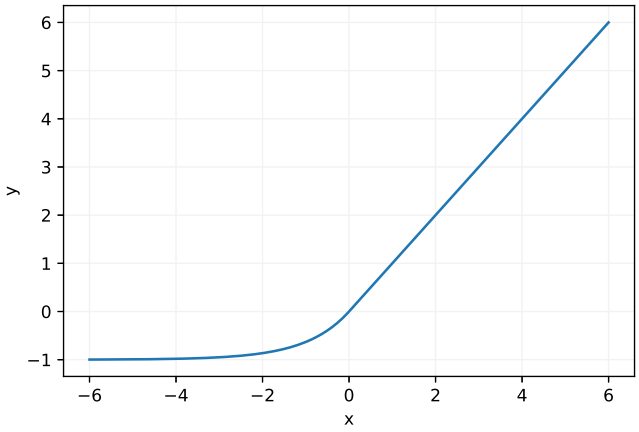


Figure 2.43. ELU function.

2.2.3.3 Loss Functions

To train a NN it is fundamental to define a proper loss function quantifying the difference between the ground truth and the predictions produced by the model. Also referred to as *cost function* or *error function*, it provides insight into the performance of the network, where higher values reflect poor predictions and lower values better ones. The cost function reduces the performance of the whole model down to a single number, a scalar value, which allows candidate solutions to be ranked and compared. Importantly, the choice of the loss function is directly related to the activation function used in the output layer of the NN. Therefore, it becomes crucial that the activation function can map the extracted features to a space where the loss function can properly quantify the network's prediction error.
The objective during the training process is to minimize the chosen loss function by computing its gradients with respect to the network weights which are adjusted accordingly. How weights are updated at each iteration is discussed in Section 2.2.3.4. Many different loss functions can be chosen depending on the application domain and the task.

Mean Squared Error

The Mean Squared Error (MSE), also known as $L2$ loss, is used in regression tasks where the targets and the model's predictions are real-valued. The MSE is computed as the average of the squared differences between the predicted values $\hat{y}_i$ and target values y_i and is formulated as

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{2.46}$$

The result is always non-negative independent from the sign of the predicted and actual values. Since the errors are squared, large deviations from the ground truth generate large errors, thus punishing the model for making large mistakes.

It is important to choose a proper activation function for the output layer of the network to map the predictions to the same space as the target values. If, for example, the target values range between -1 and 1, the Sigmoid function (see Equation 2.40) would be unsuitable since mapping the predicted values between 0 and 1. When target values are negative, the MSE would be greater or equal to the squared of the target values since the predictions can assume 0 as the smallest value. In this case, the hyperbolic tangent (see Equation 2.2.3.2) would be more suitable since mapping the outputs between -1 and 1, which coincides with the range of target values. These considerations are valid for all losses used in regression tasks.

Mean Absolute Error

The Mean Absolute Error (MAE), also known as $L1$ loss, is a loss function for regression problems. The MAE is defined as the average of the absolute difference between the actual and predicted values and is computed as

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i). \tag{2.47}$$

Differently from MSE, the MAE does not square the errors and, therefore, is more robust to outliers. On the other hand, the magnitude of the gradients is not dependent on the error size, but only on its sign, leading to large gradients even when the error is small, which can potentially lead to convergence problems for the network.

Huber Loss

The Huber loss is also typically used in regression problems. This function is less sensitive to outliers than the MSE as it squares the errors only inside a predefined interval. The error of a prediction $\hat{y}_i$ with respect to the ground truth y_i can be written as

$$L_\delta = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2, & \text{if } |y_i - \hat{y}_i| < \delta \\ \delta((y_i - \hat{y}_i) - \frac{1}{2}\delta), & \text{otherwise} \end{cases} \tag{2.48}$$

For loss values less than δ, the MSE is computed. For errors greater than δ, instead, the MAE is considered. In this way, the Huber loss function combines the advantages of both MAE and MSE, where the hyperparameter δ regulates the range of values for which the MAE and the MSE are computed. Figure 2.44 presents the MSE, MAE and Huber loss plotted together.

Binary Cross Entropy

The Binary Cross-Entropy (BCE) is a loss function used for binary classification problems where the target values can assume only two categorical values (0 or 1). The BCE calculates a score that summarizes the average difference between the actual and predicted probability distributions for predicting class 1 and can be expressed as

$$\text{BCE} = -\frac{1}{n}\sum_{i=1}^{n} y_i \cdot \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i). \tag{2.49}$$

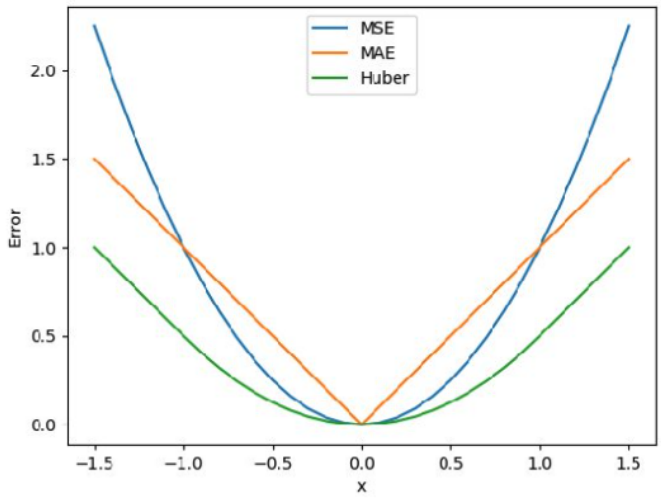


Figure 2.44. MSE, MAE and Huber losses [249].

Sigmoid is the only activation function compatible with the BCE loss function since this last needs to compute the logarithms of $\hat{y}_i$ and $(1 - \hat{y}_i)$, which are only defined when $\hat{y}_i$ is between 0 and 1. For this reason, the output layer of the network should have the sigmoid function (see Equation 2.40) when using the BCE loss function.

Categorical Cross Entropy

The Categorical Cross-Entropy (CCS) can be seen as a generalization of the BCE when the target classes are more than two. In fact, this loss function is used in multi-class classification tasks where a sample belongs to one of many possible categories. The CCS calculates a score that summarizes the average difference between the actual and predicted probability distributions for all classes in the problem and can be written as

$$\text{CCS} = -\sum_{i=1}^{n} y_i \cdot \log \hat{y}_i. \tag{2.50}$$

The softmax function defined in Equation 2.2.3.2 is the most suitable activation function to use with the CCS since the output of the network needs to be positive so that the logarithm of every output value $\hat{y}_i$ exists. Moreover, the softmax activation function also provides a proper probability distribution for all classes.

Kullback Leibler Divergence

The Kullback Leibler Divergence (KL-Divergence) measures the similarity between two probability distributions [250]. This loss function calculates how much information is lost if the predicted probability distribution is used to approximate the target distribution. A zero KL-Divergence loss suggests the distributions are identical.
The KL-Divergence between two distributions Q and P of a discrete random variable X can be written as

$$KL(P||Q) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)}, \tag{2.51}$$

For distributions Q and P of a continuous random variable, instead, the KL-Divergence is expressed as

$$KL(P||Q) = \int_{-\infty}^{+\infty} p(x) \log \frac{p(x)}{q(x)} dx, \tag{2.52}$$

where q and p are the probability density functions of Q and P, respectively.

2.2.3.4 Optimizers

Optimizers are widely used in the field of DL to minimize the loss functions defined in Section 2.2.3.3 and adjust the network weights based on the model prediction errors. The process of parameter optimization is usually associated with the image of a hiker trying to get down a mountain with a blindfold on, where the mountain represents the chosen loss function. The hiker is unable to see the destination downhill (global minima) but can tell if going down (making progress) or going up (losing progress). Eventually, if the hiker keeps taking small steps that lead downwards, it will reach its destination or a plateau (local minima).

Gradient Descent

Gradient Descent (GD) is one of the first algorithms employed for NN optimization and is very popular. GD is a first-order optimization algorithm that considers the first-order derivative of the loss function to compute the weight adjustment necessary to reach a minimum of the error function. The value of a weight $w_{t,i}$ in the network is updated at every iteration t as

$$w_{t+1,i} = w_{t,i} - \alpha \frac{\partial L}{\partial w_{t,i}}, \tag{2.53}$$

where $\frac{\partial L}{\partial w_{t,i}}$ is the partial derivative of the chosen loss function L with respect to the weight $w_{t,i}$. The weights of the whole network are optimized as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla_{\mathbf{w}_t} L, \tag{2.54}$$

where $\nabla_{\mathbf{w}_t} L$ is the gradient of L and denotes the direction of the maximum rate of change of the loss function. The hyperparameter α is the learning rate which represents the step size during the optimization process. Without the learning rate as a scaling factor for the gradients, the optimizer could take steps that are too large and skip over the optimal value for a given weight. For this reason, the learning rate has to be sufficiently small to allow convergence, but also sufficiently large to converge to the optimum as fast as possible. This parameter is usually set in the order of 0.001. Figure 2.45 shows how different learning rates can lead or not to convergence.

The GD algorithm computes the gradients on the whole dataset and, therefore, is extremely demanding both in terms of time and memory when the dataset is too large.

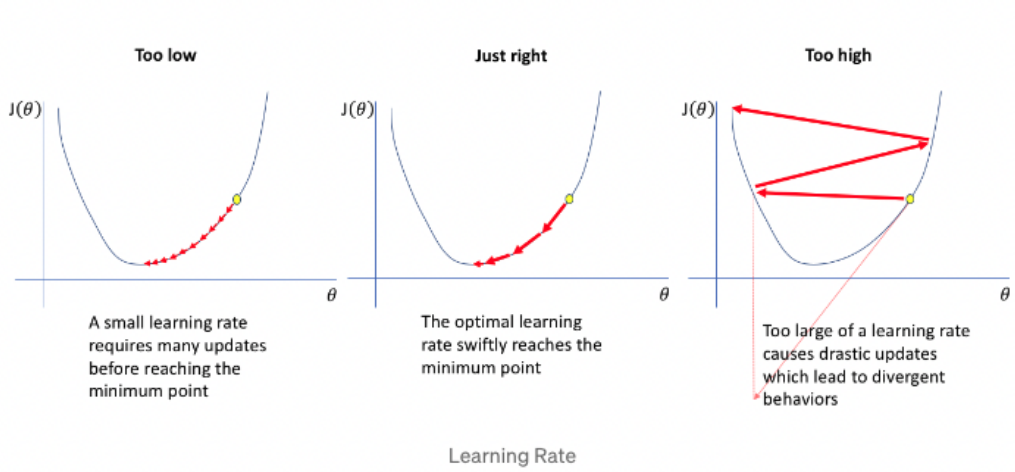


Figure 2.45. Comparison between different learning rates [251].

Stocastic Gradient Descent

Stochastic Gradient Descent (SGD), instead of considering the whole dataset, randomly selects a subset of the training samples at each iteration [252]. Frequent adjustments computed for random subsets of the dataset can lead to high variance in the network's weights and cause high loss fluctuations when using SGD.

Mini-Batch Stocastic Gradient Descent

Mini-Batch Stochastic Gradient Descent (Mini-Batch SGD) splits the dataset into multiple batches and updates the network weights after every batch. This creates a balance between the robustness of GD and the efficiency of SGD, reducing the variance of the parameters and stabilizing convergence. Since the algorithm uses batching, the entire dataset is not required to be loaded in the memory, thus making the process more efficient to implement.

Stocastic Gradient Descent with Momentum

SGD with momentum accelerates the convergence towards the relevant direction, thus reducing loss fluctuations and the number of iterations required to reach the optimal minimum. To do so, it adds a momentum term to regular SGD, simulating the inertia of an object in movement. In this way, the direction of the previous update is retained to a certain extent during the update, while the current gradient is used to correct the final direction. The network weights are updated as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{v}_t, \tag{2.55}$$

where the velocity vector $\mathbf{v}$ is computed as

$$\mathbf{v}_t = \beta \mathbf{v}_{t-1} + \alpha \nabla_{\mathbf{w}_t} L \tag{2.56}$$

β is an extra hyperparameter regulating the amount of momentum considered in the optimization process and can be interpreted as friction to control the velocity.

Adaptive Gradient Descent

All the optimizers discussed previously are limited by the fact that a single constant learning rate is used to update all parameters during the entire optimization process. Adaptive Gradient Descend (AdaGrad), instead, adapts the learning rate with larger updates for those parameters that are related to infrequent features and smaller updates for frequent ones [253]. AdaGrad updates the network weights as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \boldsymbol{\alpha}_t \nabla_{\mathbf{w}_t} L, \tag{2.57}$$

where the adaptive learning rate vector is computed as

$$\boldsymbol{\alpha}_t = \frac{\alpha}{\sqrt{\epsilon + \sum_{j=1}^{t} (\nabla_{\mathbf{w}_j} L)^2}}, \tag{2.58}$$

α is the initial learning rate hyperparameter and ϵ is a simple smoothing term that avoids division by zero.
With AdaGrad the learning rate decreases aggressively and monotonically because the squared gradients in the denominator keep accumulating at each iteration. In this way, due to small learning rates, the model eventually becomes unable to learn and the accuracy of the model is compromised.

Root Mean Square Propagation

Root Mean Square Propagation (RMSProp) is a special case of AdaGrad since the learning rate is an exponential average of the gradients instead of the cumulative sum of squared gradients. RMSProp combines the concept of momentum with AdaGrad and updates the parameters as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \boldsymbol{\alpha}_t \nabla_{\mathbf{w}} L, \tag{2.59}$$

where the adaptive learning rate vector is computed as

$$\boldsymbol{\alpha}_t = \frac{\alpha}{\sqrt{\epsilon + \mathbf{v}_t}} \tag{2.60}$$

and the vector $\mathbf{v}_t$ as

$$\mathbf{v}_t = \rho \mathbf{v}_{t-1} + (1 - \rho)(\nabla_{\mathbf{w}_t} L)^2. \tag{2.61}$$

ρ is the hyperparameter regulating the exponential smoothing and is usually set in the order of 0.9.

Adaptive Moment Optimization

Adaptive Moment Optimization (Adam) is an extension of SGD that updates the learning rate for each weight individually as done by AdaGrad or RMSProp [254]. In addition, instead of adapting learning rates based on the first moment of the gradients, it also uses the second moment, meaning that it computes not only the exponential average of the gradient but also the exponential average of the square gradients. Adam updates the weights as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \frac{\hat{\mathbf{m}}_t}{\sqrt{\epsilon + \hat{\mathbf{v}}_t}}, \tag{2.62}$$

with

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1-\beta_1^t} \tag{2.63}$$

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1-\beta_2^t} \tag{2.64}$$

and

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\nabla_{\mathbf{w}_t} L \tag{2.65}$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)(\nabla_{\mathbf{w}_t} L)^2 \tag{2.66}$$

Since the exponential average of gradients $\mathbf{m}_t$ and the exponential average of square gradients $\mathbf{v}_t$ are both initialized as 0 at the first iteration, they will be biased towards 0 as both β_1 and β_2 are usually chosen close to 1. That is why the Adam algorithm uses $\hat{\mathbf{m}}_t$ and $\hat{\mathbf{v}}_t$ in the update formula, namely a bias-corrected version of $\mathbf{m}_t$ and $\mathbf{v}_t$. The hyperparameters β_1 and β_2 which regulate the decay rate of the average of the gradients are usually set to 0.9 and 0.999, respectively.
Adam is extensively used in countless applications and is adopted as a benchmark for DL papers and recommended as a default optimization algorithm. Moreover, the method is straightforward to implement, has a fast running time, low memory requirements, and requires less tuning than most of the other optimization algorithms.

2.2.3.5 Backpropagation Algorithm

This section discusses the backpropagation algorithm, the most popular method used to compute the gradients during the optimization process of a NN [255]. The algorithm follows the use of the chain rule and product rule in differential calculus. These rules depend on the differentiation of the activation functions, therefore discontinuous and non-differentiable ones like the step function discussed in Section 2.2.3.2 are avoided.
The derivation of the backpropagation algorithm begins by applying the chain rule to the partial derivatives of the error function L. Given the weight w_{ij}^l for node j in layer l for the incoming node i, the partial derivative of the loss function L with respect to the weight can be written as

$$\frac{\partial L}{\partial w_{ij}^l} = \frac{\partial L}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{ij}^l}, \tag{2.67}$$

where z_j^l is the weighted sum of node j in layer l before it is passed to the activation function and is expressed as

$$z_j^l = \sum_{k=1}^{m^{l-1}} w_{kj}^l a_k^{l-1} + b_j^l, \tag{2.68}$$

being m^{l-1} the number of neurons in layer $l-1$ and b_j^l the bias. The activation a_j^l of neuron j in layer l is defined as

$$a_j^l = \sigma(z_j^l), \tag{2.69}$$

where σ is a generic activation function. An example of a 3-layer network is provided in Figure 2.46.

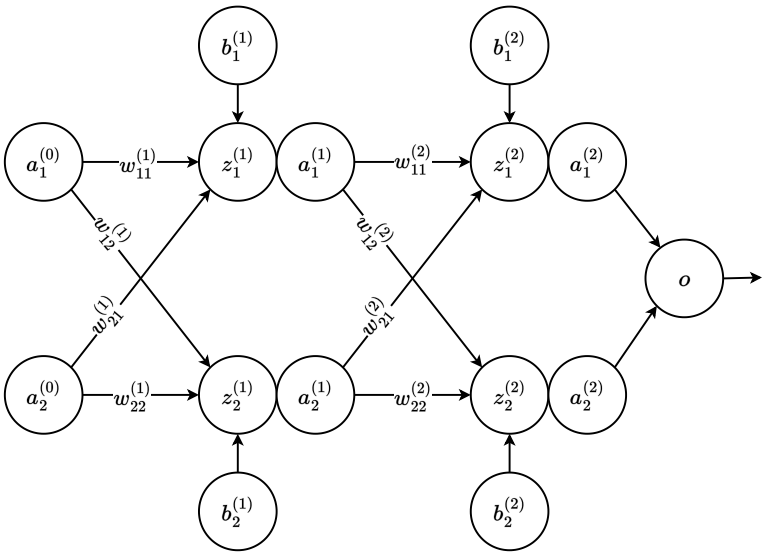


Figure 2.46. Example of a NN with two hidden layers.

Equation 2.67 says that the change in the loss function due to the weight w_{ij}^l is equal to the product of the change in the loss function due to the weighted sum z_j^l times the change in the weighted sum z_j^l due to the weight w_{ij}^l.
The first term is referred to as *error term* and can be written as

$$\delta_j^l = \frac{\partial L}{\partial z_j^l}, \tag{2.70}$$

while the second term as

$$\frac{\partial z_j^l}{\partial w_{ij}^l} = \frac{\partial}{\partial w_{ij}^l}(\sum_{k=1}^{m^{l-1}} w_{kj}^l a_k^{l-1} + b_j^l) = a_i^{l-1}. \tag{2.71}$$

In this way, Equation 2.67 can be rewritten as

$$\frac{\partial L}{\partial w_{ij}^l} = \delta_j^l a_i^{l-1} \tag{2.72}$$

so that the partial derivative of w_{ij}^l is the product of the error term δ_j^l at node j in layer l, and the activation a_i^{l-1} of node i in layer $l-1$.
The error term δ_j^l at layer l depends on the errors δ_k^{l+1} at the next layer $l+1$ as follows

$$\delta_j^l = \sum_{k=1}^{m^{l+1}} w_{jk}^{l+1} \delta_k^{l+1} \sigma'(z_j^l), \tag{2.73}$$

where σ' is the derivative of the activation function used in the network layers. Due to this recursive formulation, the errors flow backwa1rd, from the last layer to the first layer.

2.2.3.6 Vanishing and Exploding Gradients

As the backpropagation algorithm propagates the errors from the output layer toward the input layer, the gradient can become smaller and approach zero, thus leaving the weights of the first layers nearly unchanged. This phenomenon is referred to as *vanishing gradients* and can lead the optimization algorithm to never converge to the optimum [256].
On the contrary, when the gradients get larger as the backpropagation algorithm progresses, large weight updates can occur and cause the optimization algorithm to diverge. This is known as the *exploding gradients* problem [257]. One way to tackle this problem is the *gradient clipping* technique, where gradients larger than a specific threshold are clipped and set to another value.

2.2.3.7 Initalizations

When designing a NN, the weight initialization is usually considered a minor concern, even though it has serious effects on the learning phase. Tuning hyperparameters like the number of layers, neurons and activation functions is fundamental when defining a NN, together with the choice of the loss function and the optimizer. Nevertheless, the initialization of the network's weights can determine whether the algorithm converges at all, with some initial points being so unstable that the algorithm encounters numerical difficulties and fails altogether [258]. The goal is to prevent exploding or vanishing gradients during the training phase of the NN. Current weight initialization techniques vary based on the activation function used in the nodes that are being initialized.

Uniform Xavier The standard initialization of the weights belonging to nodes that use the sigmoid or hyperbolic tangent activation functions is called *Glorot* or *Xavier* [257]. This initialization computes the initial value of the weight w of a node as

$$w \sim U[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}], \tag{2.74}$$

where U is the uniform probability distribution and n is the number of inputs to the node.
To ensure that the variance of the outputs is roughly equal to the variance of the inputs with the goal to avoid the vanishing or exploding gradients problems, the initial value of the weight w of a node can be computed as

$$w \sim U[-\frac{\sqrt{6}}{\sqrt{n+m}}, \frac{\sqrt{6}}{\sqrt{n+m}}], \tag{2.75}$$

where U is the uniform probability distribution, n is the number of inputs to the node (the number of nodes in the previous layer) and m is the number of outputs from the layer (the number of nodes in the current layer).

Normalized Xavier The normalized Xavier initialization is a variant of the Uniform Xavier initialization that computes the initial value of the weight w of a

node as

$$w \sim \mathcal{N}(0, \frac{2}{n+m}), \tag{2.76}$$

where the value is drawn from a Gaussian distribution with zero mean and $\sqrt{\frac{2}{n+m}}$ as standard deviation.

He The Xavier initialization was found to be problematic when used in combination with the ReLU activation function, popular in the hidden layers of many NNs. For this reason, a modified version of the Xavier initialization was developed specifically for nodes and layers that use the ReLU activation function, called the *He* initialization [259]. This initialization computes the initial value of the weight w of a node as

$$w \sim \mathcal{N}(0, \frac{2}{n}), \tag{2.77}$$

where the value is drawn from a Gaussian distribution with zero mean and $\sqrt{\frac{2}{n}}$ as standard deviation, being n the number of inputs to the node.

2.2.3.8 Multi-Task Learning

Considering a single task during the training of a network can lead the model to ignore relevant information. When the network is trained to perform different tasks and shares representations and weights among tasks, the model might better generalize on the original task. This approach is called Multi-Task Learning (MTL) [260] [261]. In general, dealing with more than one loss function can be considered as MTL.
This approach can be interpreted as a way of introducing an *inductive bias* which induces a model to prefer some hypotheses over others. In the case of MTL, the inductive bias is provided by the auxiliary tasks and leads the model to prefer hypotheses that satisfy multiple tasks, generally leading to solutions that generalize better. When the model is a DNN, MTL is usually performed with either *hard* or *soft* parameter sharing between hidden layers.

Hard Parameter Sharing Hard parameter sharing works by sharing the hidden layers between all tasks, while every task has a separate output layer on its own, as shown in Figure 2.47. When learning multiple tasks simultaneously, the network has to adjust its parameters to capture all tasks at the same time. For this reason, hard parameter sharing can reduce the risk of overfitting.

Soft Parameter Sharing Soft parameter sharing consists in having separate parameters for every task as shown in Figure 2.48. To force the parameters of all separate tasks to be similar, the distance between the weights of the model is regularized, for example through a $L2$ regularization (explained in Section 2.2.4.2).

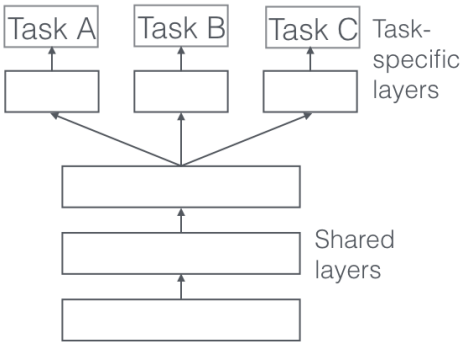


Figure 2.47. MTL with hard parameter sharing [262].

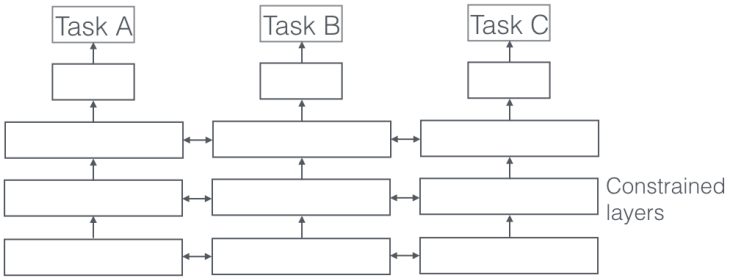


Figure 2.48. MTL with soft parameter sharing [262].

2.2.3.9 Types of Neural Networks

This section discusses different families of networks and provides an overview of the most popular types of NNs.

2.2.3.10 Feedforward Neural Networks

The Feedforward Neural Networks (FNN) was the first and simplest type of ANN where the information only flows in one direction (forward), from the input nodes to the output nodes passing through the hidden nodes. There are no cycles or loops in the network's connections.

Multi-layer Perceptron A Multi-Layer Perceptron (Multi-Layer Perceptron) is a FNN which is composed of a series of fully connected layers of neurons. MLPs are very flexible and can be used to learn a mapping from inputs to outputs for multiple tasks, from classification to regression. The number of neurons in each layer, the number of layers and the activation functions used in the network are *hyperparameters* that have to be tuned. An example of a MLP having one hidden layer with five neurons is provided in Figure 2.49.

2.2.3.11 Recurrent Neural Networks

FNNs are not designed to take into account the temporal correlation between inputs. This family of networks can not take into account the order in which the inputs are

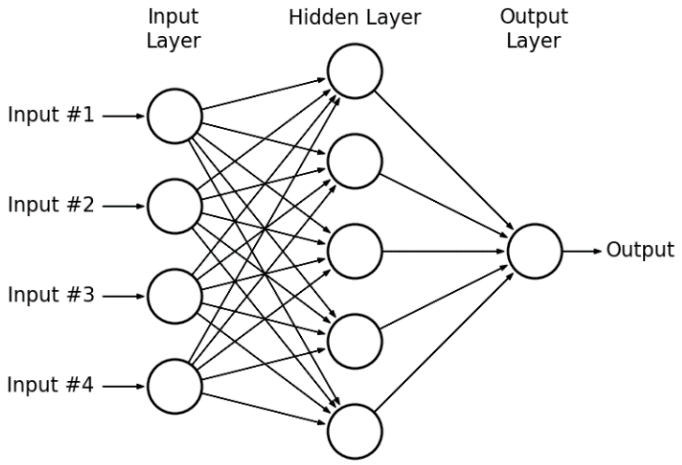


Figure 2.49. Example of MLP having one hidden layer with five neurons [263].

fed and, therefore, is unable to capture the correlation between successive inputs. It is sufficient to think about a video composed of frames. A FNN would produce an output for every single frame, but would not capture the correlation between successive frames.
In order to overcome this problem, Recurrent Neural Network (RNN) were introduced [102]. A RNN can model a sequence of t inputs and at every time step i of the sequence it takes into account not only the i-th element of the input sequence but also the state of the network from the previous time step. This allows the network to capture the temporal correlation between inputs of the same sequence.
A RNN could be seen as a FNN having a self-loop, as shown in Figure 2.50, which allows the network to feed itself its internal state across time steps. The internal state can be considered the memory of the network used to capture temporal dependencies.

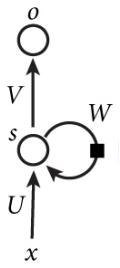


Figure 2.50. RNN with a self-loop [264].

It is possible to unroll the RNN self-loop, as shown in Figure 2.51, and consider the network as a FNN which takes in input each element of the input sequence together with the hidden internal state of the previous time step.

At every time step t we have that:

- $\mathbf{x_t}$ is the t-th element of the input sequence, i.e. the input at time step t.
- $\mathbf{h_{t-1}}$ is the hidden state of the network at the previous time step $t-1$.

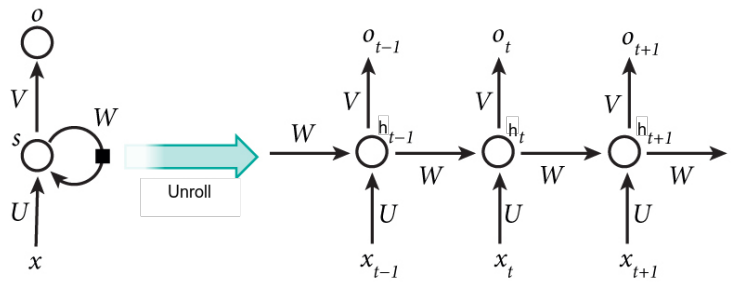


Figure 2.51. Unrolled RNN [264].

- $\mathbf{h_t} = \sigma(U \cdot \mathbf{x_t} + W \cdot \mathbf{h_{t-1}})$ is the new hidden state at time step t, where σ is an activation function.
- $\mathbf{o_t} = \sigma(V \cdot \mathbf{h_t})$ is the output of the network at time step t, where σ is an activation function.
- U, V, W are trainable weight matrices shared across all the time steps.

RNNs are trained using an extension of the backpropagation algorithm called *backpropagation through time* [265]. The temporal component is reduced to ordered series of calculations linking one time step to the following, allowing the backpropagation algorithm to optimize the network's weights. This is because a RNN, when unrolled, is nothing more than nested composite functions. Introducing the temporal component only extends the series of functions for which derivatives are calculated through the chain rule.

Bidirectional Recurrent Neural Networks A Bidirectional Recurrent Neural Network (BRNN) is obtained by combining two separate RNNs as shown in Figure 2.52 [107].

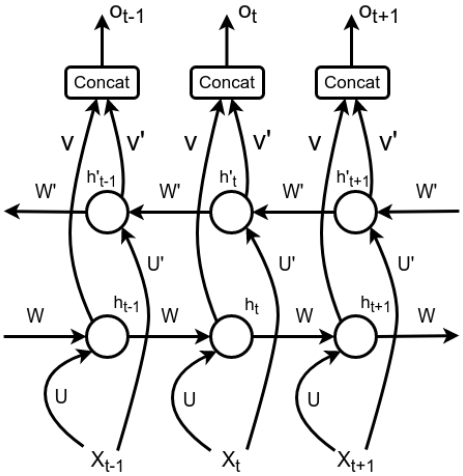


Figure 2.52. Structure of a BRNN.

The original input sequence is fed to one of the RNNs, and the sequence in reverse time order is fed to the other one. The outputs of the two networks are

usually concatenated, generating a single merged output at every time step. This particular structure allows the network to have both information from the past and the future of the sequence simultaneously at every time step. The RNN that receives the original sequence as input and provides information about the past is called the *backward network*, while the one receiving in input the sequence in reverse time order providing information about the future is called the *forward network*.
With this structure, at every time step t we have that:

- $\mathbf{x_t}$ is the t-th element of the input sequence, i.e. the input at time step t. $\mathbf{x_t}$ is fed to both the RNNs.
- $\mathbf{h_{t-1}}$ and $\mathbf{h'_{t-1}}$ are, respectively, the hidden states of the backward and forward networks at the previous time step $t-1$.
- $\mathbf{h_t} = \sigma(U \cdot x_t + W \cdot \mathbf{h_{t-1}})$ and $\mathbf{h'_t} = \sigma'(U' \cdot \mathbf{x_t} + W' \cdot \mathbf{h'_{t-1}})$ are the new hidden states of the networks at time step t, where σ and σ' are activation functions.
- $\mathbf{o_t} = Concat(\sigma(V \cdot \mathbf{h_t}), \sigma'(V' \cdot \mathbf{h'_t}))$ is the output of the whole network at time step t, obtained by concatenating the output of the two networks at time step t.
- U, V, W are the trainable weight matrices shared across all the time steps of the backward network, while U', V', W' are the weight matrices of the forward network.

BRNNs are trained similarly to normal RNNs since the two networks are separated and do not interact with each other. However, during backpropagation, some attention is required since updating input and output layers can not be done at once.

Long-Short Term Memory As discussed in Section 2.2.3.2, activation functions can map large input spaces into small input spaces, e.g. between 0 and 1. Therefore, a large change in the input of the activation function may cause a small change in the output, leading to small derivatives. Considering a DNN with multiple hidden layers all having a final activation function, small derivatives are multiplied together at every step of the backpropagation algorithm. This leads the gradient to decrease exponentially the more it propagates back to the input layers. This phenomenon is also known as the *vanishing gradient problem* (discussed in Section 2.2.3.6 and affects both FNNs as well as RNNs when dealing with long sequences [102, 266].
In general, the gradient expresses how much the weights of the network need to be adjusted with respect to the variation of the error. When the gradients are too small to adjust the weights significantly, it becomes very difficult for the optimizer to move in the direction where the error decreases, preventing the network to learn. Another issue that affects these networks is the so-called *exploding gradient problem* (also discussed in Section 2.2.3.6). When gradients are too large and cumulate across layers, they produce large updates to the weights of the network and can lead to an unstable model with poor performance. Proper training is possible when gradients are small (but not too small) and controlled.
T face these problems, Long Short-Term Memory (LSTM) cells were proposed [104]. These particular cells replace the standard memory cell of a standard RNN (as shown

in Figure 2.53) and help to preserve and control the gradient when backpropagated through time.

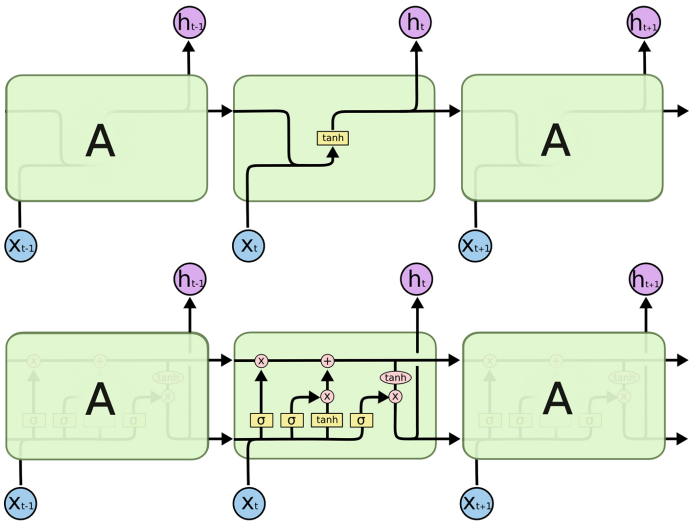


Figure 2.53. On the top the standard RNN. On the bottom, a RNN with an LSTM cell [267].

LSTMs use a *gating system* which allows them to store, write or read from the memory of the cell. Gates block or pass on information based on its magnitude and importance, through a filtering system composed of their own sets of weights which are adjusted during training. The internal structure of an LSTM cell is shown in detail in Figure 2.54.

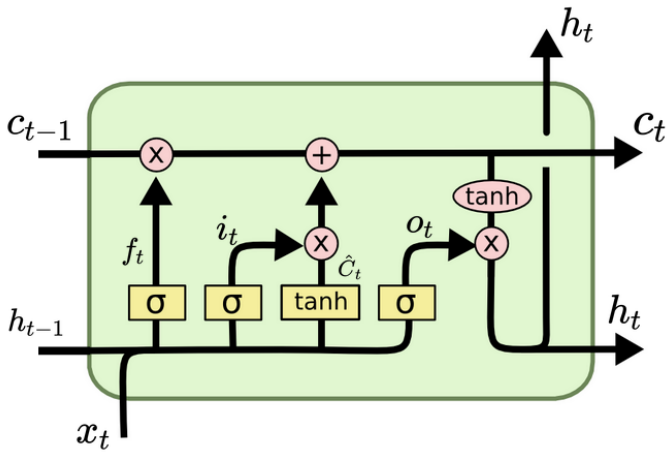


Figure 2.54. Internal structure of an LSTM cell [268].

At every time step, the cell decides how much information to keep or to discard from the cell state in the previous time step $\mathbf{c_{t-1}}$. This operation is performed through the *forget gate*

$$\mathbf{f_t} = \sigma(W_f \cdot [\mathbf{h_{t-1}}, \mathbf{x_t}] + b_f) \tag{2.78}$$

which outputs a number between 0 and 1 and determines the amount of information to keep. σ is the sigmoid activation function, $\mathbf{h_{t-1}}$ the output of the cell in the previous time step and $[\mathbf{h_{t-1}}, \mathbf{x_t}]$ the concatenation of $\mathbf{h_{t-1}}$ and $\mathbf{x_t}$ which is the current element of the input sequence. The output of the *forget gate* is multiplied by the previous cell state $\mathbf{c_{t-1}}$.
Then, the LSTM cell decides what and how much information to store in the current cell state. This is achieved by the *input gate*

$$\mathbf{i_t} = \sigma(W_i \cdot [\mathbf{h_{t-1}}, \mathbf{x_t}] + b_i) \tag{2.79}$$

which also outputs a number between 0 and 1 that determines how much information to add to the new cell state.
The candidate information that will be added to the new cell state is computed as

$$\hat{\mathbf{c}}_\mathbf{t} = tanh(W_c \cdot [\mathbf{h_{t-1}}, \mathbf{x_t}] + b_c) \tag{2.80}$$

and the new cell state is computed as

$$\mathbf{c_t} = \mathbf{f_t} \cdot \mathbf{c_{t-1}} + \mathbf{i_t} \cdot \hat{\mathbf{c}}_\mathbf{t} \tag{2.81}$$

The new candidate cell state vector $\hat{\mathbf{c}}_\mathbf{t}$ is weighted by the *input gate* $\mathbf{i_t}$ and added to what remains of the previous cell state $\mathbf{c_{t-1}}$ after passing through the *forget gate* $\mathbf{f_t}$. Finally, the output $\mathbf{h_t}$ of the cell is computed as a filtered version of the new cell state $\mathbf{c_t}$. First, $\mathbf{c_t}$ is passed through a *tanh* function to push the values between -1 and 1 and, then, it is multiplied by the *output gate*

$$\mathbf{o_t} = \sigma(W_o \cdot [\mathbf{h_{t-1}}, \mathbf{x_t}] + b_o) \tag{2.82}$$

which determines how much of the filtered version of the cell state $\mathbf{c_t}$ will contribute to the output vector. All the weights $W_f, b_f, W_i, b_i, W_o, b_o$ are adjusted during the training phase.

Gated Recurrent Unit The Gated Recurrent Unit (GRU) is a streamlined version of the LSTM cell that often achieves comparable performance but with the advantage of being faster to compute since having fewer parameters to train [269]. Differently from an LSTM cell, GRU lacks the *output gate* as shown in Figure 2.55.

At every time step, the cell decides how much information to keep or to discard from the previous hidden state $\mathbf{h_{t-1}}$. This operation is performed through the *reset gate*

$$\mathbf{r_t} = \sigma(W_r \cdot \mathbf{x_t} + U_r \cdot \mathbf{h_{t-1}} + b_r) \tag{2.83}$$

where σ is the sigmoid activation function. When off ($\mathbf{r_t}$ close to 0), the *reset gate* effectively makes the unit act as if it is reading the first element of the input sequence, allowing it to forget the previously computed state. This effectively allows the hidden state to drop any information that is found to be irrelevant later in the future, thus, allowing a more compact representation.
Then, the cell decides how much of the previous state $\mathbf{h_{t-1}}$ will contribute to the new state $\mathbf{h_t}$. This is achieved by the *update gate*

$$\mathbf{z_t} = \sigma(W_z \cdot \mathbf{x_t} + U_z \cdot \mathbf{h_{t-1}} + b_z) \tag{2.84}$$

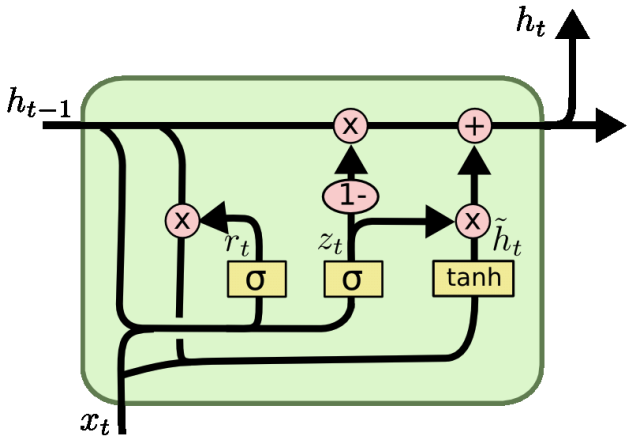


Figure 2.55. Internal structure of GRU [268].

The candidate information that will be added to the new hidden state is computed as

$$\hat{\mathbf{h}}_\mathbf{t} = tanh(W_h \cdot \mathbf{x_t} + U_h \cdot (\mathbf{r_t} \odot \mathbf{h_{t-1}}) + b_h), \tag{2.85}$$

where $\odot$ is an element-wise multiplication. Finally, the new hidden state is computed as

$$\mathbf{h_t} = \mathbf{z_t} \cdot \mathbf{h_{t-1}} + (1 - \mathbf{z_t}) \cdot \hat{\mathbf{h}}_\mathbf{t} \tag{2.86}$$

All the weights $W_r, U_r, b_r, W_z, U_z, b_z, W_h, U_h, b_h$ are adjusted during the training phase.

2.2.3.12 Attention Mechanism

Networks that deal with sequences often suffer from the incapability to remember long sequences. It is common that after a certain number of time steps it forgets part of the information relative to earlier elements in the sequence. *Attention mechanisms* were introduced to address this issue [270, 271].

Self-attention *Self-attention* allows the network to look at past and future information, even if far away, to help the network to produce better encodings for the elements in the input sequence [106]. To compute *self-attention* it is necessary to create 3 matrices W^Q, W^K, W^V which are trainable during the learning phase. Then, a *Query* vector ($\mathbf{q_i}$), a *Key* vector ($\mathbf{k_i}$) and a *Value* vector ($\mathbf{v_i}$) are computed for each of the input vectors $\mathbf{x_i}$ as

$$\mathbf{q_i} = \mathbf{x_i} \cdot W^Q, \tag{2.87}$$

$$\mathbf{k_i} = \mathbf{x_i} \cdot W^K, \tag{2.88}$$

and

$$\mathbf{v_i} = \mathbf{x_i} \cdot W^V. \tag{2.89}$$

These three vectors are abstractions necessary for calculating and understanding the concept of attention.
A *score* for every element $\mathbf{x_i}$ against every element in the sequence is computed by

performing the dot product between the *Query* vector $\mathbf{q_i}$ of the i-th element and the *Key* vectors of all the elements, one at the time. This *score* determines how much focus to place on other elements in the input sentence to produce an encoding for the current element. The score between the current element i and another element j of the sequence is computed as

$$\mathbf{s_{i,j}} = \mathbf{q_i} \cdot \mathbf{k_j}, \tag{2.90}$$

where $i = 1, \ldots, n$ and $j = 1, \ldots, n$, being n the number of elements in the sequence. Then, all the scores are divided by a constant $\sqrt{d_k}$ which is the square root of the dimension of the *Key* vectors to provide more stable gradients as follows

$$\mathbf{n_{i,j}} = \frac{\mathbf{q_i} \cdot \mathbf{k_j}}{\sqrt{d_k}}. \tag{2.91}$$

For every input $\mathbf{x_i}$, all the scores $n_{i,j}$ are passed through a softmax function which normalizes the scores making them positive and sum up to 1 as

$$\mathbf{s'_{i,k}} = \frac{e^{\mathbf{n_{i,k}}}}{\sum_{j=1}^{n} e^{\mathbf{n_{i,j}}}}, \tag{2.92}$$

where $\mathbf{s'_{i,k}}$ is the softmax score of the k-th element with respect to the current input element $\mathbf{x_i}$. For every input, the softmax score determines how important each element in the sentence is with respect to the current element. Of course, the most important score will be $\mathbf{s'_{i,i}}$, i.e. the importance of the input $\mathbf{x_i}$ with respect to itself. Afterward, normalized softmax scores are multiplied by the *Value* vectors of the elements in the sequence as

$$\mathbf{v'_{i,j}} = \mathbf{s'_{i,j}} \cdot \mathbf{v_j} \tag{2.93}$$

In this way, $\mathbf{v'_{i,j}}$ represents the original *Value* vector of the j-th element of the sequence scaled by its importance with respect to the current element of interest $\mathbf{x_i}$. The importance is established by the previously computed softmax score.
Finally, all the scaled *Value* vectors are summed up together to compute the new encoding for the input element $\mathbf{x_i}$ as

$$\mathbf{z_i} = \sum_{j=1}^{n} \mathbf{v'_{i,j}}. \tag{2.94}$$

All these computations can be performed through matrix calculations as shown in Figure 2.56. First, it is necessary to stack all the input vectors $\mathbf{x_i}$ to obtain a single matrix representation as

$$X = [\mathbf{x_1 x_2} \cdots \mathbf{x_n}]^T \tag{2.95}$$

Then, compute the *Query, Key* and *Value* matrices as

$$Q = X \cdot W^Q, \tag{2.96}$$

$$K = X \cdot W^K, \tag{2.97}$$

and

$$V = X \cdot W^V. \tag{2.98}$$

Finally, the new encodings are computed as

$$Z = softmax(\frac{Q \cdot K^T}{\sqrt{d_k}}) \cdot V \tag{2.99}$$

where

$$Z = [\mathbf{z_1 z_2} \cdots \mathbf{z_n}]^T. \tag{2.100}$$

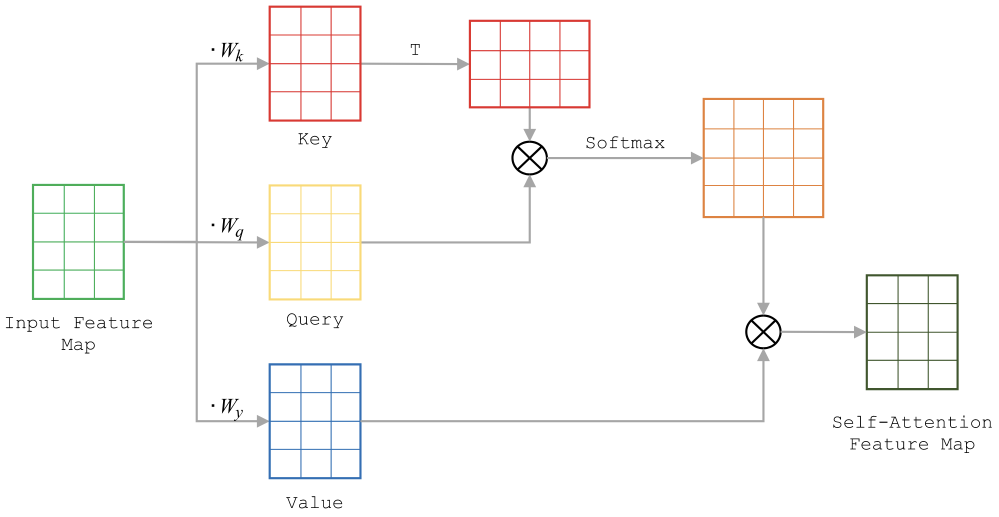


Figure 2.56. Matrix calculations to compute self-attention [272].

Multi-Head Attention Multi-Head Attention (MHA) is a module for attention mechanisms which computes *self-attention* several times in parallel [106]. The main idea is that there are multiple triples of *Query*, *Key*, and *Value* matrices, all randomly initialized. Every triple is used to project the input vectors to different representation subspaces which are combined to produce the final encodings.
Each triple (Q_i, K_i, V_i) is used to calculate a separate *self-attention* referred to as *head*. Considering h *heads*, the i-th head computes a different encoding of the input sequence as

$$Z_i = softmax(\frac{Q_i \cdot K_i^T}{\sqrt{d_k}}) \cdot V_i. \tag{2.101}$$

It is now necessary to condense the h encoding matrices by concatenation as

$$Z_c = Concat(Z_1, Z_2, \ldots, Z_h). \tag{2.102}$$

This new condensed matrix is multiplied by an additional weight matrix W_O, which linearly projects it to the final output space as

$$Z = Z_c \cdot W_0. \tag{2.103}$$

Notably, the final projected matrix

$$Z = [\mathbf{z_1 z_2} \cdots \mathbf{z_n}]^T \tag{2.104}$$

contains the encodings for all the elements in the original input sequence.

2.2.3.13 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a family of NNs that can process data having a regular and structured grid-like topology, such as images [273]. In fact, every image can be represented as a matrix of pixel values where *channels* refer to the components of an image. Standard images have three channels, namely red, green and blue (RGB), which can be imagined as three stacked 2D matrices, each having pixel values ranging from 0 to 255.
As for neurons in biological vision systems which respond only to stimuli in the restricted region of the visual field called the *receptive field*, also neurons in a CNN process data only in its *receptive field*. A CNN is structured in such a way that simpler patterns are detected in the shallow layers, like lines or curves, and more complex patterns in deeper layers, like faces or objects. Typically, a CNN has three kind of layers: *convolutional*, *pooling*, and FC.
Convolutional layers perform a dot product between the matrix of learnable parameters referred to as *kernel* and the restricted portion of input image delimited by the *receptive field* as shown in Figure 2.57.

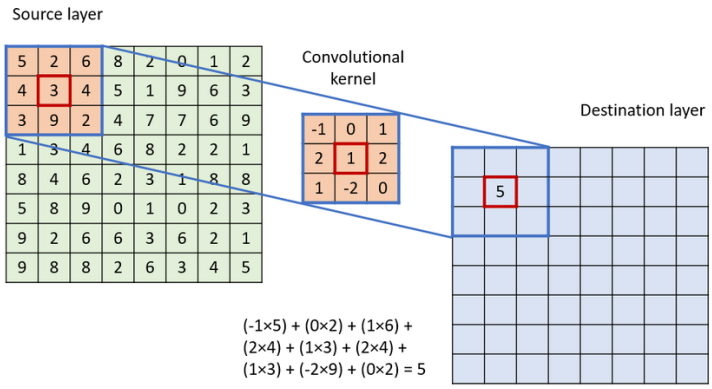


Figure 2.57. Operation performed in a convolutional layer [274].

The *kernel* is spatially smaller than the original image but extends to all *channels*. For example, if the image has three channels, the kernel height and width are spatially limited to the size of the *receptive field*, but the depth extends up to all three channels. During the forward pass, the *kernel* slides across the image from left to right until the complete width is parsed. Then, it slides down and continues from right to left and so on. This operation produces a new representation of the input that takes into account the locality of spatial features. The output is a 2D representation of the image known as an *feature map* that represents the response of the *kernel* at each spatial subregion of the input image. The sliding size of the kernel is called *stride*. To ensure that the output has the same width and height as the input image, it is common to use what is known as *padding* which consists of the addition of extra pixels on the borders of the input image (usually zero-values). The *padding* is typically one less than the *kernel* size. For example, if the *kernel* has size 3×3, a 2-pixel *padding* would be used, 1 pixel on each side of the image. Without *padding*, the border pixels are lost from the output as they participate in only a single *receptive field* instance. In Figure 2.57 the input image has size 8×8 and the output 6×6 if

no padding is added.
Being the *kernel* typically smaller than the input, instead of having a weight for each input value as for dense layers, fewer parameters are stored and shared among different *receptive field* instances, leading to sparse interaction in convolutional layers. In this way, not only the memory requirement of the model is reduced but also the statistical efficiency is improved. Due to parameter sharing, these layers have the property of equivariance to translation, meaning that if the input is changed, the output changes accordingly.
Pooling layers are responsible for reducing the spatial size of the *feature maps*. Not only do they reduce the complexity of the model, but also contribute to extracting dominant features which are rotational and positional invariant. The main types of pooling layers are *Max Pooling* and *Average Pooling*. Max Pooling returns the maximum value from the portion of the image covered by the *kernel*, while Average Pooling computes the average.
FC layers serve the purpose to map the extracted representation of the convolutional and pooling layers to the output space according to the task of interest. The final *feature maps* are usually flattened into a column vector and then fed to one or multiple FC layers to produce predictions. The general structure of a CNN is shown in Figure 2.58.

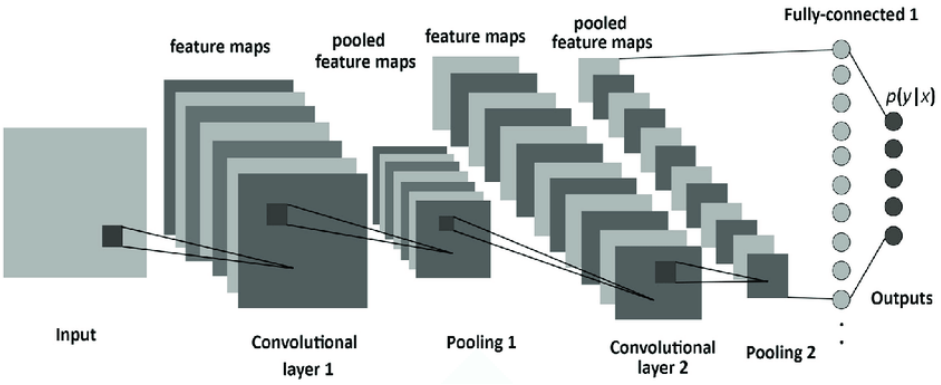


Figure 2.58. Structure of a CNN [275].

The number of layers, the number and size of the *kernels*, the type of pooling and the whole structure of the CNN can be chosen arbitrarily. Many different architectures have been proposed in the last years and have proven to be very successful among researchers and industrial applications. Some examples are ResNet, EfficientNet, and MobileNet for classification, YOLOv5, RetinaNet, and Faster R-CNN for object detection, SegNet and Mask R-CNN for semantic segmentation [276, 277, 278, 279, 280, 281, 282, 283].

1D Convolution When the input data has no 2D grid-like topology but instead is 1-dimensional like in time series, 1D CNNs can be used [284]. Each *kernel* has one dimension less with respect to 2D CNNs and slides across the data from left to right. The *kernel* is spatially smaller than the original input, extends to all *channels*, and shares the weights across different receptive *field instances* as shown in Figure 2.59.

As for 2D CNNs, 1D CNNs have *convolutional*, *pooling* and FC layers.

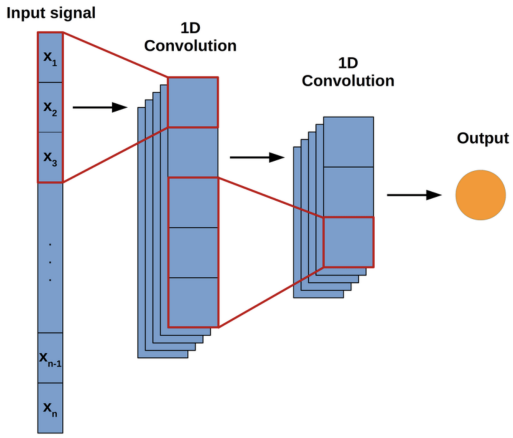


Figure 2.59. Operation performed in a 1D convolutional layer [285].

2.2.3.14 Graph Neural Networks

Graph Neural Networks (GNNs) are a class of NNs designed to perform inference on data described as graphs by leveraging their topological information [286]. In general, a graph consists of a set of nodes and a set of edges. Each node can have a fixed-length feature vector associated with it and the same goes for each edge in the graph. The basic idea behind most GNN architectures is graph convolution as an extension of the convolution operation performed in CNNs.
Even though images are usually considered as regular grids with *channels*, as discussed in the previous section, they can also be seen as graphs having a regular structure, where each pixel is a node and neighbors are determined by the *kernel* size. In this way, a 2D convolution is performing a weighted average of neighbor pixels of a node which are ordered and have a fixed size. GNNs can be thought of as a generalization of CNNs where the input can have a non-regular structure and a complex topology. In fact, in GNNs a convolutional operation can be performed by taking the average value of the node features along with its neighbors which are unordered and variable in size. The difference between the 2D and the graph convolutional operations is graphically shown in Figure 2.60.

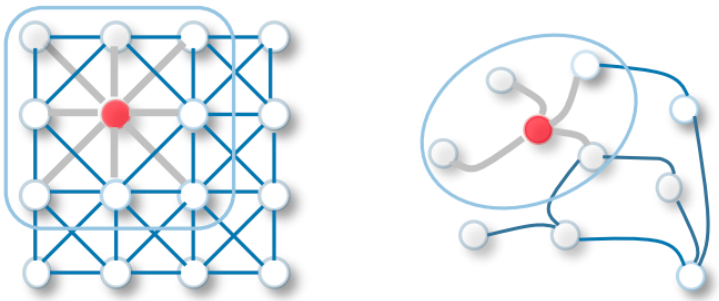


Figure 2.60. 2D convolution versus graph convolution [286].

Graph convolution computes a new latent feature vector for each node in the next layer as a function of the neighbors' features in the previous layer of the network. As

for any NN, latent representations can be used to compute predictions and, usually, GNNs are employed to perform *node*, *edge*, or *graph* classification/regression.
A large number of GNN architectures fall in the family of spectral methods which deal with the representation of a graph in the spectral domain. These methods are based on graph signal processing and define the convolution operator in the spectral domain using the Fourier transform F. The graph "signal" x is transformed to the spectral domain as

$$F(x) = U^T x. \tag{2.105}$$

Then, the transformed graph signal is multiplied by a *filter* g in the spectral domain as

$$F(g) \cdot F(x), \tag{2.106}$$

and, finally, the result is transformed back to its original domain using the inverse graph Fourier transform F^{-1} as

$$F^{-1}(x) = U^T x. \tag{2.107}$$

Given feature matrix X, an adjacency matrix A indicating whether pairs of vertices are adjacent or not, and a *filter* G, the graph convolution can be written as

$$G * X = F^{-1}(F(G) \cdot F(X)) = U(U^T G \cdot U^T X). \tag{2.108}$$

U is a matrix defined by the eigenvectors of $L = U \Lambda U^T$, being Λ a diagonal matrix containing the eigenvalues of L. L is the graph Laplacian and is computed as

$$L = D - A, \tag{2.109}$$

being D the diagonal node degree matrix where

$$D_{ii} = \sum_j A_{ij}. \tag{2.110}$$

The diagonal elements of L will have the degree of the node if A has no self-loops. The non-diagonal element $L_{ij} = -1$ $(i \neq j)$ if there is a connection. If there is no connection $L_{ij} = 0$ $(i \neq j)$. Calculating the eigenvectors of the Laplacian returns the Fourier basis for the graph.
However, computing eigenvalues and eigenvectors requires a SVD of L, which is computationally demanding. Moreover, being the filter applied on the entire graph, the notion of locality is lost for standard convolutions. Therefore, the graph convolution operation is usually approximated by a Chebyshev expansion [287]. This approximation is a recurrent expansion that is used to estimate the k-the power of L which has a direct interpretation: it traverses nodes that are up to k steps away, thus introducing the notion of locality. The bigger the power the bigger the local *receptive field* of the graph convolution.

Graph Convolutional Networks A Graph Convolutional Network (GCN) is a GNN belonging to the spectral family that uses the Chebyshev expansion approximation with $k = 1$ (it considers only direct neighbors for each node at every layer) [108]. The input of a GCN is a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ and $\mathbf{E}$ are the set of nodes and edges respectively. $\mathbf{V}$ is represented as an $N \times F$ feature matrix X, composed by the feature vectors of length F associated with each of the N nodes. The structural information of the graph enclosed in $\mathbf{E}$ is, instead, represented as an $N \times N$ adjacency matrix A.
GCNs produce a node-level output O in the form of an $N \times K$ matrix, where K is the number of output features computed for each node. This matrix can be written as a function of the feature matrix X and the adjacency matrix A as

$$O = f(X, A) = \sigma(AXW), \tag{2.111}$$

where W is a $F \times K$ trainable weight matrix.
When multiplying by A, a weighted sum is computed for each node between the feature vectors of all direct neighboring nodes excluding itself. For this reason, self-loops are added by defining $\hat{A} = A + I$, where I is the $N \times N$ identity matrix. Since A is typically not normalized, the matrix multiplications defined in Equation 2.111 can cause a change of scale in the feature vectors. To prevent numerical instabilities and vanishing/exploding gradients, the adjacency matrix can be normalized by computing $D^{\frac{1}{2}}\hat{A}D^{\frac{1}{2}}$, where D is the diagonal node degree matrix defined in Equation 2.109.
Considering the previous adjustments, the output of a GCN layer can be rewritten as

$$O = f(X, A) = \sigma(D^{\frac{1}{2}}\hat{A}D^{\frac{1}{2}}XW). \tag{2.112}$$

It is possible to define a multi-layer GCN by feeding the output feature matrix of a layer together with the adjacency matrix A as input for the next layer.

2.2.3.15 Autoencoders

An Autoencoder (AE) is a NN trained to learn a compressed representation for a set of data in an unsupervised manner [288]. First, it produces a reduced encoding for the input data. Then, it tries to reconstruct the original input from the reduced encoding. In particular, it aims at learning an identity function under specific constraints, for example with a limited number of neurons in the hidden layers. An AE consists of two parts, namely an Encoder and a Decoder.
The Encoder maps the input $\mathbf{x} \in \mathbb{R}^n$ to a latent space and, by considering a FC network, the output $\mathbf{h_e}^{(l+1)}$ of the l-th layer can be written as

$$\mathbf{h_e}^{(l+1)} = \sigma(W_e^{(l)} \cdot \mathbf{h_e}^{(l)}), \tag{2.113}$$

where $\mathbf{W_e}^{(l)}$ are the trainable weights of the layer and σ is a non-linear activation function. Considering an Encoder with L_e layers, we have that $\mathbf{h_e}^{(0)} = \mathbf{x}$ and that $\mathbf{h_e}^{(L_e)} = \mathbf{h}$, where $\mathbf{h} \in \mathbb{R}^k$ is the compressed version of the input.
The Decoder, instead, maps the compressed representation $\mathbf{h}$ back to its original space and, by considering again a FC network, we have that

$$\mathbf{h_d}^{(l+1)} = \sigma(W_d^{(l)} \cdot \mathbf{h_d}^{(l)}), \tag{2.114}$$

where $\mathbf{h_d}^{(l+1)}$ is the output of the l-th layer, $\mathbf{W}_d$ is the trainable weight matrix and σ is a non-linear activation function. Considering L_d layers, we have that $\mathbf{h_d}^{(0)} = \mathbf{h}$ and that $\mathbf{h}^{(L_d)} = \mathbf{x}'$, where $\mathbf{x}'$ is the reconstruction of the input vector $\mathbf{x}$
Since the goal of an AE is to reconstruct the input as accurately as possible (ideally $\mathbf{x}' = \mathbf{x}$), it is trained by minimizing the reconstruction error $\mathcal{L}(\mathbf{x}', \mathbf{x}) = \|\mathbf{x}' - \mathbf{x}\|$.
It is important to notice that, based on the specific application, other neural architectures different from FC networks can also be considered as Encoder or Decoder with an arbitrary number of hidden layers.

2.2.4 Training and Evaluation

This section covers data splitting for training and evaluating a ML or DL model, and the problem of overfitting and underfitting. Finally, the most commonly used evaluation metrics used to estimate the performance of a model are presented, together with data validation techniques for assessing its generalization capabilities over an out-of-sample data set.

2.2.4.1 Data Split

A dataset used to fit a model is generally split into three disjoint subsets, namely:

- *Training set*: samples used to train the model. This set is usually between 60% and 80% of the original sample.
- *Development set*: samples used to periodically evaluate the model during the training phase to select the best model. The model's parameters are not adjusted to fit this data. This set is usually between 10% and 20% of the original sample.
- *Test set*: after the whole training is completed, this set is used to perform a final evaluation of the model on unseen data. This set is usually between 10% and 20% of the original sample.

Monitoring the performance of the algorithm on the training set and the validation set (during the training phase), and on the test set (after the training phase), allows us to assess the convergence of the optimization process, the accuracy of the model, its robustness and its ability to generalize.

2.2.4.2 Overfitting and Underfitting

ML algorithms try to adjust their parameters to learn a function mapping inputs to outputs. It can happen that models adjusted their parameters excessively to correctly predict samples used for training while providing poor predictions for unseen samples in the validation and/or test set. This means that the model is unable to generalize correctly. If the predictions are excessively good on the *training set* and poor on the other sets, it means the parameters of the network have been excessively adjusted to fit only a subset of the data. This phenomenon is called *overfitting*.
If instead, the model can neither predict well on the *training set* nor generalize to new

data, it suffers from *underfitting*, thus having poor performance on the training data and the other sets. Figure 2.61 presents an example of underfitting and overfitting in a 2D scenario.

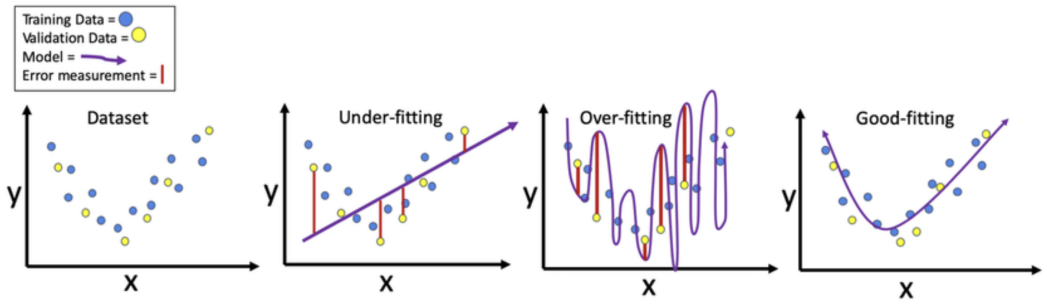


Figure 2.61. A 2D example of overfitting and underfitting [289].

When dealing with algorithms that optimize their parameters using a loss function, *overfitting* and *underfitting* can be tackled through *regularization* techniques which allow the model to better generalize. The most commonly used regularizations are $L1$, $L2$ and *dropout*.

L1 Regularization Given a loss function L quantifying the error between the prediction of the model and the correct output associated with the inputs, the $L1$ regularization computes the sum of the absolute values of all the parameters of the model and adds them to the original loss as follows

$$L_{reg} = L + \lambda \sum_{i=1}^{k} |w_i|, \tag{2.115}$$

where k is the total number of parameters and λ is the *regularization parameter* used as scaling factor for determining how much the *regularization term* $\sum_{i=1}^{k} |w_i|$ influences the total loss function [290].

L2 Regularization The $L2$ regularization, instead, adds the sum of the squared parameters to the original loss as

$$L_{reg} = L + \lambda \sum_{i=1}^{k} w_i^2. \tag{2.116}$$

When minimizing the new loss function L_{new}, the model is encouraged to keep its parameters small, preventing *overfitting* and allowing it to better generalize [291].

Dropout One of the most popular techniques used to prevent overfitting in NNs is *dropout* [292]. At every training step, a certain number of nodes are randomly selected and "deactivated", as shown in Figure 2.62, making the network ignore all their incoming and outgoing connections with other neurons.

In this way, at every iteration, the network adjusts only a random subset of its weights to generalize better. This is because the network is forced not to rely too

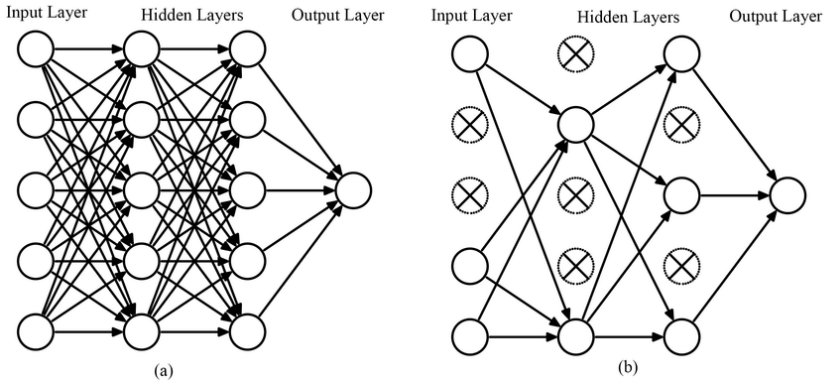


Figure 2.62. Example of a NN before (left) and after (right) dropout regularization [293].

much on any specific subset of neurons since they might be deactivated at any time. The amount of dropout, i.e. the percentage of deactivated neurons, is determined by a hyperparameter named *dropout* or *dropout rate*.

2.2.4.3 Evaluation Metrics

After training a model, it is essential to evaluate it using proper metrics. In this way, the accuracy and robustness of an algorithm can be quantified and compared to other models. The next sections describe the most popular evaluation metrics for classification and regression tasks, together with evaluation metrics used for unsupervised feature selection.

2.2.4.4 Evaluation Metrics for Classification

Metrics that quantify and compare discrete output predictions are adopted to evaluate the performance of classification models. Before diving into the different metrics, it is important to define what true positives, true negatives, false positives and false negatives are in the context of multi-class classification:

- True Positive ($\mathrm{TP_X}$): number of samples belonging to class X correctly predicted as belonging to class X.
- False Positive ($\mathrm{FP_X}$): number of samples belonging to class X incorrectly predicted as not belonging to class X.
- True Negative ($\mathrm{TN_X}$): number of samples not belonging to class X correctly predicted as not belonging to X.
- False Negative ($\mathrm{FN_X}$): number of samples not belonging to class X incorrectly predicted as belonging to class X.

Confusion Matrix The Confusion Matrix is a tabular visualization of the ground-truth labels compared to the model predictions. The entry (i, j) of the matrix represents the number of samples that belong to class i and are predicted as class

j. Ideally, the matrix has only positive values on the main diagonal (where $i = j$), meaning that the ground-truth class always coincides with the predicted class. The Confusion Matrix is not a proper performance metric but is used to compute other classification evaluation metrics. An example of normalized confusion matrix to evaluate fault classification in power systems is provided in Figure 2.63.

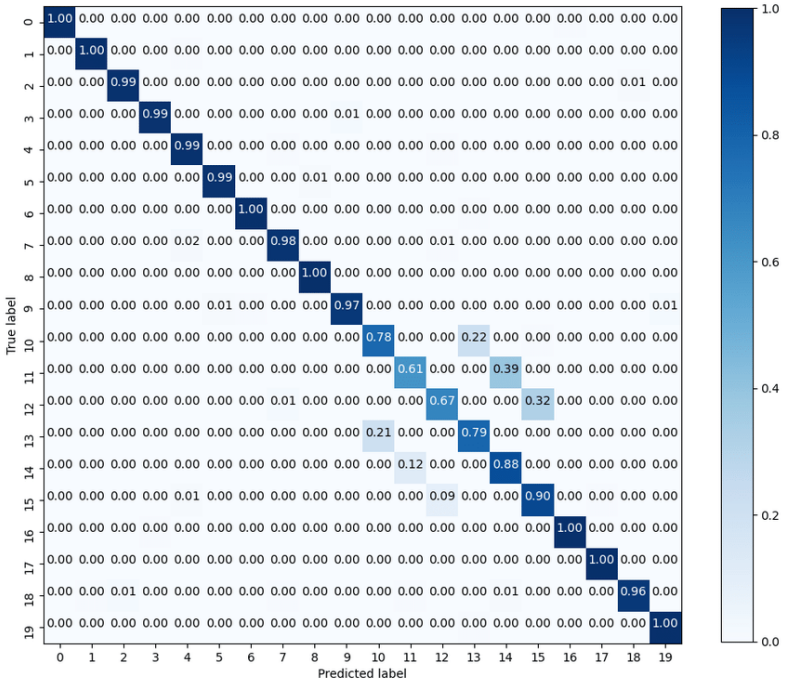


Figure 2.63. An example of normalized confusion matrix to evaluate fault classification in power systems [294]. Notably, it is a multi-class classification task with 20 classes.

Accuracy Accuracy is the ratio between the number of correct predictions and the total number of samples and can be computed for a class X as

$$\text{Accuracy}_\text{X} = \frac{\text{TP}_\text{X}}{\text{TP}_\text{X} + \text{FP}_\text{X} + \text{TN}_\text{X} + \text{FN}_\text{X}}. \tag{2.117}$$

To compute the overall accuracy of the model for all classes, three different approaches can be considered, namely *macro*, *weighted* and *micro* averaging [295]. A macro-average computes the metric independently for each class and then takes the average, thus treating all classes equally as

$$\text{Accuracy} = \frac{1}{k} \sum_\text{X} \text{Accuracy}_\text{X}, \tag{2.118}$$

where k is the number of classes and the sum goes over all possible classes. A weighted average computes the accuracy metrics for each label and finds their average weighted by support (the number of samples belonging to each label) as

$$\text{Accuracy} = \sum_\text{X} \frac{n_\text{X}}{n} \text{Accuracy}_\text{X}, \tag{2.119}$$

where n is the total number of samples and n_X is the number of samples belonging to class X. A micro-average aggregates the contributions of all classes to compute the average metric as

$$\text{Accuracy} = \frac{\sum_X \text{TP}_X}{\sum_X (\text{TP}_X + \text{FP}_X + \text{TN}_X + \text{FN}_X)}. \tag{2.120}$$

The macro, weighted and micro-averaging approaches can also be used to compute the overall performance of a classifier when using the other metrics covered in the next paragraphs. When dealing with binary classification, all metrics lose the X subscript.

Precision Precision for a class X is the ratio between the number of correct predictions for class X and the total number of samples that were predicted as belonging to class X and can be computed as

$$\text{Precision}_X = \frac{\text{TP}_X}{\text{TP}_X + \text{FP}_X}. \tag{2.121}$$

In the context of predictive maintenance, this metric is used to assess what percentage of alarms triggered by the model for a specific failure is true. A Precision value of 1 indicates that the algorithm is very robust and reliable, meaning that no false alarms are triggered. A model can be extremely precise, even though predicting a specific failure only a few times. For this reason, the following Recall metric is complementary to Precision.

Recall Recall for a class X is the ratio between the number of correct predictions for class X and the total number of samples that belong to class X, and can be computed as

$$\text{Recall}_X = \frac{\text{TP}_X}{\text{TP}_X + \text{FN}_X}. \tag{2.122}$$

In the context of predictive maintenance, this metric is used to assess what percentage of occurrences of a specific failure is correctly predicted by the model. A Recall value of 1 indicates that the algorithm can predict all failure events of a specific category. The limit case is when the model predicts all events as belonging to that specific class, thus resulting in maximum Recall. This is, of course, not a high-performing model and, therefore, Recall is usually considered together with Precision using the F1-score discussed next.

F1-Score The F1-Score is the harmonic mean between Precision and Recall and ranges between 0 and 1. This metric expresses how precise the model is in predicting a specific class, as well as how robust it is. The F1-Score for a class X can be computed as

$$\text{F1-Score}_X = \frac{2}{\frac{1}{\text{Precision}_X}\frac{1}{\text{Recall}_X}} = 2 \cdot \frac{\text{Precision}_X \cdot \text{Recall}_X}{\text{Precision}_X + \text{Recall}_X}. \tag{2.123}$$

This metric represents a balance between Precision and Recall and is recommended when dealing with imbalanced classification problems.

Area under Receiver operating characteristics curve The Area Under Receiver Operating Characteristics curve, better known as AUC-ROC curve, makes use of the concept of true positive rate and false positive rate for a class X which are defined as

$$\mathrm{TPR_X} = \frac{\mathrm{TP_X}}{\mathrm{TP_X} + \mathrm{FN_X}} \tag{2.124}$$

and

$$\mathrm{FPR_X} = \frac{\mathrm{FP_X}}{\mathrm{FP_X} + \mathrm{TN_X}}. \tag{2.125}$$

The $\mathrm{TPR_X}$ corresponds to the proportion of samples correctly classified as belonging to class X with respect to the total number of samples belonging to class X. The $\mathrm{FPR_X}$ corresponds to the proportion of samples incorrectly classified as belonging to class X with respect to the total number of samples not belonging to class X. The $\mathrm{TPR_X}$ and $\mathrm{FPR_X}$ can assume different values based on a threshold applied to the output of the predictive model. High threshold values can lead to both a low false positive rate and a low true positive rate since the model avoids predicting class X. Low thresholds, instead, can lead to both a high true positive rate and a high false positive rate since the model most likely predicts X as output label. For this reason, different thresholds are considered to generate a graph where the $\mathrm{FPR_X}$ is plotted on the x-axis and the $\mathrm{TPR_X}$ on the y-axis as shown in Figure 2.64.

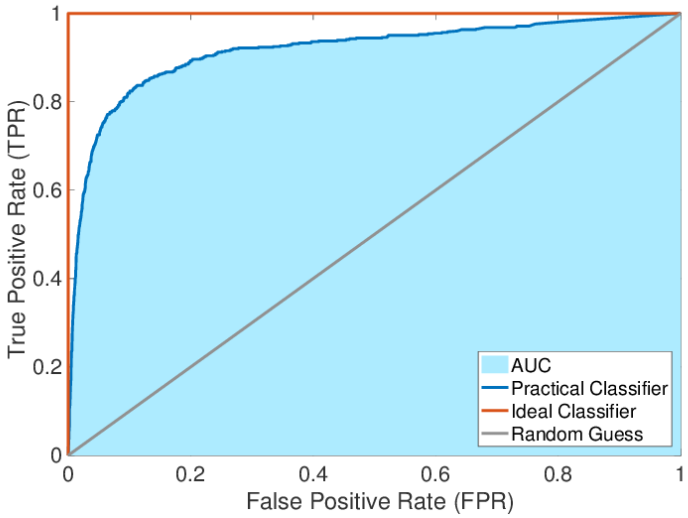


Figure 2.64. Example of an AUC-ROC curve [296].

The top left corner of the plot is the ideal point where the false positive rate is zero and a true positive rate is one. This is unrealistic, but a larger area under the curve (AUC-ROC) is usually better. This value ranges from 0 to 1 and the greater the value, the better the performance of the classifier.

2.2.4.5 Evaluation Metrics for Regression

Many evaluation metrics commonly adopted for regression tasks coincide with the loss functions used to train a NN (see Section 2.2.3.3).

Mean Squared Error The MSE is the average of the squared difference between the target value and the predicted value as defined in Equation 2.46. Due to the squaring of errors, this metric is sensitive to outliers which produce large errors, and small errors are penalized, leading to an underestimation of the model's performance. The interpretation of the MSE has to be done by keeping in mind the squaring factor of the errors.

Root Mean Squared Error The Root Mean Squared Error (RMSE) corresponds to the square root of the average of the squared difference between the target value y_i and the value predicted $\hat{y}_i$ and is computed as

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}. \tag{2.126}$$

This metric does not penalize smaller errors as the MSE and is less sensitive to outliers since it uses the square root. The interpretation is easier than the MSE since the scale of the errors is the same as the variable.

Mean Absolute Error The MAE is the average of the difference between the ground truth and the predicted values as defined in Equation 2.46. This metric is robust towards outliers since it does not square the errors as for the MSE. The MAE, since computing the absolute value, does not take into account the sign of the error and does not provide information about under-predicting or over-predicting the data. The interpretation of this metric is straightforward as the magnitude of the MAE matches the original scale of the variable.

2.2.4.6 Evaluation Metrics for Unsupervised Feature Selection

To evaluate an unsupervised feature selection method for time series, two main indicators are widely used: redundancy and information gain.

Redundancy The redundancy of information among a set of time series $y = (y_1, y_2, \ldots, y_N)$ is quantified by the metric W_I, which reads as

$$W_I(y) = \frac{1}{|y|^2}\sum_{y_i, y_j \in y} MI(y_i, y_j), \tag{2.127}$$

where $MI(y_i, y_j)$ is the mutual information between time series y_i and y_j [297]. A low value of W_I is associated with a set of time series that are maximally dissimilar to each other. It is also possible to consider the rate of variation of this metric, represented by the Redundancy Reduction Ratio (RRR)

$$RRR = \frac{W_I(y) - W_I(\bar{y})}{W_I(y)}, \tag{2.128}$$

where $\bar{y}$ is the set time series after the feature selection.

Information Gain The information gain is computed in terms of the Shannon entropy H [298], which reads as

$$H(X) = \sum_i p(x_i) \log_2(x_i), \tag{2.129}$$

where X is the data matrix associated with the set of time series y, being every row a sample of the observations and every column a different time series, and x_i is the i-th row of the matrix. The information gain is computed as the variation of entropy between the original time series and the time series after the feature selection $\bar{y}$. If the rate of variation is considered, it is possible to define the Information Gain Ratio (IGR):

$$IGR = \frac{H(X) - H(\bar{X})}{H(X)} \tag{2.130}$$

where X and $\bar{X}$ are the data matrices associated to y and $\bar{y}$.

2.2.4.7 Model Validation

Model validation is essential to assess that a model is robust enough to generalize to an unseen data set, without overfitting on the training data and achieving approximately the same performance on both samples.
As seen in Section 2.2.4.1, the available dataset is usually split into training, validation and test sets. However, by partitioning the available data into three sets, the number of samples that can be used to train the model is reduced and can lead to a problem of underfitting. Moreover, the results can depend on a particular random choice for the training and validation sets, thus generating a bias. Even though a test set should still be held out for final evaluation, the validation set is no longer a limitation when performing a procedure called Cross-Validation (CV) [299].

2.2.4.8 K-Fold Cross-Validation

In K-Fold Cross-Validation (KFCV), the training data is divided into k subsets referred to as *folds* [300]. Then, a model is trained using $k-1$ of the folds as training data and validated on the remaining part of the data. This operation is performed k times, one for each of the folds as shown in Figure 2.65.

Finally, the overall performance of the model is calculated by averaging the values computed for the k trials. Training k separate models can be computationally expensive but prevents the reduction of the training sample size as when fixing an arbitrary validation set once.

2.2.4.9 Leave-One-Out Cross-Validation

When the number of folds k is equal to the number of samples n ($k = n$), then KFCV takes the name of Leave-One-Out Cross-Validation (LOOCV) [302]. LOOCV provides a much less biased measure of the performance since n models are trained and their scores averaged, each considering $n - 1$ observations as training set and a

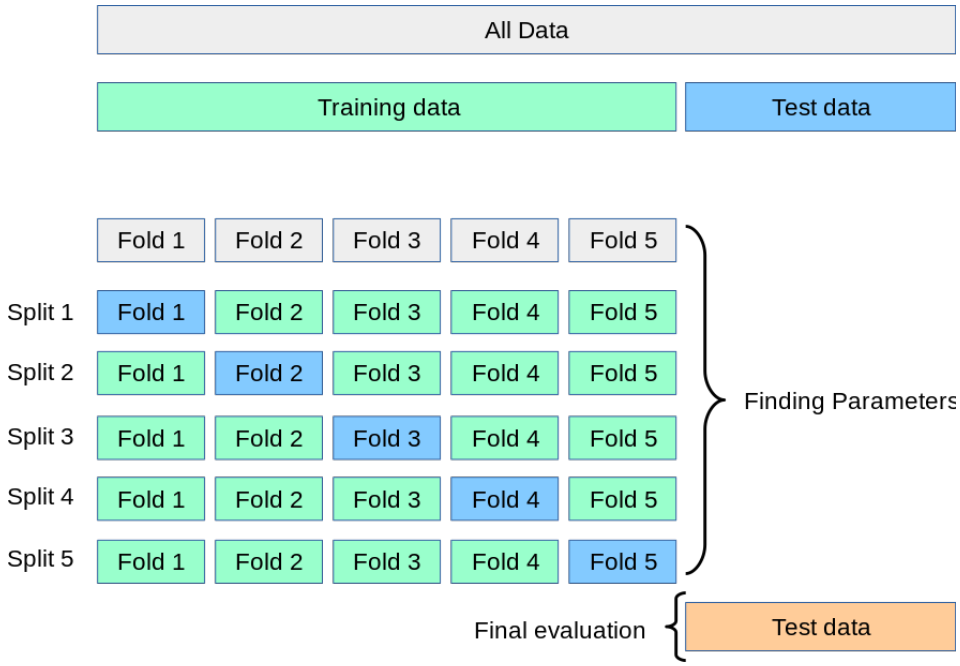


Figure 2.65. A visual representation of how KFCV operates considering 5 folds [301].

single sample as validation set. When n is large and/or the model is complex and takes longer to train, LOOCV can become extremely demanding both in terms of time and computational resources.

2.2.4.10 5×2-fold Cross-Validation

5×2-fold Cross-Validation (5x2CV) is a very popular approach for comparing the performance of two different models, especially when performing model selection [303]. 5x2CV computes 5 iterations of KFCV with $k = 2$ and, at each iteration, the available data is randomly partitioned into two equal-sized sets S_1 and S_2. Each learning algorithm (A or B) is trained on each set and tested on the other set and four error estimates are computed: $p_A^{(1)}$ and $p_B^{(1)}$ (error of model A and B, respectively, trained on S_1 and tested on S_2) and $p_A^{(2)}$ and $p_B^{(2)}$ (error of model A and B, respectively, trained on S_2 and tested on S_1). Subtracting corresponding error estimates produces two estimated differences

$$p^{(1)} = p_A^{(1)} - p_B^{(1)} \tag{2.131}$$

and

$$p^{(2)} = p_A^{(2)} - p_B^{(2)}. \tag{2.132}$$

From these two differences, the estimated variance is

$$s^2 = (p^{(1)} - \bar{p})^2 + (p^{(2)} - \bar{p})^2 \tag{2.133}$$

where

$$\bar{p} = \frac{p^{(1)} + p^{(2)}}{2} \tag{2.134}$$

Let s_i^2 be the variance computed for iteration i, and let $p_1^{(1)}$ be the $p^{(1)}$ for the very first of the 5 iterations. Then, it is possible to define the following t statistic

$$t = \frac{p_1^{(1)}}{\sqrt{\frac{1}{5}\sum_{i=1}^{5} s_i^2}}. \tag{2.135}$$

The null hypothesis is that both models have the same performance and that the score difference between the two algorithms for each fold is assumed to follow a normal distribution centered in zero. With this assumption, statistic t is assumed to follow a t distribution with 5 degrees of freedom. To test the null hypothesis the value of t is computed to check if it satisfies a t distribution with 5 degrees of freedom or if it is an outlier. When the t value is close enough to zero, the null hypothesis can not be rejected and the models are assumed to be equal. Otherwise, one of the models performs better than the other.

Chapter 3

Framework for Energy Applications

3.1 Dimensionality Reduction in Energy System Sensor Networks

Recent advances in wireless communications and electronics allowed the development of low-cost, low-power, multi-functional sensor nodes that are small in size and communicate over short distances. Each of these sensor nodes consists of sensing, data processing, and communicating components, and, when interconnected with other nodes, composes a sensor network [304].
Sensor networks usually count a large number of tiny sensor nodes that are deployed either inside the phenomenon of interest or very close to it, where each sensing device is capable of monitoring specific quantities, (e.g. temperature, pressure, vibration, etc.). One of the main advantages of sensor networks is that the location of each sensor node is not predetermined and can be engineered strategically, allowing monitoring of any quantity of interest. Moreover, since sensor nodes are fitted with an onboard processor, instead of sending the raw collected data, they use their processing abilities to locally carry out simple computations and transmit only what is useful [305].
In many energy systems, sensor networks are interfaced with a SCADA system, which stands for Supervisory Control And Data Acquisition [306]. SCADA systems are evolving rapidly and are nowadays considered a vital element penetrating the energy market that allows operators to supervise the behavior of energy systems as a whole. These systems record measurements coming from the underlying sensor network regularly (usually every 10 minutes) and allow the operator to determine if corrective actions are required. Moreover, SCADA systems also record the energy output, availability and error signals, providing a database of insightful data which can be analyzed or used to train predictive models.
On the other hand, with the possibility to install large numbers of sensors that collect measurements over months or even years, vast amounts of time series are generated and stored in databases. Moreover, when monitoring different points of an energy system, it can happen to measure redundant information or record signals that are strongly correlated or present similar trends even though having a different nature

[307]. The existence of redundant and noisy data not only makes it significantly more difficult to analyze the available data but also contributes to the degradation of the performance of learning algorithms, making them less scalable and reliable [308]. For these reasons, it is vital to detect and remove irrelevant, redundant, or noisy features, and alleviate the curse of dimensionality in the context of sensor networks installed in energy systems, allowing to improve the interpretability of predictive models, speed up their learning process, and enhance their generalization properties [297, 207].
The next sections present two papers we published in the context of dimensionality reduction for sensor networks. Specifically, the work in Section 3.1.1 [242] proposes an unsupervised algorithm based on time series clustering to group similar signals together, opening the possibility to reduce their dimensionality by selecting representative variables for each identified cluster. Even though the use case is not a, strictly speaking, RES, it involves a CHP plant using natural gas. By recovering and using heat from on-site electricity production, these systems typically achieve total system efficiencies from 65% to 80% and can reach 90%, making them suitable and necessary during the transition towards a sustainable future. Moreover, the proposed algorithm can be applied to any RE system to reduce the dimensionality of the monitored signals. The second paper presented in Section 3.1.2 [309] proposes a novel algorithm for unsupervised feature selection that leverages the predictive power of combinations of variables to assess their overall importance.

3.1.1 Time Series Clustering: A Complex Network-Based Approach for Feature Selection in Multi-Sensor Data

Introduction

The primary goal of industrial Internet of Things (IoT) has been linking operations and information technology for insight into production dynamics. This potential flexibility entails a floor of technologies made of distributed networks of physical devices embedded with sensors, edge computers, and actuators used to identify, collect, and transfer data among multiple environments. Such IoT-based Cyber-Physical Systems (CPS) establish a direct integration of engineering systems into digital computer-based ones, where the measurement or sensing technologies play an essential role in capturing the real world. Data collected are then the main ingredient to lift efficiency, accuracy, and economic benefits with the added merits of minimum human intervention [310, 311].
A consequence of such transformation is that the sensor layer, customarily used to measure, is now the means to map the actual status (of the process) into the cyber-world to derive process information, collect it in databases, and use it as a basis for the models which can be adapted (ideally by self-optimization) to the real situations. In this vein, the CPS provides a holistic view of the engineering systems and enables a bi-directional physical to digital interaction via multi-modal interfaces [312, 313]. Unlike the classic concepts derived from control theory, CPS forms the basis to describe even complex interactions and thus anticipate process deviations or interpretation and prediction of system behavior, diagnosis of exceptional events, explanation of causal mechanisms, and reactive response to urgent events [314].

On the other hand, CPS are disclosing large quantities of data to create augmented knowledge about process control. These high-dimensional datasets typically include heterogeneous measures acquired through a large variety of sensors which may have different sampling rates and communication protocols [315]. Following this trend, in recent years, there has been a rapid development of mathematical modeling techniques and analytical tools able to address the aforementioned problems, while meeting the new IoT requirements to assist process decision-makers.
In this scenario, AI is playing a major role to support the development of intelligent systems in process control engineering [316]. Examples are data preprocessing tasks, such as outlier removal [317], the replacement of missing values [318], and the definition of machine learning models for predictive and prescriptive maintenance [319, 320, 321]. Moreover, AI can be used to model a system without the computational complexity of a full simulation or a physical analysis, learning functional dependencies between system components from the data [322]. Even though most AI models are data-driven, overabundant high-dimensional data can have a negative impact on model behavior and efficiency, not only because of the computational complexity but also in terms of accuracy [315]. The existence of redundant and noisy data contributes to the degradation of the performance of learning algorithms, making them less scalable and reliable [308].
Remedial strategies must fundamentally select the most representative features of the original dataset, by using feature selection techniques. Feature selection, part of the larger family of dimensionality reduction approaches, aims at extracting a smaller set of representative features that retains the optimal salient characteristics of the data, especially in terms of global information content.
The dimensionality problem is particularly evident in data collected from control monitoring systems in engineered processes and machines, due to the strong presence of redundant data related to physical quantities, with similar trends, monitored in different points of the system, or to parameters of different nature but strongly correlated [307]. By detecting and removing irrelevant, redundant, or noisy features, feature selection can alleviate the curse of dimensionality from industrial sensor networks, improving model interpretability, speeding up the learning process, and enhancing model generalization properties [297, 207].
Feature selection techniques can be both supervised and unsupervised depending on the availability of the data class labels used for guiding the search for discriminative features [323]. Recently, unsupervised feature selection has been attracting an ever-growing interest, especially in the control and monitoring field, because of the lack of ground truth data in many real-world applications [324].
Traditional unsupervised feature selection methods based on dependence measures, such as correlation coefficients, linear dependence, or statistical redundancy, are already widely used [324]. Recently, feature clustering demonstrated its merit in terms of accuracy with respect to other unsupervised approaches [325]. In addition, clustering algorithms outperform state-of-the-art methods in detecting groups of similar features [326, 327], as well as in selecting metrics (one or more features) out of every cluster to reduce the dimensionality of the data with more or less granularity based on the application.
It has been recently demonstrated that network approaches can provide novel insights for the understanding of complex systems [238, 328], outperforming classical methods

in the ability to capture arbitrary clusters [329]. In particular, the weakness of conventional techniques resides in the use of distance functions that allow finding clusters of a predefined shape. In addition, they identify only local relationships among neighbor data samples, being indifferent to long-distance global relationships [329]. Examples of network methods for time series clustering can be found in the literature, making use of Dynamic Time Warping (DTW) and hierarchical algorithms [330] and community detection algorithms [329].
In this paper, we propose an unsupervised feature selection algorithm based on a novel feature clustering technique tailored to time series collected from real industrial IoT sensor networks. The clustering approach complements different tools from complex network theory, which are becoming promising in the field of nonlinear time series analysis for their ability to characterize the dynamical complexity of a system [238]. In particular, we used visibility graphs [237] to map time series in the network domain, then node degree sequences extraction [331] to characterize them, and, finally, community detection algorithms [332] to perform time series clustering. The proposed method was tested on the sensor network of a 1 MW Combined Head and Power (CHP) plant central monitoring system. The heterogeneous dataset includes measurements from the engine, the auxiliaries, the generator, and the heat recovery subsystem. Finally, we compared with other traditional time series clustering methods in terms of redundancy and information content for feature selection.
The rest of the paper is organized as follows. First, we present the unsupervised feature selection method, then, describe the case study. After, we report experimental results to support the proposed approach and, finally, summarize our work and draw some conclusions.

Methods

This section discusses the proposed method, starting from the problem of time series clustering up to the task of unsupervised feature selection. Given a set of N time series $y = y_1, y_2, \ldots, y_N$, the main steps of the proposed clustering approach are here summarized.

a) Remove time series noise through a low-pass filter.

b) Segment time series y_n into consecutive non-overlapping intervals $s_n^1, s_n^2, \ldots, s_n^T$ corresponding to a fixed time amplitude L, where T is the number of segments extracted for each time series.

c) Transform every signal segment s_n^t ($t = 1, \ldots, T$ and $n = 1, \ldots, N$) into a weighted natural visibility graph G_n^t.

d) Construct a feature vector $k_n^t = ((k_n^t)_1, (k_n^t)_2, .., (k_n^t)_L)$ for each visibility graph G_n^t, where $(k_n^t)_i$ is the degree centrality of the ith node in the graph and k_n^t the degree sequence of the graph.

e) Define a distance matrix D^t for every tth segment ($t = 1, \ldots, T$), where the entry d_{ij}^t is the Euclidean distance between the degree centrality vectors k_i^t

and k_j^t. Every matrix gives a measure of how different every pair of time series is in the tth segment.

f) Compute a global distance matrix D that covers the entire time period T where the entry (i, j) is computed as $d_{ij} = \frac{1}{T}\sum_{t=1}^{T} d_{ij}^t$, averaging the contributions of the individual distance matrices associated to every segment.

g) Normalize D between 0 and 1, making it possible to define a similarity matrix as $S = 1 - D$, which measures how similar every pair of time series is over the entire time period.

h) Build a weighted graph C considering S as an adjacency matrix.

i) Cluster the original time series by applying a community detection algorithm on the graph C and visualize the results through a force-directed layout.

Figure 3.1 illustrates the flowchart of the methodology.

After the initial stages of data filtering (Step a) and time series segmentation (Step b), for the transformation of every signal into the network domain (Step d), we used the natural weighted visibility graphs. The natural variant was preferred to the horizontal one because it can capture properties of the original time series with higher detail, avoiding simplified conditions. The weighted variant, on the other hand, is used to magnify the spatial distance between observations that have visibility and thus avoid binary edges in favor of weighted edges in the visibility graph.

Since we used natural weighted visibility graphs to map time series into networks, for the extraction of a feature vector for each signal segment (Step e), we considered the weighted degree centrality sequence of the network, as suggested in [333], because it can fully capture the information content included in the original time series [331, 334].

Then, after the construction of the segment distance matrices D^t and the normalized global similarity matrix S together with its graph representation C (Steps f–h), we used the modularity-based Louvain's method (Step i) for community detection since fast and well-performing in terms of modularity.

To achieve a modular visualization of the clusters detected by the discussed method and their mutual connections, we used a force-directed algorithm, namely the Frushterman–Reingold layout, as a graphical representation.

Finally, for specific unsupervised feature selection purposes, we considered a representative parameter for each cluster. Such parameters were identified based on their importance within the communities, by considering the signals with the highest total degree centrality in their respective groups.

Every part of the proposed approach was developed in Python 3.6 [335], using the Numpy [336] and NetworkX [337] packages.

Case Study

This section deals with the case study considered for the applications of the proposed method, which is an internal combustion engine used in industrial cogeneration (or CHP).

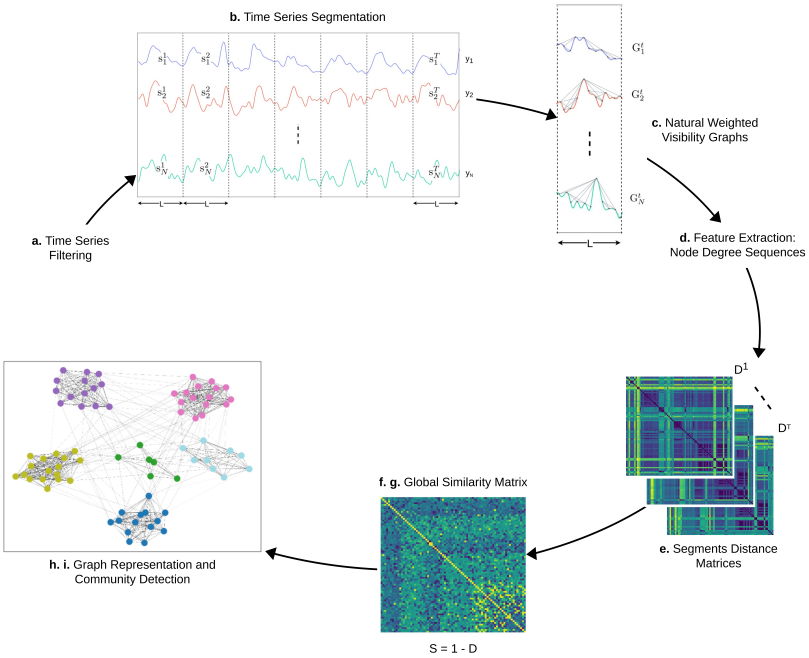


Figure 3.1. Flowchart of the proposed time series clustering methodology.

The CHP system consists of a four-stroke ignition engine ($P = 1032$ kW) fueled with vegetable oil, coupled to a three-phase synchronous generator. The electricity produced is used to meet the self-consumption of the plant and the production surplus is fed into the grid.
The heat recovery takes place both from the engine cooling water circuit and from the exhaust gases. In particular, the heat exchange with engine cooling water ($t = 65 - 80$ °C) is used both to meet part of the plant heating requirement and for the preheating of the fuel before the injection phase. The return water from the plant is cooled by a ventilation system consisting of four fans ($P = 15$ kW).
The exhaust gases, after being treated in a catalyst, are conveyed inside a boiler of 535 kW thermal power, which is used to produce steam at about 90 °C useful for different production lines. A schematic representation of the system is shown in Figure 3.2.

The system is equipped with a sensor network for condition monitoring and control purposes that samples every minute for a total of 90 physical quantities at different points.
The data used for the case study go from 25 June 2014 to 5 May 2015.
The early preprocessing phase involved the removal of the constantly flat parameters and the cumulative signals, thus reducing the number of the starting parameters to 78. The final list of monitored CHP plant variables considered for the analysis is reported in Table 3.1.
In the preprocessing phase, the outliers caused by sensor errors were also removed. To deal with unusual dynamics linked to system shutdowns, observations with zero active power were filtered out. Afterward, we resampled the data every 15 min to filter constant signal intervals and reduce the number of measurements processed

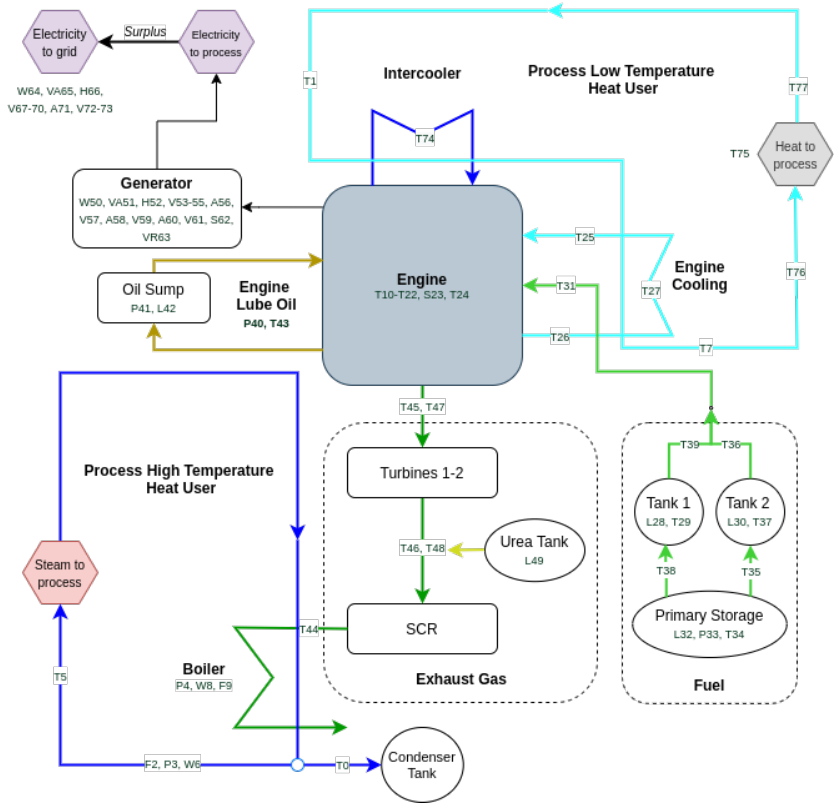


Figure 3.2. Schematic block diagram of the CHP system with measuring points.

by the algorithm. The resulting data matrix used as input for the analysis had 30,240 rows and 78 columns. Finally, we built time series segments including 24 h of observations to capture the typical daily cycle of the plant.

Results

This section provides a detailed discussion of the experimental results obtained by the proposed approach, followed by a comparison with two traditional time series clustering methods.

Figure 3.3 shows the plot of the 78 standardized signals during a representative period of about two months. Data were extracted during a total measuring time of almost 11 months.

The dataset was then analyzed by applying the method described in Section 3.1.1. After the application of a low-pass filter for noise removal, Steps b–d of the workflow, time series were segmented into non-overlapping intervals s_n^t, then mapped into natural visibility graphs G_n^t, and finally feature vectors were extracted in terms of degree sequences k_n^t. Afterward, in Steps e–g, a global distance matrix D was computed by combining the contribution of all the distance matrices D^t, followed by the definition of the similarity between all the pairs of time series. The resulting similarity matrix S is shown in Figure 3.4.

As per Step h, the similarity matrix S is represented in the form of a weighted graph, also called similarity graph C, where each node corresponds to a specific

Table 3.1. List of monitored CHP plant parameters.

ID	Variable	ID	Variable	ID	Variable
T0	Condenser Temp. [°C]	T26	Engine 1 Out Temp. [°C]	H52	Gen Frequency [Hz]
T1	Hot Water Temp. [°C]	T27	Engine 2 Out Temp. [°C]	V53	Gen L1-L2 Concat. Volt. [V]
F2	Steam Flow Rate [m3/h]	L28	Tank 1 Level [%]	V54	Gen L2-L3 Concat. Volt. [V]
P3	Steam Out Pressure [bar]	T29	Tank 1 Temp. [°C]	V55	Gen L3-L1 Concat. Volt. [V]
P4	Steam Pressure [bar]	L30	Tank 2 Level [%]	A56	Gen Phase 1 Current [A]
T5	Steam Temp. [°C]	T31	Zone 4 Temp. [°C]	V57	Gen Phase 1 Volt. [V]
W6	Steam Thermal Power [kW]	L32	Tank Level [lt]	A58	Gen Phase 2 Current [A]
T7	Cold Water Temp. [°C]	P33	Tank Pressure [bar]	V59	Gen Phase 2 Volt. [V]
W8	Thermal Power [kW]	T34	Zone 1 Temp. [°C]	A60	Gen Phase 3 Current [A]
F9	Water Flow Rate [m3/h]	T35	Zone 2 Temp. [°C]	V61	Gen Phase 3 Volt. [V]
T10	Casing Out Temp. [°C]	T36	Zone 3 Temp. [°C]	S62	Gen Power Factor
T11	Cylinder 1A Temp. [°C]	T37	Tank 2 Temp. [°C]	VR63	Gen Reactive Power [Var]
T12	Cylinder 1B Temp. [°C]	T38	Zone 5 Temp. [°C]	W64	Grid Active Power [W]
T13	Cylinder 2A Temp. [°C]	T39	Zone 6 Temp. [°C]	VA65	Grid Apparent Power [VA]
T14	Cylinder 2B Temp. [°C]	P40	Carter Pressure [mbar]	H66	Grid Frequency [Hz]
T15	Cylinder 3A Temp. [°C]	P41	Oil Pressure [bar]	V67	Grid L1-L2 Concat. Volt. [V]
T16	Cylinder 3B Temp. [°C]	L42	Sump Level [%]	V68	Grid L2-L3 Concat. Volt. [V]
T17	Cylinder 4A Temp. [°C]	T43	Oil Temp. [°C]	V69	Grid L3-L1 Concat. Volt. [V]
T18	Cylinder 4B Temp. [°C]	T44	SCR Out Temp. [°C]	V70	Grid Phase 1 Volt. [V]
T19	Cylinder 5A Temp. [°C]	T45	DX Turbine In Temp. [°C]	A71	Grid Phase 2 Current [A]
T20	Cylinder 5B Temp. [°C]	T46	DX Turbine Out Temp. [°C]	V72	Grid Phase 2 Volt. [V]
T21	Cylinder 6A Temp. [°C]	T47	SX Turbine In Temp. [°C]	V73	Grid Phase 3 Volt. [V]
T22	Cylinder 6B Temp. [°C]	T48	SX Turbine Out Temp. [°C]	T74	Intercooler In Temp. [°C]
S23	Speed [rpm]	L49	Urea Tank Level [%]	T75	Plant Delta Temp. [°C]
T24	Supercharger Temp. [°C]	W50	Gen Active Power [W]	T76	Plant In Temp. [°C]
T25	Engine 1 In Temp. [°C]	VA51	Gen Apparent Power [VA]	T77	Plant Out Temp. [°C]

signal and the edge weights quantify pairwise similarities between time series. To carry out the community detection phase, only the most important edges were taken into account. In particular, we performed edge pruning by filtering the pairwise similarities lower than the second quantile of their probability distribution.
Then, as for Step i, using Louvain's algorithm (see Section 2.2.2.5), we identified 12 different communities within the filtered similarity graph, which globally cover 70 parameters. Table 3.2 provides the detail of the variables contained in each cluster with reference to the parameter IDs presented in Table 3.1.

Table 3.2. Clusters obtained through Louvain's algorithm.

Cluster ID	Variable ID
Cluster 1	T29, T34-T39
Cluster 2	L28, L30
Cluster 3	T11-T22, T44-T48
Cluster 4	T7, T10, T24-T27, T31, T43, T74, T76
Cluster 5	W8, F9
Cluster 6	T0, F2, P3, P4, T5, W6
Cluster 7	T1, T75, T77
Cluster 8	S23, H52, H66
Cluster 9	V53-V55, V57, V59, V61, V67-V70, V72, V73
Cluster 10	W50, VA51
Cluster 11	A56, A58, A60
Cluster 12	W64, VA65, A71

The eight signals shown in Figure 3.5 were not clustered since they were characterized by independent dynamics. This subset includes engine lube oil parameters,

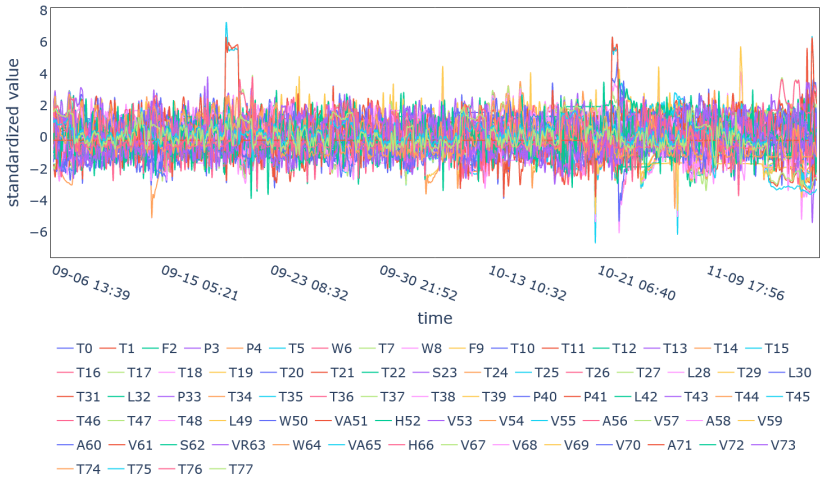


Figure 3.3. Overall standardized time series sampled from 6 September 2014 to 21 November 2014.

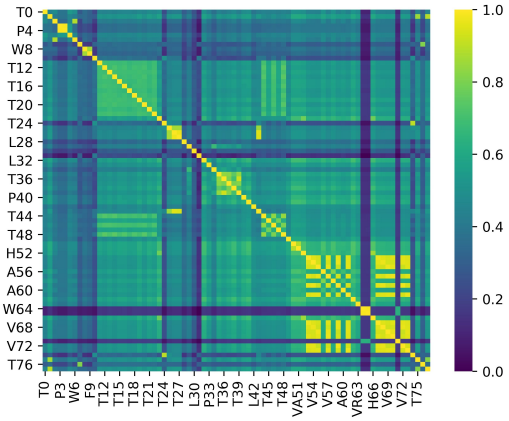


Figure 3.4. Similarity matrix represented as a heat map where the entry (i, j) quantifies the similarity between signals i and j.

i.e., carter pressure, sump level, and pressure; generator parameters, i.e., power factor and reactive power; parameters in the fuel primary storage, i.e., tank level and pressure; and parameters in the exhaust gas treatment, i.e., urea tank level.

Time series clustering results are illustrated with reference to the functional groups shown in the block diagram in Figure 3.2. Most of the fuel parameters were grouped into two distinct homogeneous clusters (see Figure 3.6). Fuel temperatures from the primary storage to the output of tanks 1 and 2 are included in Cluster 1 (Figure 3.6a), while Cluster 2 (Figure 3.6b) groups the fuel levels in the two tanks.

Engine sensor signals fall, together with other strictly related parameters, into two distinct clusters (see Figure 3.7). In particular, Cluster 3 (Figure 3.7a) includes all the cylinder temperatures and the exhaust temperatures, while Cluster 4 (Figure 3.7b) includes the casing temperatures, the supercharger temperatures, and the temperatures monitored at the engine auxiliaries, e.g., cooling water, lube oil, and intercooler subsystems. Cluster 4 also contains some parameters by the heat exchange

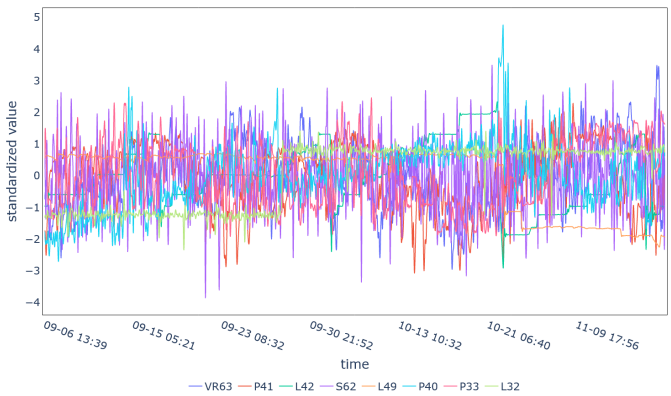
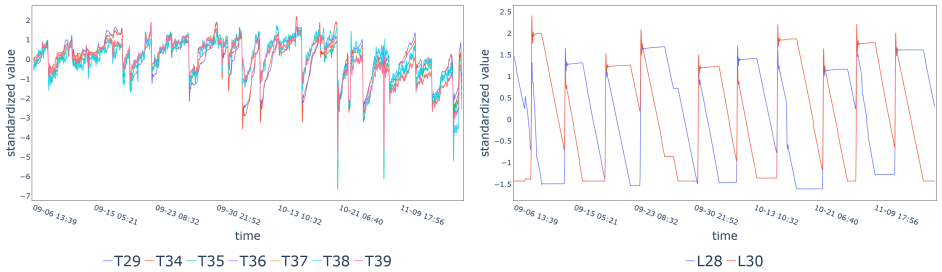


Figure 3.5. Signals that were not included in clusters: lube oil carter pressure (P40), lube oil sump level and pressure (L42 and P41), generator power factor (S62) and reactive power (VR63), fuel tank level and pressure (L32 and P33) in the primary storage, and urea tank level (L49).



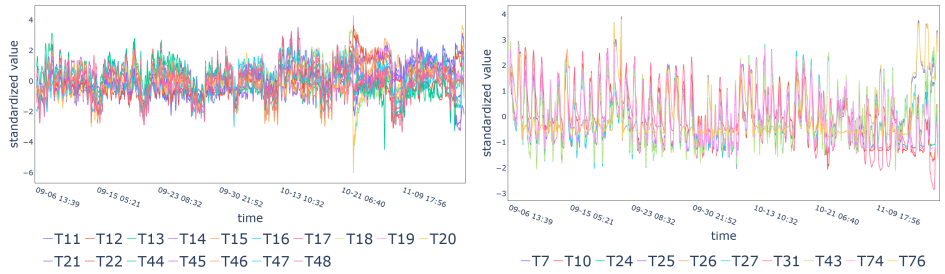
(a) Cluster 1 **(b)** Cluster 2

Figure 3.6. Fuel parameters: (**a**) the trends of the time series included in Cluster 1, i.e., temperatures of the fuel in the storage area (T29 and T34–T39); and (**b**) the trends of the time series included in Cluster 2, i.e., fuel levels in tank 1 (L28) and tank 2 (L30).

with the engine cooling, such as water inlet temperatures of the process heat circuit and inlet fuel temperature.
All the parameters of the high-temperature heat recovery circuit (process steam demand) were, instead, separated into two distinct groups (see Figure 3.8). In detail, Cluster 5 (Figure 3.8a) includes the thermal power and hot water flow rate, monitored at the boiler inlet, while, in Cluster 6 (Figure 3.8b), all the specific steam parameters are grouped together, such as steam flow rate, pressure, and thermal power, as well as the temperature of the condensed water.

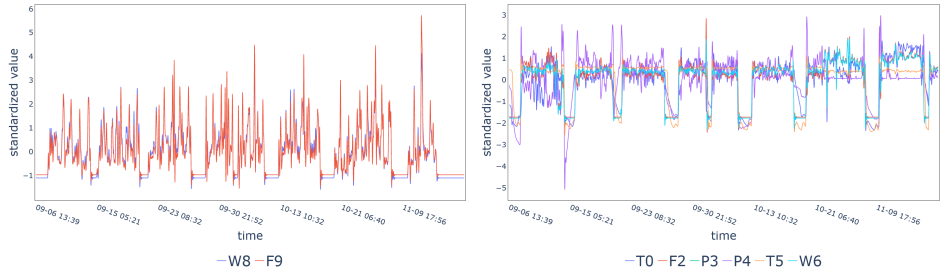
As mentioned above, low-temperature heat circuit sensor signals, measured at the plant inlet, are part of Cluster 4 together with other engine and auxiliaries signals (see Figure 3.7b), while the water temperatures at the plant outlet and the delta in–out temperature are in Cluster 7 (see Figure 3.9).

The two principal properties of the electric power supply, frequencies and voltages, were divided into two clusters (see Figure 3.10). Notably, in Figure 3.10a, it is possible to note how the engine speed was included in Cluster 8 together with the generator and grid frequencies. On the other hand, Cluster 9 includes all the



(a) Cluster 3 **(b)** Cluster 4

Figure 3.7. Engine sensor signals and strictly related parameters: (**a**) the trends of the time series included in Cluster 3, i.e., temperatures of the cylinders (T11–T22) and temperatures of the exhaust gases (T44=-T48); and (**b**) the trends of the time series included in Cluster 4, i.e., temperatures referred to the external casing of the engine (T10 and T24), engine auxiliaries (T25–T27, T43, and T44) and parameters influenced by the heat exchange with the cooling water (T7, T76, and T31).



(a) Cluster 5 **(b)** Cluster 6

Figure 3.8. Process high-temperature user parameters: (**a**) the trends of the time series included in Cluster 5, i.e., thermal power (W8) and flow rate (F9) of the hot water at the boiler inlet; and (**b**) the trends of the time series included in Cluster 6, i.e., steam parameters such as flow rate (F2), pressure (P3), temperature (T5), and thermal power (W6) as well as the condensed water temperature (T0).

generator and grid voltages.

Other electrical parameters, such as powers and currents, have instead been divided into three different clusters (see Figure 3.11). In particular, Cluster 10 (Figure 3.11a) and Cluster 11 (Figure 3.11b) distinguish, respectively, the generator powers from the generator currents, while Cluster 12 (Figure 3.11c) groups together the grid powers and currents. The latter refers only to Phase 2 current because the Phase 1 and 3 currents were removed in the preprocessing phase due to sensor malfunctions.

The clustering results show that the proposed approach is independent of the nature of the monitored parameters and their functionality within the system. For example, Clusters 1, 2, 7, 9, 10, and 11 (Figures 3.6a, 3.9a, 3.10b, and 3.11a,b) include only homogeneous variables (e.g., temperatures) belonging the same functional area (e.g., engine). Among those, it is interesting to note how the parameters within Cluster 2, i.e., the fuel levels in the tanks for primary storage, seem to be very different from the

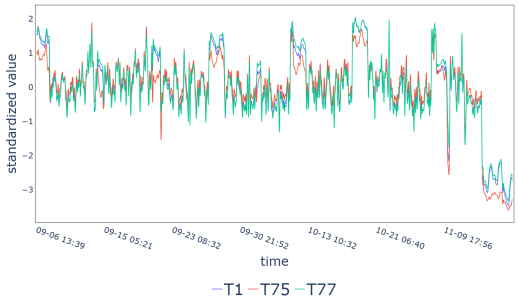
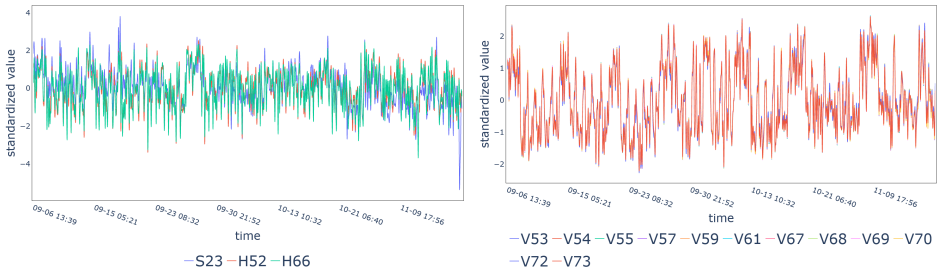


Figure 3.9. Process low-temperature user parameters: the trends of the time series included in Cluster 7, i.e., water temperatures at the plant outlet (T1, T77) and delta in–out temperature (T75).



(a) Cluster 8 **(b)** Cluster 9

Figure 3.10. Generator and Grid frequencies and voltages: (**a**) the trends of the time series included in Cluster 11, i.e., respectively engine speed (S23), and frequencies monitored both at the generator (H52) and at the grid (H66); and (**b**) the trends of the time series included in Cluster 12, i.e., voltages monitored both at the generator (V53–V55, V57, V59, and V61) and at the grid (V67–V70, V72, and V73).

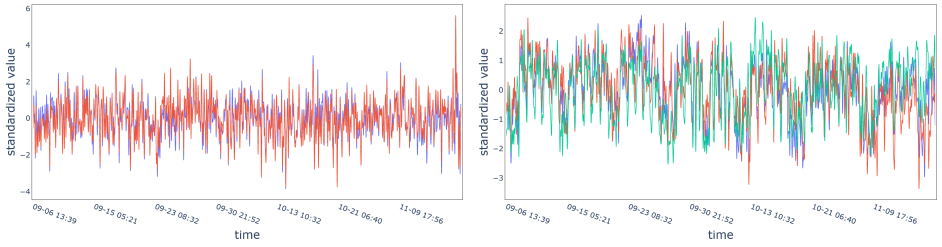
Euclidean point of view, but the method identified a similarity in their global trends. On the other hand, Clusters 5, 6, and 12 (Figures 3.8a,b, and 3.11c) represent some examples of communities populated by heterogeneous physical parameters recorded in the same functional area.

Finally, a particular interest derives from the hidden relationships identified between parameters characteristic of different functional areas. Examples are Cluster 3 (Figure 3.8a), which includes temperatures of cylinder and exhaust; Cluster 4 (Figure 3.8b), which groups together temperatures referred to the engine external casing, the engine auxiliaries, heat recovery, and fuel pre-heating systems and the inlet fuel; and Cluster 8 (Figure 3.10a), which is composed by frequencies and voltages related to both the generator and the grid.

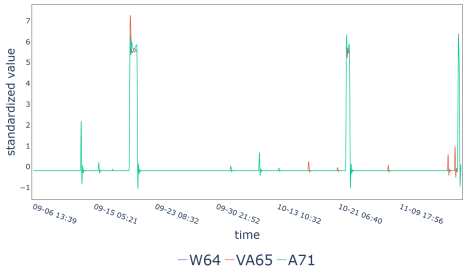
After the identification of clusters, exploratory network analysis was used to render a graphical representation of their degree of similarity (the higher the similarity between nodes, the smaller their spatial distance), thus improving the cluster visualization. The Frushterman–Reingold layout applied to the similarity graph C, after edge pruning, provided the results shown in Figure 3.12.

The force-directed layout gives evidence of a central core of strongly connected

parameters, which includes, respectively, most of the fuel temperatures in the storage area (Cluster 1), all the temperatures of cylinders and exhaust (Cluster 3), all the process low-temperature parameters (Cluster 7), and most of the generator and grid parameters (Clusters 8–11).



(a) Cluster 10 **(b)** Cluster 11



(c) Cluster 12

Figure 3.11. Generator and Grid electric powers and currents: (**a**) the trends of the time series included in Cluster 10, i.e., generator parameters such as active power (W50) and reactive power (VA51); (**b**) the trends of the time series included in Cluster 11, i.e., generator electric currents Phase 1 (A56), Phase 2 (A58), and Phase 3 (A60); and (**c**) the trends of the time series included in Cluster 12, i.e., grid parameters such as active power (W64), reactive power (VA65), and electric current Phase 2 (A71).

Notably, only two parameters of Cluster 3 are outside the central core, namely T29 and T34, measuring, respectively, the fuel temperature in the primary storage and in tank 2 (the latter being a backup tank). It is also possible to notice how the temperatures of engine cooling water (T25–T27) and lube oil (T43) subsystems represent a key group in bridging the central core to the other variables of Cluster 4. Similarly, the steam parameters in the high-temperature heat recovery (Cluster 6), although not directly included in the central core, appear to be strictly connected to it. As expected, no correlation is active among the fuel levels inside the tanks (Cluster 2), the power and flow rate of the hot water at the boiler inlet (Cluster 5), the grid power and currents (Cluster 12), and the rest of the network.
To improve the interpretation of the results by adding quantitative information to the exploratory analysis, we calculated the cumulative percentage distribution of the average degree centrality of each cluster (see Figure 3.13).

The bar chart in Figure 3.13 attributes a specific ranking to the clusters according to their average contribution to the degree centrality of the network. Overall, the results confirm the considerations made so far in relation to the core communities

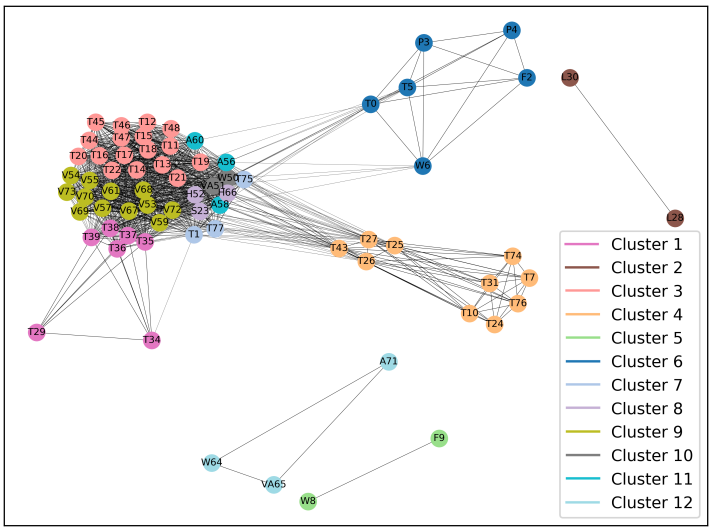


Figure 3.12. Frushterman–Reingold layout applied to the similarity graph after performing edge pruning.

(Clusters 1, 3, 7, and 8–11), to the boundary communities (Clusters 4 and 6), and the communities unrelated to the network (Clusters 2, 5, and 12).
As for the communities included in the central core, it is possible to obtain a distinction between the roles played in the network. In detail, Cluster 10, which groups engine speed and generator and grid frequencies together, is the most influential on the control and stability of the global systems, followed by Cluster 3, which includes cylinder temperatures and exhaust gases.
Finally, after cluster identification and analysis, feature selection was performed by selecting in each cluster the representative signal as the one with the highest degree contribution in its group. Table 3.3 shows the selected variables associated with each cluster, together with their degree centrality in the similarity graph, and their share contribution to the sum of the degree centralities within the reference cluster. The representative parameters shown in the table are visually confirmed by the force-directed layout in Figure 3.12. For example, variable T0 (condense temperature) appears to be the most influential node of Cluster 6 (process high-temperature user parameters), having a high number of connections not only with variables of its cluster but also with those belonging to the central core of strongly connected signals. Another example is the parameter T43 (oil temperature) with respect to Cluster 4 (parameters strictly related to the engine).
As reported in the case study, the data matrix considered as input for the analysis has 30,240 $\times$ 78 dimensions. After the application of the proposed method, by considering the 12 representative cluster variables, listed in Table 3.3, together with the 8 independent signals shown in Figure 3.5, we obtained a final data matrix of size 30,240 $\times$ 20, thus reducing the dimensionality by 74.4%.

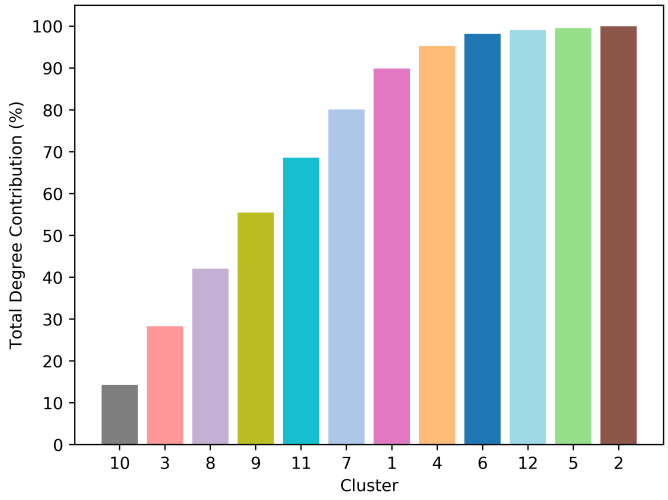


Figure 3.13. Cumulative percentage distribution of the average degree centrality of clusters.

Performance Metrics

An exhaustive evaluation of the proposed method can be obtained by appropriate measures of clustering partitioning and feature selection information content.
The lack of ground truth data in the present condition monitoring application precluded the evaluation of the clustering results through classical external indices. In addition, since we used a modularity-based method for community detection, modularity was identified as the most appropriate metric for the final clustering. A first evaluation of the clustering results was performed using the modularity measure, which quantifies the goodness of the communities on a scale that goes from -1 to 1. In particular, we obtained a modularity index of 0.72, representative of good-quality results.
Since the proposed approach belongs to the category of unsupervised feature selection methods, the final evaluation was performed in terms of Redundancy Reduction Ratio (RRR) and Information Gain Ratio (IGR), defined, respectively, in Equations (2.128) and (2.130).
The proposed method was compared with standard approaches for time series clustering (see Table 3.4). In particular, a *raw data-based* method was considered, which uses the Euclidean distance as time series similarity measure and a partitioning clustering, namely K-Means, for grouping variables. In addition, we included a *feature-based* method in the comparison, which involves the extraction of statistical parameters characteristic of the time series (i.e., average, median, standard deviation, skewness, and kurtosis) and the subsequent application of the K-Means algorithm for clustering.

Table 3.4 shows that the time series clustering approach seems to be particularly efficient in terms of feature selection, allowing a total redundancy reduction of 29.05% in the starting dataset by obtaining, at the same time, a global information gain of 10.60%.

It is also interesting to note that both performance metrics are better than those

Table 3.3. Representative signals chosen for every cluster according to their node degree centrality.

Cluster ID	Number of Elements	Most Representative Variable ID	Absolute Degree Value	Within Cluster Degree Contribution
Cluster 1	7	T38	29.44	19.51%
Cluster 2	2	L30	1.00	50.00%
Cluster 3	17	T19	34.62	6.61%
Cluster 4	10	T43	18.20	15.34%
Cluster 5	2	W8	1.00	50.00%
Cluster 6	6	T0	9.03	23.50%
Cluster 7	3	T75	25.83	33.96%
Cluster 8	3	H66	31.06	34.18%
Cluster 9	12	V72	30.91	8.70%
Cluster 10	2	VA51	31.67	50.30%
Cluster 11	3	A56	30.67	35.31%
Cluster 12	3	W64	2.00	33.33%

Table 3.4. Comparison of the feature selection performances between the proposed approach and two standard methods: a *raw data-based* method and a *feature-based* one. The evaluation was performed by considering the RRR and IGR indices.

Method	Optimal Clusters	RRR	IGR
Proposed approach	12	29.05%	10.60%
Raw-data based	12	9.52%	8.39%
Feature based	10	20.96%	7.90%

obtained with the standard approaches considered. In particular, the proposed method outperforms the *raw data-based* clustering approach in terms of both RRR and IGR indices, with an overall performance improvement of 19.53% and 2.21%, respectively. Looking at the results obtained with the *feature-based* method, also in this case the proposed approach provides better results with an increase of 8.09% and 2.70% for the RRR and IGR indices, respectively.

Conclusions

With the advent of I4.0 the increasing availability of sensor data is leading to the rapid development of models and techniques able to deal with it. In particular, data-driven AI models are becoming essential to analyze complex systems based on large data streams.

State-of-the-art models fail when dealing with overfitting in the data and suffer from performance loss when variables are highly correlated with each other. Many feature selection methods have been introduced to address these problems. Notably, it has been demonstrated that clustering-based methods for unsupervised feature selection

outperform traditional approaches in terms of accuracy.
The complexity of nonlinear dynamics associated with data streams from sensor networks makes standard clustering methods unsuitable in this context. For these reasons, in this paper, we propose a new clustering approach for time series useful for unsupervised feature selection, exploiting different complex network tools. In particular, we mapped time series segments in the network domain through natural weighted visibility graphs, extracted their degree sequences as feature vectors to define a similarity matrix between signals, used a community detection algorithm to identify clusters of similar time series, and selected a representative parameter for each of them based on the variable degree contributions.
The analysis of the results highlights two advantages deriving from the proposed method. The first is the ability to group together both homogeneous and heterogeneous physical parameters even when related to different functional areas of the system. This is obtained by capturing time series similarities not necessarily linked to the Euclidean distance. In the feature selection perspective, the approach, by considering 12 representative variables for the identified clusters and 8 independent signals that were not clustered, reduced the dimensionality of the dataset by 74.4%. Second, as an additional advantage for feature selection purposes, the method allows the discovery of hidden relationships between system components enriching the information content of the signal roles within the network.
Since the construction of a natural weighted visibility graph has time complexity $O(L^2)$, being L the number of samples in a time series interval, the proposed approach was intended as an offline filtering tool. In particular, being the visibility graph the bottleneck of the algorithm, the global time complexity is in the order of $O(TL^2)$, where T is the number of consecutive non-overlapping segments. Running the algorithm on a dataset of 11 months with time windows of 24 h took approximately 15 min. The idea is to consider the whole dataset at disposal to identify the overall most relevant signals, by averaging the contributions of all intervals. Thus, the resulting reduction in the dimensionality of data streams opens the possibility to simplify the condition monitoring system and its data.
If instead, a real-time tool for feature selection or time series clustering is of interest, it is possible to imagine the integration of the proposed algorithm into sensor network now-casting models, e.g., on a sliding window of 24 h the algorithm runs in less than 3 s.

3.1.2 Unsupervised Feature Selection of Multi-Sensor SCADA Data in Horizontal Axis Wind Turbine Condition Monitoring

Introduction

WT control and monitoring are based on multi-level Supervisory Control And Data Acquisition (SCADA) systems, able to connect individual turbines, wind farm substations, and meteorological stations to a central computer [338]. As WTs and farms grow in terms of size and complexity, SCADA systems are established as an industry standard of vital importance to guarantee effective operation, monitoring, control and reporting.

Since SCADA systems collect hundreds of data, several studies introduced health

status metrics of wind generators [339], or prediction algorithms for the power output to be used in combination with weather forecasts within the framework of non-programmable renewable power trading [340]. However, the low temporal resolution of SCADA data, together with the averaging effect, negatively affects the capabilities of SCADA-based monitoring approaches to detect some of the significant operating dynamics. In addition, the development of standardized data analytics on SCADA system may be difficult for two reasons. First, such a large amount of data might suffer from overfitting problems and, second, it may be affected by the individual evolution of turbine life cycles [341]. As a consequence, these circumstances encourage the development of data analytics approaches to determine and choose the most representative variables (from the SCADA infrastructure) that characterize WT operations.

Feature selection is typically considered a critical preprocessing step of input data sets to increase learners' performance [342]. The variable selection can be accomplished using either heuristic rules or automatic approaches based on a variety of techniques [124]. In general, there are three groups of feature selection methods: filters, wrappers and embedded methods [206]. While filters do not rely on ML but on features' correlation thresholds, wrappers use ML techniques and the selection process moves from the (out-of-sample) performance of a learning algorithm. The most commonly used techniques under wrapper methods are the Sequential Selection algorithms, e.g., Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS) [207, 208]. The SFS begins the search with an empty set of features, adding one feature at a time while trying to find the best set of combined selected parameters according to the evaluation criteria. The SBS, instead, refers to a search that begins with the full set of features, including all independent variables, and then removes the unimportant features until achieving the final set of selected significant parameters. Even though this latter approach may capture interacting features more easily, it is not fast nor computationally cost-effective [209]. Finally, embedded methods are a combination of filters and wrappers, where filters are integrated into the learner construction process [208]. This class includes a large family of DT methods. To mention but a few, XGBoost regressor [343], or RF [206].

In wind energy applications, the largest interest in feature selection algorithms is driven by the demand for forecasting wind speed as well as wind power output.

Limiting the attention to recently published papers, the survey of the literature proposes several unsupervised embedded method applications. To this end, Li Song et al. [344] proposed a Conditional Mutual Information feature selection in combination with a feedforward NN to determine wind farm generation predictors. Li Guo et al. [206] proposed the use of RF algorithm as a selector in combination with recurrent NN. A recent trend regarding wind power curve analysis involving feature selection has been presented in [345]. In this paper, the feature selection was performed through a sequential algorithm, and the results showed how the most important feature was the average wind speed on active power. In addition, the authors carried out tests on different WTs inferring that the selected features were peculiar to each turbine model.

In the context of short-term wind power forecasting, Zheng and Wu in [343] proposed the use of the XGBoost model with weather similarity analysis supported by Pearson correlation selector. Wind speed and wind direction turned out to be

the first two most influential variables. Another approach has been advanced by Qin et al. [346] where wind power prediction was weather division-based day ahead with the support of a filter feature selection. In particular, the analysis was carried out to compare the performance of six different filters. In [347] an exploration of feature selection influence on a 1 to 24-hour ahead forecast using a list of atmospheric variables was done through the SVR algorithm, varying the hyper-parameters to optimize the error. It was shown that wind prediction could be improved by more than 10% by adding and combining relevant input influential parameters. Recently, Huang et al. [342] suggested the concertation of three strategies including Maximal Information Correlation, along with Pearson and Spearman correlations for short-term wind power forecast. They showed that rotor speed, wind speed, and wind direction were the model input variables better reflecting the key features of the turbine-generated power.

The present paper proposes a novel unsupervised method for feature selection in a genuinely multivariate formulation, named Combine Predictive Power Score (CPPS). The method is cast as an embedded method combining a Pearson correlation filter with a learner based on MLP. Here the filter is intended to group homogeneous variables in the input data set to reduce the dimensionality of the search logic. The new method can be interpreted as an extension of the Predictive Power Score (PPS) approach [348], proposed as a data-type-agnostic wrapper. The CPPS augmentation resides in the selection of multi-variate WT predictors for different targets. Moreover, the search in the multi-dimensional feature space is carried out in a SFS fashion explored using q expanding subsets of variable combinations. The paper illustrates the CPPS method and the data preparation steps. Then, the case study is described which includes 9 WTs in an Italian on-shore wind farm. Finally, the results compare the performance of CPPS with the PPS baseline in the prediction of key target parameters.

Methodology

In this section, we present the workflow of the proposed methodology. In particular, we discuss data preparation which includes the creation of derived variables, outlier removal, time series clustering and scaling. Then, we introduce the PPS, a method for variable importance on which the proposed feature selection algorithm is based, namely CPPS.

Data Preparation

This phase is preliminary for the proposed feature selection method, essential to clean, enhance, and format the data properly.

Derived Variables In addition to the variables monitored by the SCADA system, we compute two derived variables, namely turbulence t and the yaw offset angle θ_o

between the wind direction θ_w and the nacelle direction θ_n, defined as:

$$t = \frac{v_{SD}}{v}$$
$$\theta_o = (\theta_w - \theta_n + 180)\%360 - 180$$

where v_{SD} and v are the wind speed standard deviation and mean in meters/seconds over the last 10 minutes.

Outlier Removal We applied both monovariate and multivariate outlier removal. First, we define a 6-hours sliding window for each signal and filter samples outside the interquartile range of the window. Then, we perform a density-based clustering [185] on the power curve of each turbine, filtering samples not belonging to the main power curve cluster, as shown in Fig 3.14.

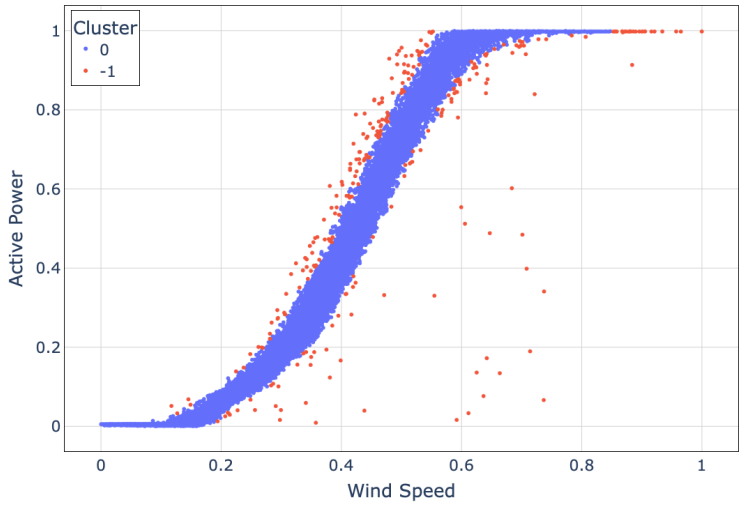


Figure 3.14. The figure shows how the density-based clustering identifies samples belonging to the power curve (cluster 0), thus isolating outliers (cluster -1).

Time Series Clustering To avoid highly correlated variables being selected as important features for each other, similar signals have been grouped using a time series clustering algorithm. In particular, we compute the correlation matrix using the Pearson correlation coefficient between pairs of variables. Then, we build a weighted graph considering the correlation matrix as an adjacency matrix and apply the Louvain method for community detection to cluster the original time series in homogeneous groups.

Figure 3.15 presents the output of time series clustering through a force-directed algorithm, namely the Fruchterman-Reingold layout. Each node is associated with a variable, while edges represent the mutual correlation between signals.

Moreover, the proposed methodology considers groups of homogeneous variables instead of single features for the process of feature selection.

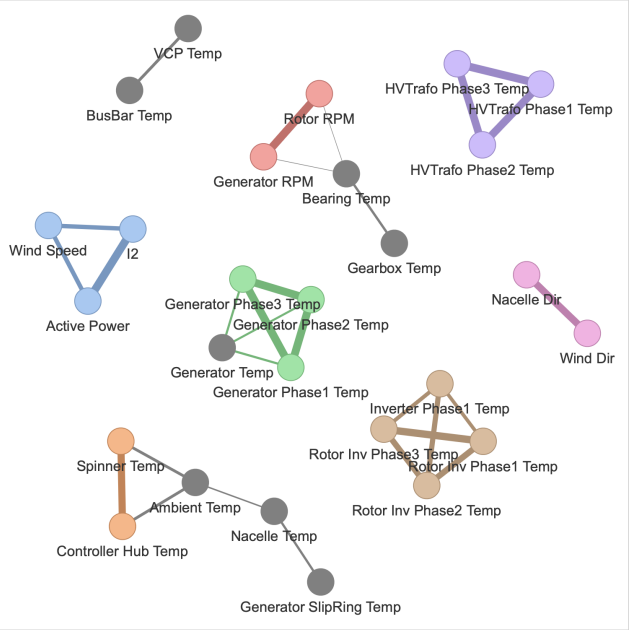


Figure 3.15. The Fruchterman-Reingold layout shows groups of homogeneous variables with the same color, as the output of the time series clustering algorithm. Grey nodes don't belong to any cluster. It is important to notice that some variables are not included in the graph since edge pruning was applied to weak connections for visualization purposes.

Time Series Scaling Since the proposed algorithm is based on a NN having multiple inputs and multiple outputs, it is necessary to scale the time series of the monitored variables. In particular, this was achieved by using the RobustScaler [349] which scales variables using statistics that are robust to outliers by removing the median and scaling data according to the interquartile range.

Feature Selection

In the next section, we describe the PPS for feature selection. Then, we present an original algorithm, namely CPPS, which extends PPS taking into account the effect of combining multiple variables.

Predictive Power Score PPS is an asymmetric, data-type-agnostic score that can detect linear or non-linear relationships between two variables [348]. The score is calculated using only one variable trying to predict another one and can be employed to select features as predictors for a target.

Since we are dealing with the regression of real-valued variables, the PPS considers the MAE as an evaluation metric. As baseline score, we calculate the MAE of a naive model ($MAE^{f_t}_{naive}$) that always predicts the median of the target variable f_t. Then, we select a candidate variable f_i as input for a model regressing the target variable f_t and compute the MAE ($MAE^{model}_{f_i,f_t}$). In this way, we can define the PPS

for a candidate feature f_t and the target variable f_t as:

$$PPS(f_i, f_t) = 1 - \frac{MAE^{model}_{f_i,f_t}}{MAE^{naive}_{f_t}} \tag{3.1}$$

The authors of the PPS selected DT as regression model [350].

The PPS ranges from 0 to 1 and assumes values close to 1 when the candidate input feature of the model has a high predictive power over the target. Otherwise, values close to 0 indicate that the candidate feature has low predictive power.

Combined Predictive Power Score In this paper, we propose a feature selection algorithm, namely the CPPS, that extends PPS by considering combinations of candidate input features $F_g = \{f_i\} \subseteq F$ for the prediction of multiple targets features $T = \{f_t\}$, where F is the set containing all input features, f_i is a candidate input and f_t a target.

In this way, the CPPS can detect linear or non-linear relationships between different groups of variables. The model employed for the definition of the CPPS has, therefore, to regress a multivariate output by considering multivariate inputs. We used a MLP having one hidden layer with $5 \cdot |F_g|$ units and ReLU activation [351].

Similarly to Equation 3.1, the CPPS for a candidate group of variables F_g and a target group T is defined as:

$$CPPS(F_g, T) = 1 - \frac{MAE^{model}_{F_g,T}}{MAE^{naive}_{T}}$$

where $MAE^{model}_{F_g,T}$ is the MAE of the MLP regression model and MAE^{naive}_{T} is the MAE of the naive model that always predicts the median of each target variable in T.

The pseudocode in Algorithm 1 shows how the combination of input variables as predictors for a specific target group T are selected by maximizing the CPPS.

Being C a set containing combinations of variables, $CPPS\_k(C, T, k)$ computes the score $CPPS(F_g, T)$ for each combinations of variables $F_g \in C$ with respect to the target group T, sorts them in descending order and selects the k-th group.

The inputs of the algorithm are F, T, q and ϵ, namely the set of original input features, the set of target features, the number of best combinations of variables to consider during each iteration and the minimum increase of the CPPS required to avoid early stopping.

At each iteration we consider the q combinations of variables in C that achieve the highest CPPS and merge them with the original features one at a time, thus generating new combinations F_g for the next iteration. At the beginning of the algorithm, C is initialized with combinations that coincide with the single variables in F.

The algorithm stops when an iteration achieves a CPPS improvement with respect to the previous iteration lower than ϵ.

To consider the output of the time series clustering algorithm previously discussed, for each feature f_i added to a new combination F_g, all variables belonging to the same homogeneous groups are added to the combination.

Data: F, T, q, ϵ
Result: F_{opt}
$C \leftarrow \{\{f_i\}, f_i \in F\}$;
do
  $C_{new} \leftarrow \{\}$;
  for $k \leftarrow 1$ **to** q **do**
    $F_k = CPPS\_k(C, T, k)$;
    foreach $f_i \in F$ **do**
      $F_g = F_k \cup \{f_i\}$;
      $C_{new} = C_{new} \cup \{F_g\}$;
    end
  end
  $C_{old} \leftarrow C$;
  $C \leftarrow C_{new}$;
while $CPPS(CPPS\_k(C, T, 1), T) - CPPS(CPPS\_k(C_{old}, T, 1), T) > \epsilon$;
$F_{opt} = CPPS\_k(C, T, 1)$;

Algorithm 1: The algorithm shows how the predictors for a target group are selected by maximizing the CPPS.

The selection process of predictors presented in Algorithm 1 provided an average speedup of 18x with respect to trying random combinations of features up to triplets.

Case Study

The dataset considered for the proposed methodology of feature selection is gathered from the SCADA system of 9 WTs belonging to the same wind farm. The data is collected every 10 minutes over one year of operation and is composed of 36 monitored variables shown in Table 3.5.

The next section will present the results relative to three of the turbines, as representative of the whole wind farm.

Results and Discussion

In this section, we present the results obtained by the proposed CPPS method, comparing them with the PPS on data collected from three turbines, namely Turbine 1, Turbine 2 and Turbine 3, as representative for the whole wind farm in terms of predictors.

Figure 3.16 shows the results for Turbine 1 in terms of CPPS evaluated for the target variable Active Power. The score is cumulative, meaning that each new variable (or group of homogeneous variables) on the x-axis is added to the previous ones, defining incremental combinations of features.

As can be seen, considering the cumulative contribution of the group consisting of the Generator RPM and Rotor RPM parameters together with the Blade Pitch Angle, a CPPS score of 0.913 is achieved. The optimal trade-off between the number of selected features and the predictive power corresponds to the point of maximum curvature of the CPPS curve, where the further addition of parameters does not improve the score significantly. It is important to note that, to avoid a signal highly

Table 3.5. Parameters monitored by the WT SCADA system.

Variable ID	Description
Active Power	Total active power
I2	Current
Wind Speed	Wind speed
Ambient Temp	Ambient temperature
Spinner Temp	Spinner temperture
Controller Hub Temp	Controller hub temperature
Bearing Temp	HSS gearbox bearing temperature
Gearbox Temp	Gearbox oil temperature
BusBar Temp	Busbar temperature
VCP Temp	Temperature on the VCP
Generator Phase1, 2, 3 Temp	Generator temperature in stator windings phase 1, 2, 3
Generator Temp	NDE generator bearing temperature
Generator RPM	Generator RPM
Rotor RPM	Rotor RPM
HVTrafo Phase1, 2, 3 Temp	Temperature in HV transformer phase 1, 2, 3
Inverter Phase1 Temp	Inverter phase 1 temperature
Rotor Inv Phase1, 2, 3 Temp	Rotor inverter phase 1, 2, 3 temperatures
Blade Pitch Angle	Blade pitch angle
Controller Top Temp	Temperature in the top nacelle controller
Generator SlipRing Temp	Temperature in the split ring chamber
Grid Choke Temp	Grid choke temperature
Hydraulic Pres	Hydraulic pressure
Hydraulic Temp	Hydraulic temperature
Nacelle Temp	Temperature in nacelle
V2	Voltage
Nacelle Dir	Nacelle direction
Wind Dir	Wind direction
Turbulence (t)	Wind turbulence
Offset Angle (θ_o)	Offset between the wind and nacelle directions

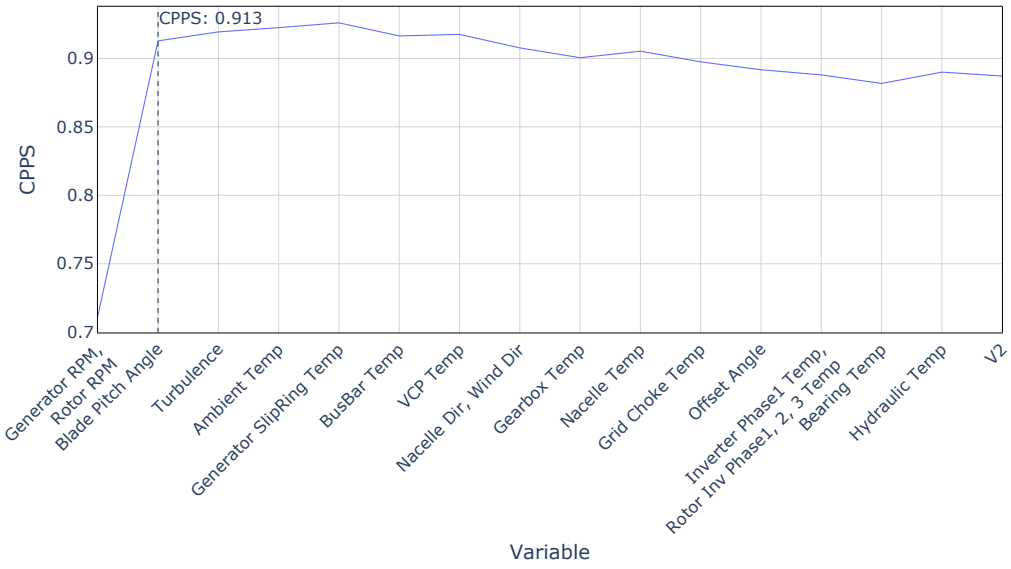


Figure 3.16. CPPS computed for Turbine 1 considering the Active Power as a target. Each new variable on the x-axis adds to the previous ones, forming the combined input group for which the CPPS is computed. The dashed line is drawn in correspondence to the point of maximum curvature of the scores.

correlated to a target being selected as its predictor, we filtered variables belonging to the same homogeneous group. In the case of the Active Power, the current I2 and the Wind Speed were filtered since trivial predictors. Combining one of these two variables with any other feature did not improve the CPPS significantly, thus leading to their selection as the only predictor. After removing the Wind Speed and I2, instead, multiple features were selected as predictors for the Active Power to achieve a similar CPPS, showing the importance of variable combinations.

Figure 3.17, on the other hand, presents the same results for the Active Power of Turbine 1 in terms of the PPS. Also in this case the variable with the highest predictive power with respect to the selected target is the Generator RPM, but the scores highlight an overall lower predictive power when compared to the proposed method. In fact, by considering the individual contribution of the three main predictors obtained with the CPPS method, we see that the Generator RPM, Rotor RPM, and Blade Pitch Angle achieve, respectively, a PPS of 0.75, 0.74 and 0.14. This means that the CPPS method achieves a score 0.16 higher than the standard PPS.

In terms of MAE, we compared the selected features by both methods using the same MLP model as a regressor. Considering the optimal combination selected using the CPPS composed by Generator RPM, Rotor RPM, and Blade Pitch Angle, the model achieves an error of 0.045. When considering, instead, the three most important features selected using the standard PPS, namely Rotor RPM, Generator RPM, and Bearing Temp, the MAE reaches 0.11, more than doubling. This confirms that, by considering the right combination of variables as input, a regression model can achieve higher performances.

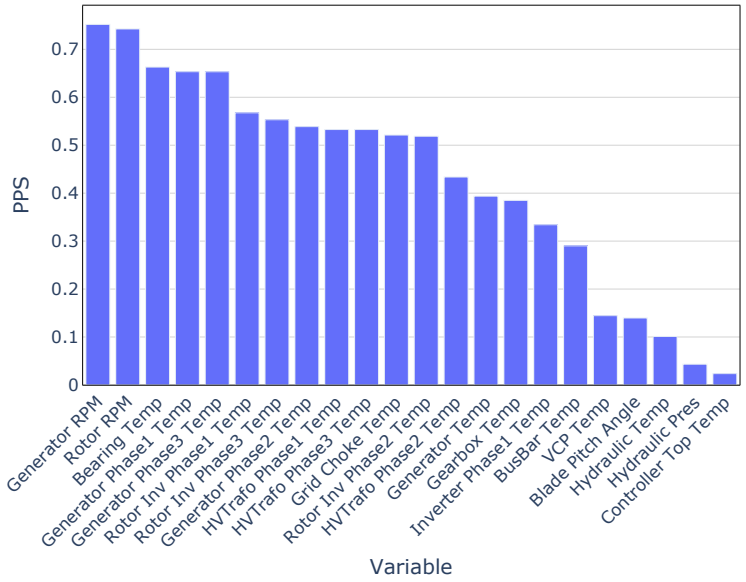


Figure 3.17. PPS computed for Turbine 1 considering the Active Power as a target.

Table 3.6 presents the CPPS scores for the three turbines. As for the plot in Figure 3.16, the variable (or group of homogeneous variables) in each row adds to the previous ones, forming the combination of features for which the CPPS is computed.

After the point of maximum curvature in correspondence with the Blade Pitch Angle, the proposed method selects the Turbulence, reaching a higher CPPS and a better performance of the predictive model. On the other hand, including this variable in the set of predictors for the Active Power only adds 0.023 points to the CPPS at most, contributing an order of magnitude less than the Blade Pitch Angle. The trade-off between the number of variables and model performance highly depends on the application. It is interesting to notice that the CPPS and the model performances start to degrade when adding too many variables to the predictors, making the feature selection process essential for the definition of an accurate regressor.

For comparison purposes, we compute the PPS for the same turbines and the results are shown in Table 3.7. The scores highlight that the Generator RPM and Rotor RPM are among the variables with the highest predictive power, reaching a maximum score of 0.796. It is worthwhile noting that the Blade Pitch Angle, included in the optimal combination of features by the CPPS method, achieves low scores, from 0.14 for Turbine 1 to 0.002 for Turbine 2. In fact, this variable on its own has almost no predictive power for the Active Power, but, when combined with the Generator RPM or Rotor RPM, achieves a CPPS over 0.9.

Notably, the multivariate output formulation of the proposed CPPS is useful when considering as target a group of correlated variables, like the Rotor RPM and the Generator RPM, the temperature in the HV transformer for all three phases of the generator temperature in the stator windings for all three phases. In this way, it is possible to select predictors for the target variables altogether, without having to compute a representative signal for the group.

Table 3.6. CPPS computed for three turbines in the same farm considering the Active Power as a target. Each new line refers to a variable that adds to the previous ones, forming the combined input group for which the CPPS is computed.

Turbine 1	CPPS	Turbine 2	CPPS	Turbine 3	CPPS
Generator RPM Rotor RPM	0.712	Generator RPM Rotor RPM	0.74	Generator RPM Rotor RPM	0.74
Blade Pitch Angle	0.913	Blade Pitch Angle	0.875	Blade Pitch Angle	0.903
Turbulence	0.919	Turbulence	0.898	Turbulence	0.923
Ambient Temp	0.922	Controller Hub Temp Spinner Temp	0.901	Nacelle Dir, Wind Dir	0.921
Generator SlipRing Temp	0.926	Generator Phase1, 2, 3 Temp	0.9	Controller Hub Temp Spinner Temp	0.913
BusBar Temp	0.916	Offset Angle	0.902	V2	0.903
VCP Temp	0.918	Nacelle Dir Wind Dir	0.905	Generator Phase1, 2, 3 Temp	0.9
Nacelle Dir Wind Dir	0.908	Nacelle Dir, Wind Dir	0.893	Offset Angle	0.9
Gearbox Temp	0.901	HVTrafo Phase1, 2, 3 Temp	0.89	Gearbox Temp	0.895
Nacelle Temp	0.905	Hydraulic Temp	0.883	Bearing Temp	0.893
Grid Choke Temp	0.897	V2	0.881	Nacelle Dir Wind Dir	0.896
Offset Angle	0.892	Bearing Temp	0.864	Generator Temp	0.892
Inverter Phase1 Temp Rotor Inv Phase1, 2, 3 Temp	0.888	Nacelle Temp	0.878	Hydraulic Temp	0.878

To provide an example of the proposed algorithm in a multivariate output scenario, the predictors for the Rotor RPM and Generator RPM were computed. As can be seen in Figure 3.18, the group composed by the Active Power, Wind Speed and I2 retains the highest CPPS of 0.89 and, by adding the Blade Pitch Angle, the score reaches 0.948. After the point of maximum curvature, the addition of further variables like V2 or Turbulence to the set of predictors degrades the CPPS, confirming that the optimal trade-off between the number of selected features and predictive power is achieved.

Being the standard formulation of the PPS limited to the computation of the predictive power with respect to a single target variable, the comparison was carried out considering both the Rotor RPM and the Generator RPM as targets. Figure 3.19 and 3.20 show that the predictors for the two variables are the same and that the scores are very similar, belonging the two features to the same homogeneous group.

Also in this case the variables with the highest scores are the Wind Speed, Active Power, and I2, with a PPS of 0.89, 0.88, and 0.88, respectively. Differently from the CPPS results, the next most important feature is the Bearing Temp with a score of 0.68 and not the Blade Pitch Angle which has no predictive power at all.

The variables selected by the CPPS and PPS methods were also compared in terms of MAE using the same MLP regressor model for the prediction of the Rotor RPM and Generator RPM. By considering the Active Power, Wind Speed, I2 and Blade Pitch Angle as predictors, the model achieves a MAE of 0.023. Training a regression model considering the Active Power, Wind Speed, I2 and Bearing Temp, instead, leads to a MAE of 0.046. The process of feature selection has improved the performance of the model and also the prediction of the Generator RPM and Rotor RPM.

These results highlight the advantages of the CPPS method which can consider multiple combinations of variables as predictors and select the optimal combination

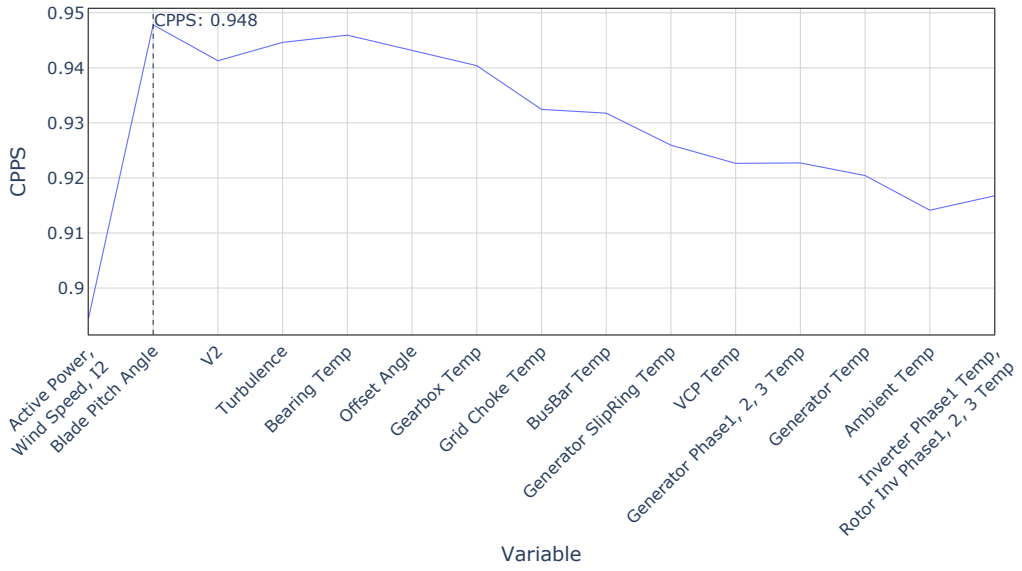


Figure 3.18. CPPS computed for Turbine 1 considering the Rotor RPM and Generator RPM as a target group. Each new variable on the x-axis adds to the previous ones, forming the combined input group for which the CPPS is computed. The dashed line is drawn in correspondence to the point of maximum curvature of the scores.

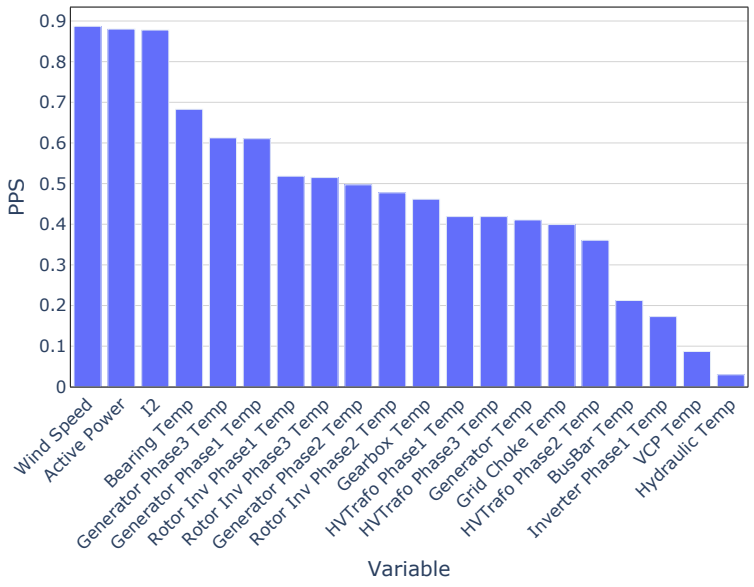


Figure 3.19. PPS computed for Turbine 1 considering the Rotor RPM as a target.

to maximize the score.

The CPPS was computed for all WTs in the wind farm and it is always the case that the score converges after finding the optimal combination of variables incrementally when the addition of further variables does not improve significantly the results in terms of MAE.

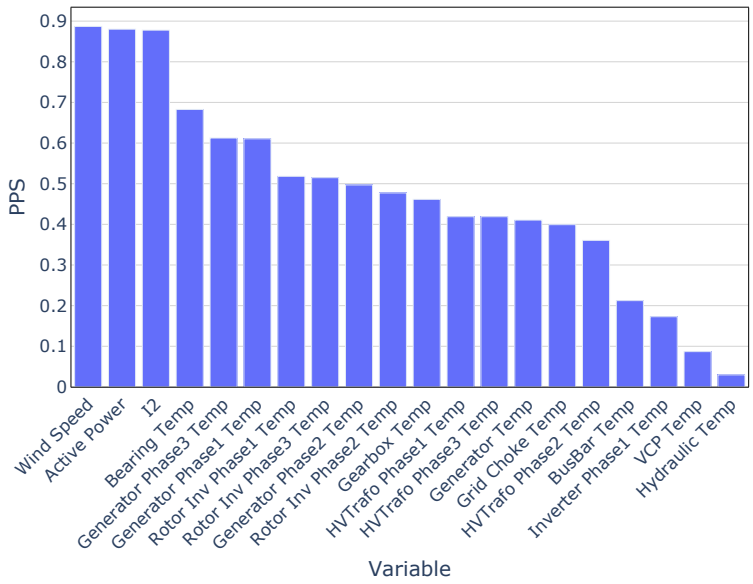


Figure 3.20. PPS computed for Turbine 1 considering the Generator RPM as a target.

Conclusions

In this paper, we propose an unsupervised method for feature selection of multi-sensor SCADA data in horizontal axis WTs.

The preliminary data preparation phase includes the definition of derived variables, outlier removal, and time series clustering for the identification of homogeneous signals and scaling of the sensor signals.

Then, we propose the CPPS, a method for feature selection with a multivariate formulation. This approach allows selecting combinations of features considering their predictive power with respect to a group of target variables. In particular, the CPPS is based on the error of a neural model that regresses the target variables by considering a combination of the input features. In this work, we also propose an algorithm for the selection of the combination of variables.

The methodology was tested on a year of data gathered from the SCADA system of 9 WTs belonging to the same wind farm and the results are reported considering the Active Power, Rotor RPM and the Generator RPM as target variables.

The results show that the CPPS method outperforms the PPS, achieving higher scores by considering combinations of features and allowing the improvement of the performance of regression models. The proposed algorithm converges to the smallest set of predictors with the highest predictive power, exploring multiple sub-sets of expanding variable combinations.

Moreover, the CPPS method is also more flexible than PPS, allowing to consider multivariate inputs and multivariate outputs.

Finally, the proposed unsupervised approach for feature selection can be applied to any set of multivariate time series, making it suitable, for example, also for other energy systems.

Table 3.7. PPS computed for three turbines in the same farm considering the Active Power as a target.

Turbine 1	PPS	Turbine 2	PPS	Turbine 3	PPS
Generator RPM	0.752	Generator RPM	0.796	Generator RPM	0.795
Rotor RPM	0.743	Rotor RPM	0.788	Rotor RPM	0.783
Bearing Temp	0.66	Generator Phase1 Temp	0.613	Generator Phase3 Temp	0.664
Generator Phase1 Temp	0.653	Generator Phase3 Temp	0.613	Generator Phase1 Temp	0.663
Generator Phase3 Temp	0.653	Bearing Temp	0.572	Bearing Temp	0.656
Rotor Inv Phase1 Temp	0.568	Generator Phase2 Temp	0.462	Generator Phase2 Temp	0.53
Rotor Inv Phase3 Temp	0.553	Grid Choke Temp	0.384	HVTrafo Phase3 Temp	0.522
Generator Phase2 Temp	0.539	HVTrafo Phase1 Temp	0.306	Grid Choke Temp	0.504
HVTrafo Phase1 Temp	0.533	Rotor Inv Phase1 Temp	0.295	Rotor Inv Phase1 Temp	0.488
HVTrafo Phase3 Temp	0.533	Rotor Inv Phase3 Temp	0.293	HVTrafo Phase1 Temp	0.487
Grid Choke Temp	0.521	HVTrafo Phase3 Temp	0.246	Rotor Inv Phase3 Temp	0.484
Rotor Inv Phase2 Temp	0.518	Rotor Inv Phase2 Temp	0.2	Rotor Inv Phase2 Temp	0.447
HVTrafo Phase2 Temp	0.433	Gearbox Temp	0.196	HVTrafo Phase2 Temp	0.435
Generator Temp	0.393	BusBar Temp	0.153	Generator Temp	0.368
Gearbox Temp	0.385	Generator Temp	0.129	Gearbox Temp	0.343
Inverter Phase1 Temp	0.334	HVTrafo Phase2 Temp	0.095	BusBar Temp	0.253
BusBar Temp	0.29	Inverter Phase1 Temp	0.053	Inverter Phase1 Temp	0.165
VCP Temp	0.144	VCP Temp	0.038	VCP Temp	0.134
Blade Pitch Angle	0.14	Hydraulic Temp	0.019	Blade Pitch Angle	0.084
Hydraulic Temp	0.101	Generator SlipRing Temp	0.008	Hydraulic Temp	0.043
Hydraulic Pres	0.06	Blade Pitch Angle	0.002	Ambient Temp	0.0

3.2 Predictive Maintenance for Renewable Energy Systems

The big amount of data collected by sensor networks, as discussed in the previous chapter, can contain information about processes, events and alarms, and provide valuable knowledge and insight into the underlying dynamical systems. In fact, by applying data-driven methods, it is possible to support strategic decision-making, resulting in a reduction of O&M costs, machine faults, repair stops, and spare parts inventory size. Moreover, predictive algorithms trained on collected data streams can increase the life of critical components, production, and safety [352, 353, 354]. All the mentioned advantages are strongly linked to maintenance procedures which directly affect the lifetime and efficiency of equipment by identifying faults and avoiding shutdowns. Nowadays, these procedures have become vital for industries, due to the growth in complexity of the interactions between different systems. Generally, maintenance is divided in four categories [355, 356], namely:

- *Corrective Maintenance* (CM): maintenance procedure applied only when equipment stops working. This is the simplest maintenance strategy since it requires stopping the process and repairing/replacing the faulty components.
- *Preventive Maintenance* (PvM): maintenance procedure performed periodically with a planned schedule in time or process iterations to anticipate equipment failures. Also referred to as scheduled maintenance, this strategy can increase O&M costs by scheduling unnecessary corrective actions.
- *Predictive Maintenance* (PdM): maintenance procedure that uses predictive algorithms to determine when maintenance actions are necessary. PdM leverages data collected from sensor networks describing the equipment condition and overall operational state of the systems, allowing the early detection of failures based on historical data.

- *Prescriptive Maintenance* (RxM): maintenance procedure that provides useful advice for making decisions and determining what actions to take to improve and optimize the process, such as mobilizing personnel and order spare parts. Not only RxM employs data-driven algorithms to predict the occurrence of failures as for PdM but also recommends which actions to take to prevent such failures.

A good maintenance strategy should improve the equipment condition, reduce failure rates and minimize O&M costs while maximizing the life of the equipment and the overall efficiency of the process. For this reason, PdM is the adopted strategy in many scenarios, due to its ability to optimize the use and management of assets [357, 358]. Moreover, since RxM is an extension of PdM that highly depends on the underlying process and the set of possible available maintenance actions, PdM is considered the preferred strategy to avoid failures and improve efficiency when it comes to directly harness data coming from sensor networks.
Many works that propose PdM strategies have been putting increasing attention on what is referred to as anomaly detection [359]. In the context of PdM, anomaly detection has to goal to identify anomalous patterns that deviate from the normal behavior of the system. Such patterns can represent early signs of failure and, if not timely addressed, can lead to breakdowns and the failure of equipment. Over the last years, PdM has been gaining prominence in multidisciplinary research groups, becoming a key component in the context of I4.0 and the RE sector [123, 130, 360].
The next sections present two papers we published in the context of predictive maintenance for RE systems. Specifically, the work in Section 3.2.1 [361] presents an algorithm for anomaly detection in a production factory of photovoltaic cells, while the paper in Section 3.2.2 [362] proposes an original unsupervised deep anomaly detection framework based on GCNs and AEs to isolate and anticipate failures in WTs.

3.2.1 Anomaly Detection in Photovoltaic Production Factories via Monte Carlo Pre-Processed Principal Component Analysis

Introduction

In the last years, PdM has been receiving ever-increasing attention and is considered fundamental in industrial applications. In fact, it contributes to guaranteeing healthy, safe and reliable systems, as well as to avoiding breakdowns that could potentially lead to a whole system shutdown.
As known, the main benefit of PCA lies in its capability to reduce the dimensionality of data by selecting the most important features that are responsible for the highest variability in the input dataset. PCA allows us to concentrate the analysis on a compressed version of the original dataset without compromising the reliability and robustness of a predictive model. Among other factors, a key quality in PCA is the inherent capability of processing large multivariate datasets as customary in industrial equipment sensor networks. As a result, PCA formed a field of choice in predictive analytics in several use cases, e.g. maritime and transport applications, as well as decision support systems in healthcare [363, 364].
On the other hand, the well-known disadvantage of PCA stems from the sensitivity

to outliers in the data. In this respect, in the literature four known algorithms have been very recently devised to sort outliers' observations out, namely the spherical principal component-based algorithm, PCA based on robust covariance matrix estimation, robust PCA and the PCA projection pursuit algorithm [365].
To this end, based on measurements collected by the sensor network of a PV production plant, the paper proposes Monte Carlo (MC) simulation as the preprocessing stage to deal with outliers before applying PCA [366, 367]. In this respect, the proposed approach is shown to be a valid alternative to relying on the classical IQR method to omit outliers when applying PCA for anomaly detection purposes.

Related works Recently, the scientific community has devoted much attention to the use of data analytics and machine learning models in the operation domains, e.g. manufacturing and energy management. In particular, many applications have focused on PdM and anomaly detection [368, 369, 370].
In this context, industrial systems have adopted PCA for detecting anomalous scenarios in their operational processes. In particular, Key Performance Indicators (KPIs) are usually defined starting from the PCA model in order to trigger alarms and prevent failures [371].
Many works focus on fault isolation techniques which are employed to classify different occurring errors and to isolate the system variables mostly affected by them [372]. Specifically, they often propose statistical methods for fault detection, like Hotelling T^2 or squared prediction errors Q [373, 374].
Even though plenty of these works deal with error classification and isolation in the context of anomaly detection and PdM, other papers and practical experiments shed light on innovative strategies to preprocess the input data that will feed the predictive model. To this end, MC simulation has been largely applied for data preprocessing to define more robust models. For example, in [375] the authors process geodetic data by applying MC simulation to perform uncertainty modelling [376].
However, choosing the statistical method for MC simulation becomes difficult when the involved dataset is highly affected by the presence of outliers. In this respect, a robust estimation procedure has been investigated in [377]: the authors exploit the median since it provides an estimator with the highest breakdown point and it always guarantees a feasible solution for the considered optimization problem.
In general, MC simulation is used as a valid preprocessing strategy to successfully manage uncertainty concerning experimental use cases in manufacturing and energy management, namely for PdM [378, 379, 380, 381] or predictive analytics purposes [382].
Moreover, the number of data points sampled by MC simulation is another crucial parameter, since it could lead to inaccurate outputs [383]. This parameter is particularly challenging to optimize since it strongly depends on the use case and the quality of data. In [384] the authors test different MC simulations to determine the relationship between the sample size and the accuracy of the sample mean and variance.
Even though larger samples could provide for a better estimation of the input

distributions, in [385] the authors demonstrated that a number of MC runs larger than the sample size can be unnecessary or even harmful to the modeling of the data.
Despite the clear advantage of such approaches, they often still need to be validated in practice. So, to the best of the authors' knowledge, this paper proposes the application of MC simulation to a real PV production scenario, as an effective way to preprocess the data stream coming from the sensors deployed throughout the production site.
The related literature also reports preprocessing techniques for similar anomaly detection scenarios based on the IQR method (e.g., [386]), which, however, offers only the property of outlier removal and not the additional benefit of outlier replacement that is consequential to applying MC simulation, as further discussed in Section 3.2.1.
The nex sections provide the use case description and the problem setting, followed by our contribution in terms of exploiting MC simulation as an innovative approach to data preprocessing with respect to the considered anomaly detection and PdM application. Then, we discuss PCA for anomaly detection, present the experimental setup and numerical results, and, finally, conclude the paper.

Problem setting

Enel Green Power needs to implement, in the production line of solar cells in the 3SUN Factory, an AI application capable of predicting faults relative to a piece of process equipment – the so-called Automatic Wet Bench (AWB) machine –, namely of predicting any malfunctioning of the fans that ventilate the different stations within the machine. The data collected on the Manufacturing Execution System (MES) are fed as input to the predictive analytics engine to predict faults.

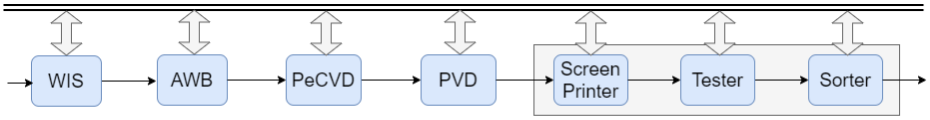


Figure 3.21. PV cell production line in the 3SUN Factory.

Use Case In Figure 3.21 we show the process steps involved in cell production. Each process equipment has a specific purpose: raw wafers enter the first machine in the line, the so-called Wafer Inspection System (WIS), to check the quality of the input wafers; then, they are subject to texturization and cleaning through the AWB equipment; next, the Plasma Enhanced Chemical Vapor Deposition (PeCVD) equipment is used for the deposition of a doped and un-doped layer of Amorphous Silicon (aSi on both side of the wafers. Then, the Physical Vapour Deposition (PVD) equipment is used for the sputtering process. Finally, the Screen Printer, Tester and Sorter equipment are responsible, respectively, for collecting the electric charge of the cell (fingers) and letting the flow between one cell and the other (Bus Bar) in the assembled modules, testing the electrical I-V measurements of the cells and classifying them depending on their performance.

The process equipment we refer to in this paper to predict the occurrence of faults is the AWB, where the wafers are chemically etched to roughen the surface to maximize the quantity of absorbed light and therefore the cell efficiency.
Along the production line, two parallel AWB machines are installed, each consisting of a loading station (the first one) and an unloading station (the last one) and, midway between the two, several stations where the chemical processes are performed. Within the AWB stage, the wafers are loaded onto specific containers called carriers, which move from one station to another until the process ends; the carriers do not enter all stations but only some of them, as the same task can be carried out indifferently by one station or another, so that the carrier is moved by the automation system to the first available station that can carry out the required task.
More specifically, the stations composing the production line serve three main purposes: pre-conditioning, texturing and cleaning. Each station is equipped with a sensor that records measurements when carriers enter and exit the station.
We now provide a brief description of the most frequently occurring fault inside the AWB and for which we design a suitable predictive analytics strategy. Such a fault is generally due to the malfunctioning of the fans that ventilate the different stations within each AWB stage.
For each AWB stage, there exist two drying tanks that must work properly in parallel and can never break down (not even alternatively), otherwise, the AWB throughput would be halved, thus compromising the whole production line. Since the fault episode is generally preceded by the occurrence of anomalous vibrations, there is room for a suitable predictive analytics strategy aimed at anticipating the occurrence of the fault through the detection of such vibrations. At a specific time slot, an unexpected error may happen in one of its machines and block production completely for several days.

Sensor Measurements The sensors mounted onto the production line stations measure several relevant parameters characterizing each station, such as station temperature, pump speed, flow speed, and ozone concentration level.
The measurements recorded by the sensors were collected only during the enter, exit and dosing phases of each carrier, thus leading to a non-constant sampling frequency. This produced many discontinuities of variable length in the sensor data streams, making standard time series analysis impossible. For this reason, the collected measurements were treated as an ordered set of samples rather than time series. To capture the time evolution of carriers going through a line, each sample is composed of the measurements coming from all the stations, collected during the enter, exit and dosing phases of a carrier.
Let k stations out of the total number N account for the main path drawn by a carrier entering the AWB stage to undergo pre-conditioning, texturing and cleaning. The remaining $(N - k)$ stations are parallel to the k principal ones and ensure the robustness of the whole AWB stage in the following way: if one of the k stations fails, there is at least a redundant station among the available $(N - k)$ that is properly working and can thus be entered by the carrier to undergo the whole production process.
For the sake of simplicity and without loss of generality, we assume to have k stations

only, and we neglect the remaining ones. Each station contains m *sensors*. Each sensor measures the carrier up to t times.
The considered dataset collects the t measurements carried out by the m sensors in the k stations over n *batches* or *carriers*, assuming a batch to account for a couple of wafers flowing through the whole production line.
So we wrap all the available data into a structured dataset represented by a matrix X with n rows and $y := k \times m \times t$ columns.
As our approach is completely data-driven, without losing generality and for the scope of the model, hereinafter we assume $k = 7$ and $m = 6$. Moreover, we assume $t = 3$, because each sensor measures the carrier three times while it is inside the considered station.

Monte Carlo based pre-preprocessing

In this section, we illustrate a novel preprocessing approach based on MC simulation and compare it with a commonly used method based on the IQR. This last is considered as a reference and the goal is to prove that our approach is a valid alternative to the IQR method. Since both these methods concern only the outlier removal phase, we also briefly describe the preliminary preprocessing steps required to standardize the data and handle missing values or flat signals.

Preliminary data cleaning Independently on the method, a preliminary data cleaning and preparation stage is required before removing outliers. The following steps are applied:

- signal filtering when the missing values are above 5% of the total number of measurements. Above this threshold, data interpolation can lead to distortions so we preferred to discard the involved signals.
- linear interpolation of signals when the missing values are less than 5% of the total number of measurements.
- flat signals removal when the derivative is zero for at least 50% of the signal length since constant measurements do not provide any meaningful information.
- signal standardization to make the scales of the different signals comparable. This operation was achieved by subtracting the mean value and dividing it by the standard deviation.

In the next sections, we describe the reference IQR method, followed by the discussion of the proposed approach based on MC simulation.

IQR method The InterQuartile Range (IQR) method is a simple but effective method used to identify outliers by isolating samples below the 25th percentile or above the 75th percentile [387].

Monte Carlo method In this paper, we propose an innovative method for removing outliers based on MC simulation, which has been largely applied in other scenarios like an estimation of sum, linear solvers, image recovery, matrix multiplication, low-rank approximation, etc. [388]. In our case, the idea is to generate new data points providing a more robust dataset by applying an estimator to random samples extracted from the original dataset.
By using the median estimator, there is no need to remove outliers from the raw data since this estimator is proven not to be affected by outliers [389].
Moreover, the size of the estimator dataset can be chosen arbitrarily, and can even be greater than that of the original one.
In the next sections we discuss the choice of the proper estimator, the number of samples used for MC simulation and the sliding window approach adopted to preserve the temporal locality of the sensor signals. Finally, we present the pseudocode illustrating the general preprocessing approach used to generate the new estimator dataset as input to the PCA model.

Mean versus Median The mean and the median are considered to be the most reliable estimators of the central tendency of a frequency distribution. Choosing the appropriate estimator is a challenging issue when using MC simulation since different results can lead to different correlations between signals, and thus different principal components when applying PCA. Let

$$x_i = (x_{p,z,w})_{\substack{p\ =\ 1,\cdots,k \\ z\ =\ 1,\cdots,m \\ w\ =\ 1,\cdots,t}} \tag{3.2}$$

denote the i-th row of the $n \times y$ data matrix X accounting for the measurement of sensor z during phase w in station p relative to batch i. In this way, each column f_j $(j = 1, \cdots, k \times m \times t)$ of X describes the temporal evolution of the measurements recorded by a specific sensor in a station during the processing of the batches.
Let

$$R^{IQR} = [r_{ij}^{IQR}] \tag{3.3}$$

with $i, j \in \{1, \ldots, n\}$, $i \neq j$, $-1 \leq r_{ij}^{IQR} \leq 1$ and

$$r_{ij}^{IQR} = \frac{\sigma_{f_i f_j}}{\sigma_{f_i} \sigma_{f_j}} \tag{3.4}$$

denote the correlation matrix computed between the columns of the dataset resulting from the IQR preprocessing. Recall that $\sigma_{f_i f_j}$ denotes the covariance between the columns f_i and f_j, whereas σ_{f_i} denotes the variance of the i-th column.
Let

$$R^{MC,median} = [r_{ij}^{MC,median}] \tag{3.5}$$

and

$$R^{MC,mean} = [r_{ij}^{MC,mean}] \tag{3.6}$$

with $(i, j \in \{1, \ldots, n\}$, $i \neq j)$, formulated as above, denote the correlation matrix computed between the columns of the dataset resulting from the median-based and

the mean-based MC simulation preprocessing methods, respectively.
Let

$$\Delta := [\delta_{ij}] = R^{IQR} - R^{MC} \tag{3.7}$$

account for the deviation between the two matrices, letting R^{MC} denote alternatively the correlation matrix relative to the median-based or the mean-based MC preprocessing method.
To evaluate which estimator suits our purpose best, we run the following statistical hypothesis test:

$$\begin{cases} H_0 : \delta_{ij} < \alpha & \forall i, j \\ H_1 : \delta_{ij} \geq \alpha & \forall i, j, \end{cases} \tag{3.8}$$

considering the difference between the correlation matrix computed after the application of the IQR method and the correlation matrix of the new dataset resulting from the previous section (that is, the MC dataset).
We can state that there exists a *significance* level α such that $\delta_{i,j}^{MC,median} < \alpha, \forall i, j$, and $\exists (i, j) : \delta_{i,j}^{MC,mean} \geq \alpha$, allowing us to choose H_0 only under the median-based MC method.
In particular, in the considered use case, the difference in the correlation matrices considering the median-based MC method is less than $\alpha = 6 \cdot 10^{-2}$ in absolute value and this proves to be a consequence of the median insensitivity to outlier observations.

Choosing the size of the Monte Carlo sample Choosing the proper number of samples has a significant effect on MC simulation since it considerably improves estimation reliability. We recall that samples are chosen out of the data matrix X, where x_i, as defined in (3.2), represents a generic row of X accounting for the measurement of sensor z during phase w in station p relative to batch i.
Up to the authors' knowledge, the literature claims that increasing the sample size reduces the variance and decreases the noise of the simulation results method [390]. Calibrating the sample size depends on many factors such as dataset size, the pursued objective and the complexity of the phenomenon the designer is modeling [391]. Therefore, we have tested different sample sizes before defining a methodology aimed at finding a suitable number of samples for each round in MC simulation.
By comparison with the highly dispersed original dataset, by increasing the number of samples we obtain a proportional decrease in variance. The desired sample size will allow the removal of only the outliers and at the same time preserve the rest of the information contained in the original dataset.
By excessively increasing the number of samples, the risk is that a significant part of the information is lost, thus affecting the accuracy of the PCA model. To select the proper sample size for MC-based outlier removal, we evaluate the impact this parameter has on the PCA model. To demonstrate that MC preprocessing is a valid alternative to the IQR-based preprocessing method, we compared the PCA models resulting from both approaches for different sample sizes, ranging from 1 to 100. In particular, we measured the proportion of the variance of the MC-PCA components that are explained by the IQR-PCA components in terms of R^2. In this way, high values of R^2 correspond to similar PCA models, thus confirming the

equivalent performance of the two preprocessing methods.
From Figure 3.22, it is evident that by considering 3 samples we obtain the highest value of R^2 (around 97.5%), thus demonstrating that, by choosing the proper sample size, the MC preprocessing method achieves very similar results to those obtained by the IQR-based preprocessing method.
Figure 3.22 presents the results of the previous steps where it is experimentally proven that PCA with 3-sample size has the best results.

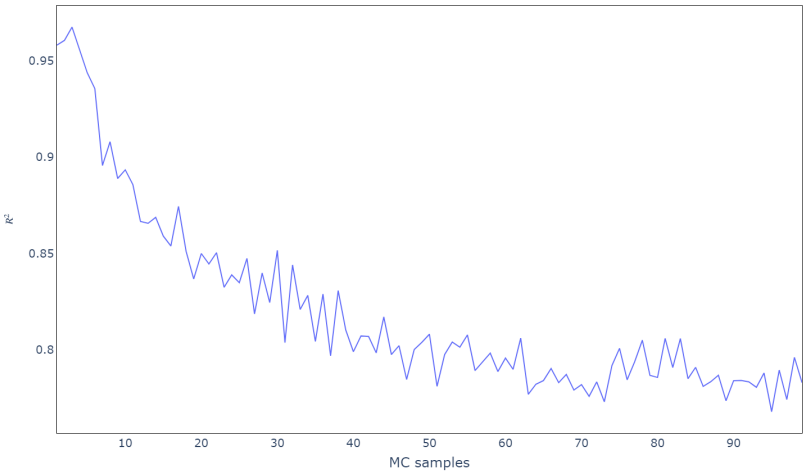


Figure 3.22. Testing R-squared for different sample sizes.

Preserving Trend Properties through a Suitable Choice of the MC Sample
Since PCA is based on the linear correlation among variables, any trends intrinsic to the signals themselves will not be considered. For this reason, random sampling among all the batches for median computation may result in the loss of the temporal dependencies characterizing signals.
Therefore, we refined the procedure for the MC sample selection accordingly. In particular, for each batch in the original dataset, we considered a time window centered around the batch itself. Samples considered for the median computation were therefore extracted inside the window, thus preserving the temporal locality among subsequent batches.

Pseudocode for the Preprocessing Method based on MC Simulation The pseudocode reported in Algorithm 2 illustrates the steps required to generate a new estimator dataset by using a preprocessing procedure based on MC simulation.

Principal component analysis for anomaly detection

PCA is a well-known method commonly used to reduce the dimensionality of a dataset, by transforming the original set of variables into a smaller one that still contains most of the information in terms of variance. In particular, it is a linear dimensionality reduction method based on SVD that projects the data on a lower

Input X : The original $n \times y$ data matrix
Output $\hat{X}$: The new estimator $\hat{n} \times y$ data matrix
Parameter $\hat{n}$: The size of the new estimator dataset
Parameter b : The number of samples considered for MC simulation

$i \leftarrow 0$
while $i < \hat{n}$ **do**
  $idx \leftarrow$ generateRandomInteger$[b, n - b - 1]$
  for j *in range[0,* $y - 1$*]* **do**
    $window \leftarrow X[idx - b : idx + b, j]$
    $\hat{X}[i, j] = \leftarrow$ median$(window)$
  end
  $i \leftarrow i + 1$
end

Algorithm 2: Preprocessing algorithm based on MC simulation.

dimensional space.
Being $\hat{n}$ the number of samples and let y the number of variables, the $\hat{n} \times y$ data matrix $\hat{X}$ is centered (by removing the mean of every feature) and SVD is applied on its covariance matrix, thus leading to a subset of orthonormal dimensions, namely the Principle Components (PCs) [392]. Since SVD computes PCs incrementally, their number depends on the pre-defined stopping criterion in searching for the next PC. A common strategy is to define the number of PCs as a function of the minimum variance information to be preserved with respect to the original dataset to compress the data sufficiently without losing too much information.
In this paper, we use PCA to perform anomaly detection. For this purpose, it is necessary to isolate a subset of data points associated with the normal behavior of the equipment. This subset is used as input to the PCA algorithm to compute a set of PCs considering as a stopping criterion high variance preservation (at least 90%). Having defined the $y \times z$ projection matrix Π composed by the z PCs, it is now possible to project each data point $\hat{x}_i$ on a lower dimensional space as

$$c_i = \hat{x}_i \Pi, \tag{3.9}$$

where c_i is the z-dimensional compressed version of $\hat{x}_i$. Then, we transform c_i back to its original space by multiplying it by the inverse of the matrix Π (being Π orthonormal, the inverse coincides with its transpose), thus obtaining the reconstructed version of the input data

$$\hat{x}'_i = c_i \Pi^T. \tag{3.10}$$

Finally, we compute the reconstruction error of the sample $\hat{x}_i$ as

$$e_i = |\hat{x}'_i - \hat{x}_i|, \tag{3.11}$$

where the vector e_i contains the residual of every input feature. Since the model is trained on normal behavior data, the reconstruction error should be low for samples belonging to the same distribution. However, during an anomalous scenario, the error is expected to be high since the associated samples will deviate from such distribution. By considering these vectors as KPIs for the stations in the production

lines, it is not only possible to detect anomalies when high errors occur, but also go back to the sensors mostly involved by inspecting the residuals of each single input feature.

Remark. *Thanks to the property of outlier replacement, the median-based approach, the optimal choice of the sample size and the preservation of any temporal dependencies characterizing the input signals, the proposed MC-based preprocessing approach turns out to be a robust alternative to IQR preprocessing. In fact, as can be seen from the experimental results, using median-based MC simulation in place of the IQR method for the preprocessing stage yields very similar results, although the number of PCs obtained when applying PCA after MC simulation is slightly higher than the number of PCs obtained when applying PCA after the IQR method.*

Remark. *The proposed preprocessing approach based on MC simulation is more suitable to the scenario of energy plants whose data require extensive cleaning. In this respect, if the input data are not cleaned enough, the IQR method, by isolating samples below the 25th percentile or above the 75th percentile, may end up removing a significant part of the original dataset, thus potentially compromising the quality of the subsequent data analytics task. Instead, MC simulation overcomes this obstacle by enabling the data scientist to tune the dimension of the dataset resulting from preprocessing according to the technical specifications of the considered task.*

Experimental Results of Anomaly Detection

In the experimental phase, we compared the results of the proposed anomaly detection approach considering both the IQR and MC preprocessing methods. In both scenarios, the relevant data were collected from the MES of the 3SUN Factory and a set of normal behavior samples was defined for training the PCA model.

Training and test sets According to the data format of the matrix X specified in Equation (3.2), we isolated a week of normal condition samples as *training set*, going from July 8th, 2020 to July 15th, 2020. This period was labeled as a period of *standard operation* by the operators working in the plant, together with other periods going from November 1st, 2020 to November 14th, 2020 and from May 1st, 2020 to May 8th, 2020, respectively, which we considered as *test sets*. The operators reported a fault in the plant on July 4th, 2020, so we isolated 24 days of data before the fault as a further test set to see if the proposed model detects the anomaly, possibly in advance.

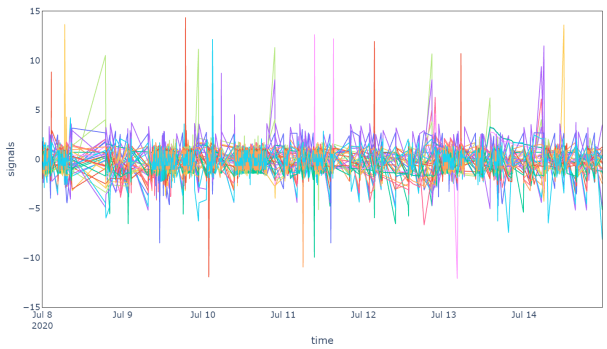
Preprocessing Phase Before the application of the anomaly detection approach based on PCA, we preprocessed the dataset as described in Section 3.2.1. In particular, 10 signals were filtered since they were completely flat, 12 signals were discarded since they presented an excessive rate of missing values, and 8 signals were linearly interpolated. After this phase, the dataset counted 36 variables to which the two outlier removal methods were applied.

Outlier Removal Results From the results, it is evident that both the IQR and MC methods were able to filter outliers successfully. In Figure 3.23**(a)**, the original sensor signals are plotted to highlight the presence of outliers, while in Figure 3.23**(b)** and Figure 3.23**(c)**, respectively, the preprocessed signals after the IQR and MC outlier removal methods are presented. It is important to notice that the IQR method does not handle the substitution of outliers (e.g., by interpolation) and it is limited to their identification and filtering. The MC method, instead, handles the presence of outliers by replacing all data points with the median over a sliding window, without requiring any additional substitution phase for the filtered values.

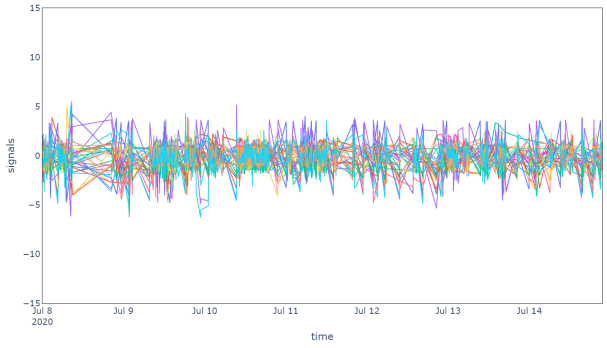
Anomaly Detection Results The PCA algorithm was run onto the two scenarios, namely considering an IQR and MC preprocessing phase, by setting as stopping criterion a minimum of 90% of explained variance. In the case of IQR, the PCs computed by the PCA algorithm were 16, while using the MC method led to 19 new dimensions.

Testing in Normal Operating Conditions The robustness of the anomaly detection model has been tested on normal behavior conditions (Figure 3.24) in a period going from November 1st, 2020 to November 14th, 2020, namely on the data collected during the week following the training period. Figure 3.24**(a)** plots the reconstruction errors of the model without preprocessing, while Figure 3.24**(b)** and Figure 3.24**(c)** display, respectively, the residuals considering IQR and MC for preprocessing. In all scenarios, the reconstruction errors are never persistently exceeding a threshold of 20 units, which was taken as a reference considering the errors computed on the training data. In fact, the operating conditions are very similar to the normal behavior period on which the model was trained and demonstrate that there are no substantial differences between the two preprocessing methods.

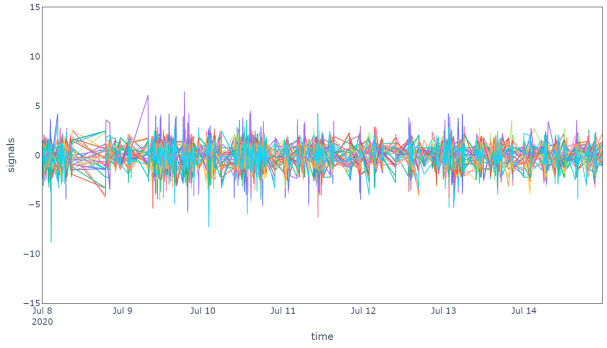
Testing in Anomalous Conditions As a final step, we evaluated the model in a critical period going from June 20th, 2020 to July 8th, 2020, during which a technical problem led to equipment failure, as reported by the operators. Figure 3.25 shows the residuals of the model considering no outlier removal phase (Figure 3.25**(a)**), the IQR (Figure 3.25**(b)**) and the MC (Figure 3.25**(c)**) preprocessing methods. In the proximity of the failure event (on July 4th, 2020), the anomaly is detected by the residuals drastically exceeding the training reference threshold of 20 units, anticipated by another reconstruction error spike on July 3rd, 2020. Without outlier removal, the residuals never persistently exceed the threshold in the period preceding the fault. When considering the IQR and MC methods, instead, residuals above 20 units are already frequent starting from June 20th, 2020, anticipating the fault by more or less two weeks. As for the normal behavior scenario, also in an anomalous period, the two preprocessing methods demonstrated their similarity by achieving comparable results.
It is important to notice that it is possible to isolate the sensors of the stations that are mostly related to anomalous conditions by inspecting the residual of each input feature of the model. In this anomalous period, stations 12 and 13 were isolated by looking at the large residuals two weeks before the fault. During the fault itself,



(a)



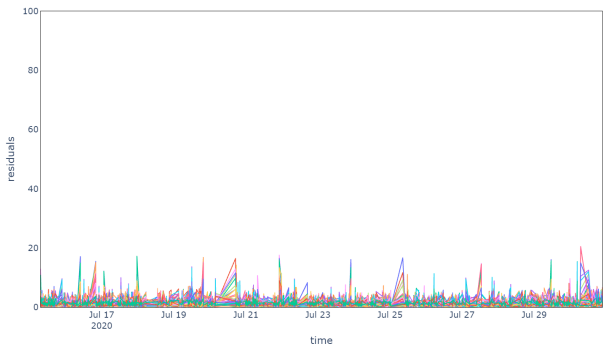
(b)



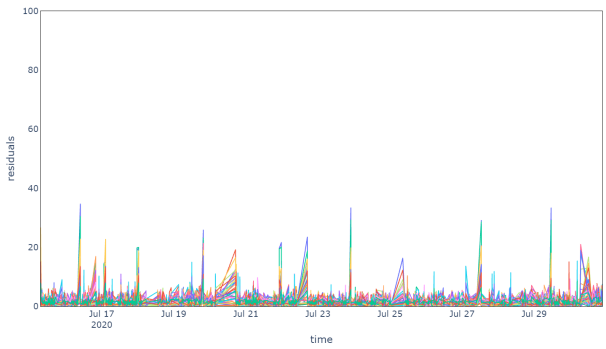
(c)

Figure 3.23. Figure (a) shows the sensor signals without the removal of outliers, while Figure (b) and Figure (c) represent the signals over time after the IQR and MC methods were applied respectively for the outlier removal phase.

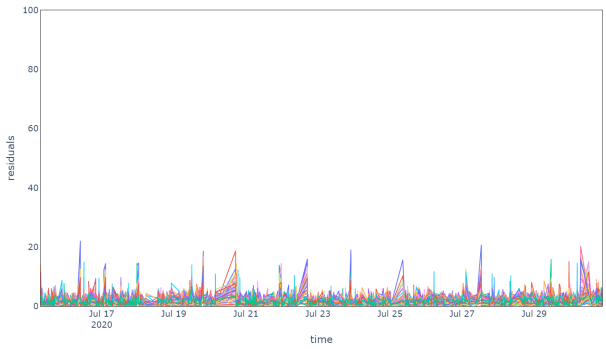
instead, stations 19 and 20 were involved according to the model reconstruction errors. The residuals two weeks before the fault and the residuals during the fault are associated with different stations since the failure of the system caused measurement errors not only to the most involved stations, namely 12 and 13, but also to other stations, namely 19 and 20.



(a)



(b)



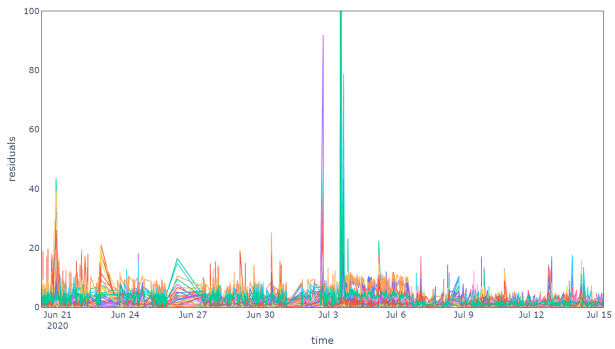
(c)

Figure 3.24. Figure (a) shows the KPIs associated with all sensors without the removal of outliers in a normal operating condition period, while Figure (b) and Figure (c) represent the KPIs (3.11) over time after the IQR and MC methods were applied respectively for the outlier removal phase.

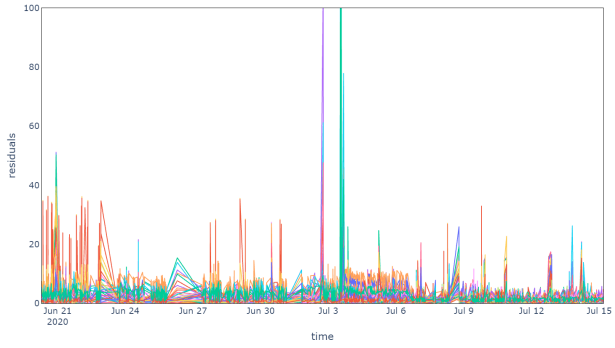
Discussion

The proposed method for data preprocessing based on MC simulation exhibits the following features:

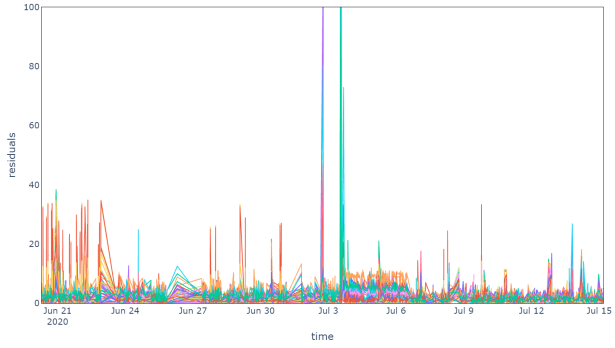
- preserving temporal locality with respect to the training dataset;



(a)



(b)



(c)

Figure 3.25. Figure (a) shows the KPIs (3.11) associated to all sensors without the removal of outliers before and after the break, while Figure (b) and Figure (c) represent the KPIs (3.11) over time after the IQR and MC methods were applied respectively for the outlier removal phase.

- outlier removal;
- outlier replacement, by contrast with traditional methods which are limited to outlier detection only (for example methods based on z-scores [393] or IQR techniques [394]).

As discussed in Section 3.2.1 and confirmed in [389], the median was chosen as the most accurate estimator to obtain a suitable dataset using MC simulation to be provided as input to the PCA-based model. In particular, the median-based MC method proved to be more effective against outlier observations with respect to the mean estimator.
Moreover, we selected the optimal sample size for MC simulation by measuring the proportion of the variance of the MC-PCA components that are explained by the IQR-PCA components in terms of R^2, thus confirming the equivalent performance of the two preprocessing methods. This was motivated by many considerations in the literature which report preprocessing techniques for similar anomaly detection scenarios based on the IQR method [386]. This analysis led to an optimal value of 3 samples to be considered for the median computation. In particular, we adopted a sliding window sampling approach to preserve the temporal locality of subsequent batches.
From the results in Section 3.2.1 it is evident that the IQR and MC-based preprocessing methods produce similar results, demonstrating their capability to successfully deal with outliers. Nevertheless, they present substantial differences. In fact, a standard method like IQR is limited to isolating outliers and possibly removing them from the dataset. This is a limitation because filtered observations generate missing values which require a substitution algorithm (e.g. mean imputation [395], KNN [396], linear interpolation [397]). The MC method, instead, intrinsically deals with outlier substitution by computing the median of randomly selected points, thus generating a new estimator dataset with an arbitrary number of samples.
The PCA models for anomaly detection demonstrated their capability to successfully anticipate a fault in the equipment as shown in several other works and practical experiments [368, 369, 370]. In particular, two PCA models were trained, respectively, on the IQR and MC preprocessed datasets. Both models highlighted an anomalous condition almost two weeks before the equipment failure by producing KPIs (residuals) above a reference threshold which was used to discriminate between healthy and anomalous states of the equipment as done in [398].
Moreover, it is important to notice that, without any preprocessing, the algorithm is unable to detect the anomalies with such an advance and is limited to spotting only the occurrence of the actual fault, which is also detected by the IQR and MC approaches.
Both models were also tested in standard operating conditions to prove their robustness to false alarms. In fact, in normal conditions, the residuals of the models never exceed the reference threshold persistently.
Finally, by inspecting the residual of each input feature of the model, the proposed approach allows the isolation of the sensors of the stations that are subject to anomalous conditions.
The authors have selected a reference period to calculate the average downtime for the AWB stage of the production line shown in Figure 3.21, and then to compute an estimate of the AWB downtime reduction resulting from the adoption of our predictive model.
Considering that only 50% of the predicted machine-down events can be avoided – in fact, only in some cases it is possible to take advantage of scheduled preventive maintenance to repair the equipment in advance –, the authors measured a reduc-

tion in AWB downtime by 0.55%. Assuming to extend the implementation of the predictive model to the entire equipment of the 3SUN production line (as shown in Figure 3.21), the authors expect an overall downtime reduction between 1% and 2%, which corresponds to an increase in the annual PV panel's production in the order of approximately 1-2 megawatts.

Conclusion

In this paper, we have presented a use case of robust anomaly detection applied to the scenario of a PV production factory – namely, Enel Green Power's 3SUN solar cell production plant in Catania, Italy – by considering a MC based preprocessing technique.
The proposed preprocessing algorithm demonstrated its ability to handle outliers like other standard methods, with the additional advantage of intrinsically dealing with outlier substitution and taking into account the temporal locality of subsequent samples.
After preprocessing, we trained an anomaly detection model based on PCA and defined a KPI for each sensor in the production line based on the model errors. In this way, by running the algorithm on unseen data streams, it was possible to isolate anomalous conditions by monitoring the KPIs and virtually trigger an alarm when exceeding a reference threshold.
The proposed approach was tested on both standard operating conditions and an anomalous scenario. In particular, it successfully anticipated a fault in the equipment with an advance of almost two weeks, but also demonstrated its robustness to false alarms during normal conditions.
Finally, given the data-driven nature of the approach and its robustness to outliers and irregular sampling frequencies, this approach could be applied to multiple lines in the production plant. In fact, as future work, we look forward to testing the proposed method on multiple types of equipment to further validate its scalability.

3.2.2 Deep Anomaly Detection in Horizontal Axis Wind Turbines using Graph Convolutional Autoencoders for Multivariate Time Series

Introduction

Wind energy is possibly one of the game-changer in future decarbonization scenarios, because of a plurality of factors. To mention but a few, incoming generations of multi-MW Wind Turbines (WTs) [399], the maturity of technology and infrastructures, and the cost competitiveness even in off-shore applications [400, 401, 402].
As reported in Hameed et al. [403], the most critical elements in WT energy converters reside in powertrain components, subject to highly irregular loads driven by wind turbulence and extreme weather conditions. As such, the fatigue loading of major structural components can be remarkably greater and peculiar when compared to other rotating machines.
Therefore, second to CAPEX investments are Operation and Maintenance (O&M) costs, being the most frequent faults on electric and control systems, followed by blades and hydraulic groups [404, 405]. In addition, failures (typically in generators

and gearboxes) entail high repair and replacement costs and result in long downtimes with significant loss of production. Remedial approaches to face O&M challenges advocate Condition-Based Monitoring (CBM) strategies capable of early detection and isolation of incipient faults. CBM is a key ingredient to enable condition-based maintenance, able to outperform the on-schedule state-of-the-art, in a view to identifying (at early stages) component degradation and limit unnecessary outage of WTs. In mechanical systems, CBM is typically based on the acquisition of high-frequency data (e.g., vibrational analysis), possibly processed through a variety of methods (see [124] for a recent review). However, this strategy suffers from several limitations as it requires the installation of additional sensors on WTs and specific data infrastructure, in fact, discouraging the implementation of high-frequency acquisition systems [403, 406].

On the other hand, modern WTs are integrated with sensor networks as part of SCADA systems for monitoring power-train status (e.g. bearing temperature, lube oil sub-system, etc.) with standard practice to record 10-minute averaged values and other statistics of the sensor time series. CBM of wind power generation plants through analysis of routinely collected SCADA data is envisaged as a viable means of forestalling expensive failures and optimizing maintenance through the identification of faults at the earliest possible stage [407, 408]. The challenge to operators is, therefore, in identifying the signature of failures within data streams and disambiguating those from other behavioral factors. The strong heterogeneity of signals, together with the loss of high-frequency temporal dependencies caused by the 10-minute averaging, makes the task very demanding [409].

In view of the lack of a comprehensive physical or mathematical model of WT operations, many data-driven methods based on 10-minute SCADA were recently proposed (see [122] for a systematic review). Probabilistic methods fail in modeling the proper temporal dependencies (and dynamics) in sensor networks [410]. For this reason, to take into account signals of mutual non-linearity and causal dependencies among WT components, most of the methods appeared to date rely on the use of NNs [411].

In the field of early fault detection, NNs are often employed to learn the normal operating conditions of the system and detect incipient faults by monitoring the real-time deviations from the standard behavior. The common assumption is that failure occurrences reflect a change of correlation among signals, causing a high multivariate reconstruction error.

To this end, several neural architectures have been proposed to capture anomalous scenarios based on prediction errors, where the neural models regress a target variable on a multivariate input. For example, approaches based on CNNs [409] have been developed, together with space-time fusion NNs combining convolutional kernels with recurrent units, such as LSTM or GRU, to extract multi-scale spatial and temporal correlations [412, 413, 414].

However, as reported in [415, 416], the standard convolutional operation of CNNs restricts the model to consider only local spatial structures in the signal time series rather than the general domain of the process. In addition, recurrent NNs for sequenced learning require iterative training, which may suffer from error accumulation, difficult training, and an increase in computational costs [415].

As a promising alternative to regression models, AE architectures have been recently

employed in unsupervised anomaly detection, given their ability to extract salient features characteristic of normal operating conditions. Examples of such architectures include deep AEs, denoising AEs, LSTM-based AEs, and CNN-based AEs [417, 157, 418, 419].
In this paper, we propose an original unsupervised deep anomaly detection framework that has at its core a neural architecture combining AEs and GCNs. Both AEs and GCNs have recently been employed for traffic forecasting or shape coding of buildings in maps where the graph formulation is intrinsic in the application domain [420, 421, 422]. Instead, we propose to adapt the formulation to multivariate time series, by modeling the sensor network as a graph where each node represents a sensor with specific feature vectors extracted from its time series. Owing to its multivariate formulation, we advocate the method to be able to analyze contextual anomalies in sensor networks [419].
In detail, we introduce a Graph Convolutional Autoencoder for Multivariate Time series (MTGCAE), composed by an encoder and a decoder based on GCNs adapted to multivariate time series. By representing the data as graphs, the structural information can be encoded to model the relations among entities and furnish more promising insights underlying the sensor data measurements, outperforming standard CNNs, especially in modeling arbitrarily structured systems like sensor networks [423, 424, 425].
To perform anomaly detection, the network is trained to learn the normal behavior of the system in an unsupervised fashion. By defining local and global indicators based on the model reconstruction errors, the framework triggers warnings after the application of a four-stage threshold method that aims at minimizing false alarms during normal operating conditions. In fact, only significant model errors are considered by filtering individual spikes and transient disturbances, allowing the generation of sensor-level warnings that isolate the assembly/sub-assembly mostly involved in the anomaly.
We tested the model on SCADA data gathered from 4 WTs belonging to the same wind farm, with a nominal power of 2 MW each [426]. The results showed that the proposed model can anticipate 10 SCADA log alarms with an average time to failure of about 23 days involving some of the most critical components, without triggering any false alarms. Furthermore, to validate the effectiveness of the model, two recently proposed neural architectures have also been applied to the same dataset, one based on an LSTM AE [418] and one on the combination of CNNs, LSTM cells and attention mechanisms [414]. The comparison confirms that the proposed model outperforms these two approaches in terms of evaluation metrics.
The rest of the paper presents the proposed MTGCAE neural architecture and discusses the building blocks of the deep anomaly detection framework. Then, we describe the case study and the obtained results, and, finally, we summarize the present work and draw our conclusions.

Neural Architecture

In this chapter, we describe the proposed neural architecture, namely a Graph Convolutional Autoencoder for Multivariate Time series, which is formulated as a combination of GCNs and AEs adapted for multivariate time series.

Graph Convolutional Autoencoder for Multivariate Time Series (MTGCAE) We propose a neural architecture based on the combination of GCNs and AEs, namely a Graph Convolutional Autoencoder for Multivariate Time series (MTGCAE), to exploit the extraction of multi-scale spatial and temporal correlations by encoding data as graphs.
In particular, as shown in Figure 3.26, the neural architecture consists of a multi-layer GCN which uses as input a graph representation of the sensor network. Each node in the graph represents one of the N signals and the edges quantify the degree of correlation between pairs of time series. More specifically, as input feature matrix $\mathbf{X}$ we consider a sliding window which is a $N \times F$ matrix. In this way, the feature vector $\mathbf{x_i}$ associated with the i-th node (i.e. the i-th sensor in the SCADA system) is composed by the values of the i-th time series in a time window of length F. As for the adjacency matrix $\mathbf{A}$, we define the entry (i, j) as the Mutual Information (MI) between the i-th and j-th signals or nodes in the graph sensor network [427].
The layers of the GCN are divided into an Encoder and a Decoder, each of which can be composed of multiple layers. Following the typical structure of AEs, the Encoder compresses the input to a latent representation, while the Decoder tries to reconstruct the original input as accurately as possible. Unlike standard AEs, here we adapt the formulation given in Equation 2.113 and 2.114 in order to consider as input a feature matrix instead of a vector.
With reference to the GCN layer formulation in Equation (2.112), the output $\mathbf{H_e}^{(l+1)}$ of the l-th Encoder layer can be written as function of the previous layer output $\mathbf{H_e}^{(l)}$ and the adjacency matrix $\mathbf{A}$ as:

$$\mathbf{H_e}^{(l+1)} = f(\mathbf{H_e}^{(l)}, \mathbf{A}) = \sigma(\mathbf{D}^{\frac{1}{2}}\hat{\mathbf{A}}\mathbf{D}^{\frac{1}{2}}\mathbf{H_e}^{(l)}\mathbf{W_e}^{(l)}) \tag{3.12}$$

where $\mathbf{W_e}^{(l)}$ is the trainable weight matrix of the layer and σ is the ReLu activation function [428]. Considering an Encoder with L_e layers, we have that $\mathbf{H_e}^{(0)} = \mathbf{X}$ and that $\mathbf{H_e}^{(L_e)} = \mathbf{H}$, where $\mathbf{H}$ is an $N \times K$ matrix representing the compressed version of the input feature matrix ($K < F$) after passing through the Encoder.
The Decoder, instead, maps the compressed feature matrix back to its original space and can be formulated in a mirrored way as:

$$\mathbf{H_d}^{(l+1)} = f(\mathbf{H_d}^{(l)}, \mathbf{A}) = \sigma(\mathbf{D}^{\frac{1}{2}}\hat{\mathbf{A}}\mathbf{D}^{\frac{1}{2}}\mathbf{H_d}^{(l)}\mathbf{W_d}^{(l)}) \tag{3.13}$$

where the subscript d is adopted to discriminate Decoder matrices. Considering L_d layers, we have that $\mathbf{H_d}^{(0)} = \mathbf{H}$ and that $\mathbf{H}^{(L_d)} = \mathbf{X'}$, where $\mathbf{X'}$ is the reconstruction of the input feature matrix $\mathbf{X}$.
Similarly to AEs, the reconstruction error $\mathcal{L} = ||\mathbf{X'} - \mathbf{X}||_F^2$ is minimized during training using the Backpropagation algorithm.
The MTGCAE is trained to reconstruct the sensor signals assuming a reference state. When reapplying it to unseen data during anomalous conditions, we expect the prediction of the trained network to deviate from the actual signals, thus generating residuals.

Deep Anomaly Detection Framework

In this chapter, we discuss the main steps of the proposed deep anomaly detection framework shown in Figure 3.27. First, we describe how the monitored signals

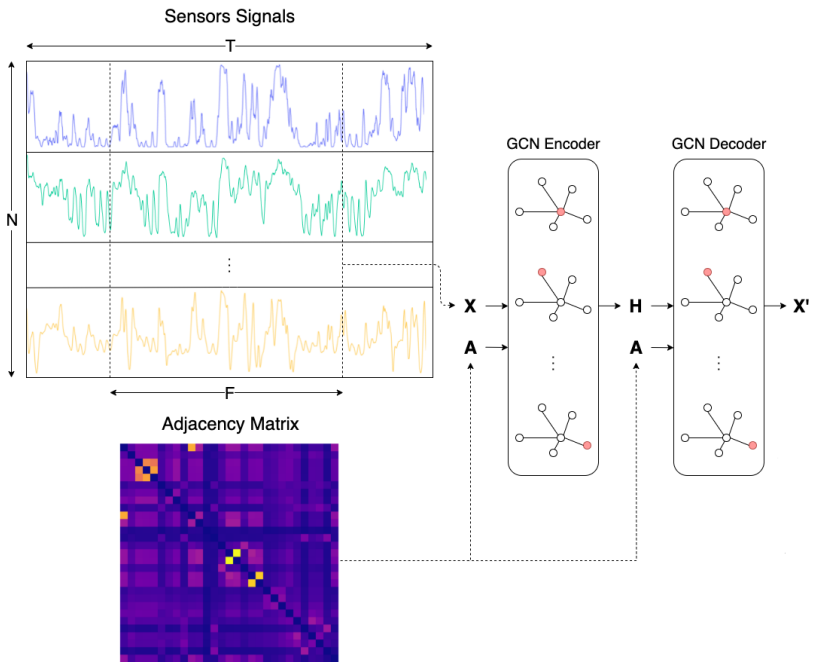


Figure 3.26. Detail of the proposed MTGCAE neural architecture. Starting from N sensor signals, the adjacency matrix $\mathbf{A}$ is computed through MI and a sliding window $\mathbf{X}$ of length F is extracted as input for the GCN Encoder. The compressed output representation $\mathbf{H}$, together with the adjacency matrix $\mathbf{A}$, are used by the GCN Decoder to produce a reconstruction of the input signals $\mathbf{X}'$.

are preprocessed and the proposed MTGCAE model application. Then, we define global and local health indicators together with a four-stage threshold approach for anomaly detection.

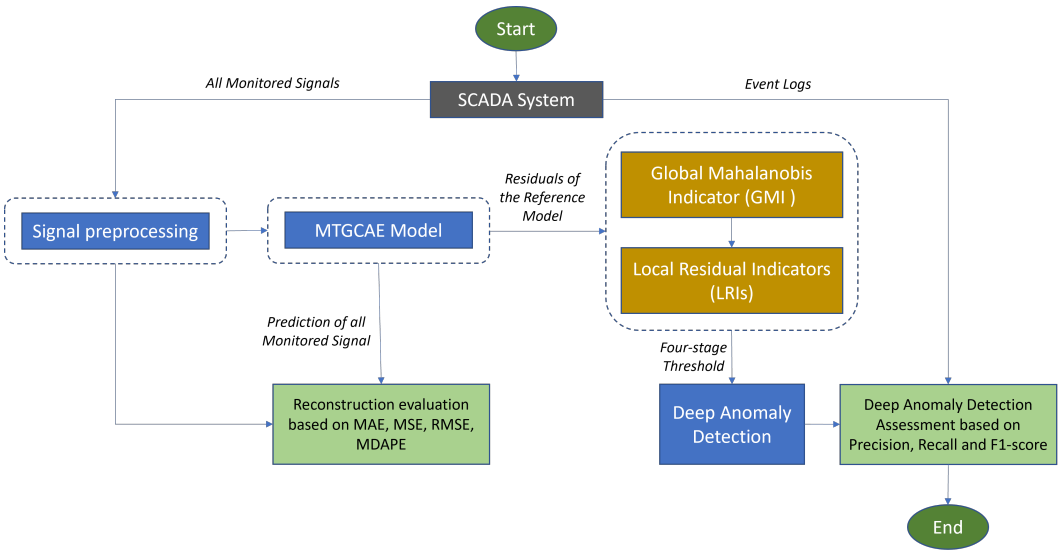


Figure 3.27. Diagram of the proposed framework for deep anomaly detection.

Signal Preprocessing Before the training of the proposed neural architecture, all monitored signals are preprocessed by deleting the records having missing values. When few isolated points, a linear interpolation was applied without introducing distortion in the data [151].
Then, signals presenting high levels of noise were smoothed using the Savitzky–Golay filter [429].
Finally, extreme outliers were filtered using the 5-sigma rule and data was scaled using the min-max normalization.

MTGCAE Model To take into account the temporal dependencies in time series, data are explored using sliding windows. Given the data matrix $\mathbf{Z} = (\mathbf{z_1}, \ldots, \mathbf{z_i}, \ldots, \mathbf{z_T})^T$, where $\mathbf{z_i}$ is the i-th N-dimensional multivariate sample, with N the number of signals and T the number of time observations, the i-th sliding window is an $N \times F$ matrix defined as:

$$\mathbf{S_i} = (\mathbf{z_{i-F}}, \mathbf{z_{i-F+1}}, \ldots, \mathbf{z_i})^T, \quad i = (F, F+1, \ldots, N) \tag{3.14}$$

where F is the length of the window. As a consequence, the dataset $\mathbf{S}$ is structured in successive $w = T - F + 1$ sliding windows:

$$\mathbf{S} = (\mathbf{S}_1, \ldots, \mathbf{S}_i, \ldots, \mathbf{S}_w)^T \tag{3.15}$$

To isolate the reference period used to train the proposed MTGCAE model for anomaly detection, as discussed in [419], we employed an unsupervised approach based on the assumption that the hidden layers of deep AEs are capable of capturing intrinsic properties of the majority of the data, representing the normal operation. In detail, to sample a subset $\mathbf{S_n}$ of normal behavior windows, we trained the proposed MTGCAE architecture on all windows in $\mathbf{S}$ to learn the most common patterns in the data. Downtimes caused by failures are not captured by the model since rare operating conditions, thus generating high model residuals during their occurrence. In this way, it is possible to isolate outages in an unsupervised manner and exclude them from $\mathbf{S_n}$.
To prevent fault precursors from being included in the dataset of standard behavior $\mathbf{S_n}$, we exclude all windows falling in a potentially anomalous time period preceding each downtime consisting of half the Mean Time Between Failure (MTBF), since further away from the last repair.
Once the normal operating conditions are isolated, the MTGCAE model is trained on $\mathbf{S_n}$ and employed for anomaly detection and early fault prediction.
Based on the above, we split $\mathbf{S_n}$ into training $\mathbf{S_{train}}$ and validation $\mathbf{S_{val}}$ both containing only standard behavior windows, and a testing set $\mathbf{S_{test}}$ which includes both normal and anomalous windows. In particular, the model is trained for E epochs using the Adam optimizer, considering early stopping to avoid overfitting [430].
It is important to notice that the latent representation $\mathbf{H}$ of the MTGCAE directly depends on the parameter K, which should be smaller than the window size F to compress the inputs properly but should also be large enough to capture the most common patterns in the data.
Another crucial parameter is the window size F, which defines the temporal depth

of the model. If too small, it will capture only small-scale local patterns and, if too large, it will process excessively wide time windows and fail to capture their temporal patterns.

Global Mahalanobis Indicator (GMI) and Local Residual Indicators (LRIs) The errors of the MTGCAE model are used to specify a rule to warn as early as possible about incipient anomalies. To this end, we define the Global Mahalanobis Indicator (GMI), reflecting the operating status of the whole sensor network, and a Local Residual Indicator (LRI), for each monitored variable.
The GMI is computed as the distance between the model multivariate reconstruction error and the reference multivariate probability distribution of the errors obtained on the validation set $\mathbf{S_{val}}$ using Minimum Covariance Determinant [157]. The LRI for each signal is, instead, defined as its specific reconstruction error.

Four-stage Threshold To generate a prompt warning before the occurrence of failures and, at the same time, reduce false alarms during normal operation, a multi-stage threshold is designed for the GMI and LRIs. In particular, as in [410], it evaluates the magnitude of the model errors to detect deviations from standard conditions but also considers their duration in time to attenuate the effect of individual spikes and transient disturbances. To make the warnings produced by the model more robust to false alarms, we apply a four-stage threshold, considering first the GMI and, then, the LRIs.
The four sequential filtering steps applied to the local and global indicators are the following:

Two-stage threshold on GMI:

1. filter GMI values below a threshold d_m to consider only significant multivariate reconstruction errors;
2. filter GMI values that have a duration less than F to consider only residuals persistent in time for at least the length of the model input sliding window;

Two-stage threshold on LRI:

3. for each signal i, filter LRI values below a threshold d_i to consider only significant reconstruction errors;
4. for each signal i, filter LRI values that have a duration less than F to consider only residuals persistent in time for at least the length of the model input sliding window.

When the conditions of all four stages are satisfied in the presented order, the model triggers a warning for the sensor having the highest LRI.

Experimental Results

In this section, the proposed fault detection framework is validated using the open dataset available at [426]. More details can be found below.

Dataset Description The data is collected from four WTs belonging to the same wind farm, each having a diameter of 90 m, a maximum rotor speed of 14.9 rpm, and a maximum rated power of 2 MW at a nominal wind speed of 12 m/s. The wind farm is ranked class 2 according to the standard IEC 61400 [431]. The complete description of the technical information of the WTs is given in Table 3.8.

Table 3.8. Technical information of each turbine.

Rated power (kW)	2000
Cut-in wind speed (m/s)	4
Rated wind speed (m/s)	12
Cut-out wind speed (m/s)	25
Rotor diameter (m)	90
Rotor swept area (m^2)	6362
Number of blades	3
Max rotor speed (rpm)	14.9
Rotor tip speed (m/s)	70
Rotor power density 1 (W/m^2)	314.4
Rotor power density 2 (m^2/kW)	3.2
Gearbox Type	Planetary/spur
Gearbox stages	3
Generator type	Asynchronous
Max generator speed (rpm)	2016
Generator voltage (V)	690
Grid frequency (Hz)	50
Hub height (m)	80

All WTs are equipped with a SCADA system for the monitoring of multiple parameters collected from the main components together with ambient measurements. In particular, for each WT we considered a separate dataset composed of 30 monitored parameters listed in Table 3.9. The dataset covers a period of about 20 months (from January 1, 2016, to September 1, 2017).

In addition to the sensor signals, we considered the event log that includes all alarms recorded by the SCADA system on the four WTs during the reported period. These events include all potential operational risks, which can be seen as anomalies reducing the remaining useful life of components. In fact, the SCADA system supervises the operating status of the wind turbines and protects them from extreme loads. In this way, when a critical signal exceeds predefined operating thresholds, an event is triggered and recorded in the log file.
Concerning the Reliawind turbine taxonomy presented in [432], the event log available in [426] mainly contains the details of the anomalies recorded at the assembly and sub-assembly levels, and only for some alarms at the component/part level. Starting from this, we filtered out all false alarms and minor events that did not lead to repair or replacement actions for the component.
Table 3.10 lists all the events we considered for this study, detailing the turbine IDs, the assembly/sub-assembly involved, the date and time of the alarm recorded by the SCADA system and the type of action taken by the operators to restore proper operation (repair or replacement). Figure 3.28, instead, shows the time distribution of the alarms for each WT over the investigated period.

From the analysis of the logs contained in Table 3.10 and Figure 3.28 it is possible to notice that most of the recorded alarms concern the drive train and

Table 3.9. Parameters monitored by WT SCADA system.

Signal ID	Description	Component
Gen_Bear_Temp	Temperature in generator bearing 1 (Non-Drive End)	Generator Bearings
Gen_Bear2_Temp	Temperature in generator bearing 2 (Drive End)	Generator Bearings
Gen_RPM	Generator rpm	Generator
Gen_Phase1_Temp	Temperature inside generator in stator windings phase 1	Generator
Gen_Phase2_Temp	Temperature inside generator in stator windings phase 2	Generator
Gen_Phase3_Temp	Temperature inside generator in stator windings phase 3	Generator
Gen_SlipRing_Temp	Temperature in the split ring chamber	Generator
Hyd_Oil_Temp	Temperature oil in hydraulic group	Hydraulic
Gear_Oil_Temp	Temperature oil in gearbox	Gearbox
Gear_Bear_Temp	Temperature in gearbox bearing on high-speed shaft	Gearbox
Nac_Temp	Temperature in nacelle	Nacelle
Nac_Direction	Nacelle direction	Nacelle
Rtr_RPM	Rotor rpm	Rotor
Amb_WindSpeed	Wind speed	Ambient
Amb_WindDir_Relative	Wind relative direction	Ambient
Amb_WindDir_Abs	Wind absolute direction	Ambient
Amb_Temp	Ambient temperature	Ambient
Prod_TotActPwr	Total active power	Production
Prod_TotReactPwr	Total reactive power	Production
Grd_Prod_PsblPwr	Grid Power Request	Grid
HVTrafo_Phase1_Temp	Temperature in HV transformer phase L1	Transformer
HVTrafo_Phase2_Temp	Temperature in HV transformer phase L2	Transformer
HVTrafo_Phase3_Temp	Temperature in HV transformer phase L3	Transformer
Cont_Top_Temp	Temperature in the top nacelle controller	Controller
Cont_Hub_Temp	Temperature in the hub controller	Controller
Cont_VCP_Temp	Temperature on the VCP-board	Controller
Cont_VCP_ChokcoilTemp	Temperature in the choke coils on the VCS-section	Controller
Cont_VCP_WtrTemp	Temperature in the VCS cooling water	Controller
Spin_Temp	Temperature in the nose cone	Spinner
Blds_PitchAngle	Blades pitch angle	Blades

power sub-systems. In particular, two repairs were carried out on the T01 turbine, one involving the gearbox and the other the transformer, in response to component level anomalies detected respectively on the gearbox pump (July 18, 2016) and the transformer fan (August 11, 2017).
As for the three alarms recorded on the T06 turbine, they refer to generator anomalies that occurred in the period June-November, 2016, and two of these required a full replacement at the assembly level.
The alarms of the T07 turbine, on the other hand, concern two episodes of high temperature recorded in the transformer on July 10, 2016, and August 23, 2016, and an anomaly in the generator that occurred about a year later (August 21, 2017), which required specific repair interventions.
Finally, in turbine T09, four distinct alarms were recorded on the generator bearings, two of which concerned high-temperature events on June 7, 2016, and August 22, 2016, requiring a repair of the damaged parts. The other two alarms, instead, were triggered on October 17, 2016, and January 25, 2017, and involved major damages that led to the replacement of the components.

MTGCAE Parameter Setting The analysis was carried out for each wind turbine, from the preprocessing and data preparation to the training of the model.

Table 3.10. Main alarms reported in the maintenance log file available in ref. [426].

Turbine ID	Alarm ID	Alarm timestamp	Assembly/Sub-assembly	Type of alarm	Type of action	
					Repair	Replacement
T01	A01GX	18/07/2016; 02:10	Gearbox	Gearbox pump damaged	x	
T01	A01T	11/08/2017; 13:14	Transformer	Transformer fan damaged	x	
T06	A06G1	11/07/2016; 19:48	Generator	Generator damaged		x
T06	A06G2	04/09/2016; 08:08	Generator	High temperature generator error	x	
T06	A06G3	27/10/2016; 16:26	Generator	Generator damaged		x
T07	A07T1	10/07/2016; 03:46	Transformer	High temperature transformer	x	
T07	A07T2	23/08/2016; 02:21	Transformer	High temperature transformer	x	
T07	A07G	21/08/2017; 14:47	Generator	Generator damaged	x	
T09	A09GB1	07/06/2016; 16:59	Generator bearings	High temperature generator bearing	x	
T09	A09GB2	22/08/2016; 18:25	Generator bearings	High temperature generator bearing	x	
T09	A09GB3	17/10/2016; 09:19	Generator bearings	Generator bearings damaged		x
T09	A09GB4	25/01/2017; 12:55	Generator bearings	Generator bearings damaged		x

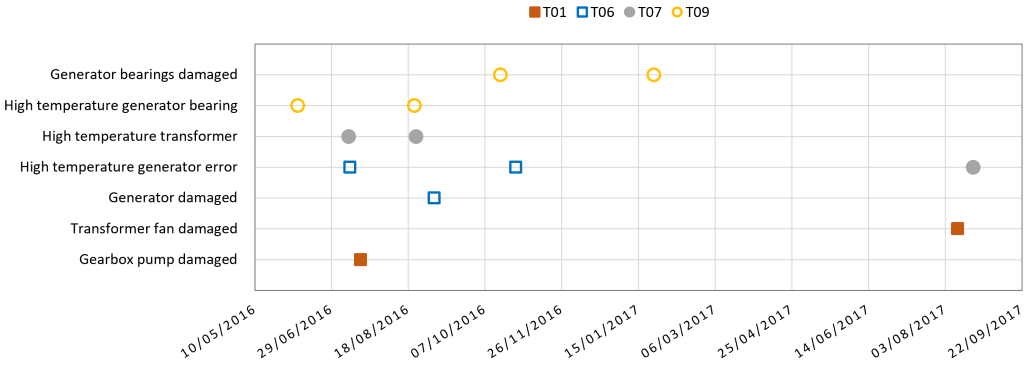


Figure 3.28. The time distribution of logs recorded by the SCADA system at sub-assembly and part level for the four WTs.

The MTGCAE was trained for $E = 50$ epochs considering early stopping to avoid overfitting. Specifically, we considered one layer for the Encoder having K output features as latent representation and one layer for the Decoder to reconstruct the input data. A single layer was sufficient for the autoencoding process and stacking more layers did not significantly improve the performance of the model.
The parameter K was set to be proportional to F so that the number of neurons scaled with the length of the input sliding window, allowing the model to increase its complexity for larger windows. In particular, we set $K = \frac{F}{6}$ to compress the inputs to a lower dimension ($K < F$) and at the same time provide the network with enough neurons to reconstruct the inputs. The sliding window size F was set to 144 (24 hours) using grid search aimed at minimizing the MAE on the validation set. This configuration allowed the model to capture the daily patterns which are also highlighted by the autocorrelation function of the signals.
To determine the standard operating conditions, we applied the unsupervised approach discussed in Section 3.2.2 by considering a MTBF of 75 days as reported in [433]. In this way, as confirmed by the event logs, the main failures and periods preceding outages were excluded from the standard behavior data which was, then, split into a training set $\mathbf{S_{train}}$ (70%) and validation set $\mathbf{S_{val}}$ (30%). The test set $\mathbf{S_{test}}$, instead, was defined by selecting periods including failure occurrences reported in the log files.
Based on the reconstruction errors, warnings were triggered by the model according to the four-stage threshold method presented in Section 3.2.2, allowing to consider

only significant residuals both in terms of magnitude and duration. In particular, the GMI threshold d_m was set to the 3rd quantile of the validation set distribution of Mahalanobis distances, and the LRI threshold d_i for the i-th signal to the 3rd quantile of its reconstruction error distribution.

Reconstruction Errors The MTGCAE model was compared in terms of reconstruction error for all SCADA signals with two other promising neural architectures recently applied for anomaly detection in wind turbines, namely LSTM-AE (introduced in [418]) and CNN-LSTM (introduced in [414]). The first is based on LSTMs and AEs, and the other is a regression model based on the combination of CNNs and LSTMs.
Table 3.11 shows the scores achieved by the three architectures, by evaluating the MAE, MSE, RMSE and the MDAPE.

Table 3.11. The table shows the reconstruction errors in terms of MAE, MSE, RMSE and MDAPE for the proposed MTGCAE model, comparing it with the LSTM-AE and CNN-LSTM architectures.

Model	MAE	MSE	RMSE	MDAPE
MTGCAE	0.038	0.004	0.061	0.058
LSTM-AE	0.053	0.007	0.085	0.08
CNN-LSTM	0.051	0.007	0.082	0.075

Even though MTGCAE performs better, the reconstruction errors of all three models are low, being the highest error 0.085, reached by the RMSE of the LSTM-AE model.
Since the training is performed on standard operating condition data, low errors are expected because all input windows are drawn from the same normal behavior distribution. High residuals are, instead, expected during anomalous conditions and failures, as confirmed by the next section.

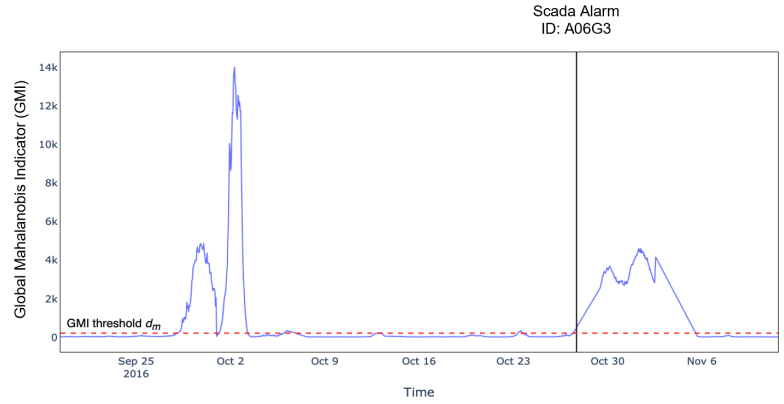
Deep Anomaly Detection Results As a first result of the proposed deep anomaly detection framework, Figure 3.29 shows the trend of the GMI and LRIs in the period that goes from September 26 to November 6, 2016, in the proximity of the IDA06G3 alarm. This alert notifies of damage on the T06 turbine generator detected by the SCADA system on October 27, 2016, at 16:26 (see Table 3.10 for details).
From Figure 3.29a, it can be seen that the GMI detects a possible anomaly from September 29 to October 3, 2016, identified by indicator values exceeding the global threshold d_m for at least 24 hours (the first two filtering steps of the four-stage threshold discussed in Section 3.2.2).
Then, looking at Figure 3.29b, we can observe high values of the LRIs in the same period as for the GMI, relative to the three phases of the generator stator windings (*Gen_Phase1_Temp, Gen_Phase2_Temp, Gen_Phase3_Temp*) and also to the temperature measured in the split ring chamber (*Gen_SplitRing_Temp*). Therefore, by applying all the filtering steps provided by the four-stage threshold, the model triggers a warning associated with the sensor having the highest LRI, namely the

Gen_Phase1_Temp. This anomaly, detected on the temperature of the first phase of the generator stator winding, anticipates by about 28 days the SCADA alarm related to the damage of the generator assembly.
It is interesting to notice that, when comparing the grid power request (Figure 3.30) and the total active power produced by the T06 turbine (Figure 3.30b), no significant mismatch is found in the proximity of the precursor (dashed box on the left), thus showing the ability of the model to capture hidden anomalies even when the turbine continues to deliver the power requested by the grid.
On the other hand, looking at the dashed box on the right after the SCADA alarm in Figure 3.30b, a long outage of about six days can be observed, needed to replace the damaged generator. This results in an anomaly in terms of expected power, which also generates significant residuals in the MTGCAE model indicators (see Figure 3.29).

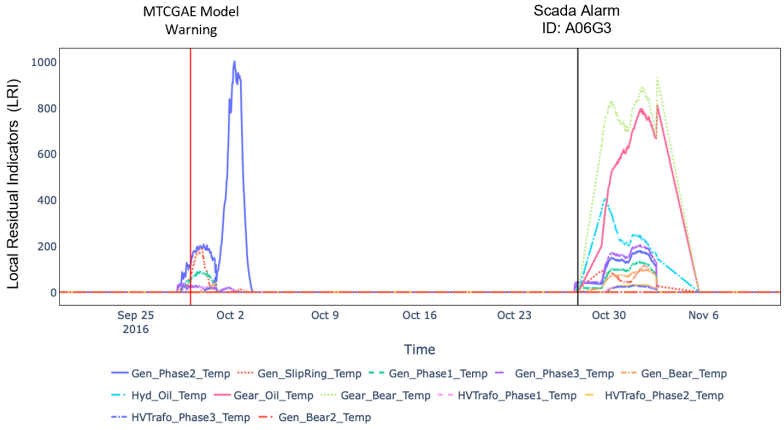
As a second result, Figures 3.31a and 3.31b show the evolution of the GMI and LRIs during a period of about three months (i.e., from May 22 to July 24, 2016), during which a high temperature anomaly was reported by the SCADA system on July 10, 2016 at 03:46 (alarm ID A07T1 in Table 3.10).
In this case, after the application of the four-stage threshold to the reconstruction errors of the MTGCAE model, an anomaly on the temperature of the T07 transformer windings (*HVTrafo_Phase1_Temp*, *HVTrafo_Phase2_Temp*, *HVTrafo_Phase3_Temp*) is isolated about 9 days before the SCADA alarm.
As for the previous example, also in this case the model is able to detect an operational anomaly even if the total active power of the T07 turbine matches the power required by the grid (see the first dashed box on the left in Figure 3.32).
About one week after the investigated event, anomalous behavior in the power delivered by the turbine T07 is observed for two days (dashed box on the right), during which significant residuals are produced by the deep anomaly detection framework at the assembly level of both the transformer and the generator (see Figure 3.31b).

As the last application, Figure 3.33 details the results in terms of the MTGCAE global and local indicators in the vicinity of a T09 generator bearing damage, reported by the SCADA system on October 17, 2016, at 09:19 and labeled as alarm ID A09GB3 in Table 3.10.
Looking at the GMI and LRIs shown, respectively, in Figure 3.33a and Figure 3.33b, the first warning is triggered about 18 days before the SCADA alarm and is associated with an anomaly isolated by the generator bearing temperature residuals (*Gen_Bear2_Temp*). Also during this anomalous period, turbine T09 seems to be unable to deliver the power required by the grid (see the first dashed box on the left in Figure 3.34).
After the SCADA alarm, Figure 3.34 shows a period of forced turbine downtime (the rightmost dashed box) required to replace the damaged generator bearings, during which high reconstruction errors are produced by the model.

Deep Anomaly Detection Assessment The best performance corresponds to the early detection of the greatest number of anomalies and faults recorded by the SCADA system that required repair or replacement at the assembly/sub-assembly level, with the minimum false alarms and the maximum time advance.



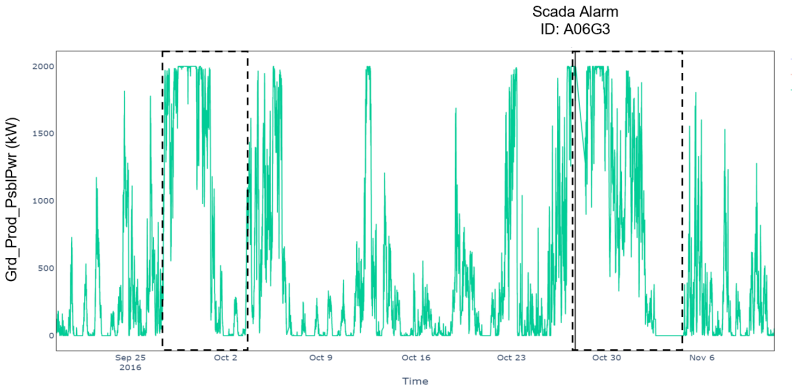
(a) GMI



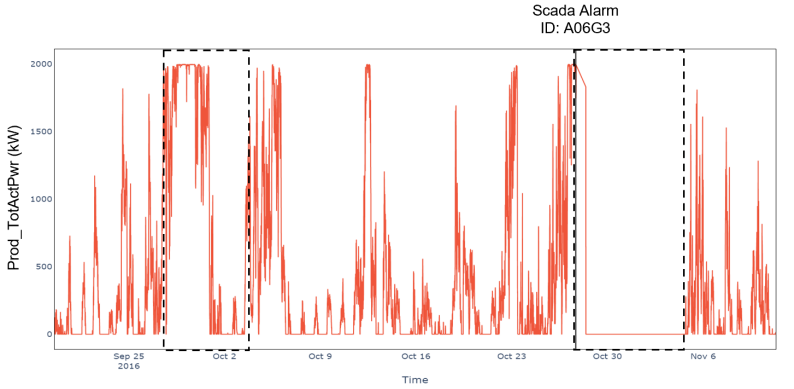
(b) LRI

Figure 3.29. 3.29a and 3.29b show, respectively, the trend of the Global Mahalanobis Indicator (GMI) and the Local Residual Indicators (LRIs) in correspondence of the alarm ID A06G3. This alarm reports a damage of the T06 generator, detected by SCADA on October 27, 2016 at 16:26. In particular, Figure 3.29a shows the trend of the global indicator and the threshold d_m applied in the first step of the four-stage filtering method. Figure 3.29b, on the other hand, reports the local residuals that satisfy all the conditions necessary to trigger an alarm based on the four-stage threshold. The vertical red line in the LRI plot represents the MTGCAE model warning associated with possible fault precursors.

In particular, we considered a discrete event evaluation of the performance where a True Positive (TP) corresponds to triggered model warnings associated with the assembly/sub-assembly that presents a SCADA alarm (with reference to the logs reported in Table 3.10) in the next time window consisting of $T_f = 4320$ samples (1 month). False Positives (FP), on the other hand, are model warnings that are not followed by a SCADA alarm associated with the involved assembly/sub-assembly. Finally, SCADA alarms not anticipated by a model warning within the reference time window are considered as False Negatives (FN).



(a) Grid Power Request



(b) Total Active Power

Figure 3.30. 3.30a and 3.30b show, respectively, the trend of the Grid Power Request and the Total Active Power of the T06 WT in correspondence of the alarm ID A06G3. The dashed boxes detail the periods in which anomalies were detected by the proposed deep anomaly detection framework (see Figure 3.29).

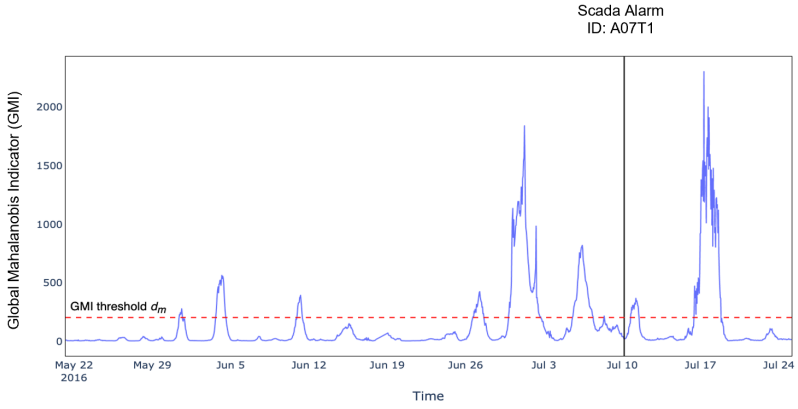
At this point, we quantified the performance of the model through classification metrics typically used in the field of Machine Learning:

$$\text{Precision } (P) = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.16}$$

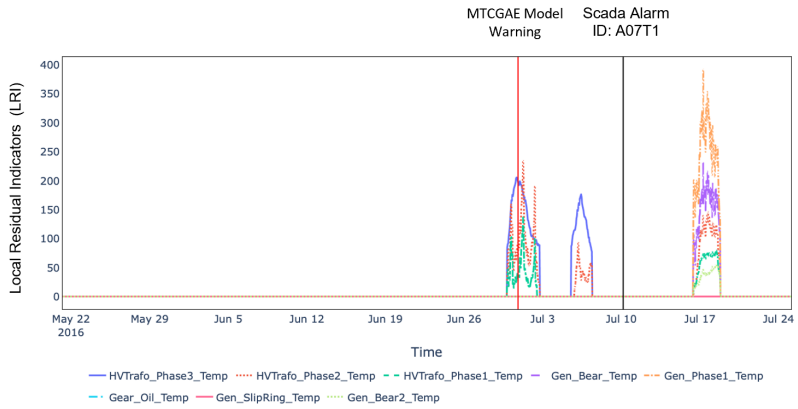
$$\text{Recall } (R) = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.17}$$

$$\text{F1-score } (F1) = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.18}$$

In addition, we also consider the average time advance (*Avg Advance*), which represents the time between the warnings triggered by the model and the reference SCADA alarm.
Based on these metrics, as for the model reconstruction errors discussed in Section 3.2.2, the deep anomaly detection capabilities of the proposed MTGCAE were



(a) GMI

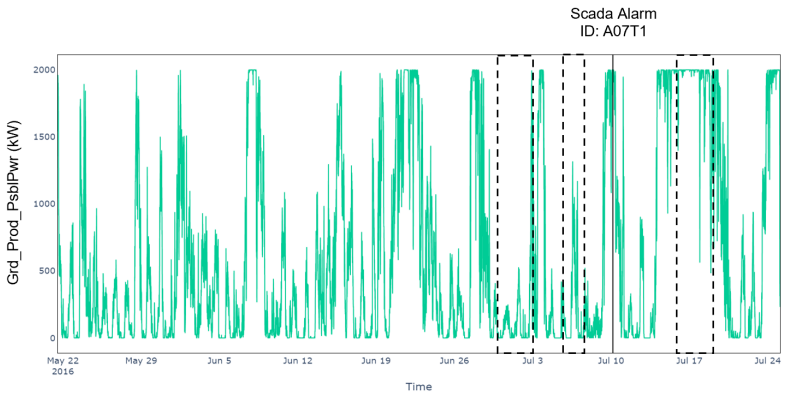


(b) LRI

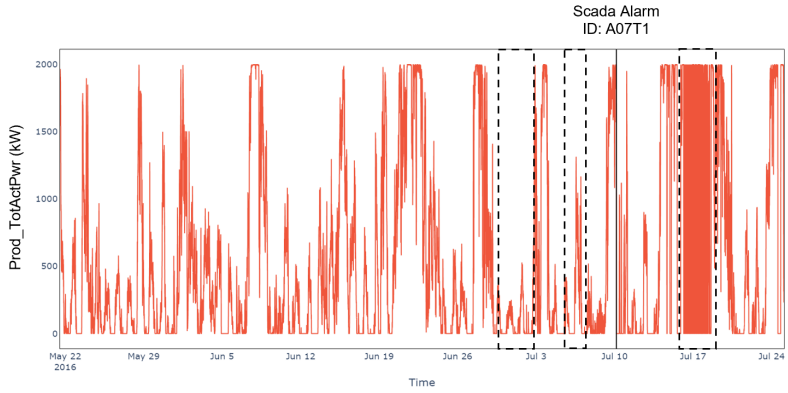
Figure 3.31. 3.31a and 3.31b show, respectively, the trend of the Global Mahalanobis Indicator (GMI) and the Local Residual Indicators (LRIs) in correspondence of the alarm ID A07T1. This alarm reports a high-temperature anomaly of the T07 transformer detected by the SCADA system on July 10, 2016, at 03:46. In particular, Figure 3.31a shows the trend of the global indicator and the threshold d_m applied in the first step of the four-stage filtering method. Figure 3.31b, on the other hand, reports the local residuals that satisfy all the conditions necessary to trigger an alarm based on the four-stage threshold. The vertical red line in the LRI plot represents the MTGCAE model warning associated with possible fault precursors.

validated against the LSTM-AE and the CNN-LSTM architectures. It is important to notice that the same framework for anomaly detection presented in Section 3.2.2 was applied when considering LSTM-AE and CNN-LSTM, to make their outputs comparable with MTGCAE.
Table 3.12 presents the evaluation metrics for the three models on the same test set.

Results show that MTGCAE can detect 10 out of 12 anomalous events without triggering any false alarm. LSTM-AE, while achieving the same TPs, counts 4 FPs, thus reducing the Precision and F1-score. Finally, the CNN-LSTM model only



(a) Grid Power Request



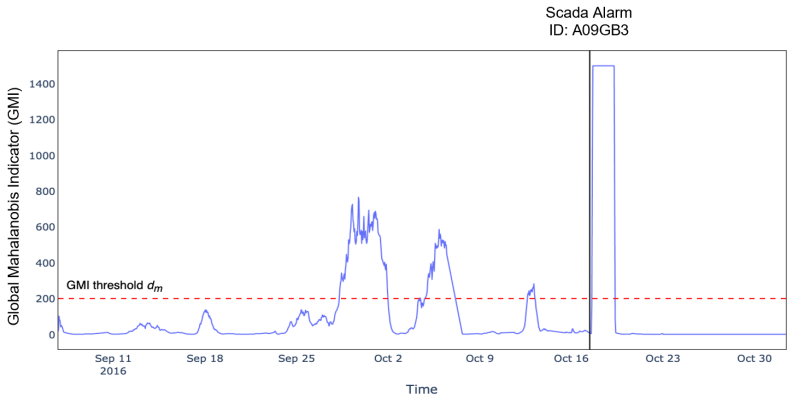
(b) Total Active Power

Figure 3.32. 3.32a and 3.32b show, respectively, the trend of the Grid Power Request and the Total Active Power of the T07 WT in correspondence of the alarm ID A07T1. The dashed boxes detail the periods in which anomalies were detected by the proposed deep anomaly detection framework (see Figure 3.31).

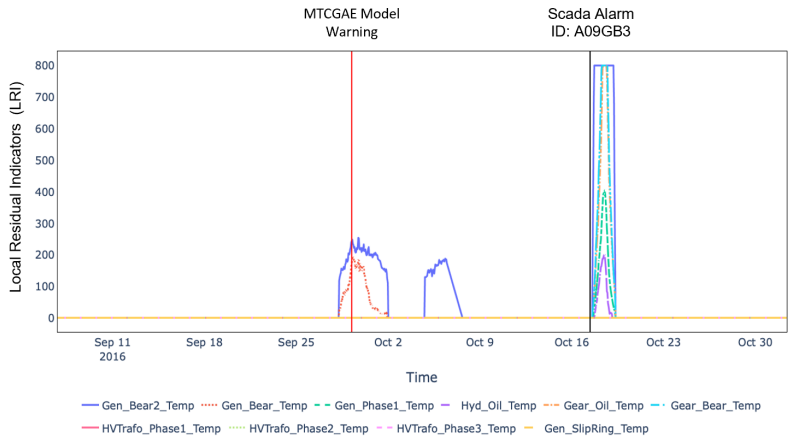
detects 9 events and presents 4 false alarms, producing lower scores also in terms of the Recall metric.
Even though MTGCAE achieves a lower average time advance (23 days) with respect to the other two models, the proposed approach seems to be more reliable given the lack of FPs, and robust provided the number of FNs.

Conclusions

In this paper, we present an original unsupervised deep anomaly detection framework in the context of horizontal axis WTs based on SCADA data. The core of the method is the proposed neural architecture, namely a Graph Convolutional Autoencoder for Multivariate Time series (MTGCAE), which models the sensor network as a dynamical functional graph. The main advantage with respect to standard AEs lies in the capability to simultaneously take into account the information content



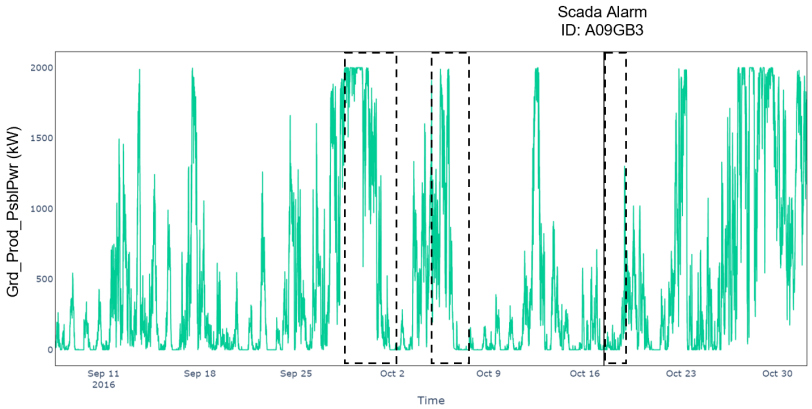
(a) GMI



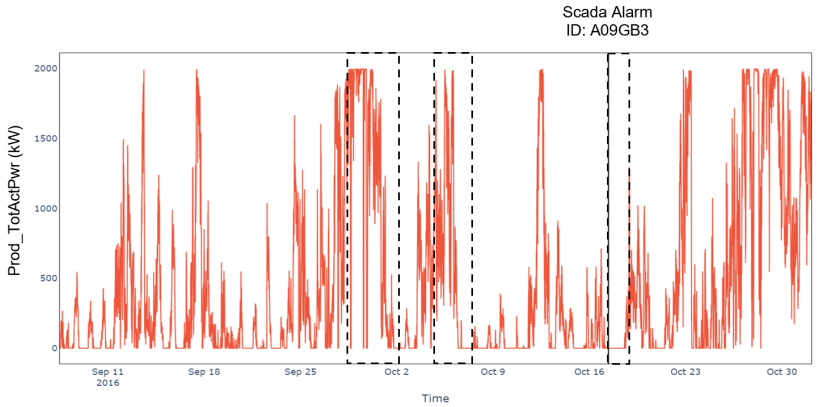
(b) LRI

Figure 3.33. 3.33a and 3.33b show, respectively, the trend of the Global Mahalanobis Indicator (GMI) and the Local Residual Indicators (LRIs) in correspondence of the alarm ID A09GB3. This alarm reports damage to the T09 generator bearings, detected by the SCADA system on October 17, 2016, at 09:19 (see Table 3.10). In particular, Figure 3.33a shows the trend of the global indicator and the threshold d_m applied in the first step of the four-stage filtering method. Figure 3.33b, on the other hand, reports the local residuals that satisfy all the conditions necessary to trigger an alarm based on the four-stage threshold. The vertical red line in the LRI plot represents the MTGCAE model warning associated with possible fault precursors.

of the individual sensors measurements (graph node features) and the nonlinear correlations existing between all pairs of sensors (graph edges).
The proposed neural architecture is trained to learn the normal behavior of the system without providing any kind of data labeling and, based on the model reconstruction errors, multiple monitoring indicators are defined, namely a GMI for the whole sensor network, and a LRI for each monitored variable. All indicators are evaluated by considering both their magnitude and duration in time by a four-stage threshold method. In this way, only significant model errors are taken into account, allowing



(a) Grid Power Request



(b) Total Active Power

Figure 3.34. 3.34a and 3.34b show, respectively, the trend of the Grid Power Request and the Total Active Power of the T09 WT in correspondence of the alarm ID A09GB3. The dashed boxes detail the periods in which anomalies were detected by the proposed deep anomaly detection framework (see Figure 3.33).

the attenuation of the effect of individual spikes and transient disturbances, thus reducing false alarms during normal operating conditions. After the four-stage threshold, a warning is triggered for the sensor having the highest reconstruction error, allowing the isolation of the assembly/sub-assembly mostly involved in the anomaly for troubleshooting purposes.

The proposed method was validated on 10-minute SCADA data collected from four WTs belonging to the same wind farm, with a rated power of 2 MW each. The dataset counts 12 failures on the most critical components (generator, gearbox, and transformer) that occurred during 20 months of operation. The model was trained on normal behavior data isolated using an unsupervised method and was tested on anomalous periods selected using the maintenance logs.

The presented model was compared with other two promising approaches, namely

Table 3.12. Results of MTGCAE, LSTM-AE and CNN-LSTM models.

Model	TP	FN	FP	Avg Advance (Time)	P	R	F1
MTGCAE	10	2	0	23 days, 0:05:42	1.0	0.83	0.91
LSTM-AE	10	2	4	27 days, 12:55:18	0.71	0.83	0.77
CNN-LSTM	9	3	4	28 days, 16:10:20	0.69	0.75	0.72

an architecture based on LSTMs and AEs (LSTM-AE) and one that combines CNNs and LSTMs (CNN-LSTM). To guarantee a reliable and robust analysis, we trained a separate model for each WT and considered all 12 anomalies for the final evaluation metrics.
The results show that the MTGCAE outperforms the other two neural architectures in terms of Precision (1.0), Recall (0.83) and F1-score (0.91). It is important to notice that the MTGCAE demonstrates the ability to capture hidden anomalies even when the turbine continues to deliver the power requested by the grid.
Since the model is unsupervised and completely data-driven, we expect it to be independent of the specific use case and potentially applicable to any WT equipped with a SCADA system.

3.3 Power Forecasting for Renewable Energy Systems

Renewable Energy (RE) is considered to be the most promising alternative to fossil fuels because of its low-carbon impact. However, RE sources bring unschedulable uncertainty due to their intermittent nature, which is an obstacle to the reliability and stability of energy grids, especially when dealing with a large-scale integration [434]. Nevertheless, the challenges coming with the penetration of RE can be alleviated if the natural sources or the power output of Renewable Energy System (RES) can be forecasted accurately [435, 436].
When reliable forecasts are available, power system operators can plan an optimal power management strategy to balance the dispatch between the intermittent power generations and the load demand while satisfying all system constraints at minimal operating costs [437, 438]. Without power forecasting, the system would suffer from over-commitment of generation and higher rates of curtailment, which would cause a rise in net generation costs. Modern dispatch decisions and scheduling strategies based on power forecasts play a significant role in Microgrid (MG) which act as local energy grids with control capability able to disconnect from the traditional grid and operate autonomously [439, 440].
When operating MGs, the uncertainties that come with REs can be compensated by performing an optimal sizing of the generation units and the Energy Storage Systems (Energy Storage Systems) as backup power supply at the time of generation deficiency [441, 442]. Moreover, since storage is the most costly element in MGs, estimating the size of the generation units and the ESSs is essential for an optimal and cost-efficient grid design. Even though the sizing of power systems is still an open problem for grid operators, research shows that better results are achieved when forecasts are included in the problem formulation [443, 444].

Finally, RE forecasting has a strong impact on energy markets and policies [445, 446]. In fact, optimal bidding strategies that incorporate power forecasts are adopted to improve trading practices and energy market policies that maximize the social welfare of the market participants [447, 448]. In particular, optimal bidding strategies can improve the competitiveness of renewable power generation and, therefore, accurate forecasts play an essential role to guarantee a rapid integration of REs in the energy market.
The next section presents a paper we submitted proposing a multi-modal spatio-temporal NN to forecast the output of multiple WTs composing a wind farm. Specifically, this work presents a neural architecture that takes in input several data sources, namely turbine-level SCADA data and Numerical Weather Prediction (NWP) maps covering a mesoscale geographical area surrounding the wind farm.

3.3.1 Multi-horizon Wind Power Forecasting Using Multi-Modal Spatio-Temporal Neural Networks

Introduction

Wind energy is promoted worldwide as a core energy source for the mitigation of climate change, allowing for the reduction of carbon emissions while having declining capital costs driven by Wind Turbine (WT) technological advancements [449, 450]. However, the rapid increase of wind power energy production on a global scale has created new challenges when it comes to grid integration, due to the non-stationarity, randomness, and intermittency of wind [451]. Accurate real-time forecasting algorithms can help to mitigate these problems and reduce the cost impacts of wind to a large extent by making wind power more schedulable. Hence, forecasting models have a significant economic and technical impact on the system by increasing wind power penetration and allowing efficient operation and maintenance, planning of unit commitment, and scheduling by system operators [131].
Wind power forecasting methods can generally be grouped into physical, data-driven, and hybrid methods, and a detailed classification is provided by [452]. This work focuses on data-driven methods, specifically approaches based on Machine Learning (ML), which have proven their ability to capture non-linear wind patterns in many recent works (see the paper of [133] for an overview). In particular, it is of interest to explore how the combination of different data sources, i.e. multi-modal data, can impact and improve predictions of data-driven algorithms. Since a single modality rarely provides complete knowledge of the phenomenon of interest, performing data fusion starting from multiple modalities can provide insights and benefits to forecasting models by contributing to a more unified picture and global view of the system [453].
Recently, multi-modal approaches are employed in a wide variety of applications concerning climate and energy forecasting. Boussix et al. [454] introduce a ML framework for tropical cyclone intensity and track forecasting, and show that, when combining historical storm data, reanalysis maps and historical operational forecasts, prediction errors comparable to current operational forecast models can be achieved while computing in seconds. Yang et al. [455] propose a multi-modal Deep Learning (DL) method for forecasting the daily power generation of small hydropower stations.

In this work, the authors combine daily power generation and precipitation data, together with the spatial distribution of precipitation observed by meteorological satellite remote sensing, and conclude that a multi-modal Neural Network (NN) can effectively improve the accuracy of forecasts. The work of Haputhanthri et al. [456] employs two Long Short-Term Memory (LSTM) networks that process a stream of sky images, time series of past solar irradiance readings, and cloud cover readings as inputs for irradiance nowcasting. Du et al. [457] present an ensemble ML-based method to forecast wind power production, which uses both the wind generation forecasted by a Numerical Weather Prediction (NWP) model and the meteorological observation data from weather stations. The experimental results show that the proposed ensemble method based on an Artificial Neural Network (ANN), Support Vector Regression (SVR), and Gaussian processes can improve the performance of 3-hour ahead wind forecasting with respect to NWP forecasts.
The present work casts the task of wind power forecasting in a multi-modal framework by considering the NWP data provided by the European Centre for Medium-Range Weather Forecasts (ECMWF) and the data collected from the Supervisory Control And Data Acquisition (SCADA) systems of the "La Haute Borne" wind farm, made publicly available by Engie [458]. Other papers have combined NWP and SCADA data but never considered NWP on a mesoscale level as input to the model together with turbine operational measurements. For example, Donadio et al. [459] proposed an ANN that takes in input meteorological variables (temperature, pressure, wind speed, and direction) interpolated at the WT locations and uses SCADA signals as turbine-level power outputs rather than predictors for the model. Another example is the work of Zheng et al. [460], where the authors use meteorological predictions obtained in the vicinity of the wind farm installation site to predict wind speed and, then, train a second model to map wind predictions to power outputs. Also in this case, the SCADA data is employed only as ground truth for the model.
This paper, instead, considers High-RESolution (HRES) NWP forecasts in the form of incrementally larger maps and SCADA time series as input for the predictive model to investigate how the area surrounding a wind farm and the turbine's internal operating conditions can impact the forecasts of the power output. NWP maps are in the form of regular square grids on a mesoscale level centered around the wind farm, and the choice of including a larger area and not only the meteorological data closest to the specific farm location was motivated by the presence of patterns that evolve both in time and space. Meteorological variables on different spatial scales, from full grids to cardinal point features, are not just downscaled to the farm location, but mapped directly to turbine-level power outputs by training a data-driven model able to capture wind patterns over larger areas. In this way, the model learns the downscaling transfer function and the power curves of the turbines altogether.
In detail, this paper proposes a spatio-temporal NN for wind power forecasting with a lead time of up to 90 hours. The model is composed of two sub-networks based on stacked Recurrent Neural Networks (RNNs) with LSTM cells that process, respectively, the SCADA and the HRES data. The outputs of the two modules are combined using a non-linear gating mechanism that regulates the flow of HRES information used for wind power forecasting based on the turbine's behavior.
The model is tested on four WTs with a rated power of 2050 kW, part of the "La

Haute Borne" wind farm, for a period that covers four years of operation.
The rest of the paper is organized as follows. Section 3.3.1 describes the different data sources and Section 3.3.1 presents the architecture of the proposed spatio-temporal NN, together with the performance metrics used to evaluate the model. Finally, Section 3.3.1 presents and discusses the wind power forecasting results, and Section 3.3.1 summarizes the present work and draws some conclusions.

Multi-modal data

As we aim to build a forecasting model that uses several data sources, this section introduces the multi-modal data considered to train and validate the model. Namely, the first source consists of turbine-level time series collected from SCADA systems, and information is provided about the wind farm size, location, and layout, together with turbine-specific details and the monitored variables. The second source consists of high-resolution NWP maps and information is provided about their size, resolution, centering position, and the variables considered for the analysis.

SCADA data The SCADA data considered is the open data of the "La Haute Borne" wind farm made available by Engie [458], located in the Meuse department of northeastern France. The wind farm is composed of four identical MM82 WTs produced by Senvion (identified as R80711, R80721, R80736, and R80790 in Figure 3.35), each having a rated power of 2050 kW. All turbines have a hub height of 80 m, a rotor diameter of 82 m, and an altitude of 411 m. The dataset includes SCADA signals collected from all four WTs sampled every 10 minutes. In addition to active power, other variables are monitored, such as wind speed, ambient temperature, or gearbox temperature.
This study considers 4 years of operation, ranging from January 2014 to December 2017. In particular, the first two years are selected as training set, the third as validation set, and the last as test set (2:1:1 ratio). In this way, each set contains a complete seasonal cycle.

NWP data The NWP data explored in this work is provided by the ECMWF. Specifically, this work considers the highest-resolution atmospheric model providing 10-day forecasts [461], which uses observations and prior information about the Earth system in the form of physical and dynamic representations of the atmosphere. This model produces 4 forecast runs per day (midnight, 6 am, noon, and 6 pm) with hourly steps to step 90 for all four runs, 3-hourly steps from step 93 to 144 and 6-hourly steps from step 150 to 240 for the midnight and noon runs. Forecasts are available on a regular grid for different variables like temperature, total precipitation, and wind speed.
In this paper, midnight runs are considered on a $0.25° \times 0.25°$ resolution grid up to step 90, thus having hourly high-resolution forecasts. In particular, a 17×17 map centered around the "La Haute Borne" wind farm (approximately a 308 km $\times$ 445 km patch as shown in Figure 3.36) is considered with the variables horizontal speed of air moving towards the east (U wind speed component) and the north (V wind speed component), at a height of 100 m above the surface of the Earth, together with wind gusts at 10 m height, at each grid point.

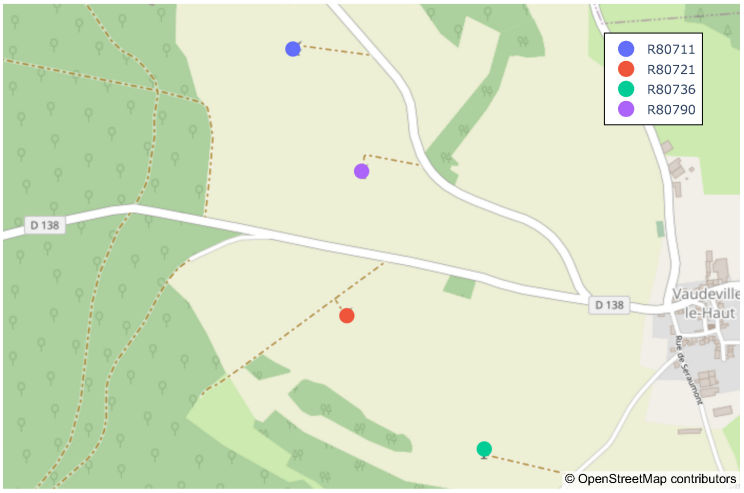


Figure 3.35. Layout of the "La Haute Borne" wind farm in France. The farm is composed by four identical MM92 WTs produced by Senvion identified as R80711, R80721, R80736, and R80790.

Data processing

Both data sources are preprocessed and augmented before the training of the proposed spatio-temporal NN.
The SCADA sensor signals are standardized, namely by removing the mean and scaling to unit variance. Then, outlier removal is carried out using the 6-sigma rule to filter out only extreme outliers caused by sensor measurement errors [462]. Since the proposed model integrates turbine-level knowledge into the forecasts, it is also of interest to include anomalous samples and conditions in the training set to capture possible machine inefficiencies and failures. Finally, SCADA signals are linearly interpolated to deal with missing values and resampled hourly to match the NWP prediction step.
Normalization is applied to the HRES variables, considering samples from the whole grid at all prediction steps. Since weather data has a clear daily periodicity, two extra synthetic signals are generated as input for the NN. In particular, they are sine and cosine transformations of time and read as

$$f_{sin}(s) = sin(s \cdot \frac{2\pi}{86400}),$$

$$f_{cos}(s) = cos(s \cdot \frac{2\pi}{86400}),$$

where s represents time measured in seconds and 86400 is the total number of seconds in a day. These two synthetic signals are added as extra features to each HRES grid point.
Since the dataset presents only a few short periods of turbine shutdowns compared to normal operating conditions, the training set is augmented to allow the model to capture these scenarios. In fact, for each training year, data is duplicated and the active power is set to zero. The monitored wind speed is left unaltered, enforcing a

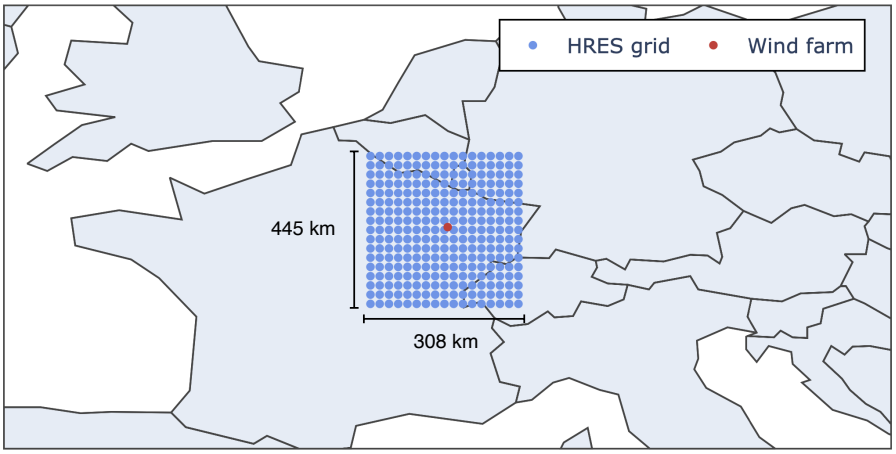


Figure 3.36. Map centered around the "La Haute Borne" wind farm in France. The blue dots represent the HRES 17×17 grid, while the red dot refers to the location of the wind farm.

condition in which turbines do not produce power even though the wind speed falls between the cut-in and cut-off range. The spatio-temporal NN is, then, trained on both the original and the augmented versions of the data.

Spatio-temporal neural network

The proposed multi-modal NN processes and combines the two different data sources previously discussed to produce wind power forecasts. The first includes variables monitored by the SCADA system in the form of time series and provides the model information about the internal behavior and performance of the WTs. The second source includes HRES forecasts in the form of regular square maps centered around the wind farm of interest, one for each lead time. In this way, the network can access the meteorological forecasts associated not only with the location of the wind farm but also with the surrounding area, thus capturing patterns that evolve both in time and space.

The network is composed of two sub-networks, referred to as SCADA sub-network and HRES sub-network, that process, respectively, the two data sources, and combine them using a gating mechanism, as shown in Figure 3.37. The building block of each sub-network is a RNN with a LSTM cell [463]. Even though the HRES maps are in the form of images, an LSTM-based neural architecture is preferred over standard Convolutional Neural Networks (CNNs) since the latter require longer training time and leads to worse results, probably due to the small map sizes. Even more complex architectures like Convolutional LSTM networks [464] perform worse on this type of data with respect to the proposed architecture.

The network is trained using the Backpropagation algorithm considering the L_1 loss function, the Adam optimizer, and early stopping to avoid overfitting. Next, the neural architecture is described in detail.

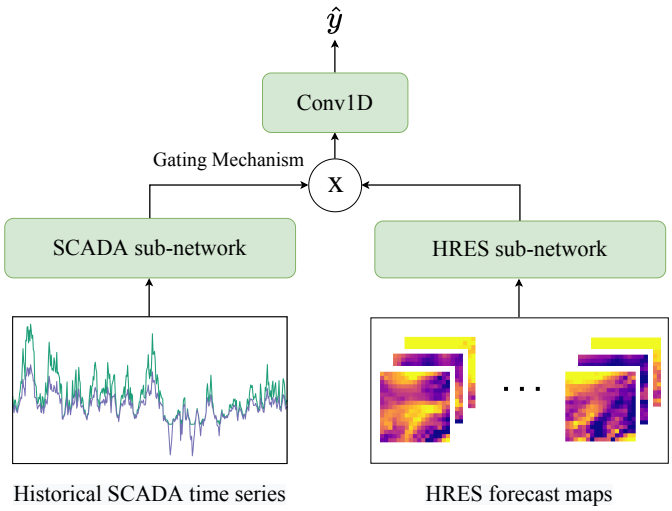


Figure 3.37. The complete architecture of the proposed spatio-temporal NN.

LSTM LSTMs use a memory cell and a gating mechanism that contains three non-linear gates, namely, an input gate i_t, an output gate o_t and a forget gate f_t. These gates allow regulating the flow of information into and out of the cell, to capture both short and long-term dependencies. LSTM cells are formulated as follows

$$f_t = \sigma(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f)$$

$$i_t = \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i)$$

$$o_t = \sigma(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o)$$

$$\tilde{c}_t = tanh(W_c \cdot x_t + U_c \cdot h_{t-1} + b_c)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$

$$h_t = o_t \circ tanh(c_t),$$

where x_t, h_t and c_t are the input, hidden state vector and cell state vector at time step t, respectively, σ and $tanh$ are the sigmoid and hyperbolic activation functions, and $\circ$ denotes the Hadamard product. W_f, U_f, b_f, W_i, U_i, b_i, W_o, U_o, b_o, W_c, U_c, b_c represent the trainable weights of the network. Moreover, residual connections are added as follows

$$\bar{h}_t = x_t + h_t, \tag{3.19}$$

where $\bar{h}_t$ represents the output of the LSTM cell at time step t, as shown also in Figure 3.38. In this way, gradients can flow through the network directly, without passing through non-linear activation functions, thus preventing gradients to explode or vanish. It is possible to stack multiple LSTM layers by feeding the output $\bar{h}_t$ of the previous layer as input x_t for the next.

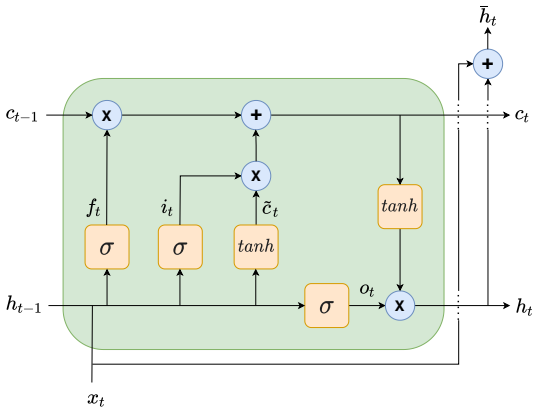


Figure 3.38. LSTM cell internal structure.

SCADA sub-network The SCADA sub-network is composed of an encoder and a decoder, as shown in Figure 3.39. The encoder takes $x = (x_1, \ldots, x_t, \ldots, x_T)$ as input, where x_t represents a multivariate sample of SCADA measurements at time step t, being T the length of the input window. The encoder has L stacked LSTM layers and produces $\bar{h}^{\text{enc}} = (\bar{h}^{\text{enc}}_{1,L}, \ldots, \bar{h}^{\text{enc}}_{t,L}, \ldots, \bar{h}^{\text{enc}}_{T,L})$ as output, together with a hidden state vector $h^{\text{enc}}_{T,l}$ and a cell state vector $c^{\text{enc}}_{T,l}$ for each layer l. The decoder has a symmetrical formulation with respect to the decoder. In fact, it is composed of L layers and the LSTM cell of layer l is initialized using the hidden state vector and the cell state vector produced by the encoder at layer $L - l$. The decoder takes $x^{\text{dec}} = (\bar{h}^{\text{enc}}_{T,L}, \ldots, \bar{h}^{\text{enc}}_{T,L}, \ldots, \bar{h}^{\text{enc}}_{T,L})$ as input, namely the output of the last encoder layer L at the last time step T repeated for each time step t, where t goes from 1 to the forecast horizon H. The decoder produces $\bar{h}^{\text{SCADA}} = (\bar{h}^{\text{dec}}_{1,L}, \ldots, \bar{h}^{\text{dec}}_{t,L}, \ldots, \bar{h}^{\text{dec}}_{H,L})$ as output, where $\bar{h}^{\text{dec}}_{t,L}$ is the output vector computed for time step t.

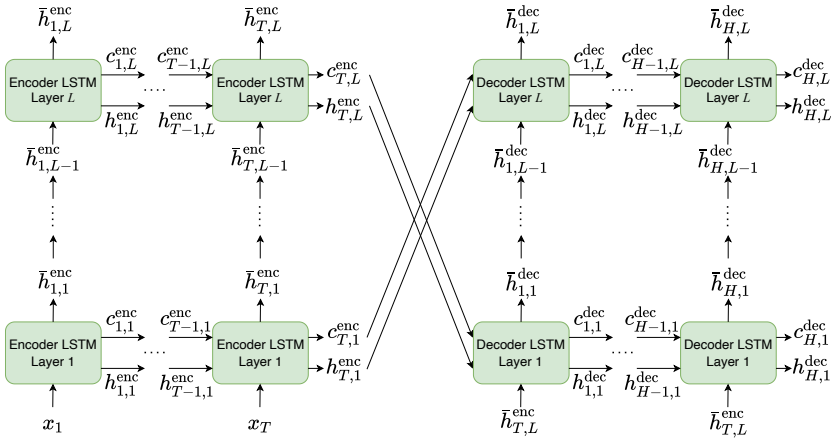


Figure 3.39. The neural architecture of the sub-network processing SCADA data, composed of an encoder (left) and a decoder (right).

HRES sub-network The HRES sub-network has no decoder component since the input sequence length matches the output length H. It takes in input $m =$

$(m_1, \ldots, m_t, \ldots, m_T)$, where m_t represents a HRES multivariate forecast map or a spatial sub-section of it at time step $t = 1, \ldots, H$. The network has L stacked LSTM layers and produces $\bar{h}^{\text{HRES}} = (\bar{h}_{1,L}, \ldots, \bar{h}_{t,L}, \ldots, \bar{h}_{H,L})$ as output, where $\bar{h}_{t,L}$ is the output vector computed for time step t.

Both sub-networks are bidirectional to allow the model to retain information from

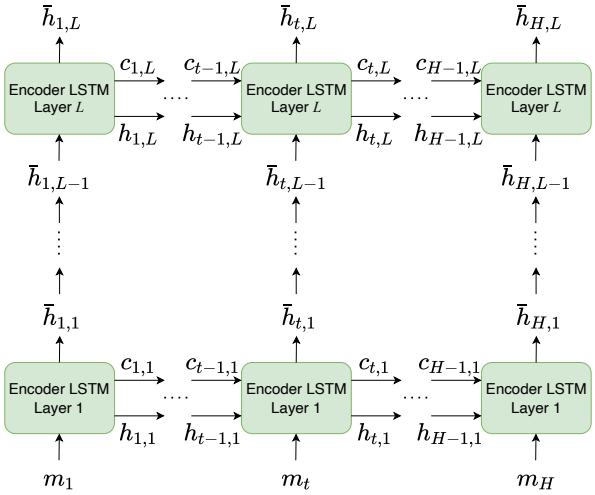


Figure 3.40. The neural architecture of the sub-network processing HRES data.

past (backward) and future (forward) time steps. In fact, the outputs of two separate networks, one processing the original input sequences and one the input sequences in reversed order, are concatenated.

Gating mechanism If the output tensors $\bar{h}^{\text{SCADA}}$ and $\bar{h}^{\text{HRES}}$ have the same dimension, they can be combined as

$$\bar{h} = \sigma(\bar{h}^{\text{SCADA}}) \circ \bar{h}^{\text{HRES}},$$

where σ is the sigmoid activation function and $\circ$ denotes the Hadamard product. In this way, the SCADA hidden vector acts as a non-linear gate, allowing the regulation of the flow of HRES information considered for wind power forecasting based on the turbine operating conditions. This gating mechanism is possible when the two sub-networks have the same number of neurons in the last layer.

Output Layer Finally, $\bar{h}$ passes through a 1D Convolutional layer and is mapped to the output dimension. Wind power forecasts are produced for each lead time up to step H, making the proposed NN effectively a multi-horizon model.

Spatial features analysis

To explore the spatial input features, different incremental map sizes are considered in the form of regular grids and then for each map size, a subset of features is

selected to train different models, thereby reducing the input's dimensionality and improving the learner's performance.
Considering different map sizes helps to evaluate and understand how the area surrounding the wind farm impacts wind power forecasting, and nine separate models can be trained using incrementally larger HRES forecast maps, with sizes from 1×1 to 17×17. Figure 3.41 shows an example of a map of size 9×9, where thus a square grid of 81 points is provided as spatial features to the NN. In addition

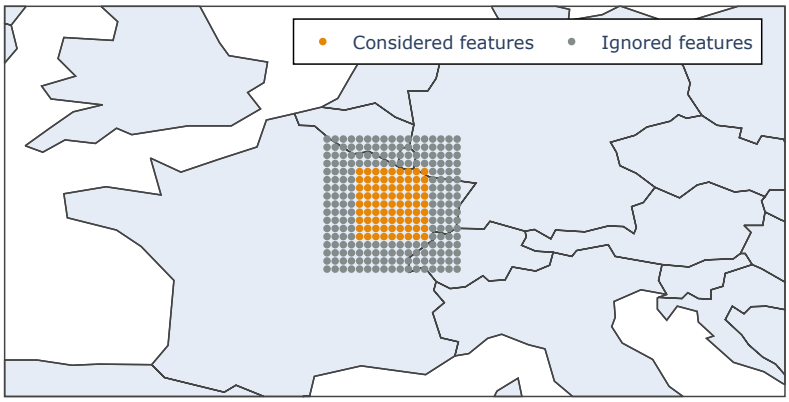


Figure 3.41. An example of a 9×9 square grid centered around the wind farm. Features belonging to the selected grid are considered input features for the model, while the others are ignored.

to changing the whole grid size, the number of features considered from the grid can also be reduced. For each incremental square grid, alternative models can be trained using a subset of features belonging to the perimeter of the forecast map or features relative to the main wind directions (one for each cardinal point), as shown in Figure 3.42.

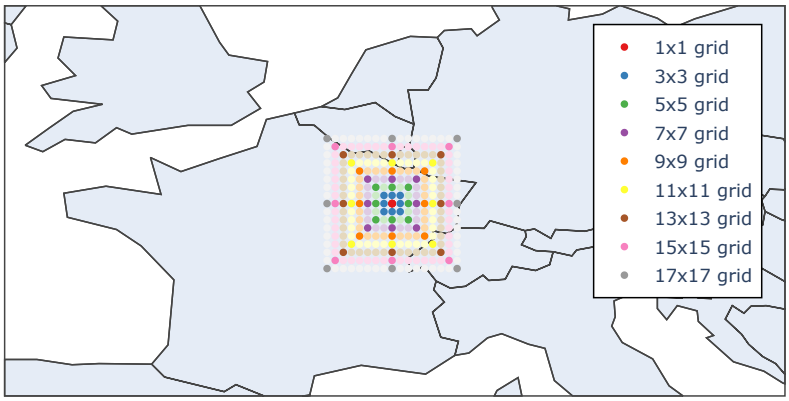


Figure 3.42. Perimeters of incrementally larger features maps (dark-colored and light-colored) and features associated with the main wind directions (dark-colored), one for each cardinal point.

Evaluation

This section presents the evaluation of the wind power forecasting model on the "La Haute Borne" wind farm. The proposed spatio-temporal NN was trained and evaluated on all four turbines, identified as R80711, R80721, R80736, and R80790, considering lead times up to 90 hours.
The model is evaluated with respect to both the spatial and operational input features. To evaluate the spatial input features, different map sizes and reduced maps are analyzed, while for the operational input features the SCADA data is considered. All evaluations are quantified using the same set of error metrics, which are therefore introduced in the following.

Evaluation metrics The performance of the proposed spatio-temporal NN for wind power forecasting is evaluated using different metrics. In particular, the Mean Absolute Error (MAE), the Root Mean Squared Error (RMSE), and the Median Absolute Error (MDAE) are considered. Each metric is computed for all lead times individually to provide a better insight into the model prediction errors and, then, normalized by the nominal power of the WTs which is 2050 kW. Considering n samples for each lead time t, the error measures can be described as

$$\mathrm{MAE}_t = \frac{1}{2050n} \sum_{i=1}^{n} |y_{i,t} - \hat{y}_{i,t}|,$$

$$\mathrm{RMSE}_t = \frac{1}{2050} \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_{i,t} - \hat{y}_{i,t})^2},$$

$$\mathrm{MDAE}_t = \frac{1}{2050} \mathrm{Median}(|y_{1,t} - \hat{y}_{1,t}|, \ldots, |y_{n,t} - \hat{y}_{n,t}|),$$

where $y_{i,t}$ is the ground truth wind power and $\hat{y}_{i,t}$ is the proposed model prediction for sample i at lead time t. Finally, to evaluate the overall model performance with respect to a baseline, a Skill Score (SS) is computed with

$$\mathrm{SS}_t = 1 - \frac{1}{n} \sum_{i=1}^{n} \frac{|y_{i,t} - \hat{y}_{i,t}|}{|y_{i,t} - \bar{y}_{i,t}|},$$

where $\bar{y}_{i,t}$ is the prediction of the Extreme Gradient Boosting (XGBoost) regressor proposed by [465], widely used in many applications involving regression tasks, including wind power forecasting as done by [466], [467] or [468]. This evaluation metric expresses the percentage of improvement achieved by the proposed model with respect to a baseline, which in this case is the XGBoost regressor. When computing overall performance metrics, each measure is averaged over all lead times.

Evaluation of spatial input features Having trained the model with different map sizes and different amounts of features as described in Section 3.3.1, the performance on the forecasting task can then be compared. The comparison between full maps, perimeters, and cardinal points is carried out to investigate if a subset of features covering all wind directions is sufficient to produce accurate forecasts.

This preliminary analysis is performed without considering the SCADA data and sub-network to focus exclusively on the spatial information included in the forecast maps.
The HRES sub-network used for this task counts 11 layers with 64 neurons each and the hyperparameters are selected using a grid search. The analysis is mainly presented for turbine R80721 which is representative of the whole wind farm, to avoid an excessive amount of similar results for the reader. Nevertheless, a general quantitative overview is provided at the end of the chapter for all turbines composing the wind farm of interest.
As can be seen in Figure 3.43, models trained on perimetric or cardinal point features have comparable or even better performances with respect to models trained on full grids. Notably, the 1×1 map size achieves the same MAE in all three scenarios since the input, namely the central pixel, is the same for all models. The lowest overall error is achieved by using perimetric or cardinal point features (which coincide also in this case) with 3×3 grids. Considering full maps is not only slower from a computational point of view, but also degrades the model performances for most map sizes, except for the 5×5 and 7×7 maps. Training a one-layer network using 17×17 full grids takes 1:03 minutes, compared to 37 seconds achieved by using cardinal point features (speedup of 1.7x).
It is interesting to notice that, when using cardinal point features, the lowest MAE (0.063) achieved using 3×3 grids is only 0.004 lower than the highest MAE (0.067), obtained using the largest map size of 17×17. This means that 8 features, one for each wind direction, even when spatially distant from the wind farm, are sufficient to achieve relatively accurate forecasts, thanks to the bidirectional temporal capabilities of the model. In fact, by looking at HRES meteorological forecasts associated with the cardinal points from the past and future of each lead time, the network is supposedly able to model the dynamics of wind and its evolution in time over the entire geographical area. Figure 3.44 presents the same results in terms of the SS

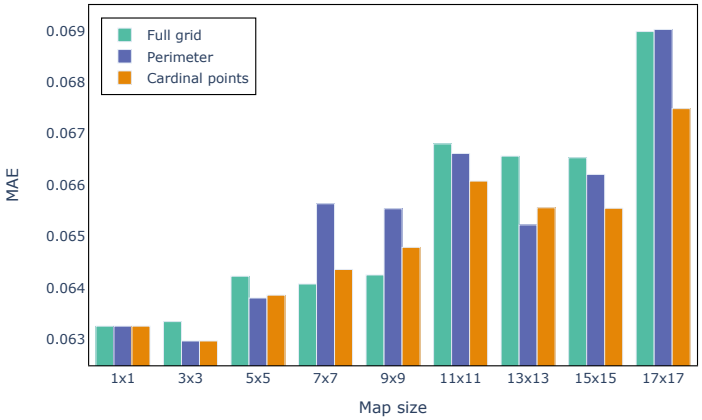


Figure 3.43. The bar chart shows the MAE averaged over lead times achieved by training separate networks, one for each map size, from 1×1 to 17×17. In particular, green (left) bars refer to the results obtained by training the networks on full feature maps, purple (central) on perimetric features, and orange (right) on cardinal point features.

evaluation metric. As a baseline for the score, an XGBoost regression was trained for each lead time selecting as input the whole HRES forecast map. The best results were also achieved by using the 3×3 grid with a SS of 25%, and both perimetric and cardinal point features perform on average better than full maps.

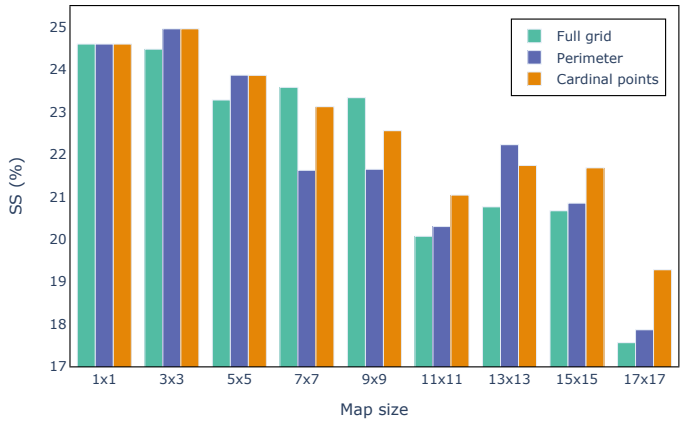


Figure 3.44. The bar chart shows the SS achieved by training separate networks, one for each map size, from 1×1 to 17×17. In particular, green (left) bars refer to the results obtained by training the networks on full feature maps, purple (central) on perimetric features, and orange (right) on cardinal point features.

Evaluation of operational input features After selecting the optimal grid configuration, namely perimetric/cardinal point features of 3×3 maps, the SCADA data and sub-network were integrated into the training process to produce the final wind power forecasts. This sub-network counts 11 layers with 64 neurons each and considers a window of 24 hours of observed historical measurements to capture the last daily cycle of the turbine's operating conditions. Also in this case, the hyperparameters are selected using a grid search. The whole model takes approximately 30 minutes to train on CPU for 60 epochs using an Apple M1 using the augmented dataset described in Section 3.3.1 and a batch size of 64. Inference, once the network is trained, takes only a few seconds, making it suitable for real-time forecasting.

When integrating the SCADA data and sub-network, the model performance for turbine R80721 increased by only 0.005 in terms of SS, since no major failures or inefficiencies occurred during the four years of operation considered for this study. However, it is interesting to notice that without the SCADA component, the network always predicts a non-zero active power based on the wind speed, even when the turbine is shut down and outputs no power. When the operating conditions of the turbine are considered using the SCADA sub-network, instead, the model can successfully predict a zero active power. An example of such a scenario is provided in Figure 3.45.

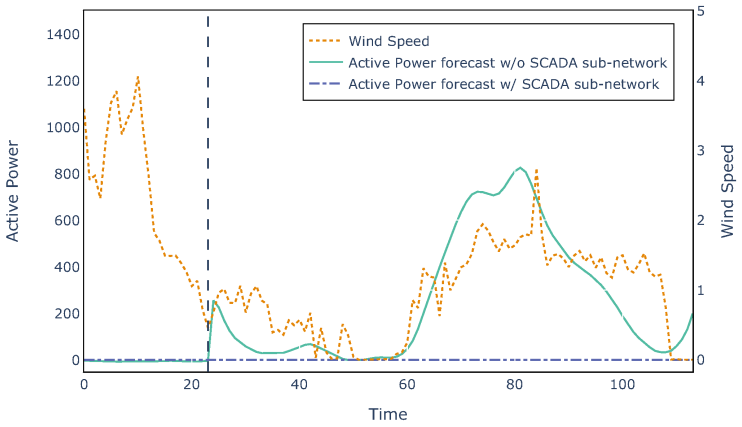


Figure 3.45. Wind power forecast produced by the spatio-temporal NN for turbine R80721 in a downtime scenario, namely when the turbine is shut down and produces no power. The plot shows both the forecast produced by the model trained without considering the turbine's operational conditions and the output of the NN when trained using the SCADA data and sub-network. The x-axis ranges from 0 to 114 and includes the observed history of measurements (0-23) and the 90 forecast lead times (24-113), separated by a vertical dashed line.

Comparison to benchmarks In order to compare the performance of the spatio-temporal NN with benchmark algorithms, the MAE for all lead times was computed for the proposed model, linear regression, the XGBoost regression, and another promising neural architecture, namely a Convolutional LSTM (ConvLSTM) network [469]. For both benchmarks, a separate model was trained for each lead time providing as input the whole 17×17 forecast map. As shown in Figure 3.46, the network outperforms the other two regression models overall, achieving an error of 0.044. It is interesting to notice that the MAE for the last two lead times (89 and 90) of the spatio-temporal network increases and becomes comparable to the other models. This is probably because the bidirectional neural architecture has no access to information from the future and, therefore, produces worse results. Moreover, forecasting wind power with lead times close to 90 hours in the future is a challenging task by itself, which is confirmed by the fact that the temporal resolution of NWP forecasts decreases. Notably, even though the HRES maps are in the form of images, the spatio-temporal neural network outperforms a more complex architecture based on Convolutional LSTMs both in terms of time and prediction error. In fact, the proposed network requires 30 minutes to train for 60 epochs, while the ConvLSTM takes 300 minutes, achieving a speedup of 10x. In addition, the lead time errors of the ConvLSTM are always higher than the proposed network, probably due to the excessive complexity of the model when dealing with small map sizes.
Looking at Figure 3.46, it can be seen that the errors follow a daily pattern, with low MAEs during morning lead times and high MAEs during evenings. For example, during the first daily cycle (lead times from 1 to 24), the minimum MAE (0.044) corresponds to 4 AM forecasts and the maximum (0.067) to 9 PM predictions, with a difference of 0.023 during 24 hours. This result is in line with the strong diurnal

cycle of wind forecasts resulting in usually lower predictability from the afternoon up to midnight due to the presence of atmospheric convection [470, 471]. Even though there is a daily cycle, the lead time errors follow an overall increasing trend correlated to the temporal distance from the midnight forecast origin.

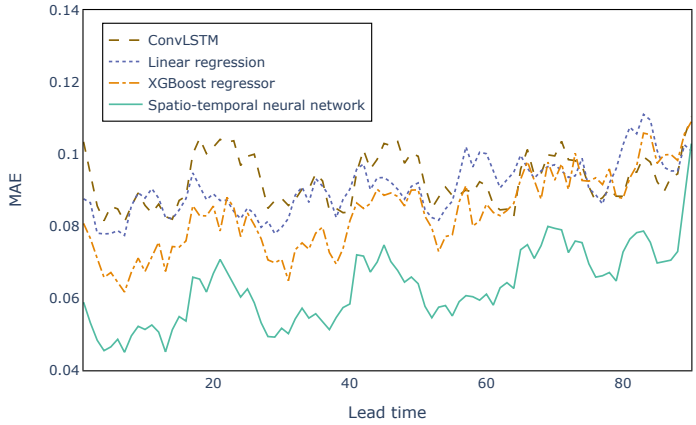


Figure 3.46. Comparison between the proposed spatio-temporal neural network, linear regression, the XGBoost regressor, and the ConvLSTM network. In particular, MAEs are plotted for each lead time, from 1 to 90.

An example of wind power forecasts produced by the proposed spatio-temporal NN is presented in Figure 3.47. The model is not able to predict the fluctuations of higher frequency which are hard to capture, especially when the meteorological data is not specific to the wind farm location, but rather to a regular grid surrounding it. Nevertheless, the network can predict the general trend of the active power produced by the WT up to almost 4 days in the future.

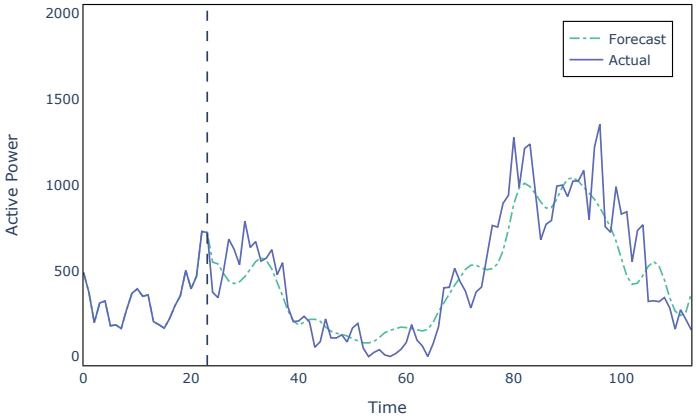


Figure 3.47. Example of wind power forecasts produced by the spatio-temporal NN for turbine R80721. The x-axis ranges from 0 to 114 and includes the observed history of measurements (0-23) and the 90 forecast lead times (24-113), separated by a vertical dashed line in correspondence with the forecast origin.

Finally, to provide a general quantitative analysis for the whole wind farm, Table 3.13 presents the evaluation metrics computed for all four WTs. The scores are comparable among turbines and achieve an average SS of 0.251, meaning that the proposed spatio-temporal NN performs 25.1% better on the whole wind farm with respect to an XGBoost regressor trained on the same data.

Table 3.13. The table presents the prediction errors in terms of MAE, RMSE, and MDAE averaged over lead times, together with the overall SS, for all four WTs.

Turbine ID	MAE	RMSE	MDAE	SS
R80711	0.078	0.121	0.045	0.236
R80721	0.063	0.103	0.035	0.255
R80736	0.066	0.109	0.036	0.265
R80790	0.072	0.118	0.040	0.248

Conclusions

This paper proposed a multi-modal spatio-temporal NN for multi-horizon wind power forecasting considering a lead time of up to 90 hours. In particular, the model combined high-resolution NWP forecast maps with turbine-level SCADA data, and explored how meteorological variables on different spatial scales together with the turbines' internal operating conditions impact wind power forecasts.
The spatial analysis of HRES maps showed that a subset of features associated with all wind directions can produce more accurate forecasts with respect to full grids and reduce computation times. In this way, when no regular grid data is available in the immediate surrounding of the wind farm, it is still possible to forecast wind power by considering features corresponding to the main wind directions, even if spatially distant.
The spatio-temporal NN was compared to linear regression and XGBoost and outperformed both on all lead times. More specifically, the proposed model improved the XGBoost baseline with an average skill score of 25.1%.
For future work, it is of interest to extend the analysis to other wind farms where more anomalies, failures and downtimes occurred and are reported in a maintenance log. In this way, the network can model these scenarios and improve the wind power forecasts based on the turbine's operating conditions. Moreover, it would be of interest to train the proposed NN on other wind farms having similar wind conditions to understand if the same subset of NWP maps, namely 3×3 cardinal point features, leads to the best results, or if it depends on the topology of the area surrounding the wind farm.

Acknowledgements

The research detailed in the current paper was based on data from the ECMWF obtained through an academic license for research purposes. This work was supported by the German Research Foundation (DFG) under Germany's Excellence Strategy - EXC number 2064/1 - Project number 390727645.

Chapter 4

Conclusions

Climate action is not an option anymore as constantly reminded by the IPCC reports on climate change. We are witnessing rising sea levels, heat waves, events of flooding, droughts and desertification resulting in the loss of lives and damage to livelihoods, especially in countries of the Global South. Therefore, it is of uttermost importance that the commitments laid out in Paris in 2015 are respected by all parties to guarantee that humankind as a whole can enjoy peace and prosperity on this planet. The cumulative efforts of conferences held in Rio, Stockholm, Kyoto, Johannesburg and Paris identified sustainable energy development as a key requirement for sustainable development, linking the energy sector directly to the environmental, economical, and social dimensions of society. Nowadays, the energy sector is responsible for more than two-thirds of global GreenHouse Gas (GHG) emissions and decarbonization of the global energy systems has become one of the greatest and most important challenges in the 21st Century requiring international cooperation and agreements complemented by regional policies and investments.

To mitigate climate change and protect the planet from irreversible consequences, it is evident that part of the solution is to reduce emissions coming from the energy sector. To this end, large-scale penetration of Renewable Energy Systems (RESs) into the energy market plays a key role in the energy transition toward a sustainable future by replacing fossil fuels. Renewable Energy (RE) sources not only contribute to the mitigation of GHG emissions, climate change, and environmental pollution but also improve access to energy since locally available, with socio-economic benefits. The technical feasibility of 100% RESs has been extensively scrutinized by the literature and many countries and cities have already declared their commitment towards net zero GHG emissions or 100% RE for all energy consumption by 2050. Moreover, with the increasing presence of distributed RE generation systems, centralized paradigms of power transmission and distribution are no longer valid, creating new opportunities for less hierarchical and more flexible energy and infrastructure management. This is leading to decentralized small-scale energy systems that can offer a resilient answer to the global energy crisis characterizing the current historical period, which is witnessing the COVID-19 pandemic emergency and the war in Ukraine.

With the advent of Industry 4.0, IoT technologies have been increasingly applied to the energy sector introducing the concept of smart grid or, more in general, Internet

of Energy (IoE), where an intelligent electricity distribution infrastructure that uses two-way flows of electricity and information is used to create a widely distributed automated energy delivery network. The rapid digitalization of the energy sector and the interconnection of RESs are contributing to the flexibility and resilience of the next-generation power grids. In this way, smart grids and IoE are steering the industry towards more efficient, reliable, safe, and sustainable solutions with huge environmental and social potential benefits. These are not only technical challenges and, therefore, require a collective and political effort to transform the current energy market into a sustainable alternative by implementing substantial changes to electricity infrastructure, business models, and regulations.

To realize the concept of smart grids and IoE, new information technologies are required and among the most promising possibilities is Artificial Intelligence (AI) or Machine Learning (ML) which in many countries has already revolutionized the energy industry by incorporating learning algorithms to perform different types of tasks, such as controlling, forecasting, and operating energy systems. The potential and capabilities of AI have become essential to allow RESs to penetrate the energy market by mitigating the variability of RE sources, by providing better Operation and Maintenance (O&M) decisions, monitoring the power infrastructure, increasing the security of system operations, and new market designs.

This thesis presented different ML algorithms and methods for the implementation of new strategies to make RESs more efficient and reliable. It described and analyzed various learning algorithms, highlighted their advantages and limits, and evaluated their application for different tasks. In addition, different techniques were presented for the preprocessing and cleaning of time series, nowadays collected by sensor networks mounted on every RES.

With the possibility to install large numbers of sensors that collect vast amounts of time series, it is vital to detect and remove irrelevant, redundant, or noisy features, and alleviate the curse of dimensionality, thus improving the interpretability of predictive models, speeding up their learning process, and enhancing their generalization properties. Therefore, this thesis discussed the importance of dimensionality reduction in sensor networks mounted on RESs and presents an unsupervised algorithm based on time series clustering to group similar signals, opening the possibility to reduce their dimensionality by selecting representative variables for each identified cluster. Then, it proposed a novel algorithm for unsupervised feature selection that leverages the predictive power of combinations of variables to assess their overall importance.

Data-driven approaches for dimensionality reduction can alleviate data storage, processing, and transmission issues, which are critical in energy monitoring and control applications. However, these techniques face several challenges when it comes to real-time applications. When a large volume of data streams is collected continuously over time, the computational complexity of dimensionality reduction algorithms can become a time-consuming bottleneck for the whole data pipeline, especially when predictive algorithms leverage the incoming data to train models, leading to delays and potential data loss.

Selecting the most relevant features of variables from the sensor data can be a good strategy because it is a one-time procedure performed during a preliminary offline data analysis phase. In contrast, reducing the number of real-time measurements

recorded from the sensors is a continuous procedure that involves reducing the frequency of the data. This strategy can help reduce the number of samples that need to be processed and transmitted but can also lead to loss of information, especially for sensors where a high temporal resolution, like vibrational sensors, is critical. The chosen strategy or combination of approaches highly depends on the specific application and the available computational resources. When the computational resources are limited, feature selection can greatly reduce the number of processed signals. Reducing the number of incoming real-time sensor measurements is also a viable strategy but leads inevitably to a trade-off between data reduction and information loss.

Over the past years, edge computing has become a promising solution for real-time applications based on sensor networks by allowing dimensionality reduction algorithms to operate closer to the data streams' source [472]. This computing strategy based on edge devices mounted close to the sensors reduces the amount of transmitted and stored data, leading to lower network traffic and decreased latency. In this way, edge computing is particularly beneficial in energy system applications where timely decisions and actions are critical. Moreover, this computing paradigm can enhance the security and privacy of the sensor data by reducing the exposure of the actual monitored data to external networks [473].

However, implementing an edge computing solution for dimensionality reduction faces several challenges, including edge devices' limited computational and storage resources. Additionally, edge devices may have limited power sources, which can constrain their processing capabilities. To address these challenges, researchers are developing specialized hardware and software to deploy distributed dimensionality reduction techniques that can reduce the processing load on individual devices by enabling collaboration between edge devices [474]. Edge computing offers a promising approach for real-time dimensionality reduction in energy system sensor networks by enhancing sensor data processing efficiency, security, and privacy while reducing network traffic and latency. However, it still requires a complete paradigm shift in the context of power grids and energy production and the installation of edge devices on the monitored energy systems.

Among the data collected by sensor networks, there is information about processes, events and alarms, that provide valuable knowledge and insight into the underlying dynamical systems. Therefore, data-driven methods open the possibility to support strategic decision-making, resulting in a reduction of O&M costs, machine faults, repair stops, and spare parts inventory size. These procedures are vital for industries, due to the growth in complexity of the interactions between different systems. This thesis presents two approaches in the context of predictive maintenance to improve the lifetime and efficiency of the equipment. The first work proposes an algorithm for anomaly detection in a production factory of photovoltaic cells, while the second paper presents an original unsupervised deep anomaly detection framework based on Graph Convolutional Networks and Autoencoders to isolate and anticipate failures in Wind Turbines (WTs).

As discussed in these works, data-driven predictive maintenance solutions can support informed decision-making, reduce maintenance costs and benefit the overall life span of assets. Still, several limitations arise when working with RESs in the real world.

One major challenge involves the need for historical data containing maintenance events. Even though many of the modern machines are equipped with SCADA systems that monitor and collect standardized measurements relative to different parameters, it is not always the case that maintenance events are recorded or available. Many RESs are still relatively new, with limited long-term performance and maintenance interventions. In addition, when available, maintenance data is often stored in different systems and formats, including handwritten notes by technicians, making it difficult to integrate and analyze effectively. These records are usually not standardized when provided by the operators and lack consistency in format, content, or level of detail, leading to several challenges. For example, it can be difficult to identify the root cause of a problem if the maintenance records do not provide enough detail or to evaluate how well a predictive maintenance algorithm performs over time.

Moreover, maintenance records do not always reflect the actual maintenance events on the assets. Monitoring systems or technicians may not accurately record a maintenance event's occurrence time and duration or could report only high-level or undetailed information related to the failures. This can lead to inconsistencies and inaccuracies in the maintenance records, making it challenging to analyze the data and make informed decisions about maintenance and operation. Standardizing maintenance record-keeping practices and improving data collection processes could address some of these challenges. Therefore, it is necessary to provide standardized training for technicians on recording maintenance events accurately and consistently using electronic systems that can capture real-time data and integrate it with other systems. Another possibility is to leverage Data Mining and Natural Language Processing techniques to extract patterns from unstructured maintenance data and produce standardized maintenance logs.

Even though the standardized format of data provided by the SCADA systems is highly desirable, as discussed previously, these systems usually record samples at 10-minute intervals. While this is sufficient for many applications, it may not be adequate to detect performance issues occurring on a faster time scale. For example, detecting patterns of a vibrating WT blade can be challenging when processing 10-minute data and delay identifying and addressing the problem, thus causing increased maintenance costs and decreased system performance.

Installing vibrational sensors can help monitor critical components' conditions in real time. These sensors measure the amplitude, frequency, and other characteristics of vibrations in the system, providing detailed information about the state of individual subsystems. When combining vibrational data with other SCADA measurements, it is possible to identify potential problems using data-driven models and take corrective action before the machine fails. Unfortunately, installing vibrational sensors is only sometimes possible or practical since installing and maintaining them can be expensive, particularly in large-scale systems where many sensors may be required. Moreover, installing and maintaining the sensors is not always possible when assets are located in difficult or unsafe areas. For example, WT blades can be more than 100 meters above the ground, making it challenging for technicians to have access. Moreover, the amount of vibrational data can quickly become unmanageable when collected from multiple sensors at high frequencies. Storing the data becomes challenging, especially when stored for long periods and used for future analysis and

reference. Cloud-based storage solutions can provide a scalable and cost-effective way to store and manage vibrational data. Another challenge is related to the real-time processing of vibrational data. As discussed for dimensionality reduction techniques, edge computing is a viable solution to address these limitations.
In some cases, it may be possible to use alternative technologies to monitor the condition of components. For example, acoustic sensors can detect changes in sound patterns that may indicate a problem with an element, such as a gearbox or generator. Infrared thermography can also detect temperature changes that may indicate a problem with electrical connections or other components. However, it is essential to note that these alternative technologies may not be as effective as vibrational sensors in detecting specific performance issues. In some cases, it may be necessary to use a combination of different technologies to understand system performance comprehensively.
Another difficulty in data-driven predictive maintenance resides in the complexity of RESs. These systems usually consist of multiple components and subsystems with complex interactions, making it challenging to identify the root cause of an anomaly. This complexity can make it hard to develop reliable predictive models capable of capturing highly non-linear phenomenons, especially when the monitored variables do not fully describe the state of the asset.
When it comes to REs, the unschedulable uncertainty due to their intermittent nature represents an obstacle to the reliability and stability of energy grids, especially when dealing with large-scale integration. Nevertheless, these challenges can be alleviated if the natural sources or the power output of RES can be forecasted accurately, allowing power system operators to plan optimal power management strategies to balance the dispatch between the intermittent power generations and the load demand while satisfying all system constraints at minimal operating costs. Moreover, accurate forecasts can contribute to estimating the size of the generation units and Energy Storage Systems for an optimal and cost-efficient grid design with a backup power supply at the time of generation deficiency. RE forecasting has a strong impact on energy markets and policies and contributes to the competitiveness of renewable power generation and penetration in the energy market. This thesis proposes a multi-modal spatio-temporal Neural Network to forecast the output of a wind farm. Specifically, it presents a neural architecture that takes in input several data sources, namely turbine-level SCADA data and NWP maps covering a mesoscale geographical area surrounding the wind farm.
Multi-modal approaches are effective in many fields, including RES, because combining multiple data types can improve predictions and reduce the model's uncertainty. Relying solely on one data type is often insufficient and can lead to poor power forecasts. For example, using only meteorological data to forecast energy production can be insufficient since the model needs to account for other factors impacting energy output, such as changes in energy demand or equipment failures. Furthermore, multi-modal approaches can also help to identify potential correlations and patterns between the different types of data that would not be evident when considering only one type of data. For example, by combining energy demand data with geographic data, multi-modal approaches could identify regions where energy demand is highest and optimize energy production accordingly.
To train a multi-model power forecasting model, a list of potential data sources is:

- Sensor data: This includes data measured from sensor networks mounted on board of RESs such as wind speed, active power, generator RPM, or bearing temperatures. By monitoring these parameters, multi-modal approaches can make more accurate forecasts and identify potential issues such as equipment failures or maintenance needs.

- Meteorological data: Weather conditions play a significant role in determining the power output of RES since they are usually directly influenced by them. For example, information about wind speed in the surrounding area, surface temperature, or solar radiation can help the model better forecast the power output of the energy system.

- Energy production data: This includes historical data on the power output of RESs such as solar panels, WTs, or hydroelectric plants. By including this data, multi-modal models can identify patterns and trends in energy production and leverage them to generate future estimates.

- Energy demand data: This data includes energy consumption patterns, such as hourly, daily, or seasonal fluctuations in energy demand. By including this data in multi-modal approaches, energy grid operators can better anticipate future energy needs and balance energy supply and demand more effectively.

- Market data: This includes data on energy prices, regulatory policies, or other market factors that can impact energy production and consumption. By considering market data, multi-modal approaches can help energy grid operators make informed decisions about energy production and distribution.

- Geographic data: This includes data on the location and terrain of RESs, such as the angle and orientation of solar panels or the elevation of WTs. By analyzing this data, multi-modal approaches can optimize the placement and operation of renewable energy systems to maximize energy production.

It would be interesting to explore different multi-modal approaches that include multiple or even all the sources mentioned above in future work.
All the approaches presented in this thesis have been developed and tested separately on different RESs. As future work, it is of interest to develop a complete pipeline that incorporates dimensionality reduction, predictive maintenance, and power forecasting for RESs to improve the performance and robustness of the predictive models. A possible pipeline could be the following:

1. Data Preprocessing: This step includes handling missing values, outlier detection and removal, time series smoothing, transformations, scaling, and data formatting.

2. Dimensionality Reduction: This step aims at reducing the number of features or samples in the preprocessed data while preserving the most important information, thus reducing the computation time and storage requirements.

3. Predictive Maintenance: This involves using preprocessed and reduced data to train algorithms that predict when and how a system will likely fail. Main-

tenance can be scheduled in advance using the models' predictions, avoiding unexpected downtime and minimizing maintenance costs.

4. Power Forecasting: This step employs data-driven algorithms to predict the amount of power a RES will likely generate. Considering anomalous behaviors in the energy systems identified in the previous step can improve power forecasts and help the operation of RESs, improve efficiency, and reduce costs.

5. Continuous Improvement: The final step in the pipeline involves monitoring the models' performance and making necessary updates to ensure the pipeline remains effective as data and operating conditions change.

Testing such a pipeline's robustness and generalization capabilities across different energy systems is vital to ensure that it can be applied more broadly and is not limited to a particular system or application. Therefore, some considerations can be taken into account:

- Collecting data from diverse RESs, such as solar, wind, hydroelectric, or geothermal, is essential to test the pipeline on various operating conditions, including different weather patterns, environmental factors, system configurations, and locations.

- It is necessary to preprocess the data consistently across different RESs to ensure the results are comparable across different applications.

- Predictive maintenance models must be tuned for each energy system to ensure they perform optimally. Nevertheless, the model's structure and assumptions must be similar across different systems of the same type to ensure robustness and generality.

- The pipeline's performance on all RESs should be measured using appropriate metrics consistent across different systems to enable meaningful comparisons.

- The pipeline's generalization capabilities should be tested by applying it to data collected from previously unseen energy systems to evaluate its ability to generalize to new data and identify any potential limitations or biases.

- The pipeline's robustness should be tested by introducing variations in the data, such as adding noise or introducing missing values, simulating realistic scenarios when dealing with sensor networks mounted on energy systems. This will help to evaluate the pipeline's ability to handle noisy or incomplete data and to identify any instabilities in the model.

RESs are becoming important as the world moves towards clean and sustainable power sources. Data-driven and ML approaches have the potential to optimize these systems, enhance efficiency, and reduce maintenance costs. However, implementing these methods faces several challenges due to, in many cases, the reluctance of energy providers and other players to share data and knowledge. This is usually caused by concerns about privacy, competitiveness, and the potential misuse of sensitive information, leading to a lack of data and transparency. Therefore, it is

essential to encourage collaboration between energy providers, researchers, and other stakeholders through data-sharing agreements that protect privacy and ensure data security. Moreover, it is crucial to establish open-data initiatives and public-private partnerships to facilitate sharing of anonymized or aggregated data when privacy issues are insurmountable.

The world has experienced two major energy transitions in the past. The first replaced wood with coal, and, during the second, oil and gas replaced coal as the main energy source. Today, the world is undergoing a third energy transition with the main goal to tackle global climate change through decarbonization of the energy supply and consumption patterns. For most of history, the idea of a broad-scale energy transition was unthinkable as decisions were made at the local, regional or individual level, with limited or no coordination. Today, it is not only possible thanks to global cooperation and agreements between parties, power generation systems advancements, and IoT and AI technologies but also necessary to prevent the severe and irreversible consequences of climate change that are threatening life on the planet as we know it.

I conclude this thesis with the hope that it can become a reference to researchers that want to contribute to the sustainable energy transition and are approaching the field of AI in the context of RESs.

Glossary

5x2CV 5×2-fold Cross-Validation. 99

AC Alternating Current. 20, 25

AdaGrad Adaptive Gradient Descend. 72

Adam Adaptive Moment Optimization. 72, 73

AE Autoencoder. 90, 91, 131, 147–149, 151, 156, 160, 161, 164, 183

AI Artificial Intelligence. v, vi, 22–26, 103, 116, 133, 182, 188

ANN Artificial Neural Network. 22, 25, 26, 59, 60, 77, 166

ARIMA Autoregressive Integrated Moving Average. 26

aSi Amorphous Silicon. 133

AUC-ROC Area Under Receiver Operating Characteristics. 96

AWB Automatic Wet Bench. 133, 134, 145, 146

BCE Binary Cross-Entropy. 68, 69

BEM Buildings Energy Management. 18

BRNN Bidirectional Recurrent Neural Network. 23, 79, 80

CAPEX CAPital EXpenditures. 146

CBM Condition-Based Monitoring. 24, 147

CCS Categorical Cross-Entropy. 69

CH4 Methane. 1, 2

CHP Combined Head and Power. 102, 104–108

CM Corrective Maintenance. 130

CMS Control Monitoring System. 9

CNA Complex Network Analysis. 49, 53

CNN Convolutional Neural Network. 22, 26, 86–88, 147, 148, 156, 160, 164, 169

CO2 Carbon Dioxide. 1, 2

ConvLSTM Convolutional LSTM. 177, 178

COP21 21st United Nations Climate Change Conference of Parties. 3

COP26 26st United Nations Climate Change Conference of Parties. 3

COVID-19 COVID-19. 3, 6, 7, 181

CPPS Combine Predictive Power Score. v, 119, 121–123, 125–129

CPS Cyber-Physical System. 17, 102, 103

CPU Central Processing Unit. 176

CSD Commission on Sustainable Development. 3

CV Cross-Validation. 98

DBSCAN Density-Based Spatial Clustering of Applications with Noise. 35, 36, 49–51

DC Direct Current. 20

DGC Digital Grid Consortium. 20

DL Deep Learning. 22, 44, 59, 70, 73, 91, 165

DM Data Mining. 184

DNN Deep Neural Network. 22, 24, 26, 60, 61, 76, 80

DT Decision Tree. 24, 34, 35, 44–48, 52, 118, 122

DTW Dynamic Time Warping. 104

EC Energy Community. 6

ECMWF European Centre for Medium-Range Weather Forecasts. 166, 167, 179

ELU Exponential Linear Unit. 66, 67

ESS Energy Storage System. 164, 185

EU Europe. 20

EV Eletric Vehicle. 18–20

FC Fully-Connected. 60, 86, 87, 90, 91

FEM Factory Energy Management. 18

FN False Negative. 158

FNN Feedforward Neural Networks. 77, 78, 80

FP False Positive. 158, 160, 161

FREEDM Future Renewable Electric Energy Delivery. 20

GB Gradient Boosting. 47, 48

GCN Graph Convolutional Network. 23, 90, 131, 148–150, 183

GD Gradient Descent. 70, 71

GEI Global Energy Internet. 20

GHG GreenHouse Gas. v, 1–6, 11, 14, 181

GMI Global Mahalanobis Indicator. 152, 156–158, 160, 162

GNN Graph Neural Network. 88–90

GPU Graphics Processing Units. 22

GRU Gated Recurrent Unit. 82, 83, 147

HAWT Horizontal-Axis Wind Turbine. 7, 9, 11

HEM Home Energy Management. 18

HRES High-RESolution. 166, 168, 169, 171–173, 175–177, 179

I4.0 Industry 4.0. v, 16, 17, 23, 116, 131, 181

IF Isolation Forest. 34, 35

IGR Information Gain Ratio. 98, 115, 116

IIoT Industrial Internet of Things. 16–18, 20

IoE Internet of Energy. v, 18–20, 26, 181, 182

IoT Internet of Things. v, vi, 16, 18, 20, 102–104, 181, 188

IP Internet Protocol. 20

IPCC Intergovernmental Panel on Climate Change. v, 1, 181

IQR InterQuartile Range. 34, 42, 132, 133, 135–138, 140–145

KFCV K-Fold Cross-Validation. 98, 99

KL-Divergence Kullback Leibler Divergence. 69, 70

KNN K-Nearest Neighbors. 25, 44, 145

KPI Key Performance Indicator. 132, 139, 143–146

LOCF Last Observation Carried Forward. 29, 31

LOOCV Leave-One-Out Cross-Validation. 98, 99

LR Linear Regression. 22, 44, 45

LRI Local Residual Indicator. 152, 156–158, 160, 162

LSTM Long Short-Term Memory. 23, 26, 80–82, 147, 148, 156, 160, 164, 166, 169–172, 177

M2M Machine to Machine. 16

MAE Mean Absolute Error. 68, 69, 97, 121, 122, 125, 127, 128, 155, 156, 174, 175, 177–179

MC Monte Carlo. 132, 133, 135–146

MDAE Median Absolute Error. 174, 179

MDAPE Median Absolute Percentage Error. 156

MES Manufacturing Execution System. 133, 140

MG Microgrid. 164

MHA Multi-Head Attention. 85

MI Mutual Information. 149

ML Machine Learning. v, 22, 24–27, 37, 40, 42–44, 46, 48, 52, 59, 60, 91, 118, 165, 166, 182, 187

MLP Multi-Layer Perceptron. 77, 78, 119, 122, 125, 127

MSE Mean Squared Error. 48, 67–69, 97, 156

MTBF Mean Time Between Failure. 151, 155

MTGCAE Graph Convolutional Autoencoder for Multivariate Time series. 148–152, 154–162, 164

MTL Multi-Task Learning. 76, 77

NLP Natural Language Processing. 184

NN Neural Network. 24, 25, 42, 44, 60–63, 67, 70, 73–77, 86, 88–90, 92, 93, 96, 118, 121, 147, 165–170, 172–174, 177–179, 185

NOCB Next Observation Carried Backward. 30, 31

NOx Nitrous Oxides. 1, 2

NWP Numerical Weather Prediction. 165–168, 177, 179, 185

O&M Operation and Maintenance. v, 15, 23–25, 130, 131, 146, 147, 182, 183

PC Principle Component. 139–141

PCA Principal Component Analysis. vi, 53, 54, 131–133, 136–141, 145, 146

PdM Predictive Maintenance. 130–133

PeCVD Plasma Enhanced Chemical Vapor Deposition. 133

PLC Programmable Logic Controller. 17

PPS Predictive Power Score. 119, 121–123, 125–130

PV PhotoVoltaic. 7, 11–15, 24–26, 132, 133, 146

PVD Physical Vapour Deposition. 133

PvM Preventive Maintenance. 130

RE Renewable Energy. vi, 5–7, 11, 18–20, 23, 26, 102, 131, 164, 165, 181, 182, 185

ReLU Rectified Linear Unit. 60, 65, 66, 76, 122

RES Renewable Energy System. v, vi, 5, 102, 164, 181–188

RF Random Forest. 25, 44, 52, 118

RL Reinforcement Learning. 26

RMSE Root Mean Squared Error. 97, 156, 174, 179

RMSProp Root Mean Square Propagation. 72

RNN Recurrent Neural Network. 22, 23, 78–81, 166, 169

RRR Redundancy Reduction Ratio. 97, 115, 116

RxM Prescriptive Maintenance. 131

S2S Sequence-to-Sequence. 23

SBS Sequential Backward Selection. 52, 118

SCADA Supervisory Control And Data Acquisition. vi, 24, 101, 117–119, 123, 124, 129, 147–149, 153–169, 171, 172, 174–177, 179, 184, 185

SDG Sustainable Development Goal. 4, 5

SFS Sequential Forward Selection. 52, 118, 119

SGD Stochastic Gradient Descent. 71, 72

SO2 Sulphurdioxide. 1

SS Skill Score. 174–176, 179

STC Standard Test Conditions. 13, 14

SVD Singular Value Decomposition. 53, 89, 138, 139

SVM Support Vector Machine. 22, 24–26

SVR Support Vector Regression. 25, 26, 44, 119, 166

TP True Positive. 158, 160

UN United Nations. 3

UNCED United Nations Conference on Environment and Development. 3

UNEP United Nations Environment Programme. 3

UNFCCC United Nations Framework Convention on Climate Change. 3

US United States. 8, 12, 20

USB Universal Serial Bus. 20

WIS Wafer Inspection System. 133

WSSD World Summit on Sustainable Development. 3

WT Wind Turbine. 7–11, 24, 25, 35, 36, 46, 117–119, 123, 124, 128, 129, 131, 146–148, 153–155, 159, 161, 163–169, 174, 178, 179, 183, 184, 186

XGBoost Extreme Gradient Boosting. 48, 52, 118, 174, 176, 177, 179

Bibliography

[1] M. Allen, P. Antwi-Agyei, F. Aragon-Durand, M. Babiker, P. Bertoldi, M. Bind, S. Brown, M. Buckeridge, I. Camilloni, A. Cartwright, et al., Technical summary: 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, Intergovernmental Panel on Climate Change (2019).

[2] IPCC, Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change, Vol. In Press, Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, 2021. doi:10.1017/9781009157896.

[3] A. A. Marcus, Controversial issues in energy policy, Vol. 2, Sage, 1992.

[4] T. J. Wallington, J. Srinivasan, O. J. Nielsen, E. J. Highwood, Greenhouse gases and global warming, Environmental and ecological chemistry 1 (2009) 36–63.

[5] G. I. Tunc, S. Türüt-Aşık, E. Akbostancı, Co2 emissions vs. co2 responsibility: an input–output approach for the turkish economy, Energy Policy 35 (2) (2007) 855–868.

[6] R. Lindsey, Climate change: Atmospheric carbon dioxide, climate.gov (2022).

[7] H. K. Wang, Climate Change and Clean Energy Management: Challenges and Growth Strategies, Routledge, 2019.

[8] M. Kabeyi, O. Olanrewaju, Sustainable energy transition for renewable and low carbon grid electricity generation and supply. front, Energy Res 9 (2022) 743114.

[9] S. Fawzy, A. I. Osman, J. Doran, D. W. Rooney, Strategies for mitigation of climate change: a review, Environmental Chemistry Letters 18 (6) (2020) 2069–2094.

[10] E. Vasseur, United nations conference on the human environment: Stockholm, 5–16 june 1972, Water Research 7 (8) (1973) 1227–1233.

[11] A. 21, United nations conference on environment & development, Rio de Jeneiro, Brazil (1992).

[12] I. Von Frantzius, World summit on sustainable development johannesburg 2002: A critical analysis and assessment of the outcomes, Environmental Politics 13 (2) (2004) 467–473.

[13] K. Protocol, Kyoto protocol, UNFCCC Website. Available online: http://unfccc. int/kyoto_protocol/items/2830. php (accessed on 1 January 2011) (1997).

[14] G. Erbach, Doha amendment to the kyoto protocol, EPRS (2015).

[15] P. Agreement, Paris agreement, in: Report of the Conference of the Parties to the United Nations Framework Convention on Climate Change (21st Session, 2015: Paris). Retrived December, Vol. 4, HeinOnline, 2015, p. 2017.

[16] C. C. COP, United nations climate change conference, COP (2021).

[17] U. Desa, et al., Transforming our world: The 2030 agenda for sustainable development, Desa (2016).

[18] H. Ritchie, M. Roser, P. Rosado, Co2 and greenhouse gas emissions, Our World in DataHttps://ourworldindata.org/co2-and-other-greenhouse-gas-emissions (2020).

[19] S. A. Solarin, M. O. Bello, F. V. Bekun, Sustainable electricity generation: the possibility of substituting fossil fuels for hydropower and solar energy in italy, International Journal of Sustainable Development & World Ecology 28 (5) (2021) 429–439.

[20] IEA, World energy outlook 2019, IEA (2019).
URL `https://www.iea.org/reports/world-energy-outlook-2019`

[21] IEA, Key world energy statistics 2021, IEA (2021).
URL `https://www.iea.org/reports/key-world-energy-statistics-2021`

[22] K. Hansen, C. Breyer, H. Lund, Status and perspectives on 100% renewable energy systems, Energy 175 (2019) 471–480.

[23] P. del Río, L. Janeiro, Overcapacity as a barrier to renewable energy deployment: the spanish case, Journal of Energy 2016 (2016).

[24] G. Liu, Development of a general sustainability indicator for renewable energy systems: A review, Renewable and sustainable energy reviews 31 (2014) 611–621.

[25] L. Jaramillo-Nieves, P. Del Río, Contribution of renewable energy sources to the sustainable development of islands: An overview of the literature and a research agenda, Sustainability 2 (3) (2010) 783–811.

[26] F. J. Zarco-Soto, P. J. Zarco-Periñán, J. L. Martínez-Ramos, Centralized control of distribution networks with high penetration of renewable energies, Energies 14 (14) (2021) 4283.

[27] IEA, The covid-19 crisis and clean energy progress, IEA (2020).
URL https://www.iea.org/reports/the-covid-19-crisis-and-clean-energy-progress

[28] J. Tollefson, et al., What the war in ukraine means for energy, climate and food, Nature 604 (7905) (2022) 232–233.

[29] J. Zhong, M. Bollen, S. Rönnberg, Towards a 100% renewable energy electricity generation system in sweden, Renewable Energy 171 (2021) 812–824.

[30] IEA, Danish energy agreement of 29 june 2018, IEA (2018).
URL https://www.iea.org/policies/12144-danish-energy-agreement-of-29-june-2018-only

[31] REN21, Renewables 2018 global status report, IEA (2018).
URL https://www.ren21.net/gsr-2018/

[32] B. D. Solomon, K. Krishna, The coming sustainable energy transition: History, strategies, and outlook, Energy Policy 39 (11) (2011) 7422–7431.

[33] V. Smil, Energy transitions: history, requirements, prospects, ABC-CLIO, 2010.

[34] IEA, Global energy review 2021, IEA (2021).
URL https://www.iea.org/reports/global-energy-review-2021

[35] Enerdata, Share of wind and solar in electricity production, Enerdata (2021).
URL https://www.enerdata.net/publications/reports-presentations/world-energy-trends.html

[36] J. Cao, Y. Shi, H. Zhang, Wind energy: History, current design principles, market penetration, environmental impacts, in: 2021 International Conference on Control Science and Electric Power Systems (CSEPS), IEEE, 2021, pp. 398–405.

[37] M. S. Energy, Leading innovation: Mingyang smart energy launches myse 16.0-242, the world's largest offshore hybrid drive wind turbine, MingYang Smart Energy (2018).
URL http://www.myse.com.cn/en/jtxw/info.aspx?itemid=825

[38] R. Wiser, J. Rand, J. Seel, P. Beiter, E. Baker, E. Lantz, P. Gilman, Expert elicitation survey predicts 37% to 49% declines in wind energy costs by 2050, Nature Energy 6 (5) (2021) 555–565.

[39] R. Wiser, M. Bolinger, B. Hoen, D. Millstein, J. Rand, G. Barbose, N. Darghouth, W. Gorman, S. Jeong, B. Paulos, Land-based wind market report: 2022 edition, Tech. rep., Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States) (2022).

[40] Y. Tumewu, C. Petrone, M. Sivaselvan, Numerical simulation of the influence of platform pitch motion on power generation steadiness in floating offshore wind turbines, International Journal of Environmental Science & Sustainable Development 1 (2) (2017).

[41] M.-L. Chanin, A. Garnier, A. Hauchecorne, J. Porteneuve, A doppler lidar for measuring winds in the middle atmosphere, Geophysical research letters 16 (11) (1989) 1273–1276.

[42] K. Zipp, Turbine of the month: A turbine with top power curve acciona aw3000, Windpower Engineering & Development (2012).
URL https://www.windpowerengineering.com/turbine-of-the-month-a-turbine-with-top-power-curve-acciona-aw3000116/

[43] J. F. Manwell, J. G. McGowan, A. L. Rogers, Wind energy explained: theory, design and application, John Wiley & Sons, 2010.

[44] M. Becquerel, Mémoire sur les effets électriques produits sous l'influence des rayons solaires, Comptes rendus hebdomadaires des séances de l'Académie des sciences 9 (1839) 561–567.

[45] W. Smith, Effect of light on selenium, Nature 7 (1873) 303.

[46] W. G. Adams, R. E. Day, V. the action of light on selenium, Proceedings of the Royal Society of London 25 (171-178) (1877) 113–117.

[47] C. E. Fritts, On a new form of selenium cell, and some electrical discoveries made by its use, American Journal of Science 3 (156) (1883) 465–472.

[48] L. M. Fraas, History of solar cell development, in: Low-cost solar electric power, Springer, 2014, pp. 1–12.

[49] A. Einstein, et al., On the motion of small particles suspended in liquids at rest required by the molecular-kinetic theory of heat, Annalen der physik 17 (549-560) (1905) 208.

[50] D. C. Brock, Useless no more: Gordon k. teal, germanium, and single-crystal transistors, Chemical Heritage Newsmagazine (Chemical Heritage Foundation) 24 (1) (2007).

[51] D. M. Chapin, C. S. Fuller, G. L. Pearson, A new silicon p-n junction photocell for converting solar radiation into electrical power, Journal of applied physics 25 (5) (1954) 676–677.

[52] F. Venn, The oil crisis, Routledge, 2016.

[53] IRENA, Renewable power generation costs in 2021, IRENA (2021).
URL https://irena.org/publications/2022/Jul/Renewable-Power-Generation-Costs-in-2021

[54] A. Fernandez, How a solar cell works, American Chemical Society Website (2014).
URL https://www.acs.org/content/acs/en/education/resources/highschool/chemmatters/past-issues/archive-2013-2014/how-a-solar-cell-works.html

[55] J. Svarc, Most powerful solar panels 2022, Clean Energy Reviews (2021). URL https://www.cleanenergyreviews.info/blog/most-powerful-solar-panels

[56] M. A. Green, High efficiency silicon solar cells, in: Seventh EC Photovoltaic Solar Energy Conference, Springer, 1987, pp. 681–687.

[57] U. D. of Energy, The history of solar, U.S. Department of Energy (2002). URL https://www1.eere.energy.gov/solar/pdfs/solar_timeline.pdf

[58] U. of New South Wales, Milestone in solar cell efficiency achieved: New record for unfocused sunlight edges closer to theoretic limits, ScienceDaily (2016). URL www.sciencedaily.com/releases/2016/05/160517121811.htm

[59] J. F. Geisz, R. M. France, K. L. Schulte, M. A. Steiner, A. G. Norman, H. L. Guthrey, M. R. Young, T. Song, T. Moriarty, Six-junction iii–v solar cells with 47.1% conversion efficiency under 143 suns concentration, Nature energy 5 (4) (2020) 326–335.

[60] S. Ghazi, K. Ip, The effect of weather conditions on the efficiency of pv panels in the southeast of uk, Renewable energy 69 (2014) 50–59.

[61] B. Willockx, B. Herteleer, J. Cappelle, Combining photovoltaic modules and food crops: first agrovoltaic prototype in belgium, Renewable Energy & Power Quality Journal (RE&PQJ) 18 (2020).

[62] B. Willockx, B. Herteleer, J. Cappelle, Theoretical potential of agrovoltaic systems in europe: a preliminary study with winter wheat, in: 2020 47th IEEE Photovoltaic Specialists Conference (PVSC), IEEE, 2020, pp. 0996–1001.

[63] M. Burhan, R. A. Rehman, B. Khan, B.-S. Kim, Iot elements, layered architectures and security issues: A comprehensive survey, Sensors 18 (9) (2018) 2796.

[64] K. Rose, S. Eldridge, L. Chapin, The internet of things: An overview, The internet society (ISOC) 80 (2015) 1–50.

[65] L. Da Xu, W. He, S. Li, Internet of things in industries: A survey, IEEE Transactions on industrial informatics 10 (4) (2014) 2233–2243.

[66] Y. Shahzad, H. Javed, H. Farman, J. Ahmad, B. Jan, M. Zubair, Internet of energy: Opportunities, applications, architectures and challenges in smart industries, Computers & Electrical Engineering 86 (2020) 106739.

[67] P. O'brien, The micro foundations of macro invention: the case of the reverend edmund cartwright, Textile History 28 (2) (1997) 201–233.

[68] R. Batchelor, Henry Ford, mass production, modernism, and design, Vol. 1, Manchester University Press, 1994.

[69] P. Waurzyniak, Masters of manufacturing: Richard morley, Manufacturing Engineering 131 (1) (2003) 35–35.

[70] R. Y. Zhong, X. Xu, E. Klotz, S. T. Newman, Intelligent manufacturing in the context of industry 4.0: a review, Engineering 3 (5) (2017) 616–630.

[71] H. Sasajima, T. Ishikuma, H. Hayashi, Future iiot in process automation—latest trends of standardization in industrial automation, iec/tc65, in: Proceedings of the 54th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Hangzhou, China, 2015, pp. 28–30.

[72] F. Stefano, F. Benzi, E. Bassi, Iiot based efficiency optimization in logistics applications, Asian Journal of Basic Science & Research 2 (4) (2020) 59–73.

[73] M. Cakir, M. A. Guvenc, S. Mistikoglu, The experimental application of popular machine learning algorithms on predictive maintenance and the design of iiot based condition monitoring system, Computers & Industrial Engineering 151 (2021) 106948.

[74] H. Li, Research on safety monitoring system of workers in dangerous operation area of port, in: 2017 4th International Conference on Transportation Information and Safety (ICTIS), IEEE, 2017, pp. 400–408.

[75] B. N. Silva, M. Khan, K. Han, Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities, Sustainable Cities and Society 38 (2018) 697–713.

[76] S. Bhowmick, B. Biswas, M. Biswas, A. Dey, S. Roy, S. K. Sarkar, Application of iot-enabled smart agriculture in vertical farming, in: Advances in Communication, Devices and Networking, Springer, 2019, pp. 521–528.

[77] X. Fang, S. Misra, G. Xue, D. Yang, Smart grid—the new and improved power grid: A survey, IEEE communications surveys & tutorials 14 (4) (2011) 944–980.

[78] J. Rifkin, The third industrial revolution: how lateral power is transforming energy, the economy, and the world, Macmillan, 2011.

[79] G. Tulemissova, The impact of the iot and ioe technologies on changes of knowledge management strategy, in: ECIC2016-Proceedings of the 8th European Conference on Intellectual Capital: ECIC2016, Academic Conferences and publishing limited, 2016, p. 300.

[80] T. L. Vu, N. T. Le, Y. M. Jang, et al., An overview of internet of energy (ioe) based building energy management system, in: 2018 International Conference on Information and Communication Technology Convergence (ICTC), IEEE, 2018, pp. 852–855.

[81] Y. Kafle, K. Mahmud, S. Morsalin, G. Town, Towards an internet of energy, in: 2016 IEEE International Conference on Power System Technology (POWERCON), IEEE, 2016, pp. 1–6.

[82] Y. Xu, J. Zhang, W. Wang, A. Juneja, S. Bhattacharya, Energy router: Architectures and functionalities toward energy internet, in: 2011 IEEE International Conference on Smart Grid Communications (SmartGridComm), IEEE, 2011, pp. 31–36.

[83] G. Xu, W. Yu, D. Griffith, N. Golmie, P. Moulema, Toward integrating distributed energy resources and storage devices in smart grid, IEEE internet of things journal 4 (1) (2016) 192–204.

[84] Z. Zhang, Y. Zhang, M.-Y. Chow, Distributed energy management under smart grid plug-and-play operations, in: 2013 IEEE Power & Energy Society General Meeting, IEEE, 2013, pp. 1–5.

[85] O. Vermesan, L.-C. Blystad, P. Hank, R. Bahr, R. John, A. Moscatelli, Smart, connected and mobile: Architecting future electric mobility ecosystems, in: 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2013, pp. 1740–1744.

[86] Artemis, Internet of energy for electric mobility, Artemis (2011).
URL `http://www.artemis-ioe.eu/index.htm`

[87] A. Q. Huang, M. L. Crow, G. T. Heydt, J. P. Zheng, S. J. Dale, The future renewable electric energy delivery and management (freedm) system: the energy internet, Proceedings of the IEEE 99 (1) (2010) 133–148.

[88] H.-J. Belitz, S. Winter, C. Rehtanz, Load shifting of the households in the e-energy project e-dema, in: 2013 IEEE Grenoble Conference, IEEE, 2013, pp. 1–6.

[89] L. L. Lai, Global energy internet and interconnection, IEEE Smart Grid Newsletter (2015).

[90] R. Abe, H. Taoka, D. McQuilkin, Digital grid: Communicative electrical grids of the future, IEEE Transactions on Smart Grid 2 (2) (2011) 399–410.

[91] R. Nishant, M. Kennedy, J. [illegible] telligence for sustainability: Challenges, opportunities, [illegible]da, International Journal of Information Management 5[illegible]

[92] P. L. Donti, J. Z. Kolter, M[illegible] sustainable energy systems, Annual Review of Environm[illegible]6 (2021) 719–747.

[93] K. W. Kow, Y. W. Wong, [illegible] K. Rajkumar, A review on performance of artificial intelligence and conventional method in mitigating pv grid-tied related power quality events, Renewable and Sustainable Energy Reviews 56 (2016) 334–346.

[94] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (3) (1995) 273–297.

[95] T. G. Dietterich, Ensemble methods in machine learning, in: International workshop on multiple classifier systems, Springer, 2000, pp. 1–15.

Time series clustering: A complex network-based

[96] W. S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, The bulletin of mathematical biophysics 5 (4) (1943) 115–133.

[97] P. Werbos, Beyond regression : new fools for prediction and analysis in the behavioral sciences, PhD thesis, Harvard University (1974).
URL `https://ci.nii.ac.jp/naid/10010414900/en/`

[98] B. C. Csáji, Approximation with artificial neural networks, MSc Thesis, Eötvös Loránd University (ELTE), Budapest, Hungary 24 (2001) 48.

[99] K. Fukushima, Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, Biological cybernetics 36 (4) (1980) 193–202.

[100] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al., Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[101] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[102] S. Hochreiter, Untersuchungen zu dynamischen neuronalen netzen, Diploma, Technische Universität München 91 (1) (1991).

[103] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112.

[104] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

[105] T. Luong, H. Pham, C. D. Manning, Effective approaches to attention-based neural machine translation, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015, 2015, pp. 1412–1421.
URL `http://aclweb.org/anthology/D/D15/D15-1166.pdf`

[106] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).

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

[108] T. N. Kipf, M. Welling, Semi-Supervised Classification with Graph Convolutional Networks, in: Proceedings of the 5th International Conference on Learning Representations, ICLR '17, 2017, p. 1.

[109] L. Cozzi, D. Turk, T. Abergel, J. Bartos, E. Bellevrat, S. Bennett, T. Berly, S. Bouckaert, J. Dulac, C. F. Alvarez, et al., Digitalization and Energy, OECD, 2017.

[110] A. Gilchrist, Industry 4.0: the industrial internet of things, Springer, 2016.

[111] A. Ustundag, E. Cevikcan, Industry 4.0: managing the digital transformation, Springer, 2017.

[112] R. K. Mobley, An introduction to predictive maintenance, Elsevier, 2002.

[113] E. Lughofer, M. Sayed-Mouchaweh, Predictive maintenance in dynamic systems: advanced methods, decision support tools and real-world applications, Springer, 2019.

[114] R. Akerkar, P. Sajja, Knowledge-based systems, Jones & Bartlett Publishers, 2009.

[115] M. Sayed-Mouchaweh, E. Lughofer, Learning in non-stationary environments: methods and applications, Springer Science & Business Media, 2012.

[116] P. Marti-Puig, A. Blanco-M, J. J. Cárdenas, J. Cusidó, J. Solé-Casals, Effects of the pre-processing algorithms in fault diagnosis of wind turbines, Environmental modelling & software 110 (2018) 119–128.

[117] J. Chen, W. Chen, J. Li, P. Sun, A generalized model for wind turbine faulty condition detection using combination prediction approach and information entropy., Journal of Environmental Informatics 32 (1) (2018).

[118] Y. Cui, P. Bangalore, L. B. Tjernberg, An anomaly detection approach based on machine learning and scada data for condition monitoring of wind turbines, in: 2018 IEEE International Conference on Probabilistic Methods Applied to Power Systems (PMAPS), IEEE, 2018, pp. 1–6.

[119] A. Blanco-M, K. Gibert, P. Marti-Puig, J. Cusidó, J. Solé-Casals, Identifying health status of wind turbines by using self organizing maps and interpretation-oriented post-processing tools, Energies 11 (4) (2018) 723.

[120] K. Leahy, C. Gallagher, K. Bruton, P. O'Donovan, D. T. O'Sullivan, Automatically identifying and predicting unplanned wind turbine stoppages using scada and alarms system data: Case study and results, in: Journal of Physics: Conference Series, Vol. 926, IOP Publishing, 2017, p. 012011.

[121] P. Sun, J. Li, C. Wang, Y. Yan, Condition assessment for wind turbines with doubly fed induction generators based on scada data, Journal of Electrical Engineering and Technology 12 (2) (2017) 689–700.

[122] J. Maldonado-Correa, S. Martín-Martínez, E. Artigao, E. Gómez-Lázaro, Using scada data for wind turbine condition monitoring: A systematic literature review, Energies 13 (12) (2020) 3132.

[123] J. Tautz-Weinert, S. J. Watson, Using scada data for wind turbine condition monitoring–a review, IET Renewable Power Generation 11 (4) (2017) 382–394.

[124] A. Stetco, F. Dinmohammadi, X. Zhao, V. Robu, D. Flynn, M. Barnes, J. Keane, G. Nenadic, Machine learning methods for wind turbine condition monitoring: A review, Renewable energy 133 (2019) 620–635.

[125] L. B. Bosman, W. D. Leon-Salas, W. Hutzel, E. A. Soto, Pv system predictive maintenance: Challenges, current approaches, and opportunities, Energies 13 (6) (2020) 1398.

[126] E. Klugmann-Radziemska, Degradation of electrical performance of a crystalline photovoltaic module due to dust deposition in northern poland, Renewable Energy 78 (2015) 418–426.

[127] A. Bouraiou, M. Hamouda, A. Chaker, A. Neçaibia, M. Mostefaoui, N. Boutasseta, A. Ziane, R. Dabou, N. Sahouane, S. Lachtar, Experimental investigation of observed defects in crystalline silicon pv modules under outdoor hot dry climatic conditions in algeria, Solar Energy 159 (2018) 475–487.

[128] M. Díez-Mediavilla, M. Dieste-Velasco, M. d. C. Rodríguez-Amigo, T. García-Calderón, C. Alonso-Tristán, Performance of grid-tied pv facilities based on real data in spain: Central inverter versus string system, Energy Conversion and Management 86 (2014) 1128–1133.

[129] R. Platon, J. Martel, N. Woodruff, T. Y. Chau, Online fault detection in pv systems, IEEE Transactions on Sustainable Energy 6 (4) (2015) 1200–1207.

[130] M. De Benedetti, F. Leonardi, F. Messina, C. Santoro, A. Vasilakos, Anomaly detection and predictive maintenance for photovoltaic systems, Neurocomputing 310 (2018) 59–68.

[131] X. Wang, P. Guo, X. Huang, A review of wind power forecasting models, Energy procedia 12 (2011) 770–778.

[132] S. Hanifi, X. Liu, Z. Lin, S. Lotfian, A critical review of wind power forecasting methods—past, present and future, Energies 13 (15) (2020) 3764.

[133] K. L. Jørgensen, H. R. Shaker, Wind power forecasting using machine learning: State of the art, trends and challenges, in: 2020 IEEE 8th International Conference on Smart Energy Grid Engineering (SEGE), IEEE, 2020, pp. 44–50.

[134] D. Rangel-Martinez, K. Nigam, L. A. Ricardez-Sandoval, Machine learning on sustainable energy: A review and outlook on renewable energy systems, catalysis, smart grid and energy storage, Chemical Engineering Research and Design 174 (2021) 414–441.

[135] W. Yaïci, E. Entchev, Performance prediction of a solar thermal energy system using artificial neural networks, Applied thermal engineering 73 (1) (2014) 1348–1359.

[136] M. W. Ahmad, J. Reynolds, Y. Rezgui, Predictive modelling for solar thermal energy systems: A comparison of support vector regression, random forest, extra trees and regression trees, Journal of cleaner production 203 (2018) 810–821.

[137] C. Voyant, G. Notton, S. Kalogirou, M.-L. Nivet, C. Paoli, F. Motte, A. Fouilloy, Machine learning methods for solar radiation forecasting: A review, Renewable Energy 105 (2017) 569–582.

[138] K. Amarasinghe, D. L. Marino, M. Manic, Deep neural networks for energy load forecasting, in: 2017 IEEE 26th international symposium on industrial electronics (ISIE), IEEE, 2017, pp. 1483–1488.

[139] H. Naganathan, W. O. Chong, X. Chen, Building energy modeling (bem) using clustering algorithms and semi-supervised machine learning approaches, Automation in Construction 72 (2016) 187–194.

[140] R. Dobbe, O. Sondermeijer, D. Fridovich-Keil, D. Arnold, D. Callaway, C. Tomlin, Toward distributed energy services: Decentralizing optimal power flow with machine learning, IEEE Transactions on Smart Grid 11 (2) (2019) 1296–1306.

[141] Z. Zhang, D. Zhang, R. C. Qiu, Deep reinforcement learning for power system applications: An overview, CSEE Journal of Power and Energy Systems 6 (1) (2019) 213–225.

[142] M. Glavic, (deep) reinforcement learning for electric power system control and related problems: A short review and perspectives, Annual Reviews in Control 48 (2019) 22–35.

[143] M. Glavic, R. Fonteneau, D. Ernst, Reinforcement learning for electric power system decision and control: Past considerations and perspectives, IFAC-PapersOnLine 50 (1) (2017) 6918–6927.

[144] Y. Juan, Y. Dai, Y. Yang, J. Zhang, Accelerating materials discovery using machine learning, Journal of Materials Science & Technology 79 (2021) 178–190.

[145] Ç. Odabaşı, R. Yıldırım, Machine learning analysis on stability of perovskite solar cells, Solar Energy Materials and Solar Cells 205 (2020) 110284.

[146] H. Li, Z. Liu, K. Liu, Z. Zhang, Predictive power of machine learning for optimizing solar water heater performance: the potential application of high-throughput screening, International journal of photoenergy 2017 (2017).

[147] J. Han, J. Pei, H. Tong, Data mining: concepts and techniques, Morgan kaufmann, 2022.

[148] A. R. T. Donders, G. J. Van Der Heijden, T. Stijnen, K. G. Moons, A gentle introduction to imputation of missing values, Journal of clinical epidemiology 59 (10) (2006) 1087–1091.

[149] J. M. Lachin, Fallacies of last observation carried forward analyses, Clinical trials 13 (2) (2016) 161–168.

[150] J. M. Engels, P. Diehr, Imputation of missing longitudinal data: a comparison of methods, Journal of clinical epidemiology 56 (10) (2003) 968–976.

[151] M. Lepot, J.-B. Aubin, F. H. Clemens, Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment, Water 9 (10) (2017) 796.

[152] F. Pukelsheim, The three sigma rule, The American Statistician 48 (2) (1994) 88–91.

[153] B. Wang, W. Shi, Z. Miao, Confidence analysis of standard deviational ellipse and its extension into higher dimensional euclidean space, PloS one 10 (3) (2015) e0118537.

[154] R. Dawson, How significant is a boxplot outlier?, Journal of Statistics Education 19 (2) (2011).

[155] N. A. Bakar, S. Rosbi, Identification of non-equilibrium growth for bitcoin exchange rate: Mathematical derivation method in islamic financial engineering, International Journal of Scientific Research and Management (IJSRM) 5 (12) (2017) 7772–7781.

[156] F. T. Liu, K. M. Ting, Z.-H. Zhou, Isolation forest, in: 2008 eighth ieee international conference on data mining, IEEE, 2008, pp. 413–422.

[157] J. Chen, J. Li, W. Chen, Y. Wang, T. Jiang, Anomaly detection for wind turbines based on the reconstruction of condition parameters using stacked denoising autoencoders, Renewable Energy 147 (2020) 1469–1480.

[158] R. J. Hyndman, Moving averages. (2011).

[159] R. W. Schafer, What is a savitzky-golay filter?[lecture notes], IEEE Signal processing magazine 28 (4) (2011) 111–117.

[160] S. Liu, Y. Zhang, X. Du, T. Xu, J. Wu, Short-term power prediction of wind turbine applying machine learning and digital filter, Applied Sciences 13 (3) (2023) 1751.

[161] R. J. Hyndman, G. Athanasopoulos, Forecasting: principles and practice, OTexts, 2018.

[162] I.-K. Yeo, R. A. Johnson, A new family of power transformations to improve normality or symmetry, Biometrika 87 (4) (2000) 954–959.

[163] A. Bauer, Automated hybrid time series forecasting: Design, benchmarking, and use cases, Ph.D. thesis, Universität Würzburg (2021).

[164] G. E. Box, D. R. Cox, An analysis of transformations, Journal of the Royal Statistical Society: Series B (Methodological) 26 (2) (1964) 211–243.

[165] P. Li, Box-cox transformations: an overview, presentation, http://www. stat. uconn. edu/˜ studentjournal/index_files/pengfi_s05. pdf (2005).

[166] S. Weisberg, Yeo-johnson power transformations, Department of Applied Statistics, University of Minnesota. Retrieved June 1 (2001) 2003.

[167] S. Patro, K. K. Sahu, Normalization: A preprocessing stage, arXiv preprint arXiv:1503.06462 (2015).

[168] C.-S. J. Chu, Time series segmentation: A sliding window approach, Information Sciences 85 (1-3) (1995) 147–173.

[169] S. B. Kotsiantis, I. Zaharakis, P. Pintelas, et al., Supervised machine learning: A review of classification techniques, Emerging artificial intelligence applications in computer engineering 160 (1) (2007) 3–24.

[170] D. R. Cox, The regression analysis of binary sequences, Journal of the Royal Statistical Society: Series B (Methodological) 20 (2) (1958) 215–232.

[171] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, et al., Top 10 algorithms in data mining, Knowledge and information systems 14 (1) (2008) 1–37.

[172] T. K. Ho, Random decision forests, in: Proceedings of 3rd international conference on document analysis and recognition, Vol. 1, IEEE, 1995, pp. 278–282.

[173] N. S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, The American Statistician 46 (3) (1992) 175–185.

[174] M. Diskin, Definition and uses of the linear regression model, Water Resources Research 6 (6) (1970) 1668–1673.

[175] J. Ranstam, J. Cook, Lasso regression, Journal of British Surgery 105 (10) (2018) 1348–1348.

[176] G. C. McDonald, Ridge regression, Wiley Interdisciplinary Reviews: Computational Statistics 1 (1) (2009) 93–100.

[177] E. Ostertagová, Modelling using polynomial regression, Procedia Engineering 48 (2012) 500–506.

[178] A. J. Smola, B. Schölkopf, A tutorial on support vector regression, Statistics and computing 14 (3) (2004) 199–222.

[179] A. C. Todd, M. Optis, N. Bodini, M. J. Fields, J. Perr-Sauer, J. C. Lee, E. Simley, R. Hammond, An independent analysis of bias sources and variability in wind plant pre-construction energy yield estimation methods, Wind Energy 25 (10) (2022) 1775–1790.

[180] I. Abdallah, V. Dertimanis, H. Mylonas, K. Tatsis, E. Chatzi, N. Dervili, K. Worden, E. Maguire, Fault diagnosis of wind turbine structures using decision tree learning algorithms with big data, in: Safety and Reliability–Safe Societies in a Changing World, CRC Press, 2018, pp. 3053–3061.

[181] J. H. Friedman, Greedy function approximation: a gradient boosting machine, Annals of statistics (2001) 1189–1232.

[182] A. Terko, E. Žunić, D. Đonko, A. Dželihodžić, Credit scoring model implementation in a microfinance context, in: 2019 XXVII International Conference on Information, Communication and Automation Technologies (ICAT), IEEE, 2019, pp. 1–6.

[183] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, New York, NY, USA, 2016, pp. 785–794. doi:10.1145/2939672.2939785.
URL http://doi.acm.org/10.1145/2939672.2939785

[184] G. Mesnil, Y. Dauphin, X. Glorot, S. Rifai, Y. Bengio, I. Goodfellow, E. Lavoie, X. Muller, G. Desjardins, D. Warde-Farley, et al., Unsupervised and transfer learning challenge: a deep learning approach, in: Proceedings of ICML Workshop on Unsupervised and Transfer Learning, JMLR Workshop and Conference Proceedings, 2012, pp. 97–110.

[185] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, X. Xu, Dbscan revisited, revisited: why and how you should (still) use dbscan, ACM Transactions on Database Systems (TODS) 42 (3) (2017) 1–21.

[186] A. Amini, T. Y. Wah, H. Saboohi, On density-based data streams clustering algorithms: A survey, Journal of Computer Science and Technology 29 (1) (2014) 116–141.

[187] M. N. Gaonkar, K. Sawant, Autoepsdbscan: Dbscan with eps automatic for large dataset, International Journal on Advanced Computer Theory and Engineering 2 (2) (2013) 11–16.

[188] B. Rahdari, T. Arabghalizi, Event-based user profiling in social media using data mining approaches, 2017.

[189] J. Wu, Advances in K-means clustering: a data mining thinking, Springer Science & Business Media, 2012.

[190] S. K. Popat, M. Emmanuel, Review and comparative study of clustering techniques., International journal of computer science and information technologies 5 (2014) 805–812.

[191] A. Fujita, J. Sato, M. Demasi, M. Sogayar, C. Ferreira, S. Miyano, Comparing pearson, spearman and hoeffding's d measure for gene expression association analysis, Journal of bioinformatics and computational biology 7 (2009) 663–84.

[192] F. Iglesias, W. Kastner, Analysis of similarity measures in times series clustering for the discovery of building energy patterns, Energies 6 (2) (2013) 579–597.

[193] L. Jing, M. Ng, J. Huang, An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data, IEEE Transactions on knowledge and data engineering 19 (2007) 1026–1041.

[194] X. Huang, Y. Ye, H. Zhang, Extensions of kmeans-type algorithms: A new clustering framework by integrating intracluster compactness and intercluster separation, IEEE Transactions on Neural Networks and Learning Systems 25 (2014) 1433–1446.

[195] R. Baragona, A simulation study on clustering time series with metaheuristic methods, Quaderni di Statistica 3 (01 2001).

[196] M. Ramoni, P. Sebastiani, P. R. Cohen, Multivariate clustering by dynamics, in: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, AAAI Press, 2000, pp. 633–638.

[197] D. Tran, M. Wagner, Fuzzy c-means clustering-based speaker verification, in: Proceedings of the AFSS International Conference on Fuzzy Systems, Springer, 2002, pp. 318–324.

[198] K. Bandara, C. Bergmeir, S. Smyl, Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach, Expert Systems with Applications 140 (2020) 112896.

[199] C. Shaw, G. King, Using cluster analysis to classify time series, Physica D: Nonlinear Phenomena 58 (1992) 288–298.

[200] M. Vlachos, J. Lin, E. Keogh, D. Gunopulos, A wavelet-based anytime algorithm for k-means clustering of time series, in: Proceedings of the Workshop on Clustering High Dimensionality Data and Its Applications, 2003, pp. 23–30.

[201] V. Kavitha, M. Punithavalli, Clustering time series data stream – a literature survey, International Journal of Computer Science and Information Security 8 (04 2010).

[202] S. Rani, G. Sikka, Recent techniques of clustering of time series data: A survey, International Journal of Computer Applications 52 (2012) 1–9.

[203] S. Aghabozorgi, A. S. Shirkhorshidi, T. Wah, Time-series clustering - a decade review, Information Systems 53 (05 2015).

[204] M. Köppen, The curse of dimensionality, in: 5th online world conference on soft computing in industrial applications (WSC5), Vol. 1, 2000, pp. 4–8.

[205] S. Khalid, T. Khalil, S. Nasreen, A survey of feature selection and feature extraction techniques in machine learning, in: 2014 science and information conference, IEEE, 2014, pp. 372–378.

[206] G. Li, C. Wang, D. Zhang, G. Yang, An improved feature selection method based on random forest algorithm for wind turbine condition monitoring, Sensors 21 (16) (2021) 5654.

[207] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Computers & Electrical Engineering 40 (1) (2014) 16–28.

[208] X. Peng, K. Cheng, J. Lang, Z. Zhang, T. Cai, S. Duan, Short-term wind power prediction for wind farm clusters based on sffs feature selection and blstm deep learning, Energies 14 (7) (2021) 1894.

[209] R. Kohavi, G. H. John, Wrappers for feature subset selection, Artificial intelligence 97 (1-2) (1997) 273–324.

[210] B. Fritzke, Unsupervised clustering with growing cell structures, in: Proceedings of the IJCNN-91-Seattle International Joint Conference on Neural Networks, Vol. 2, 1991, pp. 531–536.

[211] B. Clarkson, A. Pentland, Unsupervised clustering of ambulatory audio and video, in: Proceedings of the 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing., Vol. 6, IEEE, 1999, pp. 3037–3040.

[212] K. P. F.R.S., Liii. on lines and planes of closest fit to systems of points in space, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2 (11) (1901) 559–572. doi:10.1080/14786440109462720.

[213] J. J. Gerbrands, On the relationships between svd, klt and pca, Pattern recognition 14 (1-6) (1981) 375–381.

[214] M. Hesami, A. M. P. Jones, Application of artificial intelligence models and optimization algorithms in plant cell and tissue culture, Applied Microbiology and Biotechnology 104 (22) (2020) 9449–9485.

[215] Z.-K. Gao, M. Small, J. Kurths, Complex network analysis of time series, EPL (Europhysics Letters) 116 (5) (2017) 50001.

[216] D. Koschützki, F. Schreiber, Centrality analysis methods for biological networks and their application to gene regulatory networks, Gene regulation and systems biology 2 (2008) GRSB–S702.

[217] P. Bonacich, Some unique properties of eigenvector centrality, Social networks 29 (4) (2007) 555–564.

[218] L. C. Freeman, Centrality in social networks conceptual clarification, Social networks 1 (3) (1978) 215–239.

[219] M. Rubinov, O. Sporns, Complex network measures of brain connectivity: uses and interpretations, Neuroimage 52 (3) (2010) 1059–1069.

[220] M. E. Newman, M. Girvan, Finding and evaluating community structure in networks, Physical review E 69 (2) (2004) 026113.

[221] M. E. Newman, Modularity and community structure in networks, Proceedings of the national academy of sciences 103 (23) (2006) 8577–8582.

[222] A. Clauset, M. E. Newman, C. Moore, Finding community structure in very large networks, Physical review E 70 (6) (2004) 066111.

[223] S. Fortunato, Community detection in graphs, Physics reports 486 (3-5) (2010) 75–174.

[224] C. G. Antonopoulos, Dynamic range in the c. elegans brain network, Chaos: An Interdisciplinary Journal of Nonlinear Science 26 (1) (2016) 013102.

[225] B. W. Kernighan, S. Lin, An Efficient Heuristic Procedure for Partitioning Graphs, Bell System Technical Journal 49 (2) (1970) 291–307.

[226] E. Barnes, An algorithm for partitioning the nodes of a graph, in: Proceedings of the 1981 20th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes, IEEE, 1981, pp. 303–304.

[227] S. P. Lloyd, Least Squares Quantization in PCM, IEEE Transactions on Information Theory 28 (2) (1982) 129–137.

[228] J. Friedman, T. Hastie, R. Tibshirani, The elements of statistical learning, Vol. 1, Springer series in statistics New York, 2001.

[229] M. E. Newman, Fast algorithm for detecting community structure in networks, Physical review E 69 (6) (2004) 066133.

[230] R. Guimera, M. Sales-Pardo, L. A. N. Amaral, Modularity from fluctuations in random graphs and complex networks, Physical Review E 70 (2) (2004) 025101.

[231] J. Duch, A. Arenas, Community detection in complex networks using extremal optimization, Physical review E 72 (2) (2005) 027104.

[232] M. E. Newman, M. Girvan, Finding and evaluating community structure in networks, Physical review E 69 (2) (2004) 026113.

[233] W. E. Donath, A. J. Hoffman, Lower bounds for the partitioning of graphs, in: Selected Papers Of Alan J Hoffman: With Commentary, World Scientific, 2003, pp. 437–442.

[234] M. B. Hastings, Community detection as an inference problem, Physical Review E 74 (3) (2006) 035102.

[235] M. E. Newman, E. A. Leicht, Mixture models and exploratory analysis in networks, Proceedings of the National Academy of Sciences 104 (23) (2007) 9564–9569.

[236] T. Lozano-Pérez, M. A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, Communications of the ACM 22 (10) (1979) 560–570.

[237] L. Lacasa, B. Luque, F. Ballesteros, J. Luque, J. C. Nuño, From time series to complex networks: The visibility graph, Proceedings of the National Academy of Sciences 105 (13) (2008) 4972–4975.

[238] Y. Zou, R. Donner, N. Marwan, J. Donges, J. Kurths, Complex network approaches to nonlinear time series analysis, Physics Reports 787 (11 2018).

[239] B. Luque, L. Lacasa, F. Ballesteros, J. Luque, Horizontal visibility graphs: Exact results for random time series, Physical review E 80 (2009) 046103.

[240] L. Lacasa, R. Toral, Description of stochastic and chaotic series using visibility graphs, Physical review E 82 (3) (2010) 036120.

[241] F. M. Bianchi, L. Livi, C. Alippi, R. Jenssen, Multiplex visibility graphs to investigate recurrent neural network dynamics, Scientific Reports 7 (1) (2017) 1–13.

[242] F. Bonacina, E. S. Miele, A. Corsini, Time series clustering: A complex network-based approach for feature selection in multi-sensor data, Modelling 1 (1) (2020) 1–21.

[243] V. Sakkalis, Review of advanced techniques for the estimation of brain connectivity measured with eeg/meg, Computers in biology and medicine 41 (12) (2011) 1110–1117.

[244] T. M. Fruchterman, E. M. Reingold, Graph drawing by force-directed placement, Software: Practice and experience 21 (11) (1991) 1129–1164.

[245] C. Schmidt, B. Pester, N. Schmid-Hertel, H. Witte, A. Wismüller, L. Leistritz, A multivariate granger causality concept towards full brain functional connectivity, PloS one 11 (4) (2016) e0153105.

[246] J. M. Zurada, Introduction to artificial neural systems, Vol. 8, West publishing company St. Paul, 1992.

[247] M. Merenda, C. Porcaro, D. Iero, Edge machine learning for ai-enabled iot devices: A review, Sensors 20 (9) (2020) 2533.

[248] P. Werbos, Beyond regression : new fools for prediction and analysis in the behavioral sciences, PhD thesis, Harvard University (1974).
URL `https://ci.nii.ac.jp/naid/10010414900/en/`

[249] T. Théate, D. Ernst, An application of deep reinforcement learning to algorithmic trading, Expert Systems with Applications 173 (2021) 114632.

[250] S. Kullback, R. A. Leibler, On information and sufficiency, The annals of mathematical statistics 22 (1) (1951) 79–86.

[251] S. F. Zaman, Automated liver segmentation from mr-images using neural networks (2019).

[252] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprint arXiv:1609.04747 (2016).

[253] A. Lydia, S. Francis, Adagrad—an optimizer for stochastic gradient descent, Int. J. Inf. Comput. Sci 6 (5) (2019) 566–568.

[254] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[255] H. J. Kelley, Gradient theory of optimal flight paths, Ars Journal 30 (10) (1960) 947–954.

[256] S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6 (02) (1998) 107–116.

[257] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: International conference on machine learning, PMLR, 2013, pp. 1310–1318.

[258] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.

[259] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.

[260] Y. S. Abu-Mostafa, Learning from hints in neural networks, Journal of complexity 6 (2) (1990) 192–198.

[261] S. C. Suddarth, Y. Kergosien, Rule-injection hints as a means of improving network performance and learning time, in: European Association for Signal Processing Workshop, Springer, 1990, pp. 120–129.

[262] S. Ruder, An overview of multi-task learning in deep neural networks, arXiv preprint arXiv:1706.05098 (2017).

[263] H. Mohamed, A. Negm, M. Zahran, O. C. Saavedra, Assessment of artificial neural network for bathymetry estimation using high resolution satellite imagery in shallow lakes: case study el burullus lake, in: International water technology conference, 2015, pp. 12–14.

[264] S. Rajaram, P. Gupta, B. Andrassy, T. Runkler, Neural architectures for relation extraction within and across sentence boundaries in natural language text (2018).

[265] P. J. Werbos, et al., Backpropagation through time: what it does and how to do it, Proceedings of the IEEE 78 (10) (1990) 1550–1560.

[266] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, et al., Gradient flow in recurrent nets: the difficulty of learning long-term dependencies (2001).

[267] G. Hoffman, Introduction to lstms with tensorflow (2018).

[268] J. V. Tembhurne, T. Diwan, Sentiment analysis in textual, visual and multimodal inputs using recurrent neural networks, Multimedia Tools and Applications 80 (5) (2021) 6871–6910.

[269] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078 (2014).

[270] M.-T. Luong, H. Pham, C. D. Manning, Effective approaches to attention-based neural machine translation, arXiv preprint arXiv:1508.04025 (2015).

[271] D. Hu, An introductory survey on attention mechanisms in nlp problems, in: Proceedings of SAI Intelligent Systems Conference, Springer, 2019, pp. 432–448.

[272] Z. Mi, X. Jiang, T. Sun, K. Xu, Gan-generated image detection with self-attention mechanism against gan generator defect, IEEE Journal of Selected Topics in Signal Processing 14 (5) (2020) 969–981.

[273] K. O'Shea, R. Nash, An introduction to convolutional neural networks, arXiv preprint arXiv:1511.08458 (2015).

[274] D. Podareanu, V. Codreanu, T. Sandra Aigner, G. C. van Leeuwen, V. Weinberg, Best practice guide-deep learning, Partnership for Advanced Computing in Europe (PRACE), Tech. Rep 2 (2019).

[275] S. Albelwi, A. Mahmood, A framework for designing the architectures of deep convolutional neural networks, Entropy 19 (6) (2017) 242.

[276] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[277] M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in: International conference on machine learning, PMLR, 2019, pp. 6105–6114.

[278] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861 (2017).

[279] G. J. et. al., ultralytics/yolov5: v6.2 - yolov5 classification models, apple m1, reproducibility, clearml and deci.ai integrations (August 2022). doi:10.5281/zenodo.7002879.
URL https://doi.org/10.5281/zenodo.7002879

[280] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2980–2988.

[281] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, Advances in neural information processing systems 28 (2015).

[282] V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: A deep convolutional encoder-decoder architecture for image segmentation, IEEE transactions on pattern analysis and machine intelligence 39 (12) (2017) 2481–2495.

[283] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961–2969.

[284] B.-J. Singstad, C. Tronstad, Convolutional neural network and rule-based algorithms for classifying 12-lead ecgs, in: 2020 Computing in Cardiology, IEEE, 2020, pp. 1–4.

[285] A. Shenfield, M. Howarth, A novel deep learning model for the detection and identification of rolling element-bearing faults, Sensors 20 (18) (2020) 5112.

[286] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S. Y. Philip, A comprehensive survey on graph neural networks, IEEE transactions on neural networks and learning systems 32 (1) (2020) 4–24.

[287] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Advances in neural information processing systems 29 (2016).

[288] P. Baldi, Autoencoders, unsupervised learning, and deep architectures, in: Proceedings of ICML workshop on unsupervised and transfer learning, JMLR Workshop and Conference Proceedings, 2012, pp. 37–49.

[289] C. Kiourt, G. Pavlidis, S. Markantonatou, Deep learning approaches in food recognition, in: Machine Learning Paradigms, Springer, 2020, pp. 83–108.

[290] Y. Zhang, J. D. Lee, M. I. Jordan, l1-regularized neural networks are improperly learnable in polynomial time, in: International Conference on Machine Learning, PMLR, 2016, pp. 993–1001.

[291] T. Van Laarhoven, L2 regularization versus batch and weight normalization, arXiv preprint arXiv:1706.05350 (2017).

[292] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15 (1) (2014) 1929–1958.

[293] Z.-S. Wang, J. Lee, C. G. Song, S.-J. Kim, Efficient chaotic imperialist competitive algorithm with dropout strategy for global optimization, Symmetry 12 (4) (2020) 635.

[294] J. Hong, Y.-H. Kim, H. Nhung-Nguyen, J. Kwon, H. Lee, Deep-learning based fault events analysis in power systems, Energies 15 (15) (2022) 5539.

[295] M. Grandini, E. Bagli, G. Visani, Metrics for multi-class classification: an overview, arXiv preprint arXiv:2008.05756 (2020).

[296] H. Chen, D. S. Boning, Machine learning approaches for ic manufacturing yield enhancement, in: Machine Learning in VLSI Computer-Aided Design, Springer, 2019, pp. 175–199.

[297] S. Acid, L. M. De Campos, M. Fernández, Minimum redundancy maximum relevancy versus score-based methods for learning markov boundaries, in: Proceedings of the 2011 11th International Conference on Intelligent Systems Design and Applications, IEEE, 2011, pp. 619–623.

[298] C. E. Shannon, A mathematical theory of communication, Bell system technical journal 27 (3) (1948) 379–423.

[299] P. Refaeilzadeh, L. Tang, H. Liu, Cross-validation., Encyclopedia of database systems 5 (2009) 532–538.

[300] T. Fushiki, Estimation of prediction error by using k-fold cross-validation, Statistics and Computing 21 (2) (2011) 137–146.

[301] R. Maredia, Analysis of google play store data set and predict the popularity of an app on google play store (2020).

[302] T.-T. Wong, Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation, Pattern Recognition 48 (9) (2015) 2839–2846.

[303] T. G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural computation 10 (7) (1998) 1895–1923.

[304] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, IEEE Communications magazine 40 (8) (2002) 102–114.

[305] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, IEEE Communications magazine 40 (8) (2002) 102–114.

[306] A. Daneels, W. Salter, What is SCADA?, Conf. Proc. C 991004 (1999) 339–343.

[307] J. Frolik, M. Abdelrahman, Synthesis of quasi-redundant sensor data: a probabilistic approach, in: Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334), Vol. 4, IEEE, 2000, pp. 2917–2921.

[308] S. Chormunge, S. Jena, Correlation based feature selection with clustering for high dimensional data, Journal of Electrical Systems and Information Technology 5 (3) (2018) 542–549.

[309] E. S. Miele, F. Bonacina, A. Corsini, A. Peruch, M. Cannarozzo, D. Baldan, F. Pennisi, Unsupervised feature selection of multi-sensor scada data in horizontal axis wind turbine condition monitoring, in: Turbo Expo: Power for Land, Sea, and Air, Vol. 86137, American Society of Mechanical Engineers, 2022, p. V011T38A015.

[310] M. Mohammadi, A. Al-Fuqaha, S. Sorour, M. Guizani, Deep Learning for IoT Big Data and Streaming Analytics: A Survey., IEEE Communications Surveys & Tutorials 20 (2018) 2923–2960.

[311] P. Asghari, A. M. Rahmani, J. H. H. S., Internet of Things applications: A systematic review., Computer Networks 148 (2019) 241–261.

[312] D. Imkamp, J. Berthold, M. Heizmann, K. Kniel, E. Manske, M. Peterek, R. Schmitt, J. Seidler, K.-D. Sommer, Challenges and trends in manufacturing measurement technology-the" industrie 4.0" concept, Journal of Sensors and Sensor Systems 5 (2) (2016) 325.

[313] Y. Lu, Industry 4.0: A survey on technologies, applications and open research issues, Journal of Industrial Information Integration 6 (2017) 1–10.

[314] B. Hayes-Roth, R. Washington, R. Hewett, M. Hewett, A. Seiver, Intelligent monitoring and control., in: Proceedings of the IJCAI, Vol. 89, 1989, pp. 243–249.

[315] M. Verleysen, D. François, The curse of dimensionality in data mining and time series prediction, in: International Work-Conference on Artificial Neural Networks, Springer, 2005, pp. 758–770.

[316] V. Uraikul, C. W. Chan, P. Tontiwachwuthikul, Artificial intelligence for monitoring and supervisory control of process systems, Engineering applications of artificial intelligence 20 (2) (2007) 115–131.

[317] H.-X. Tian, X.-J. Liu, M. Han, An outliers detection method of time series data for soft sensor modeling, in: Proceedings of the 2016 Chinese Control and Decision Conference (CCDC), IEEE, 2016, pp. 3918–3922.

[318] J. Kaiser, Dealing with missing values in data, Journal of systems integration 5 (1) (2014) 42–51.

[319] R. Liu, B. Yang, E. Zio, X. Chen, Artificial intelligence for fault diagnosis of rotating machinery: A review, Mechanical Systems and Signal Processing 108 (2018) 33–47.

[320] J.-R. Ruiz-Sarmiento, J. Monroy, F.-A. Moreno, C. Galindo, J.-M. Bonelo, J. Gonzalez-Jimenez, A predictive model for the maintenance of industrial machinery in the context of industry 4.0, Engineering Applications of Artificial Intelligence 87 (2020) 103289.

[321] F. Ansari, R. Glawar, T. Nemeth, Prima: a prescriptive maintenance model for cyber-physical production systems, International Journal of Computer Integrated Manufacturing 32 (4-5) (2019) 482–503.

[322] X. Jin, J. Shao, X. Zhang, W. An, R. Malekian, Modeling of nonlinear system based on deep learning framework, Nonlinear Dynamics 84 (3) (2016) 1327–1340.

[323] D. You, X. Wu, L. Shen, Y. He, X. Yuan, Z. Chen, S. Deng, C. Ma, Online streaming feature selection via conditional independence, Applied Sciences 8 (12) (2018) 2548.

[324] S. K. Pal, P. Mitra, Pattern Recognition Algorithms for Data Mining: Scalability, Knowledge Discovery, and Soft Granular Computing, Chapman & Hall, Ltd., 2004.

[325] Q. Song, J. Ni, G. Wang, A fast clustering-based feature subset selection algorithm for high-dimensional data, IEEE transactions on knowledge and data engineering 25 (1) (2011) 1–14.

[326] L. D. Baker, A. K. McCallum, Distributional clustering of words for text classification, in: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, 1998, pp. 96–103.

[327] N. Slonim, N. Tishby, The power of word clusters for text classification, in: Proceedings of the 23rd European Colloquium on Information Retrieval Research, Vol. 1, 2001, p. 200.

[328] M. Zanin, D. Papo, P. A. Sousa, E. Menasalvas, A. Nicchi, E. Kubik, S. Boccaletti, Combining complex networks and data mining: why and how, Physics Reports 635 (2016) 1–44.

[329] L. Ferreira, L. Zhao, Time series clustering via community detection in networks, Information Sciences 326 (08 2015).

[330] X. Zhang, J. Liu, Y. Du, T. Lu, A novel clustering method on time series data, Expert Systems with Applications 38 (2011) 11891–11900.

[331] D. Koschützki, F. Schreiber, Centrality analysis methods for biological networks and their application to gene regulatory networks, Gene regulation and systems biology 2 (2008) GRSB–S702.

[332] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of statistical mechanics: theory and experiment 2008 (10) (2008) P10008.

[333] B. Luque, L. Lacasa, Canonical horizontal visibility graphs are uniquely determined by their degree sequence, The European Physical Journal Special Topics 226 (05 2016).

[334] A. Corsini, F. Bonacina, S. Feudo, A. Marchegiani, P. Venturini, Internal combustion engine sensor network analysis using graph modeling, Energy Procedia 126 (2017) 907–914.

[335] G. Van Rossum, F. L. Drake Jr, Python reference manual, Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[336] T. E. Oliphant, A guide to NumPy, Vol. 1, Trelgol Publishing USA, 2006.

[337] A. Hagberg, P. Swart, D. Chult, Exploring network structure, dynamics, and function using networkx, in: Proceedings of the 7th Python in Science Conference, 2008, pp. 11–15.

[338] X. Wen, M. Xie, Performance evaluation of wind turbines based on scada data, Wind Engineering (2020) 0309524X20968935.

[339] H. Chen, C. Xie, J. Dai, E. Cen, J. Li, Scada data-based working condition classification for condition assessment of wind turbine main transmission system, Energies 14 (21) (2021) 7043.

[340] J. M. G. Sopeña, V. Pakrashi, B. Ghosh, Can we improve short-term wind power forecasts using turbine-level data? a case study in ireland, in: 2021 IEEE Madrid PowerTech, IEEE, 2021, pp. 1–6.

[341] X. Jin, Z. Xu, W. Qiao, Condition monitoring of wind turbine generators using scada data analysis, IEEE Transactions on Sustainable Energy 12 (1) (2020) 202–210.

[342] H. Huang, R. Jia, X. Shi, J. Liang, J. Dang, Feature selection and hyper parameters optimization for short-term wind power forecast, Applied Intelligence (2021) 1–19.

[343] H. Zheng, Y. Wu, A xgboost model with weather similarity analysis and feature engineering for short-term wind power forecasting, Applied Sciences 9 (15) (2019) 3019.

[344] S. Li, P. Wang, L. Goel, Wind power forecasting using neural network ensembles with feature selection, IEEE Transactions on sustainable energy 6 (4) (2015) 1447–1456.

[345] D. Astolfi, F. Castellani, A. Lombardi, L. Terzi, Multivariate scada data analysis methods for real-world wind turbine power curve monitoring, Energies 14 (4) (2021) 1105.

[346] L. Qin, Y. Xiong, K. Liu, Weather division-based wind power forecasting model with feature selection, IET Renewable Power Generation 13 (16) (2019) 3050–3060.

[347] N. Botha, C. M. van der Walt, Forecasting wind speed using support vector regression and feature selection, in: 2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech), IEEE, 2017, pp. 181–186.

[348] F. Wetschoreck, T. Krabel, S. Krishnamurthy, 8080labs/ppscore: zenodo release (Oct. 2020). doi:10.5281/zenodo.4091345.
URL https://doi.org/10.5281/zenodo.4091345

[349] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[350] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, et al., Top 10 algorithms in data mining, Knowledge and information systems 14 (1) (2008) 1–37.

[351] S. Haykin, Neural networks: a comprehensive foundation, Prentice Hall PTR, 1994.

[352] R. S. Peres, A. D. Rocha, P. Leitao, J. Barata, Idarts–towards intelligent data analysis and real-time supervision for industry 4.0, Computers in industry 101 (2018) 138–146.

[353] E. Sezer, D. Romero, F. Guedea, M. Macchi, C. Emmanouilidis, An industry 4.0-enabled low cost predictive maintenance approach for smes, in: 2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), IEEE, 2018, pp. 1–8.

[354] S. Biswal, G. Sabareesh, Design and development of a wind turbine test rig for condition monitoring studies, in: 2015 international conference on industrial instrumentation and control (icic), IEEE, 2015, pp. 891–896.

[355] T. P. Carvalho, F. A. Soares, R. Vita, R. d. P. Francisco, J. P. Basto, S. G. Alcalá, A systematic literature review of machine learning methods applied to predictive maintenance, Computers & Industrial Engineering 137 (2019) 106024.

[356] T. Zonta, C. A. Da Costa, R. da Rosa Righi, M. J. de Lima, E. S. da Trindade, G. P. Li, Predictive maintenance in the industry 4.0: A systematic literature review, Computers & Industrial Engineering 150 (2020) 106889.

[357] A. Jezzini, M. Ayache, L. Elkhansa, B. Makki, M. Zein, Effects of predictive maintenance (pdm), proactive maintenace (pom) & preventive maintenance (pm) on minimizing the faults in medical instruments, in: 2013 2nd International conference on advances in biomedical engineering, IEEE, 2013, pp. 53–56.

[358] A. Kumar, R. B. Chinnam, F. Tseng, An hmm and polynomial regression based approach for remaining useful life and health state estimation of cutting tools, Computers & Industrial Engineering 128 (2019) 1008–1014.

[359] P. Kamat, R. Sugandhi, Anomaly detection for predictive maintenance in industry 4.0-a survey, in: E3S Web of Conferences, Vol. 170, EDP Sciences, 2020, p. 02007.

[360] S. Wang, K. Wang, Z. Li, A review on data-driven predictive maintenance approach for hydro turbines/generators, in: 6th International Workshop of Advanced Manufacturing and Automation, Atlantis Press, 2016, pp. 30–35.

[361] E. Arena, A. Corsini, R. Ferulano, D. A. Iuvara, E. S. Miele, L. Ricciardi Celsi, N. A. Sulieman, M. Villari, Anomaly detection in photovoltaic production factories via monte carlo pre-processed principal component analysis, Energies 14 (13) (2021) 3951.

[362] E. S. Miele, F. Bonacina, A. Corsini, Deep anomaly detection in horizontal axis wind turbines using graph convolutional autoencoders for multivariate time series, Energy and AI 8 (2022) 100145.

[363] H. Wang, G. Ni, J. Chen, J. Qu, Research on rolling bearing state health monitoring and life prediction based on pca and internet of things with multi-sensor, Measurement 157 (2020) 107657.

[364] D. Kimera, F. N. Nangolo, Improving ship yard ballast pumps' operations: A pca approach to predictive maintenance, Maritime Transport Research 1 (2020) 100003.

[365] X. Chen, B. Zhang, T. Wang, A. Bonni, G. Zhao, Robust principal component analysis for accurate outlier sample detection in rna-seq data, BMC bioinformatics 21 (1) (2020) 1–20.

[366] S. Kim, J. Hur, A probabilistic modeling based on monte carlo simulation of wind powered ev charging stations for steady-states security analysis, Energies 13 (20) (2020) 5260.

[367] J. E. Yoo, M. Rho, Large-scale survey data analysis with penalized regression: A monte carlo simulation on missing categorical predictors, Multivariate Behavioral Research 57 (4) (2022) 642–657.

[368] M. De Benedetti, F. Leonardi, F. Messina, C. Santoro, A. Vasilakos, Anomaly detection and predictive maintenance for photovoltaic systems, Neurocomputing 310 (2018) 59–68.

[369] N. Bashir, D. Chen, D. Irwin, P. S.-T. Shenoy, A data-driven toolkit for solar pv performance modeling and forecasting, in: Proceedings of the 2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Monterey, CA, USA, 2019, pp. 4–7.

[370] F. Bonacina, A. Corsini, L. Cardillo, F. Lucchetta, Complex network analysis of photovoltaic plant operations and failure modes, Energies 12 (10) (2019) 1995.

[371] F. Zhou, J. H. Park, Y. Liu, Differential feature based hierarchical pca fault detection method for dynamic fault, Neurocomputing 202 (2016) 27–35.

[372] Z. Chen, X. Li, C. Yang, T. Peng, C. Yang, H. Karimi, W. Gui, A data-driven ground fault detection and isolation method for main circuit in railway electrical traction system, ISA transactions 87 (2019) 264–271.

[373] M.-F. Harkat, A. Kouadri, R. Fezai, M. Mansouri, H. Nounou, M. Nounou, Machine learning-based reduced kernel pca model for nonlinear chemical process monitoring, Journal of Control, Automation and Electrical Systems 31 (5) (2020) 1196–1209.

[374] F. Bencheikh, M. Harkat, A. Kouadri, A. Bensmail, New reduced kernel pca for fault detection and diagnosis in cement rotary kiln, Chemometrics and Intelligent Laboratory Systems 204 (2020) 104091.

[375] W. Niemeier, D. Tengen, Stochastic properties of confidence ellipsoids after least squares adjustment, derived from gum analysis and monte carlo simulations, Mathematics 8 (8) (2020) 1318.

[376] M. L. Zelditch, D. L. Swiderski, H. D. Sheets, Geometric morphometrics for biologists: a primer, academic press, 2012.

[377] X. Fang, W. Zeng, Y. Zhou, B. Wang, On the total least median of squares adjustment for the pattern recognition in point clouds, Measurement 160 (2020) 107794.

[378] G. Leger, Combining adaptive alternate test and multi-site, in: 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2015, pp. 1389–1394.

[379] S. Parashar, A. Swarnkar, K. Niazi, N. Gupta, Optimal integration of electric vehicles and energy management of grid connected microgrid, in: 2017 IEEE Transportation Electrification Conference (ITEC-India), IEEE, 2017, pp. 1–5.

[380] T. M. L. Fernando, L. G. E. Marcelo, V. M. H. David, Substation distribution reliability assessment using network reduction and montecarlo method, a comparison, in: 2019 FISE-IEEE/CIGRE Conference-Living the energy Transition (FISE/CIGRE), IEEE, 2019, pp. 1–7.

[381] X. Kong, X. Tong, Monte-carlo tree search for graph coalition structure generation, in: 2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), IEEE, 2020, pp. 1058–1063.

[382] P. Saracco, M. Batic, G. Hoff, M. Pia, Uncertainty quantification (uq) in generic montecarlo simulations, in: 2012 IEEE Nuclear Science Symposium and Medical Imaging Conference Record (NSS/MIC), IEEE, 2012, pp. 651–656.

[383] H. García-Alfonso, D.-M. Córdova-Esparza, Comparison of uncertainty analysis of the montecarlo and latin hypercube algorithms in a camera calibration model, in: 2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA), IEEE, 2018, pp. 1–5.

[384] Y. Chen, R. Sun, J. Borken-Kleefeld, On-road no x and smoke emissions of diesel light commercial vehicles–combining remote sensing measurements from across europe, Environmental Science & Technology 54 (19) (2020) 11744–11752.

[385] R. Heijungs, On the number of monte carlo runs in comparative probabilistic lca, The International Journal of Life Cycle Assessment 25 (2) (2020) 394–402.

[386] P. Nair, I. Kashyap, Hybrid pre-processing technique for handling imbalanced data and detecting outliers for knn classifier, in: 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), IEEE, 2019, pp. 460–464.

[387] D. Zwillinger, S. Kokoska, CRC standard probability and statistics tables and formulae, Crc Press, 1999.

[388] H. Ji, Y. Li, Monte carlo methods and their applications in big data analysis, in: Mathematical Problems in Data Science, Springer, 2015, pp. 125–139.

[389] C. von Brömssen, E. Röös, Why statistical testing and confidence intervals should not be used in comparative life cycle assessments based on monte carlo simulations, The International Journal of Life Cycle Assessment 25 (11) (2020) 2101–2105.

[390] F. Dongxiao, P. Chuan, Z. Guoxing, Z. Rui, L. Fang, D. Zhenhua, M. Hongliang, Research on simulation method for reliability prediction of pyrotechnical system, in: 2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan), IEEE, 2020, pp. 556–559.

[391] B. Meuleman, J. Billiet, A monte carlo sample size study: How many countries are needed for accurate multilevel sem?, in: Survey Research Methods, 2009, pp. 45–58.

[392] S. Fadhel, C. Delpha, D. Diallo, I. Bahri, A. Migan, M. Trabelsi, M. F. Mimouni, Pv shading fault detection and classification based on iv curve using principal component analysis: Application to isolated pv system, Solar Energy 179 (2019) 1–10.

[393] V. Aggarwal, V. Gupta, P. Singh, K. Sharma, N. Sharma, Detection of spatial outlier by using improved z-score test, in: 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), IEEE, 2019, pp. 788–790.

[394] H. Vinutha, B. Poornima, B. Sagar, Detection of outliers using interquartile range technique from intrusion dataset, in: Information and decision sciences, Springer, 2018, pp. 511–518.

[395] A. R. T. Donders, G. J. Van Der Heijden, T. Stijnen, K. G. Moons, A gentle introduction to imputation of missing values, Journal of clinical epidemiology 59 (10) (2006) 1087–1091.

[396] R. Malarvizhi, A. S. Thanamani, K-nearest neighbor in missing data imputation, International Journal of Engineering Research and Development 5 (1) (2012) 5–7.

[397] N. M. Noor, M. M. Al Bakri Abdullah, A. S. Yahaya, N. A. Ramli, Comparison of linear interpolation method and mean method to replace the missing values in environmental data set, in: Materials Science Forum, Vol. 803, Trans Tech Publ, 2015, pp. 278–281.

[398] M. Parzinger, L. Hanfstaengl, F. Sigg, U. Spindler, U. Wellisch, M. Wirnsberger, Residual analysis of predictive modelling data for automated fault detection in building's heating, ventilation and air conditioning systems, Sustainability 12 (17) (2020) 6758.

[399] M. Liserre, R. Cardenas, M. Molinas, J. Rodriguez, Overview of multi-mw wind turbines and wind parks, IEEE Transactions on Industrial Electronics 58 (4) (2011) 1081–1095.

[400] K. T. Fung, R. L. Scheffler, J. Stolpe, Wind energy - a utility perspective, IEEE Transactions on Power Apparatus and Systems PAS-100 (3) (1981) 1176–1182.

[401] E. Sesto, C. Casale, Exploitation of wind as an energy source to meet the world's electricity demand, Journal of Wind Engineering and Industrial Aerodynamics 74 (1998) 375–387.

[402] H.-M. Wee, W.-H. Yang, C.-W. Chou, M. V. Padilan, Renewable energy supply chains, performance, application barriers, and strategies for further development, Renewable and Sustainable Energy Reviews 16 (8) (2012) 5451–5465.

[403] Z. Hameed, Y. Hong, Y. Cho, S. Ahn, C. Song, Condition monitoring and fault detection of wind turbines and related algorithms: A review, Renewable and Sustainable energy reviews 13 (1) (2009) 1–39.

[404] K. Kim, G. Parthasarathy, O. Uluyol, W. Foslien, S. Sheng, P. Fleming, Use of scada data for failure detection in wind turbines, in: Energy Sustainability, Vol. 54686, 2011, pp. 2071–2079.

[405] C. Dao, B. Kazemtabrizi, C. Crabtree, Wind turbine reliability data review and impacts on levelised cost of energy, Wind Energy 22 (12) (2019) 1848–1871.

[406] A. Zaher, S. McArthur, D. Infield, Y. Patel, Online wind turbine fault detection through automated scada data analysis, Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology 12 (6) (2009) 574–593.

[407] A. Lebranchu, S. Charbonnier, C. Bérenguer, F. Prévost, A combined mono- and multi-turbine approach for fault indicator synthesis and wind turbine monitoring using scada data, ISA transactions 87 (2019) 272–281.

[408] D. Menezes, M. Mendes, J. A. Almeida, T. Farinha, Wind farm and resource datasets: A comprehensive survey and overview, Energies 13 (18) (2020) 4702.

[409] M. Ulmer, E. Jarlskog, G. Pizza, J. Manninen, L. Goren Huber, Early fault detection based on wind turbine scada data using convolutional neural networks, in: 5th European Conference of the Prognostics and Health Management Society, Virtual Conference, 27-31 July 2020, Vol. 5, PHM Society, 2020, p. 9.

[410] Y. Cui, P. Bangalore, L. Bertling Tjernberg, A fault detection framework using recurrent neural networks for condition monitoring of wind turbines, Wind Energy (2021).

[411] G. Pang, C. Shen, L. Cao, A. V. D. Hengel, Deep learning for anomaly detection: A review, ACM Computing Surveys (CSUR) 54 (2) (2021) 1–38.

[412] Y. Pang, Q. He, G. Jiang, P. Xie, Spatio-temporal fusion neural network for multi-class fault diagnosis of wind turbines based on scada data, Renewable Energy 161 (2020) 510–524.

[413] Z. Kong, B. Tang, L. Deng, W. Liu, Y. Han, Condition monitoring of wind turbines based on spatio-temporal fusion of scada data by convolutional neural networks and gated recurrent units, Renewable Energy 146 (2020) 760–768.

[414] L. Xiang, P. Wang, X. Yang, A. Hu, H. Su, Fault detection of wind turbine based on scada data analysis using cnn and lstm with attention mechanism, Measurement 175 (2021) 109094.

[415] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI'18, AAAI Press, 2018, pp. 3634–3640.

[416] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, C. Zhang, Connecting the dots: Multivariate time series forecasting with graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 753–763.

[417] N. Renström, P. Bangalore, E. Highcock, System-wide anomaly detection in wind turbines using deep autoencoders, Renewable Energy 157 (2020) 647–659.

[418] H. Chen, H. Liu, X. Chu, Q. Liu, D. Xue, Anomaly detection and critical scada parameters identification for wind turbines based on lstm-ae neural network, Renewable Energy 172 (2021) 829–840.

[419] R. Chalapathy, S. Chawla, Deep learning for anomaly detection: A survey, arXiv preprint arXiv:1901.03407 (2019).

[420] J. J. Q. Yu, J. Gu, Real-time traffic speed estimation with graph convolutional generative autoencoder, IEEE Transactions on Intelligent Transportation Systems 20 (10) (2019) 3940–3951.

[421] Y. Hu, A. Qu, D. Work, Graph convolutional networks for traffic anomaly, arXiv preprint arXiv:2012.13637 (2020).

[422] X. Yan, T. Ai, M. Yang, X. Tong, Graph convolutional autoencoder model for the shape coding and cognition of buildings in maps, International Journal of Geographical Information Science (2020) 1–23.

[423] S. Zhang, H. Tong, J. Xu, R. Maciejewski, Graph convolutional networks: a comprehensive review, Computational Social Networks 6 (1) (2019) 1–23.

[424] G. Li, M. Muller, A. Thabet, B. Ghanem, Deepgcns: Can gcns go as deep as cnns?, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 9267–9276.

[425] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S. Y. Philip, A comprehensive survey on graph neural networks, IEEE transactions on neural networks and learning systems (2020).

[426] EDP, Wind farm 1 - failures, data retrieved from EDP Open Data, `https://opendata.edp.com/explore/dataset/htw-failures-2016/information/` (2016).

[427] B. C. Ross, Mutual information between discrete and continuous data sets, PloS one 9 (2) (2014) e87357.

[428] R. Arora, A. Basu, P. Mianjy, A. Mukherjee, Understanding deep neural networks with rectified linear units, arXiv preprint arXiv:1611.01491 (2016).

[429] A. Savitzky, M. J. Golay, Smoothing and differentiation of data by simplified least squares procedures., Analytical chemistry 36 (8) (1964) 1627–1639.

[430] L. Prechelt, Early stopping-but when?, in: Neural Networks: Tricks of the trade, Springer, 1998, pp. 55–69.

[431] P. H. Madsen, D. Risø, Introduction to the iec 61400-1 standard, Risø National Laboratory, Technical University of Denmark (2008).

[432] M. Wilkinson, K. Harman, B. Hendriks, F. Spinato, T. van Delft, G. Garrad, U. Thomas, Measuring wind turbine reliability-results of the reliawind project, in: EWEA Conference, 2011, pp. 1–8.

[433] M. Sahnoun, F. Bagui, M. Messaadia, Failure analysis of onshore wind farms based on experimental data, in: Mediterranean Conference on Information & Communication Technologies' 2015, 2015, pp. –.

[434] H. Wang, Z. Lei, X. Zhang, B. Zhou, J. Peng, A review of deep learning for renewable energy forecasting, Energy Conversion and Management 198 (2019) 111799.

[435] Q. Wang, H. Wu, A. R. Florita, C. B. Martinez-Anido, B.-M. Hodge, The value of improved wind power forecasting: Grid flexibility quantification, ramp capability analysis, and impacts of electricity market operation timescales, Applied Energy 184 (2016) 696–713.

[436] Z. Shen, M. Ritter, Forecasting volatility of wind power production, Applied energy 176 (2016) 295–308.

[437] L. Jebaraj, C. Venkatesan, I. Soubache, C. C. A. Rajan, Application of differential evolution algorithm in static and dynamic economic or emission dispatch problem: A review, Renewable and Sustainable Energy Reviews 77 (2017) 1206–1220.

[438] M. Khalid, R. P. Aguilera, A. V. Savkin, V. G. Agelidis, On maximizing profit of wind-battery supported power station based on wind power and energy price forecasting, Applied Energy 211 (2018) 764–773.

[439] S. A. Arefifar, Y. A.-R. I. Mohamed, T. H. El-Fouly, Supply-adequacy-based optimal construction of microgrids in smart distribution systems, IEEE transactions on smart grid 3 (3) (2012) 1491–1502.

[440] J. Ma, X. Ma, A review of forecasting algorithms and energy management strategies for microgrids, Systems Science & Control Engineering 6 (1) (2018) 237–248.

[441] K. Baker, G. Hug, X. Li, Energy storage sizing taking into account forecast uncertainties and receding horizon operation, IEEE Transactions on Sustainable Energy 8 (1) (2016) 331–340.

[442] C. Matke, D. Bienstock, G. Munoz, S. Yang, D. Kleinhans, S. Sager, Robust optimization of power network operation: storage devices and the role of forecast errors in renewable energies, in: International Workshop on Complex Networks and their Applications, Springer, 2016, pp. 809–820.

[443] S. Mazzola, C. Vergara, M. Astolfi, V. Li, I. Perez-Arriaga, E. Macchi, Assessing the value of forecast-based dispatch in the operation of off-grid rural microgrids, Renewable Energy 108 (2017) 116–125.

[444] P. Haessig, B. Multon, H. B. Ahmed, S. Lascaud, P. Bondon, Energy storage sizing for wind power: impact of the autocorrelation of day-ahead forecast errors, Wind Energy 18 (1) (2015) 43–57.

[445] F. Nieuwenhout, A. Brand, The impact of wind power on day-ahead electricity prices in the netherlands, in: 2011 8th International Conference on the European Energy Market (EEM), IEEE, 2011, pp. 226–230.

[446] Q. Wang, C. B. Martinez-Anido, H. Wu, A. R. Florita, B.-M. Hodge, Quantifying the economic and grid reliability impacts of improved wind power forecasting, IEEE Transactions on sustainable energy 7 (4) (2016) 1525–1537.

[447] A. Kaur, L. Nonnenmacher, H. T. Pedro, C. F. Coimbra, Benefits of solar forecasting for energy imbalance markets, Renewable energy 86 (2016) 819–830.

[448] L. Exizidis, J. Kazempour, P. Pinson, Z. De Grève, F. Vallée, Impact of public aggregate wind forecasts on electricity market outcomes, IEEE Transactions on Sustainable Energy 8 (4) (2017) 1394–1405.

[449] D. G. Caglayan, D. S. Ryberg, H. Heinrichs, J. Linßen, D. Stolten, M. Robinius, The techno-economic potential of offshore wind energy with optimized future turbine designs in europe, Applied energy 255 (2019) 113794.

[450] R. Belmans, Integration of large-scale renewable energy into bulk power systems: From planning to operation., Economics of Energy & Environmental Policy 8 (2) (2019) 201–203.

[451] G. Notton, M.-L. Nivet, C. Voyant, C. Paoli, C. Darras, F. Motte, A. Fouilloy, Intermittent and stochastic character of renewable energy sources: Consequences, cost of intermittence and benefit of forecasting, Renewable and sustainable energy reviews 87 (2018) 96–105.

[452] M. U. Yousuf, I. Al-Bahadly, E. Avci, Current perspective on the accuracy of deterministic wind speed and power forecasting, IEEE Access 7 (2019) 159547–159564.

[453] D. Lahat, T. Adali, C. Jutten, Multimodal data fusion: an overview of methods, challenges, and prospects, Proceedings of the IEEE 103 (9) (2015) 1449–1477.

[454] L. Boussioux, C. Zeng, T. Guénais, D. Bertsimas, Hurricane forecasting: A novel multimodal machine learning framework, Weather and Forecasting (2022).

[455] S. Yang, H. Wei, L. Zhang, S. Qin, Daily power generation forecasting method for a group of small hydropower stations considering the spatial and temporal distribution of precipitation—south china case study, Energies 14 (15) (2021) 4387.

[456] D. Haputhanthri, D. De Silva, S. Sierla, D. Alahakoon, R. Nawaratne, A. Jennings, V. Vyatkin, Solar irradiance nowcasting for virtual power plants using multimodal long short-term memory networks, Frontiers in Energy Research (2021) 469.

[457] P. Du, Ensemble machine learning-based wind forecasting to combine nwp output with data from weather station, IEEE Transactions on Sustainable Energy 10 (4) (2018) 2133–2141.

[458] Engie, The la haute borne wind farm, `https://opendata-renewables.engie.com/` (2018).

[459] L. Donadio, J. Fang, F. Porté-Agel, Numerical weather prediction and artificial neural network coupling for wind energy forecast, Energies 14 (2) (2021) 338.

[460] D. Zheng, M. Shi, Y. Wang, A. T. Eseye, J. Zhang, Day-ahead wind power forecasting using a two-stage hybrid modeling approach based on scada and meteorological information, and evaluating the impact of input-data dependency on forecasting accuracy, Energies 10 (12) (2017) 1988.

[461] ECMWF, Atmospheric model high resolution 10-day forecast (set i - hres), `https://www.ecmwf.int/en/forecasts/datasets/set-i` (2022).

[462] J. Litten, Applying sigma metrics to reduce outliers, Clinics in Laboratory Medicine 37 (1) (2017) 177–186.

[463] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

[464] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, W.-c. Woo, Convolutional lstm network: A machine learning approach for precipitation nowcasting, Advances in neural information processing systems 28 (2015).

[465] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, New York, NY, USA, 2016, pp. 785–794. `doi:10.1145/2939672.2939785`.
URL `http://doi.acm.org/10.1145/2939672.2939785`

[466] Q.-T. Phan, Y.-K. Wu, Q.-D. Phan, A comparative analysis of xgboost and temporal convolutional network models for wind power forecasting, in: 2020 International Symposium on Computer, Consumer and Control (IS3C), IEEE, 2020, pp. 416–419.

[467] R. Cai, S. Xie, B. Wang, R. Yang, D. Xu, Y. He, Wind speed forecasting based on extreme gradient boosting, IEEE Access 8 (2020) 175063–175069.

[468] Q. T. Phan, Y. K. Wu, Q. D. Phan, A hybrid wind power forecasting model with xgboost, data preprocessing considering different nwps, Applied Sciences 11 (3) (2021) 1100.

[469] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, W.-c. Woo, Convolutional lstm network: A machine learning approach for precipitation nowcasting, Advances in neural information processing systems 28 (2015).

[470] S. Alessandrini, S. Sperati, P. Pinson, A comparison between the ecmwf and cosmo ensemble prediction systems applied to short-term wind power forecasting on real data, Applied energy 107 (2013) 271–280.

[471] B. Kosovic, S. E. Haupt, D. Adriaansen, S. Alessandrini, G. Wiener, L. Delle Monache, Y. Liu, S. Linden, T. Jensen, W. Cheng, et al., A comprehensive wind power forecasting system integrating artificial intelligence and numerical weather prediction, Energies 13 (6) (2020) 1372.

[472] W. Dai, H. Nishi, V. Vyatkin, V. Huang, Y. Shi, X. Guan, Industrial edge computing: Enabling embedded intelligence, IEEE Industrial Electronics Magazine 13 (4) (2019) 48–56.

[473] J. Zhang, B. Chen, Y. Zhao, X. Cheng, F. Hu, Data security and privacy-preserving in edge computing paradigm: Survey and open issues, IEEE access 6 (2018) 18209–18237.

[474] M. D. de Assuncao, A. da Silva Veith, R. Buyya, Distributed data stream processing and edge computing: A survey on resource elasticity and future directions, Journal of Network and Computer Applications 103 (2018) 1–17.