data

MDPI

Review

Machine Learning-Based Algorithms to Knowledge Extraction from Time Series Data: A Review

Giuseppe Ciaburro * and Gino Iannace

Department of Architecture and Industrial Design, Università degli Studi della Campania, Luigi Vanvitelli, Borgo San Lorenzo, 81031 Aversa, Italy; gino.iannace@unicampania.it

* Correspondence: giuseppe.ciaburro@unicampania.it; Tel.: +39-0818122530

Abstract: To predict the future behavior of a system, we can exploit the information collected in the past, trying to identify recurring structures in what happened to predict what could happen, if the same structures repeat themselves in the future as well. A time series represents a time sequence of numerical values observed in the past at a measurable variable. The values are sampled at equidistant time intervals, according to an appropriate granular frequency, such as the day, week, or month, and measured according to physical units of measurement. In machine learning-based algorithms, the information underlying the knowledge is extracted from the data themselves, which are explored and analyzed in search of recurring patterns or to discover hidden causal associations or relationships. The prediction model extracts knowledge through an inductive process: the input is the data and, possibly, a first example of the expected output, the machine will then learn the algorithm to follow to obtain the same result. This paper reviews the most recent work that has used machine learning-based techniques to extract knowledge from time series data.

Keywords: time series data; machine learning; classification; regression; review

Citation: Ciaburro, G.; Iannace, G. Machine Learning-Based Algorithms to Knowledge Extraction from Time Series Data: A Review. *Data* **2021**, *6*, 55. https://doi.org/10.3390/data6060055

Academic Editor: Joaquín Torres-Sospedra

Received: 4 May 2021
Accepted: 22 May 2021
Published: 25 May 2021

1. Introduction

Time series are used in various fields of application to understand and analyze the evolution of a phenomenon over time [1,2]. A characteristic of these data is the dependence between successive observations, which we try to capture using suitable models. The need to understand past characteristics translates into the much more difficult phase of forecasting the possible future trend of the studied series. The most obvious purpose of the forecast is to try to dominate randomness and uncertainty to exploit them to one's advantage. This task is particularly difficult because the analysis is carried out on very unpredictable data [3,4].

In this context, the measurement of the phenomena that evolve over time is carried out with the aim of observing, and possibly predicting, the future dynamics of the phenomenon under study [5]. One of the objectives of the time series analysis is to identify the components that determine the evolution of the phenomena of time. A time series can be defined as a succession of numerical data in which each datum is associated with a particular instant or time interval. If only one phenomenon is observed at each point or time interval, the series is called univariate. If the observed variables are more than one, the series is called multiple or multivariate [6,7].

The first step in the study of temporal data is to understand the link between the variable and time: the values are realized with the passage of time intervals and these are therefore the units on which the variable of interest is observed [8]. In a time series, the order of the data is substantial and not accidental as in the case of a random sample of observations, and the recognition of this order is one of the most important characteristics of the series itself. It can be assumed that there is a kind of dependence between successive observations and that it is linked to the position of the observation in the sequence: the data detected at a given instant t are more like those detected at instant $t-1$ rather than

in epochs distant [9]. The study of this dependence and the consequent possibility of exploiting it for forecasting purposes represent the heart of the analysis of the time series. The time parameter, which defines the ordering of the data, belongs to a parametric set that can be continuous or discrete. The timing with which to observe a phenomenon depends on the nature of the phenomenon itself. The most used time intervals are days, weeks, months, quarters, and years [10,11].

Most of the time series is, however, of the stochastic type, in the sense that the future is determined by the past only partially, thus making it impossible to draw up completely error-free forecasts [12,13]. A stochastic process is a family of random variables ordered with respect to a parameter. Being a finite set of observations relating to a certain phenomenon, ordered in time, a time series can therefore be a finite realization of a stochastic process. The objective of the time series analysis is therefore to use the observed data to identify the generating stochastic process to be used to make future predictions. Stationary stochastic processes enjoy probabilistic properties, which are useful for modeling many time series encountered in practice [14,15].

In real cases, the trend of a phenomenon over time is probabilistic, that is, its future movement cannot be predicted without errors, unlike what happens for deterministic temporal trends [16,17]. The study of time series can highlight some recurring behaviors so that the process can be decomposed into random parts and deterministic parts with which to elaborate a future forecast [18–20]. The problem of prediction without a priori knowledge of a model of the system requires suitable investigation methods that identify the structures and characteristics underlying the time series. Methods that capture these constructs inductively are those based on machine learning [21–23]. Machine learning is a branch of artificial intelligence that studies algorithms capable of improving their performance by refining knowledge and increasing the notions learned through experience. At the basis of the technique lies the belief that even a machine can constantly learn and improve its performance during construction by drawing knowledge from experimental data and analytical observation of the facts [24,25].

Machine learning distorts the traditional paradigm that assumes an output from the input data and an algorithm that explains how to use it. In new systems, however, knowledge is an inductive process: the input is the data and, possibly, a first example of the expected output, so it will be the machine that will learn the algorithm to follow to obtain the same result [26,27]. The information underlying the knowledge is literally extracted from the data themselves, which are explored and analyzed with data mining techniques in search of recurring patterns or to discover hidden causal associations or relationships [28]. Machine learning techniques can be used for classification through supervised learning: the computer processes a rule for assigning the input data classes, after training the model with an already labeled training set [29–31].

In this work we will analyze and describe methods based on machine learning for the extraction of knowledge from times series data. The paper is structured as follows: Section 2 describes the times series data in detail, analyzing its characteristics and peculiarities that make mathematical modeling extremely complex. Section 3 analyzes the most popular methods of analysis of the Stoic series based on machine learning. Finally, Section 4 summarizes the results obtained in applying these methods to real cases, highlighting their advantages, and listing their limits.

2. Times Series Analysis

Often the detected values constitute, from a statistical point of view, the observations of a random sample of independent random variables. The analysis of the time series is based on the nature of the dependence between the members of the sequence. This approach modifies the treatment of many phenomena as randomly distributed over time around a substantially stable average. The study of time series must be centered on the concept of memory or persistence or hysteresis [32,33].

2.1. Time Series Background

The tool we use to model the observed time series is the stochastic process. An intuitive definition of a stochastic process sees it as an infinitely long sequence of random variables, or a random vector of infinite size [34,35]. A sample of t consecutive observations over time is therefore not thought so much as a realization of t distinct random variables, but rather as part of a single realization of a stochastic process, whose memory is given by the degree of connection between the random variables that make it up [36,37].

The classical approach to time series assumes that the process includes a deterministic part and a random disturbance component [38,39]. We can therefore represent the data of a time series as the sum of two contributions as follows:

$$Y_t = f(t) + w(t), \quad (1)$$

In Equation (1),

- $f(t)$ is the deterministic component;
- $w(t)$ is the random noise component.

The deterministic component is responsible for the temporal evolution of the phenomenon according to a certain law, while the random component contains a set of information of negligible entities [40,41]. This random component is attributed to chance and assimilated to a series of accidental errors. We can therefore consider it a stochastic component generated by a white noise process, that is, by a sequence of independent and identically distributed random variables of zero mean and constant variance. In the classical approach, the study focuses on the deterministic component, while the random component is considered a process with unrelated components and therefore negligible [42,43].

In a time series, time is the parameter that determines the sequence of events, and it cannot be neglected, so it is also necessary to know the position of the observation along the time dimension. To represent the pair of values (time and observation), a Cartesian diagram with a solid line graph is used as if the phenomenon was detected continuously. Figure 1 shows the trend of milk production monitored from January 1962 to December 1975 [44].

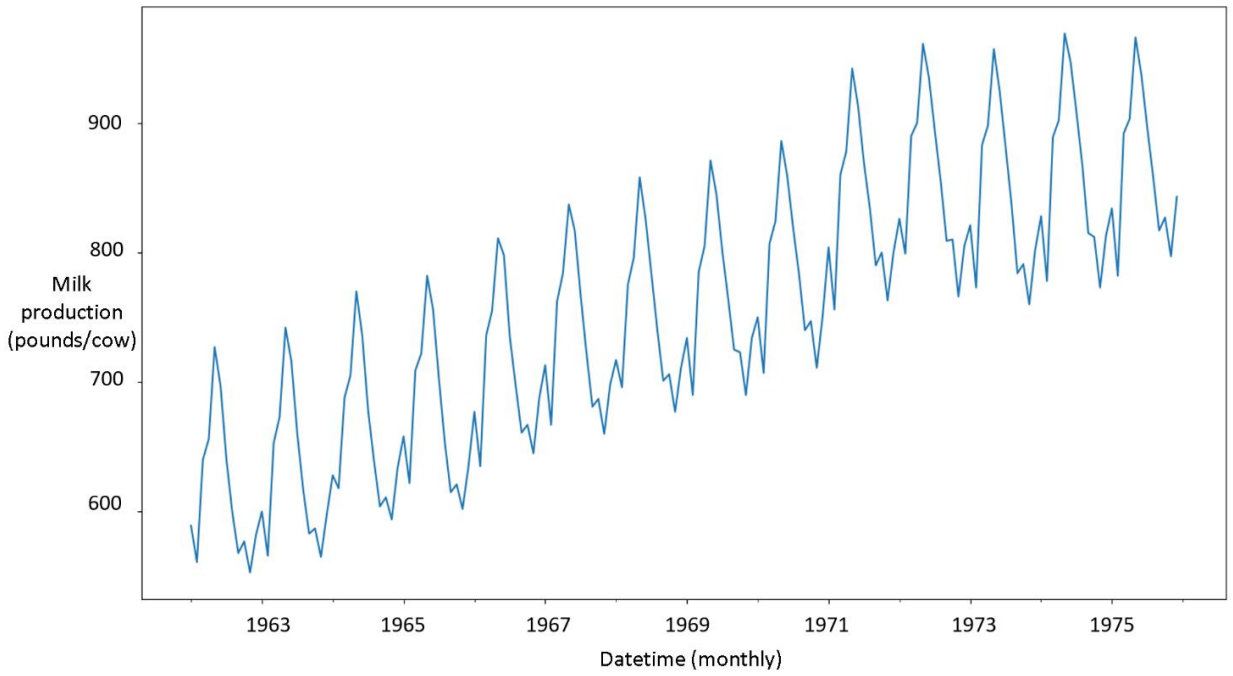

Figure 1. Trend of milk production monitored from January 1962 to December 1975.

From the analysis of Figure 1, it is possible to notice regular trends or fluctuations and other systematic trends over time. The previous graph shows the annual data with a systematically increasing trend in the long term. It returns a zig-zag trend, and since the data are monthly, we can clearly distinguish the phenomenon called seasonality. In fact, it can be noted that high peaks are always recorded in those months in which the calving of cows is expected.

The time series show oscillations around a long-term trend, which are called components of the series [45–47]. Four main types of components can be distinguished (Figure 2):

- Trend (T): monotonous, long-term underlying trend movement, which highlights a structural evolution of the phenomenon due to causes that act systematically on it. For this reason, it is usual to assume that the trend values are expressible through a sufficiently regular and smooth function, for example, polynomial or exponential functions. If it is assumed that the function to be used is monotonous, then the trend can be an increasing or decreasing function of the phenomenon in the analyzed period [48].
- Seasonality (S): consisting of the movements of the phenomenon during the year, which, due to the influence of climatic and social factors, tend to repeat themselves in an almost similar manner in the same period (month or quarter) of subsequent years. These fluctuations originate from climatic factors (alternation of seasons) and/or social organization [49].
- Cycle (C): originating from the occurrence of more or less favorable conditions, of expansion and contraction, of the context in which the phenomenon under examination is located. It consists of fluctuations or alternating upward and downward movements, attributable to the succession in the phenomenon considered of ascending and descending phases, generally connected with the expansion and contraction phases of the entire system. The difference with respect to seasonality, which from a numerical point of view is equal to cyclicality, is the fact that the first is in the order of years for long-term cycles, while the second is in the order of months or at most one year [50].
- Accidentality or disturbance component (E): is given by irregular or accidental movements caused by a series of circumstances, each of a negligible extent. It represents the background noise of the time series, that is, the unpredictable demand component, given by the random fluctuation of the demand values around the average value of the series. The random fluctuation is detected after removing the three regular components of the time series, that is, having isolated the average demand. It should be noted that the random component is not statistically predictable. However, if it is numerically relevant, it is possible to apply linear regression models in which several independent variables are tested on the time series with only the random component. This is useful for identifying correlations between the random component and measurable input variables for which prediction for future values is also available [51].

The components we have described can be combined in different ways, so we can represent the time series as a function of those components, as represented by the following equation:

$$Y_t = f(T_t,\ C_t,\ S_t, E_t), \tag{2}$$

In Equation (2),

- T_t is the trend component;
- C_t is the cycle component;
- S_t is the seasonality component;
- E_t is the accidentality component.

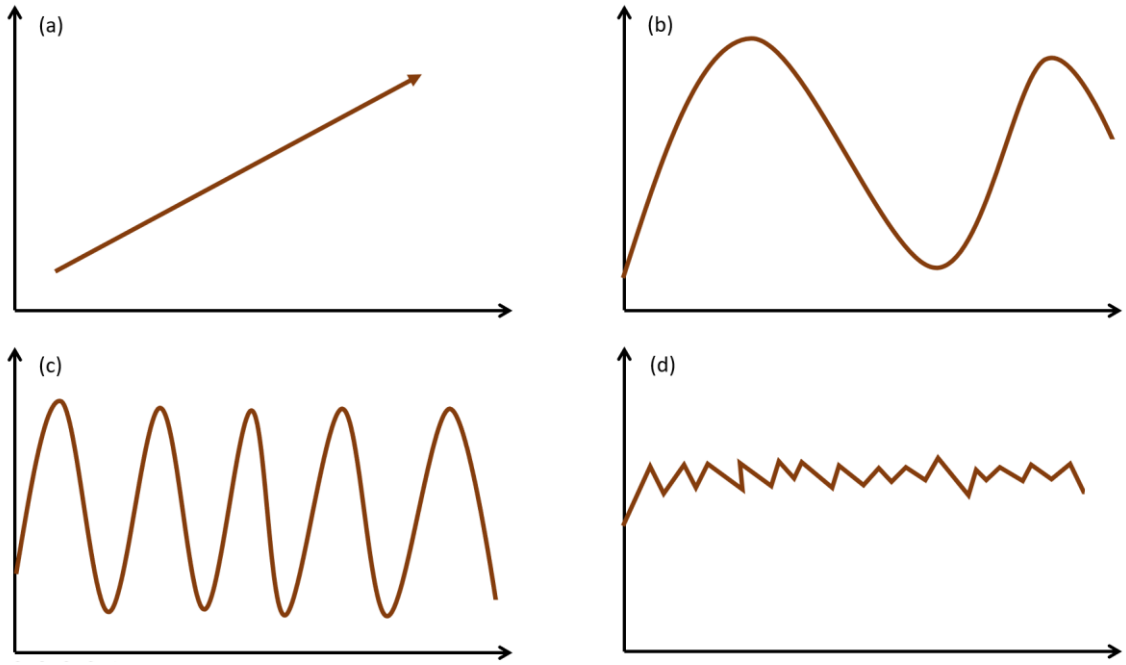

Figure 2. Times series components: (**a**) trend; (**b**) seasonality; (**c**) cycle; (**d**) disturbance.

Once the trend as well as the cyclical and seasonal components have been identified, the classic approach allows you to automatically determine the data of the time series without a random component. In this approach, time is treated directly or indirectly as an independent variable. In fact, it is with the passage of time that the trend evolves, that the cycle assumes an oscillating trend, and that the seasonal component, if present, assumes regular trends. Time is therefore the fundamental variable for the origin of the forces that determine the data, and for this reason, it is the main variable that explains the phenomenon [52].

The modern, or stochastic, approach proposed by Box and Jenkins [8,53,54] has as its final objective the understanding of the probabilistic structure underlying the data and to attempt to reproduce the stochastic process that is supposed to have generated the time series, that is, the set of forces that contributed to producing the data according to probabilistic laws. In the modern approach, it is assumed that the deterministic component is missing or has already been eliminated. The focus is on the stochastic component, which is assumed to be a process with controlled components. The stochastic process can never be identified exactly since we have a finite set of observations available, what we can do is try to simulate it through a model [55].

However, to be able to identify with a good approximation a model that simulates the data-generating process, it must have some characteristics: stationarity, invertibility, the process must be Gaussian, and it must also be possible that, from the time series, it is possible to obtain the properties of the whole process, such as the moments calculated on the sample must converge to those of the entire population [56].

2.2. Times Series Datasets

To apply machine learning-based algorithms for the processing of forecast scenarios on time series, it is necessary to have a quality data archive. For the algorithms to be able to identify recurring behaviors so that the process can be broken down into random parts and deterministic parts, it is necessary to have data distributed over a consistent time window. The collection of information must be carried out in a timely manner, and the data obtained must undergo an adequate validation process. Various archives are available in the literature that contain time series datasets on the most diverse fields of application.

We have already referred to this data store in the previous section (Figure 1): I am referring to the Time Series Data Library (TSDL) [44] created by Rob Hyndman, a professor of statistics at Monash University, Australia. From this library we have extracted data relating to milk production. It represents a collection of datasets that the author has extracted from textbooks or that were sent directly by the creators. The library collects 648 datasets on time series data covering 24 fields of applications, ranging from agriculture to physics through finance and health, just to name a few.

The UCR time series archive [57] collects 128 datasets. This repository is available for free and has a web page to support all the datasets made available [58]. The archive has already been used in numerous studies published in various scientific journals. Some of these studies have formulated numerous criticisms on the quality of the data contained in the repository, and for this reason, the authors in the paper responded to these criticisms by proposing updates and corrections to some datasets.

Kleiber et al. [59] have collected numerous time series datasets contained in the AER package [60] available for the r platform. The package contains approximately 100 datasets taken from a wide variety of sources suitable for processing time series models. Here are some datasets: Consumer Price Index in Argentina, Monthly Averages of the Yield on a Moody's Aaa-Rated Corporate Bond, Real National Income in China per Section, Properties of a Fast-Moving Consumer Good, Dow Jones Index Time Series computed at the end of the week, and TV and Radio Advertising Expenditures Data.

Numerous datasets on financial time series are also contained in the R package called FinTS [61] edited by Spencer Graves. It is a collection of datasets, functions, and script files supporting the book by Ruey S. Tsay [11]. The datasets are many and concern different financial fields, we mention only a few to frame the topics: Simple daily and monthly returns for financial Time Series extracted from IBM, Microsoft, Intel, SP (Standard & Poors), and EW, starting from 2003 Cisco Daily Register Returns for Each Dealing Day in 1999; IBM transactions data; Change series of weekly US 3-year maturity interest rates.

Another substantial source of data is contained in the R Ecdat package [62]. These are datasets for econometrics, including political science. The collection consists of about 120 datasets arranged in alphabetical order concerning very different fields, just to name a few: Accountants and Auditors in the US Labor Force 1850–2016, Air Quality for Californian Metropolitan Areas, Computer Security Violations of Medical Records of 500 or More People as of 27 June 2014, and Seasonally Incorrect Quarterly Data on Available Income and Expenses.

The ANES Time Series Study [63] collected data relating to the US presidential elections from 1948 to 2020. These data are regularly used to analyze public opinion and voting behavior in the US presidential election. The data were collected through pre-election and post-election surveys carried out in person or in recent years via the Internet. The surveys were carried out on an adequately represented population sample.

Schlittgen et al. [64] have developed the tsapp package, which contains more than 40 ready-to-use datasets already loaded, that can be used by simply installing it on our r platform. The datasets were collected by Rainer Schlittgen and Cristina Sattarhoff together with the text [64]. In addition, in this case, the fields of application are varied, to name a few: Monthly Numbers of Road Traffic Accidents with Personal Injury, Alcohol Demand, UK, 1870–1938, Weekly Number of Births in New York from 1961 to 1966, Annual Coffee-Consumption USA from 1910 to 1971, Incidences of Insulin-Dependent Diabetes Mellitus, Temperature and Consumption of Ice Cream, and Annual Amount of Rainfall in Los Angeles.

Harvard Dataverse [65] is an online data repository open to all researchers, both inside and outside the Harvard community. Its use makes it possible to share, store, quote, explore, and analyze research data. In order to use the services of this repository, the user must use the Dataverse open-source web application, developed at the Institute for Quantitative Social Science. This application facilitates the provision of data and allows you to replicate jobs more easily. The data are made visible with the academic credits

related to the authors. The repository hosts several virtual archives called dataverses, each of which contains datasets with descriptive metadata, accompanying documentation, and any code. Harvard Dataverse currently contains over 2000 dataverses, 75,000 datasets, and 350,000+ files. The repository offers the ability to set a filter that returns only datasets classified as time series data.

For government datasets, Data.gov [66] allows you to download data from many US government agencies. Data.gov collects over 1000 different sources of U.S. federal government open data and also provides access to many nonfederal and local government open data resources. Data.gov does not host the data directly, but rather aggregates the metadata about the open data resources in a centralized location. Once an open data source meets the required format and metadata requirements, the Data.gov team can collect the metadata directly, syncing that source's metadata to Data.gov every 24 h.

Another very rich repository of time series datasets is contained in the UCI Machine Learning Repository [67]. It is a collection of 585 well-structured datasets already divided by analysis technique and field of application. For each dataset, there are also numerous references to relevant literature. This archive was created in 1987 so some datasets are a bit dated, but for algorithm applications they are just fine. To speed up the search, you can set filters by number of variables, number of cases, and by type of variables (qualitative, quantitative, or mixed). For each dataset, you will also find references to the relevant literature. An entire section dedicated to time series is available, which collects 119 datasets, for which the type (univariate or multivariate), the type of variables, the number of instances, the number of variables, and finally the year are shown.

Cryer et al. [68] collected about 42 datasets on the time series that they used for the analyses proposed in their book. The data were made available as a package r with the name TSA [69]. These are data concerning different fields of applications, to name a few: Monthly U.S. Airline Passenger-Miles, Monthly Carbon Dioxide Levels in Northern Canada, Electroencephalogram (EEG) Test Data, Monthly U.S. Electricity Generation, Daily Price of Gold, and Monthly Spot Price for Crude Oil.

In 2018, Google launched Google Dataset Search [70], a service that complements Google Scholar, created specifically for students, researchers, scientists, and data journalists. Google has indexed over 25 million datasets, which you can search as you normally would on the search engine and filter by date, format, topic, usage rights, and whether they are open and free or not.

Shumway et al. [71] collected a series of time series datasets that they used in the time series analysis and application examples proposed in their book. The datasets and numerous scripts for the analysis of the time series were collected in a package called astsa [72].

Kaggle [73] is an online platform where you can find more than 30,000 datasets of all sizes and formats taken from real and often recent cases on the issues more varied: from medicine to sport, from science to economics. Over time, many datasets have generated real communities where you can find interesting discussions on the possible analyses that can be carried out with that data and in some cases even the code ready to be created on software such as R or Phyton. On Kaggle, real competitions are also held periodically in which the best types of analysis are awarded to solve a complex problem proposed by a company or a research center. Moreover, in this case it is possible to set a filter that returns only the datasets classified as time series data.

There are also two interesting packages that contain the data used in Makridakis competitions. The first Mcomp package [74] collects the data used in the first Makridakis competition, called M-Competition [75], which was held in 1982 and used 1001 time series. Furthermore, this package also collects the data used in the third competition, called IJF-M3 competition [76], which used a total of 3003 time series. The time series collected in these archives have been elaborated, starting from data in different fields of application: micro, industry, macro, finance, demography, and more. The second package called M4comp [77] collects about 10,000 time series from the M4 time series forecasting.

3. Machine Learning Methods for Time Series Data Analysis

Machine Learning-based algorithms autonomously develop their knowledge thanks to the data patterns received, without the need to have specific initial inputs from the developer [78]. In these models, the machine can establish by itself the patterns to follow to obtain the desired result, therefore, the real factor that distinguishes artificial intelligence is autonomy. In the learning process that distinguishes these algorithms, the system receives a set of data necessary for training, estimating the relationships between the input and output data: these relationships represent the parameters of the model estimated by the system [79] (Figure 3).

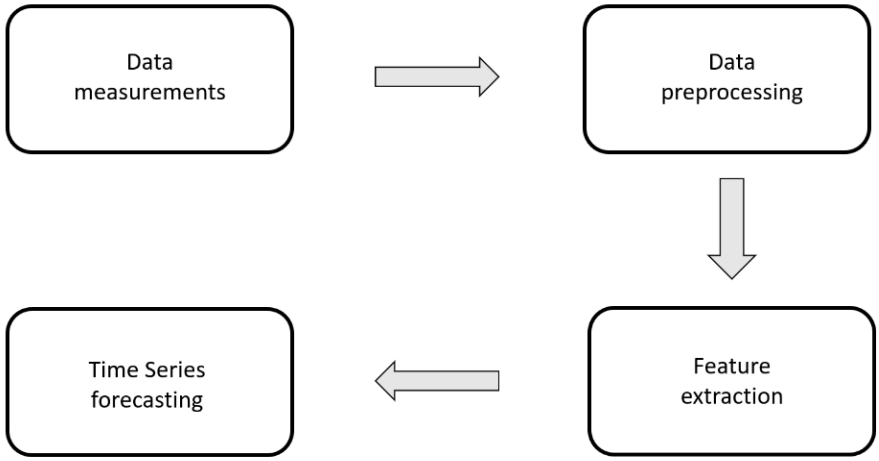

Figure 3. Block diagram of the machine-learning-based knowledge extraction for time series forecasting.

Sometimes, the input data are supplied with the respective output data, that is, the system receives initial data and is given a precise future realization determined by the actions taken. In these cases, we speak of supervised learning [80], in the sense that the input data are provided with a label that represents the output of the model: the system learns to recognize the relationships that bind the input data to the labels. In other cases, the data are not labeled, so we speak of unsupervised learning [81], the system receives only the input datasets without any indication of the outputs to be obtained. The purpose of this learning method is to trace hidden patterns, or to identify a logical structure in the input data without having previously been classified. Finally, to the two learning paradigms just mentioned, a third one is added: the reinforcement learning [82]. In this paradigm, the system interacts with a dynamic environment and with a system of rewards and punishments that guide it towards improving its performance. Once the input data are obtained, the computer must reach a goal that is established on the basis of a value function that only tells it if the goal has been achieved. In the following subsections, we will analyze the most common methods based on machine learning that the scientific community has adopted to solve numerous problems of forecasting time series.

3.1. Artificial Neural Network-Based Methods

Neural computation is inspired by biological neural systems, whose structure it tries to model to simulate their basic functions. An artificial neural network (ANN), inspired by biological systems, considers many processors, called artificial neurons, having an elementary computational capacity connected to other units of the same type [83]. This model was therefore born as an attempt to emulate and better understand the functioning of the human brain, organized in biological neural networks. The brain is a highly complex, nonlinear system capable of processing information in parallel. In this way, it can perform some functions, such as pattern recognition and movement management, faster than any modern computer, but above all it can learn from past experience [84].

A model based on artificial neural networks is therefore made up of neurons, which, through connections, can transmit signals to nearby neurons. In this structure we can distinguish between the following:

- Input neurons: they receive information from the environment;
- Hidden neurons: they are used by the input neurons to communicate with the output neurons or with other hidden neurons;
- Output neurons: they emit responses to the environment.

The mechanism that regulates the functioning of such a system is the transmission of signals between neurons when the signal they receive exceeds a certain activation threshold. The activity of an artificial neuron can therefore be summarized through a mathematical function described by the following equation:

$$y_i = f(x_i) \tag{3}$$

The function f is typically nonlinear and is called the activation function. In Equation (3), the term x_i is represented by Equation (4).

$$x_i = \sum(w_{ji} + b_i) \tag{4}$$

The terms present in Equation (4) are defined as follows:

- w_{ji} = connection weights;
- b_i = bias.

The parameters described in Equations (3) and (4) are fundamental for the success of analyses based on neural networks, as they summarize the information extracted in the learning process. During the learning process, the parameters are updated to get closer to ever more correct and precise prediction models [85].

Neural networks are therefore made up of central blocks called neurons, organized in layers that, starting from the so-called input layer, can process the information produced by the intermediate hidden layers to produce the desired output (Figure 4).

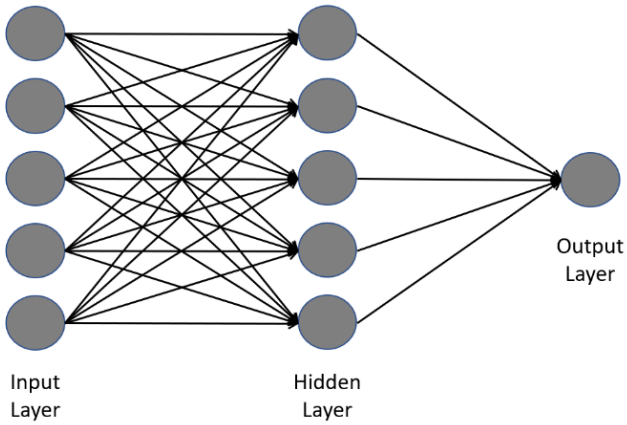

Figure 4. Artificial neural network architecture for time series data. The input dataset is divided into sequential groups with a fixed number of components. Each of these vectors constitutes a network input, to which the correct output is matched. Therefore, neuron 1 will be sent the vector that contains the first k data (from x_1 to x_k), neuron 2 instead will be sent the vector that contains data from x_2 to x_{k+1}, and so on (Equation (5)).

Figure 4 shows the architecture of a simple artificial neural network of the feed-forward type. Feed-forward networks foresee unidirectional connections between neurons belonging to immediately consecutive layers, with a connection that proceeds forward, that is, from layer n to layer n + 1. In this architecture, therefore, the inputs are connected

only to the neurons of the hidden layer (there can also be more than one), which in turn are joined to the neurons of the output layer [86].

In Equation (4), the weight assumed by the connections is weighted through a simple linear combination of the input variables. It is understood that such a simple structure cannot model complex systems with nonlinearity characteristics. This simple discriminant function can be generalized by transforming the result through a nonlinear function called activation function. The activation function conditions the relationship between the input and output neurons when the former receives a signal from the outside, above a threshold, and propagate it towards the inside of the network. When the activation threshold is reached, then the neuron is activated and participates in the processing of the model output [87]. Otherwise, the neuron remains inactive and does not participate in the construction of the output signal. Depending on the structure used, different activation functions are distinguished that describe the relationship between the different neurons. The activation function is a nonlinear but continuous and differentiable function. Nonlinear because if we want a network to be able to perform a complex mapping of the input information, nonlinearity characteristics are necessary. The characteristics of continuity and differentiation are instead necessary for the retro-propagation of the error [88].

After having appropriately chosen the activation function of the neurons and the architecture of the network, the weights must be adjusted so that the network works optimally. One method for training a neural network is supervised learning, which presents as an input a set of inputs, or training set, in which the network tries to understand how the phenomenon evolves over time to estimate the parameters of the model [89]. The response obtained with these inputs is then compared with the observed response to calculate the error committed and subsequently adjust the weights. In this way, the most suitable model can be tested on a sample, the test set, not used in the training phase to evaluate the generalization capacity of the neural network [90]. If this network is unable to provide adequate outputs for inputs never presented, the training phase will have to be repeated (Figure 5).

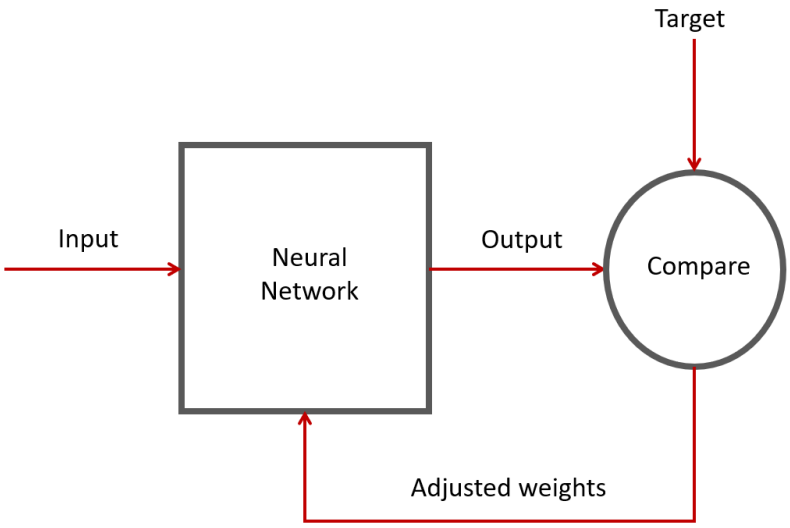

Figure 5. Artificial neural network training procedure.

To use a model based on artificial neural networks for the prediction of time series, it is necessary to proceed in advance with the division of the input data [91]. The approach to follow is to divide the set of data into sequential groups with a fixed number of components. Suppose we have a distribution of univariate time series data and denote by x_i with

$i = 1, 2, \ldots, n$ the explanatory variable of the phenomenon of interest [92]. Suppose we want to group the data into subsets of k elements, obtaining the following distribution:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ x_k \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \\ x_4 \\ \cdots \\ x_{k+1} \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \\ x_5 \\ \cdots \\ x_{k+2} \end{bmatrix} \ldots\ldots \begin{bmatrix} x_{n-(k-1)} \\ x_{n-(k-2)} \\ x_{n-(k-3)} \\ \cdots \\ x_n \end{bmatrix} \quad (5)$$

Each of these vectors constitutes an input of the network, to which the correct output is matched. The output layer of the network is formed by a single neuron, so the first term of the series not contained in the input data is chosen as the correct value [93]. This means that, considering the vectors shown in Equation (5), they will be associated with an output belonging to the following sequence:

$$x_{k+1}, x_{k+2}, x_{k+3}, \ldots\ldots, x_{n+1} \quad (6)$$

We denote by w_s the window size parameter defined as the number of elements of each subset plus 1, this value representing the output value. In this way, the training set will consist of $n - (w_s - 1)$ pairs: an input vector, to which the correct output is associated [94]. More briefly, a matrix of input values is constructed, represented by the values expressed in Equation (5) and a vector containing the correct output values represented in the sequence of Equation (6).

The architecture of the network will be modulated to have $w_s - 1$ neuron in the first layer, and a neuron in the last layer. After presenting all the patterns included in the training set, the weights will be updated, having concluded a complete iteration (epoch). The number of iterations carried out before finishing learning is an arbitrary parameter, but in choosing it, you must keep in mind the problems arising from overfitting [95]. Once the network training process is complete, the model obtained can be used to make predictions.

In recent years, several researchers have used models based on artificial neural networks for the prediction of time series. Azadeh et al. [96] adopted a model based on a supervised multilayer perceptron (MLP) for the prediction of electricity consumption. Electricity consumption modeling is difficult to predict with conventional methods due to various seasonal and monthly variations. The authors used the data extracted from the time series to train the model. This study shows the advantage of the ANN methodology through the analysis of variance with ANOVA technology [97]. In addition, the actual data are compared with the ANN and the conventional regression model. Hill et al. [98] compared the results obtained by modeling time series data with ANNs with those from six statistical time series methods. The data used in this work were extracted from the first Makridakis competition, called M-Competition [75]. In the monthly and quarterly time series, neural networks performed significantly better than traditional methods, proving particularly effective for discontinuous time series. Zhang [99] instead proposed a hybrid methodology that combines ARIMA [100] and ANN models to exploit the potential of both technologies in linear and nonlinear modeling. Experimental results with real datasets indicate that the combined model can be an effective way to improve the accuracy of the prediction obtained from one of the models used separately. Jain et al. [101] used a hybrid methodology capable of exploiting the strengths of traditional time series and ANN approaches. They first performed trend reduction and deseasonalization using an approach based on conventional techniques, and the results obtained were used to train the ANN. The model was tested on monthly data of current flow in part of the Colorado River in the United States. The model developed proved capable of capturing the nonlinear nature of complex time series and thus produce more accurate predictions. Tseng et al. [102] have also adopted a hybrid approach, this time by exploiting the seasonal adjustment potential of the SARIMA model [103] and the generalization capacity of the ANNs. The hybrid model was tested on seasonal machinery manufacturing data and soft drink manufacturing data.

Khashei et al. [104] adopted a hybrid model based on the integrated self-regressive moving average algorithm (ARIMA) and artificial neural networks. The aim is to improve the predictive ability of neural networks for time series. The results obtained using real datasets indicate that the proposed model can be an effective way to improve the accuracy of the prediction obtained from artificial neural networks. Chaudhuri et al. [105] applied models based on multilayer feed forward neural network for forecasting the exchange rate in a multivariate framework. To evaluate its performance, they then compared it with other models based on traditional statistical techniques. In comparison, the model based on ANNs proved to be more efficient. Aras et al. [106] proposed a new strategy for selecting an ANN-based forecasting model. This approach first determines the number of input and hidden units, then selects a neural network model from various trials caused by different initial weights considering the validation and training performance of each neural network model. This approach statistically improves the performance of neural networks compared to the classical model selection method in simulated and real datasets. Furthermore, it shows some robustness with respect to the size of the validation dataset. Doucoure et al. [107] developed a forecasting method for renewable energy sources to achieve intelligent management of a microgrid system and promote the use of renewable energy in isolated as well as grid-connected power systems. The proposed method is based on the multiresolution analysis of time series by wavelet decomposition and the use of artificial neural networks. Lohani et al. [108] used ANNs to model hydrological time series. Chicea et al. [109] modeled the time series of dynamic light dispersion with the use of ANNs. The network was trained with data from simulated time series autocorrelation series for monodisperse spherical particles with diameters in the range 10–1200 μm. The accuracy of the neural network output was tested on both simulated and experimental time series recorded on fluids containing nanoparticles and microparticles. Horák et al. [110] compared a method of exponential time series alignment with time series alignment using artificial neural networks as tools for predicting future share price trends. Liu et al. [111] compared three hybrid models to predict wind speed: wind speed prediction is important for the safety of wind energy integration. Wang et al. [112] developed a new method of diagnosing failures in rotating machines using neural network time series analyses.

3.2. Time Series Claustering Methods

The theory of prediction is that branch of statistics that tries, starting from completely or partially known information, to deduce the value of other currently unknown quantities of interest to us. The way in which this information is deduced characterizes the extraction process and governs the related methodologies [113]. Previously, we spoke of supervised learning in the case of a classification of labeled examples that a hypothetical supervisor has provided us. However, when we face an unsupervised problem, we do not have a set of labeled examples available. This means that in the training set, there is no target that we can use as an example. In these cases, we must adopt a different learning paradigm, for example, we can apply cluster analysis [114].

It is a multivariate analysis technique through which it is possible to group the statistical units, to minimize the logical distance within each group and to maximize that between the groups (Figure 6) [115]. The logical distance is quantified by means of similarity/dissimilarity measures defined between the statistical units [116]. Cluster analysis therefore tends to identify groups of units like each other with respect to a set of characters taken into consideration, and according to a specific criterion. The goal is essentially to bring together heterogeneous units into multiple subsets that tend to be homogeneous and mutually exhaustive. The statistical units are, in other words, divided into a certain number of groups according to their level of similarity, evaluated starting from the values that a series of selected variables assume in each unit [117].

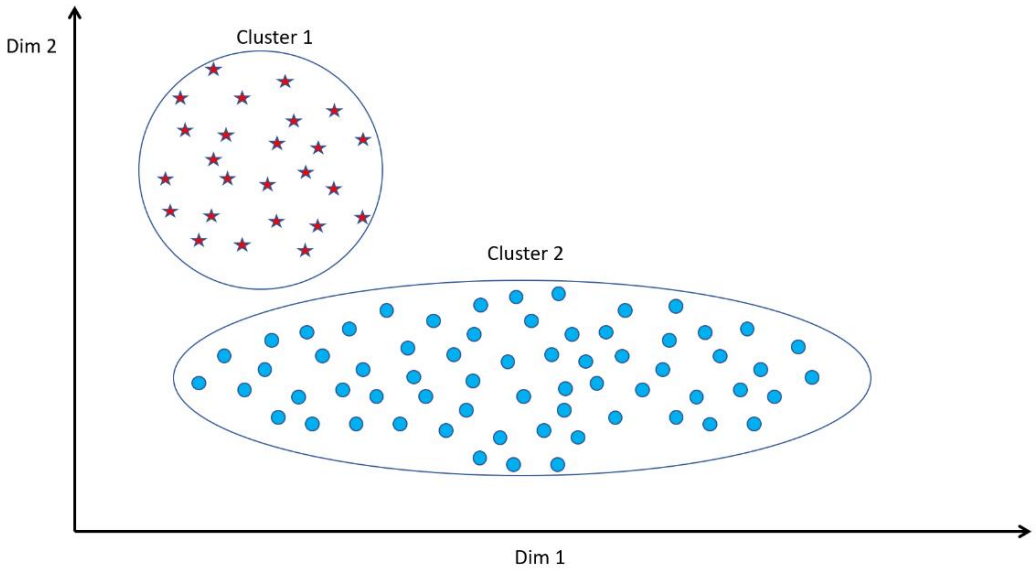

Figure 6. Clustering grouping methodology example in a new plane defined by two new dimensions identified by the algorithm.

Cluster analysis techniques can be divided into two broad categories: hierarchical analysis and nonhierarchical analysis. In the first type of analysis, each class is part of a larger class, which is in turn contained in a class of greater size, and so on, progression up to the class that contains the entire set of data. In the second type of analysis, however, the groups are not hierarchical, and for this reason the number of groups must be decided a priori. There are also fuzzy methods for which the assignment of a unit to a group is not certain and unique. The choice of a clustering algorithm depends on both the type of data available and the purpose to be achieved. Sometimes, several choices are applicable, and a priori arguments are not enough to restrict the choice to a single method. In these cases, it is useful to apply more than one algorithm, carefully analyze and compare the resulting classifications, making use of their graphical results [118].

Nonhierarchical clustering directly divides objects into a prespecified number of clusters without a hierarchical structure, while hierarchical clustering groups data with a sequence of nested partitions, starting from many clusters, each having a single object, up to having a single cluster with all the objects of the dataset for agglomerative hierarchical clustering, or vice versa for divisive hierarchical clustering [119]. Both types of hierarchical clustering methods organize data into hierarchical structures based on the proximity matrix. The results of hierarchical clustering are usually represented by a binary tree or dendrogram (Figure 7). The initial node, which is called the root of a dendrogram, represents the entire dataset, and each leaf is considered as a single object of the dataset. The intermediate nodes, consequently, describe how similar the objects are to each other, and the height of the dendrogram expresses the distance between each pair of objects or clusters or the distance between an object and a cluster. Final clustering results can be obtained by cutting the dendrogram into different levels [120].

This representation provides very informative descriptions and views of clustering structures.

Hierarchical analysis techniques can be further distinguished as follows:

- Agglomerative: if they foresee a succession of mergers of the n units, starting from the basic situation in which each constitutes a group and up to the stage $(n - 1)$ in which a group is formed which includes them all.
- Divisive: when the set of n units, in $(n - 1)$ steps, is divided into groups that are, at each step of the analysis, subsets of a group formed at the previous stage of analysis, and which ends with the situation in which each group is composed of a unit.
- Nonhierarchical analysis techniques can be divided into the following:

- Partitions: mutually exclusive classes, such that, for a predetermined number of groups, it is possible to classify an entity into one and only one class.
- Overlapping classes: for which it is admitted the possibility that an entity may belong to more than one class at the same time.

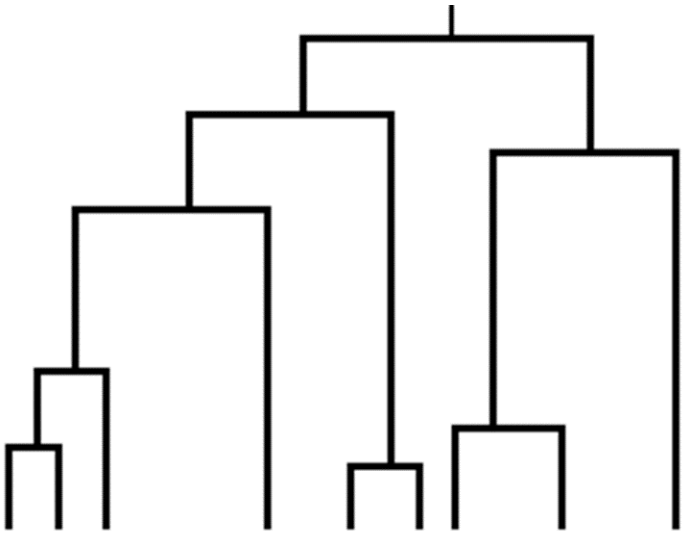

Figure 7. Dendrogram representation with nodes, claves, and leaves.

The use of cluster analysis poses problems relating to the choice of a metric that can synthetically express the distance between the elements of the sample that are to be grouped. In the specific context of time series, the concept of similarity is particularly complex given the dynamic characteristics of the data [121]. The similarity measures generally considered in cluster analysis for independent data do not work adequately with data that have a time dependency because they ignore the interdependent relationships between values. It is therefore advisable to adopt distance measures that consider the behavior of the series over time. In the context of time series, however, it is not sufficient to calculate the Euclidean distance between the different values of the time series over time [122]. This is because, in this way, we are mistakenly treating temporal observations as independent variables (Figure 8).

To carry out cluster analysis between time series, it is necessary to introduce measures specifically designed for this area that consider the intrinsic time dependence in time series data [123].

The measures present in the literature are often divided into three large groups:

- Model-free approaches, that is, methods that are based only on specific characteristics of the series;
- Model-based approaches, that is, methods that are based on parametric assumptions on time series;
- Prediction-based approaches, that is, methods that are based on forecasts of time series.

The model-free approach does not require assumptions on any models for time series. A first method involves the measurement of distance based on correlation. The classical concept of correlation was introduced to characterize processes that have a similar course. Given a set of time series, the correlation or variance–covariance matrix measures, in fact, the degree of similarity between the time series. This distance now enjoys the typical properties of a dissimilarity measure. Another model-free approach involves the measurement of distance based on autocorrelation. Analogous distances can be constructed by considering partial autocorrelation functions instead of global autocorrelation functions [124].

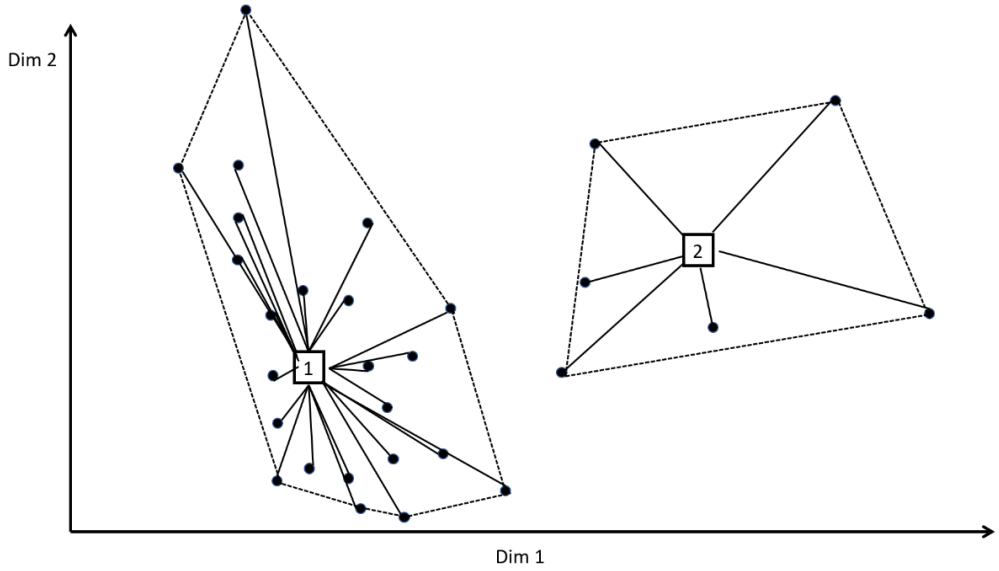

Figure 8. An example of clustering of a dataset using two classes in a new bidimensional plane.

In the model-based approach, the evaluation of the measures requires the development of a statistical model for the description of the observations. The Piccolo distance [125] defines a measure of dissimilarity in the class of invertible ARMA processes as the Euclidean distance between the coefficients of the AR representation (∞) and the corresponding ARMA measures. Piccolo argues that autoregressive coefficients convey all useful information on the stochastic structure of this type of process. When dealing with historical financial series, a relevant measure is given of the risk, which can be measured through the volatility, and therefore through the conditional variance. Otranto et al. [126] proposes an extension of the dissimilarity measure proposed by Piccolo, which considers the Euclidean distance of the parameters of an autoregressive conditional heteroskedasticity process (GARCH), therefore a function of the values assumed by the process in the previous instants.

In the prediction-based approach, two time series will be considered similar if their forecast, for a given time instant, is close. Obviously, a clustering procedure of this type can produce different results from those generated with model-free hierarchical methods. For example, two time series from the same generation process will be able to produce different predictions at prespecified horizons, so these time series may not be grouped together according to this new dissimilarity criterion [127].

Iglesias et al. [128] applied several similarity measures in the grouping of time series for the elaboration of energy models for buildings. The aim was to evaluate the effect of similarity measures in the application of clustering techniques for strongly correlated data. The authors proposed cluster vector balancing as a validation technique for comparing clustering performance. Gopalapillai et al. [129] used clustering to support decision-making in rescue robots management. The authors used a robot equipped with four sensors, collecting data as it moves along a path in a straight line. Wismüller et al. [130] applied cluster analysis for self-organized segmentation of time series data derived from biomedical images.

Guo et al. [131] used clustering techniques for stock data analysis. In a first phase, the authors transformed the raw data into vectors of smaller features using independent component analysis (ICA) [132]. In the second phase of the study, they applied the modified k-mean algorithm [133] to the vectors of the extracted features. Shumway [134] applied time–frequency clustering to classify earthquakes and mineral explosions at regional distances. Elangasinghe et al. [135] applied k-means clustering to model PM 10 and PM 2.5 concentration levels in a coastal New Zealand location. The authors exploited the results obtained with clustering as an input of an ANN-based model, thus obtaining a significant

improvement in performance. Möller-Levet et al. [136] exploited fuzzy clustering to group short time series or those with nonequidistant sampling points. Rebbapragada et al. [137] applied clustering to detect anomalous presences in the light curve data to identify new classes of astronomical objects.

Paparrizos et al. [138] proposed two new algorithms based on time series clustering. These algorithms use a shape-based measure of distance between clusters (SBD) [139], which represents a normalized version of the measure of cross-correlation. Sardá-Espinosa [140] compared several algorithms based on time series clustering, creating a specific R package easily usable by researchers. Zhang et al. [141] proposed a new model based on clustering for identifying the shapelets of the time series without the effort of labeling. Chen et al. [142] exploited clustering techniques to generate a risk profile for lane-changing (LC) maneuvers performed by a car driver. Steinmann et al. [143] addressed the Scenario Discovery problem using time series clustering. Identifying common properties of the input space between sets of exploratory model runs represents a widely used methodology in the field of decision support. Kuschnerus et al. [144] applied time series clustering for the elaboration of coastal change models using permanent laser scan data. Motlagh et al. [145] applied clustering techniques to group the customers of electricity supply companies to reduce the extreme dimensionality of the load time series. Hallac et al. [146] proposed a new method based on a correlation network, or Markov random field (MRF), which characterizes the interdependencies between the different observations in a subsequence typical of a specific cluster. McDowell et al. [147] pooled gene expression time series data using an infinite Gaussian process mixture model.

3.3. Convolutional Neural Network for Time Series Data

In Section 3.1, we introduced the ANNs specifying that a typical architecture of these algorithms includes one or more hidden layers. In convolutional networks, in fact, there are several hidden layers that give the network its depth characteristics. Convolutional neural networks (CNN) were born in 1990 from the research of Yann LeCun and his team based on the functioning of the visual cortex of the human brain [148]. Thanks to the excellent performance that has been achieved especially in the field of image recognition, even today CNNs are considered the state of the art in terms of pattern and image recognition. Yann LeCun received the Turing Award in 2018 for his work on deep neural networks. CNNs can automatically understand the most relevant traits and learn their patterns [149]. For this reason, normally the layers of CNNs are considered as blocks to detect the traits. The first layers, those immediately after the entry layer, are considered low-level features extractors, while the last layers, usually completely connected like those of the ANNs, are considered high-level features extractors [150].

In the images, the low-level features are, for example, the edges or the blobs that are reworked to form high-level features, which, once passed to the last layers of the network, will be able to create, for example, the outlines of houses, dogs, cats, or whatever was present in the original image [151,152]. CNNs process maps of traits where each element corresponds to pixels in the original image. To obtain this result it is necessary to carry out an operation called convolution [153].

There are two concepts behind how CNNs work:

- Sparse connectivity: a single element in the trait map is connected only to small groups of pixels.
- Parameter-sharing: the same weights are used for different groups of pixels of the main image.

Replacing a perceptron multilayer type layer with a convolutive one involves a drastic decrease in the number of parameters within the network and, therefore, an improvement in the ability to extract relevant traits. CNNs consist of various convolutional layers and subsampling layers followed by fully connected layers at the end [154]. The convolutional and fully connected layers have weights and activations like those present in perceptrons, and these are optimized for learning. On the other hand, pooling layers are not [155].

The layers that follow one another in a typical convolutional neural network architecture are the following (Figure 9) [156]:

- Input level: represents the set of numbers that make up the image to be analyzed by the computer.
- Convolutional level: it is the main level of the network. Its goal is to identify patterns, such as curves, angles, and squares. There are more than one, and each of them focuses on finding these characteristics in the initial image. The greater the number, the greater the complexity of the feature they can identify.
- ReLu level (rectified linear units): the objective is to cancel negative values obtained in previous levels.
- Pool level: allows you to identify if the study feature is present in the previous level.
- Fully connected level: connects all the neurons of the previous level in order to establish the various identifying classes according to a certain probability. Each class represents a possible final answer.

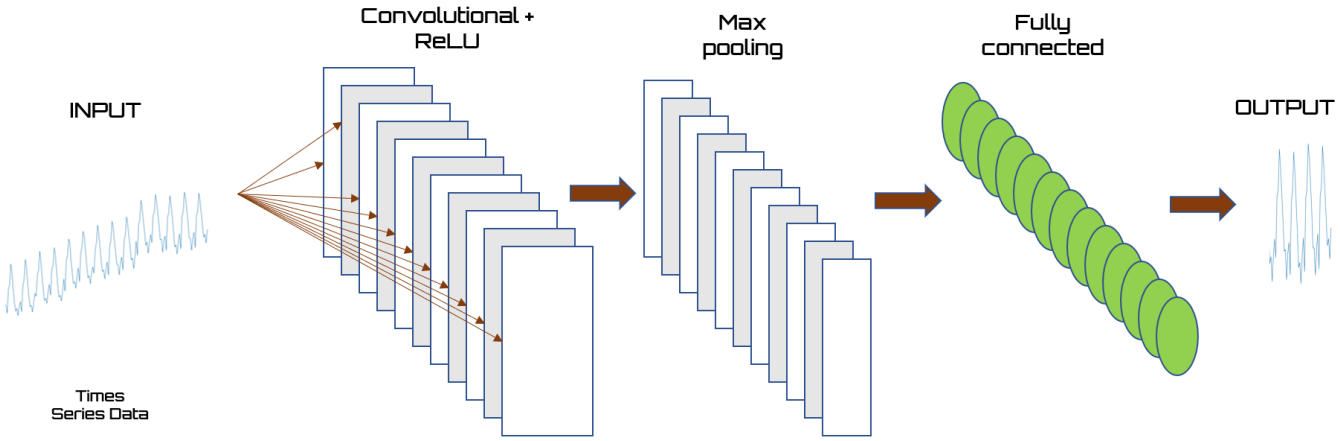

Figure 9. Convolutional neural network architecture.

A multivariate time series can be effectively treated with the use of a convolutional neural network [157]. These algorithms can extract knowledge from spatial and temporal models with the use of appropriate trainable filters. Such filters, which in traditional classification models must be processed in a preventive manual supervision phase, in the case of convolutional networks are learned automatically by the same algorithm [158]. This important function is effectively dealt with in the convolutional layer. Performing the convolution product means sliding one function over another, making the integral [159]. In this case, the first function represents the time series. A second function runs on it, which takes the name of filter, which has the function of identifying structures or patterns within the first signal (the time series) [160]. Therefore, the function of this network is precisely to recognize patterns recurring within the input signal, and it is based on what it is possible to exploit it in the prediction of time series [161].

The first to indicate convolutional networks as an important tool for extracting knowledge from time series was the creator of these algorithms, Yann LeCun [162], who used CNNs for the classification of extracted time series data from satellite images. The authors exploited the images detected by modern remote sensing sensors housed on the satellites. Such images exhibit spatial, spectral, and temporal characteristics. Convolutional layers applied to the time dimension can automatically learn temporal and spectral characteristics. Zhao et al. [163] showed how convolution and pooling operations can extract the suitable internal structure to automatically generate deep characteristics of the input time series. The network used was trained on eight different datasets extracted from the UCR Time Series Classification Archive [164]. Liu et al. [165] proposed a model that uses a deep learning architecture based on a multivariate convolutional neural network. This model was used by

the authors for the classification of multivariate time series, exploiting the multivariate and lag-feature characteristics as input to the network. Data extracted from the 2015 PHM Data Challenge [166] were used to evaluate the model's performance. Cui et al. [167] presented a model based on a multiscale convolutional neural networks. The use of this model allows to obtain the extraction and classification of the characteristics in a single structure. The model has several branches that perform different transformations of the time series, in this way, characteristics at different frequencies and time scales are extracted.

Borovykh et al. [168] treated the conditional time series by exploiting the deep convolutional WaveNet architecture [169] and adapting it to this type of data. The structure uses dilated layers of convolutions capturing the history of the data during the forecast. Furthermore, the activation function ReLU, applied in parallel to the convolutional filters, allows the exploitation of the correlation structure between the multivariate time series. The evaluation of the proposed model was performed on financial time series data such as that of the volatility index, S & P500, and the CBOE interest rate. Yang et al. [170] addressed the problem of human activity recognition (HAR), where inputs are multichannel time series signals captured by a series of body-worn inertial sensors and outputs are predefined human activities. The authors used a CNN to automate feature learning from raw inputs in a systematic way. Two datasets were used to evaluate the proposed technology: Opportunity Activity Recognition Dataset [171] and the Hand Gesture Dataset [172]. Le Guennec et al. [173] proposed data-augmentation and a semisupervised approach to training a CNN. These techniques aim to overcome the problem of the large amount of data that is required for the training of a convolutional network.

Hatami et al. [174] applied CNN for image recognition. The authors transformed one-dimensional time series images into 2D texture images using recurrence (RP) plots [175]. Sezer et al. [176] used CNNs for financial trading. The approach followed involves a conversion of time series data into 2D images, exploiting 15 different technical indicators applied to a time horizon of 15 days. This returns 15×15 images that can be passed as input to a CNN. Hong et al. [177] exploited CNNs for the diagnosis and explanation of physiological time series. The methodology is of particular importance given the large amount of data that can be recorded by smart bracelets or smartwatches. Lu et al. [178] diagnosed faults in photovoltaic systems by applying CNNs to the time series of electricity production. The authors first transformed the sequential current and voltage data produced by the photovoltaic system into two-dimensional electrical time series graphs. These graphs were proposed as input to CNN with an architecture with nine convolutional layers, nine max-pooling layers, and a fully connected layer.

Han et al. [179] developed a CNN-based methodology for the detection of mixed gases. Data were collected from eight gas sensors with time series characteristics. The authors applied as many as five CNN architectures, obtaining a final gas recognition rate of 96.67%. Gao et al. [180] used CNNs to detect time series anomalies. The authors first performed a decomposition of seasonal trends in the time series data and subsequently applied a convolutional neural network-based algorithm for anomaly detection. Kashiparekh et al. [181] proposed a pretrained model for classifying time series. This model exploits the potential of convolutional filters to capture temporal characteristics on multiple time scales. Tang et al. [182] presented algorithms for identifying features in ARMA time series models. Such algorithms exploit the potential of CNNs for statistical inference in cases where classical methods based on likelihood are expensive to implement.

3.4. Recurrent Neural Network

A recurrent neural network (RNN) is a particular family of ANNs in which there are some feedback connections, or cycles or loops within the network structure [183]: the presence of cycles allows to analyze time sequences. In other words, they are able to keep the status in memory from one iteration to the next, using their output as input for the next step [184]. An RNN then receives the inputs, produces an output, and sends the latter back to itself (Figure 10). Unrolling it along the time axis, it can be thought of as a concatenation

of neural networks that take as input values the inputs and also the outputs of the network at the previous time, in order to also keep memory of the past history of the data [185].

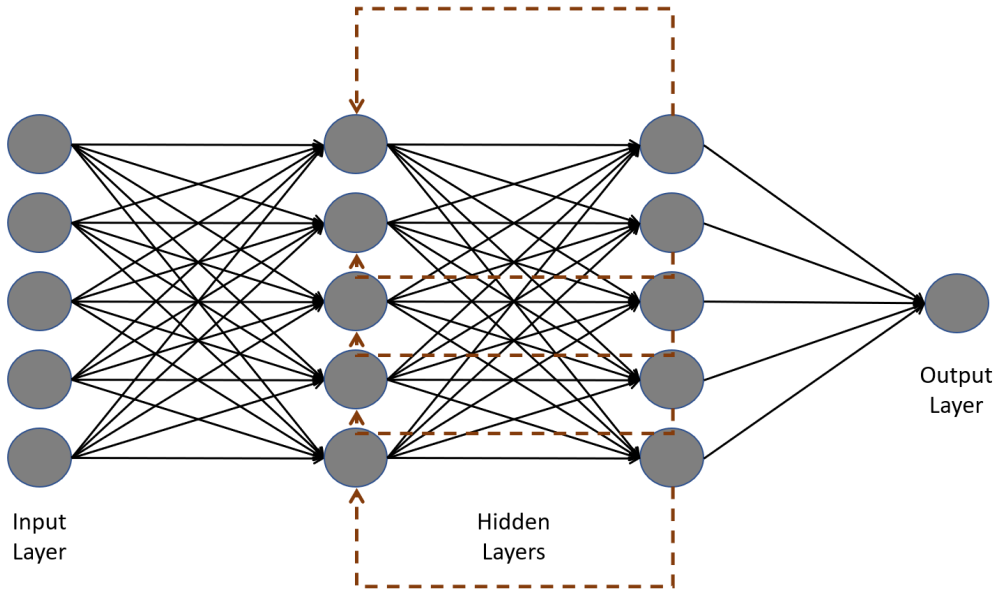

Figure 10. Recurrent neural network architecture.

One of the most used recurrent networks is that of Hopfield conceived in 1982 [186], able to simulate the capacity of the human brain to remember things and to produce an associative memory, suitable for reconstructing distorted images and recovering some missing information; this type of network represents an example of unsupervised learning [187].

From the point of view of the computational graph, it is possible to perform the so-called unfolding of the structure to obtain a feedforward version of the network of arbitrary length that depends on a sequence of inputs (Figure 11) [188]. While the weights and biases of block S are shared, each output h_t depends on the processing by the network of all inputs x_i with $i \in [0; t]$. The number of blocks of the unfolded version essentially depends on the length of the sequence to be analyzed.

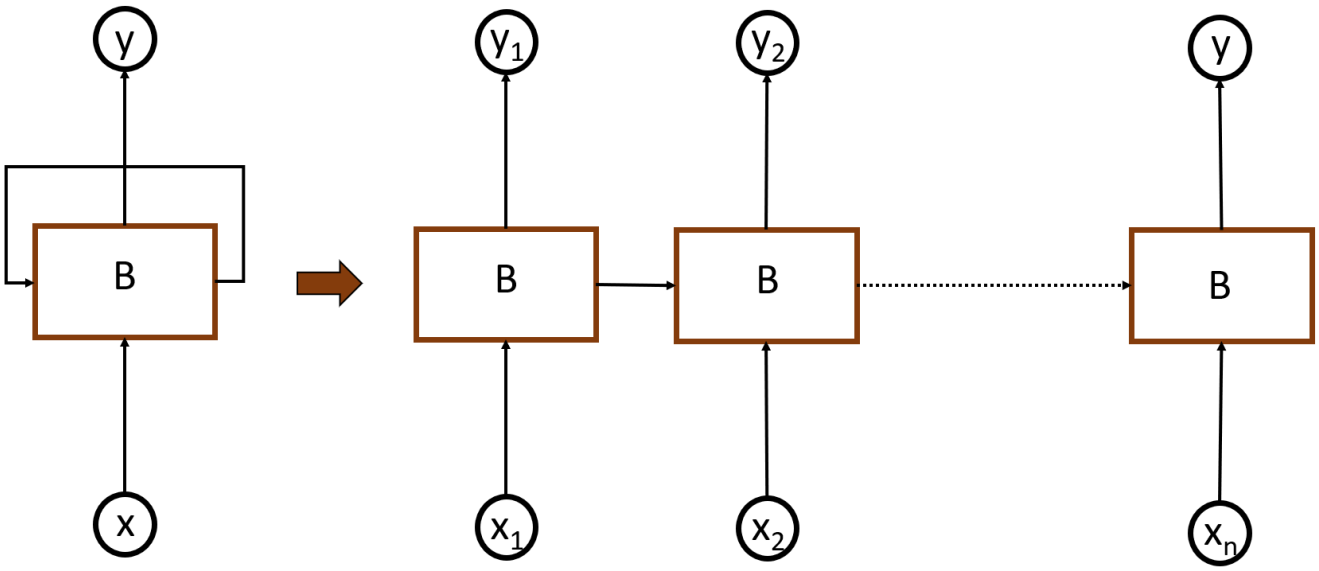

Figure 11. Recurrent neural network unfolded scheme.

The major limitation of this type of network, in the case of a learning algorithm based on backpropagation [189], is however the vanishing/exploding gradient [190]. In the learning phase, the error gradients are propagated back to the initial layer by passing through the continuous matrix multiplications, and after a certain number of epochs, the weights diverge to infinity or converge to zero. In the case of the vanishing gradient, the weights converge to zero and therefore the network, when it makes the predictions, will consider only the most recent events, while in the case of an exploding gradient, the value of the weights explodes, tending to infinity. There will therefore be no past memory and the gradients will be characterized only by indefinite values.

The long short-term Memory (LSTM) represents a particular recurrent network proposed by Hochreiter and Schmidhuber [191] in order to solve the vanishing and exploding gradient problem that compromised the effectiveness of RNNs. The principle behind LSTM is the memory cell, which keeps the state out of the normal flow of the recurring network. The state in fact has a direct connection with itself as it is simply a sum between the previous state and the current input processing, modulated through masks so that only part of them is remembered. Since the activation function for updating the status is in fact the identity, the derivative will be equal to 1, and therefore the gradient in the backpropagation will not vanish or explode but will remain constant through all the time instants of the unfolded network [192].

For the modulation of the data, the implementation of the LSTM foresees the presence of layers called gates, which apply an activation function of the sigmoid type to the concatenation of the input and output, obtaining an output vector whose elements belong to the range (0; 1). This vector can therefore be used as a mask to be multiplied elementwise by the vectors concerned. Using a sigmoid instead of a simple binary-step is more effective, both because it is differentiable and because it indicates a quantity of the data to remember instead of offering a dichotomous output that only indicates whether to remember or not to remember [193]. From a structural point of view, the standard LSTM network consists of four fundamental elements:

- Cell status: represents the element that carries the information that can be modified by the gates.
- Forget gate: represents the bridge where information passes through the sigmoid function. The output values are between 0 and 1. If the value is closer to 0, it means forget the information, if it is closer to 1, it means keep the information.
- Input gate: represents the element that updates the state of the cell. The sigmoid function is still present, but in combination with the tanh function. The tanh function reduces the input values to between -1 and 1. Then the output of tanh is multiplied with the output of the sigmoid function, which will decide what information is important to keep from the tanh output.
- Output gate: represents the element that decides how the next hidden state should be, which remembers the information on previous inputs to subsequent time cells. First, the previous hidden state and the current input pass into a sigmoid function, whose output, that is, the cell state just modified, passes through the tanh function. This output is then multiplied with the sigmoid output to decide what information should be contained in the hidden state. The new cell state and the new hidden state are then carried over to the next time step.

Unlike RNNs, LSTMs, thanks to the use of memory cells and gate functions, which can be considered as neurons, remember information for long periods. These structures naturally lend themselves to the processing of data with temporal characteristics such as those relating to time series [194].

Connor et al. [195] used recurrent neural networks to make predictions on the electric load time series. The authors first filtered the outliers from the data and then estimated the parameters from the filtered data. Qin et al. [196] developed a two-phase recurring network model to address the long-term dependency capture problems of time series and to select essential elements. Che et al. [197] proposed a model based on recurrent neural

networks for dealing with multivariate time series with missing values. The authors used partially masked data at different time intervals and modeled it to capture long-term time dependencies to improve predictions. Chandra et al. [198] used Elman's recurrent neural networks [199] for the prediction of chaotic time series. In this work, the RNNs have been applied through a cooperative coevolution process that breaks down a problem into subcomponents and employs evolutionary algorithms to solve them. Hüsken et al. [200] exploited RNNs to classify input sequences into different specified classes. The authors enhanced the network learning process with auxiliary training and used the network output to estimate the degree of reliability of the classification result. Hermans et al. [201] investigated the effect of a hierarchy of recurrent neural networks on time series processing. The architecture of the model developed provides for a series of levels based on a recurring network that receives the hidden state of the previous level as input. This model allows you to perform hierarchical processing on difficult time tasks and to acquire the structure of time series in a more natural way.

The scientific community has also extensively used LSTMs for the prediction of time series. Hua et al. [202] used LSTMs for time series prediction, applying stochastic connectivity to conventional LSTM neurons. Song et al. [203] used LSTMs for forecasting oil production. To extract useful information on oil extraction from fractured horizontal wells in a volcanic reservoir, the authors applied the particle swarm optimization (PSO) algorithm to optimize the essential configuration of the LSTM model. Yang et al. [204] applied LSTM-based algorithms to predict the displacement of landslides. The movement of landslides accumulated over time has been broken down into a trend term and a periodic term. A cubic polynomial function was selected to predict the trend shift. Sahoo et al. [205] developed an LSTM-based model for making predictions of low-flow hydrological time series. The authors applied this model to the daily downloaded data collected from the Basantapur measurement station located in the Mahanadi River basin in India. Benhaddi et al. [206] made predictions of urban air quality by exploiting the potential of recurrent and convolutional networks, comparing the results. Kong et al. [207] exploited the features of LSTMs to analyze changes in satellite images. Fires, floods, deforestation, and urbanization can be considered as disturbances in the slow natural change in the Earth's surface, and as such can be detected by an LSTM-based algorithm. Lei et al. [208] applied LSTMs to diagnose wind turbine failures. The model developed by the authors can capture the long-term dependencies of the data stored by the turbine, through recurrent behaviors and gate mechanisms.

3.5. Autoencoders Algorithms in Time Series Data Processing

Autoencoder (AE) has played a very important role in the area of machine learning. Not only in the reduction of dimensionality, but also as an initializer of the weights of deep neural networks [209]. An autoencoder represents a network made up of two substructures, an encoder and a decoder: the encoder receives the data in input and produces a compressed representation, the decoder receives the aforementioned representation in input and reconstructs the source data. The fundamental objective of an autoencoder is therefore to learn an efficient coding of a set of data, and generally to produce a reconstruction that respects some fundamental properties. It is an artificial neural network architecture very similar to that of MLP, the main difference of which is in the hidden layer, where there are fewer neurons than in the output layer. This type of architecture can have a greater number of hidden layers but maintain the symmetry [210]. This type of network aims to learn the compressed and distributed representation of a dataset, in other words, the output is a reconstruction of the input, but having fewer neurons in the hidden layers compresses and reduces the dimensionality of the data. To train an autoencoder, the backpropagation algorithm is used, but in an unsupervised way, therefore, the network receives the data, compresses it, and brings it back to its size, and the weights are updated taking into account that the desired outputs are the same inputs, based on global error. In most cases, it achieves optimal parameter values, which directly influence any subsequent activity. If the capacity of the autoencoder, specifically of the encoder and decoder, is too large compared

to that of the latent space, the network could return a good output, but not be able to extract any information from the data [211]. That is, he would have learned to copy the input and return it to the output without learning any features of the data. Starting from an autoencoder, it is possible to define a variation, called denoising autoencoder (DAE) [212], whose purpose is to learn and eliminate a noise component in the input images.

Variational autoencoders (VAEs) have been found to be one of the most popular unsupervised learning approaches [213]. They take the name of autoencoder since, like these, they consist of an encoder and a decoder. This type of network is called generative model as it allows to obtain a distribution of a model, defined on a set of data in a high-dimensional space. The goal, however, is not to reproduce exactly the data received, but to generate many examples that are similar to those already present in the database. The training of a VAE is fast and is performed through backpropagation.

Mehdiyev et al. [214] have exploited autoencoders (AE) for the classification of time series data detected in the process industry. This is a study on the multivariate classification of time series that proposes a framework for extracting functionality in an unsupervised way. LSTMs and autoencoders are combined for compressed data representation, which is then sent to a deep feedforward neural network for classification. Corizzo et al. [215] used autoencoders for gravitational wave encoding. Gravitational waves are detected on a daily basis, giving rise to a large amount of data that are difficult to analyze with traditional techniques. The authors developed models based on deep autoencoders to analyze these data and associate a classification label with them. Yang et al. [216] investigated a new method for classifying arrhythmia in the electrocardiogram based on autoencoders. Arrhythmia detection is a complex procedure due to the presence of heartbeats that partially cover the signal. The authors used autoencoders to hierarchically extract high-level functionality from huge amounts of data. Rußwurm et al. [217] applied autoencoders for the classification of land use of the Earth. Modern sensors are capable of detecting a large amount of data with time series characteristics. The authors developed an encoder-based model with convolutional recurrent layers to approximate a phenological model for vegetation classes. The input data were obtained from the images detected by the Sentinel 2 satellite.

3.6. Automated Features Extraction from Time Series Data

Feature extraction is a methodology that returns the best features that solve a specific problem starting from data. New features are created that encapsulate the central properties of a dataset and represent it in a low-size space that facilitates the learning process. The use of this method is necessary in the case of a dataset too large with too burdensome computational costs. By resorting to the feature extraction, you get a subset of more manageable data but equally representative of the input variables. The extraction of these data can help you make better decisions, identifying the connections that led to an output starting from an input [218].

Feature extraction maps the signals acquired in an M-dimensional space in an N-dimensional space, with $n < m$, in order to provide a synthetic representation of the raw signals, preserving the original structure and content information about it. In this process, significant signal characteristics that allow the different behaviors of the machine in question are extracted.

Data mining tools widely used by the scientific community show the obvious limits in the processing of time series data. In these cases, it is necessary to reduce the dimensions through techniques of extraction of the characteristics and map each time series in a lower-dimensional space. For time series, this methodology can be performed using various data analysis techniques and decomposition. In addition, the characteristics can be extracted using comparison techniques of sequences and techniques of discovery of subsequences [219].

The feature extraction procedure is essential in the machine learning-based algorithms. Effectively identifying the features that best describe the characteristics of the data will decisively help to extract knowledge from the data. It is to be specified that the deep

learning-based algorithms do not require the extraction of the features, this is because the input filters are able to identify effects in an automatic way, identifying those that represent the best features able to represent the phenomenon under investigation.

4. Conclusions

The use of time series to predict the behavior of a system over time is a consolidated approach in various sectors of the production world. Time series data show a strong dependence between successive observations, which can be exploited to predict possible future trends. In machine learning, knowledge is extracted through an inductive process: the input provided by the data is combined with some examples of the expected output, so that in the training phase the machine can learn to recognize patterns from the structures it receives. The time series shows recurring structures that such models can recognize and exploit to build predictions about the future behavior of the system.

Over the years, researchers have leveraged machine learning-based algorithms to extract knowledge about time series data. Algorithms based on artificial neural networks were initially widely used. It was immediately seen that to take advantage of these algorithms it is necessary to proceed in advance with the subdivision of the input data: the set of data is then divided into sequential groups with a fixed number of components. To improve the model's performance, researchers have often adopted a hybrid model that exploits traditional statistical models with ANNs [220]. Subsequently, the focus was on unsupervised learning models that allow the extraction of recurring structures in the data without the need for prior labeling. For this type of model, the choice of a method for measuring the distances between data that can consider the dynamic characteristic of the data is crucial. The adoption of distance measurement metrics that consider the interdependence relationships between values is essential.

The development of models based on deep learning has also contributed massively to the use of these algorithms for the prediction of time series. At first, the researchers focused on convolutional networks, exploiting these algorithms to extract knowledge from spatial and temporal models with the use of appropriate trainable filters. These filters, which in traditional classification models must be processed in a preventive manual supervision phase, in the case of convolutional networks are learned automatically by the same algorithm. Subsequently, attention turned to models based on recurring networks. RNNs are equipped with feedback connections that allow you to analyze time sequences. The essential feature of these models is the ability to keep the state in memory from one iteration to the next, using its output as input for the next step. An RNN then receives the inputs, produces an output, and sends the latter back to itself. This structure made it possible to develop models that returned time series forecasts with excellent performance. The accuracy of these methods is very high and depends on the method used and how the data were preprocessed. In all cases, it is greater than 90%, and in the case of methods based on CNN and RNN, it reaches even 99%.

The studies presented in this work have highlighted the great potential that the methods based on machine learning have in the treatment of time series. Scrolling through the chronology of the researchers' activities, we could see that these methods have improved over time, providing more and more precise results. A great contribution to this advance was provided by algorithms based on deep learning, which relieved researchers from the onerous task of feature extraction. In fact, in these models, the extraction of the features takes place automatically, thus succeeding in capturing correlations that would otherwise be difficult to identify. Future advances will most likely be directed towards the search for faster learning processes, also given the technological evolution of processors that allow for a significant reduction in computational times. Another point that may interest scholars may concern the interpretability of the results provided by these models. Algorithms based on artificial neural networks have been associated with the concept of the black box. This is because it was not clear in a mathematical way what was the process that led to the formation of the output once an input was sent: developing easily interpretable

models represents a future challenge. Finally, we have seen that several authors have used hybrid models to exploit the characteristics of different algorithms in order to create a better performing model. This approach will certainly lead to an evolution of methodologies by exploiting the potential offered by reinforcement learning and meta-learning.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wei, W.W. Time series analysis. In *The Oxford Handbook of Quantitative Methods in Psychology*; Oxford University Press: New York, NY, USA, 2006; Volume 2.
2. Lütkepohl, H. *New Introduction to Multiple Time Series Analysis*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2005.
3. Chatfield, C.; Xing, H. *The Analysis of Time Series: An Introduction with R*; CRC Press: Boca Raton, FL, USA, 2019.
4. Hamilton, J.D. *Time Series Analysis*; Princeton University Press: Princeton, NJ, USA, 2020.
5. Brillinger, D.R. *Time Series: Data Analysis and Theory*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2001.
6. Granger, C.W.J.; Newbold, P. *Forecasting Economic Time Series*; Academic Press: Cambridge, MA, USA, 2014.
7. Cryer, J.D. *Time Series Analysis*; Duxbury Press: Boston, MA, USA, 1986; Volume 286.
8. Box, G.E.; Jenkins, G.M.; Reinsel, G.C.; Ljung, G.M. *Time Series Analysis: Forecasting and Control*; John Wiley & Sons: Hoboken, NJ, USA, 2015.
9. Madsen, H. *Time Series Analysis*; CRC Press: Boca Raton, FL, USA, 2007.
10. Fuller, W.A. *Introduction to Statistical Time Series*; John Wiley & Sons: Hoboken, NJ, USA, 2009; Volume 428.
11. Tsay, R.S. *Analysis of Financial Time Series*; John Wiley & Sons: Hoboken, NJ, USA, 2005; Volume 543.
12. Harvey, A.C. *Forecasting, Structural Time Series Models and the Kalman Filter*; Cambridge University Press: Cambridge, UK, 1990.
13. Kantz, H.; Schreiber, T. *Nonlinear Time Series Analysis*; Cambridge University Press: Cambridge, UK, 2004; Volume 7.
14. Shumway, R.H.; Stoffer, D.S.; Stoffer, D.S. *Time Series Analysis and Its Applications*; Springer: New York, NY, USA, 2000; Volume 3.
15. Fahrmeir, L.; Tutz, G.; Hennevogl, W.; Salem, E. *Multivariate Statistical Modelling Based on Generalized Linear Models*; Springer: New York, NY, USA, 1990; Volume 425.
16. Kirchgässner, G.; Wolters, J.; Hassler, U. *Introduction to Modern Time Series Analysis*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012.
17. Hannan, E.J. *Multiple Time Series*; John Wiley & Sons: Hoboken, NJ, USA, 2009; Volume 38.
18. Brown, R.G. *Smoothing, Forecasting and Prediction of Discrete Time Series*; Courier Corporation: Chelmsford, MA, USA, 2004.
19. Rao, S.S. *A Course in Time Series Analysis*; Technical Report; Texas A & M University: College Station, TX, USA, 2008.
20. Schreiber, T.; Schmitz, A. Surrogate Time Series. *Phys. D Nonlinear Phenom.* **2000**, *142*, 346–382. [CrossRef]
21. Zhang, C.; Ma, Y. (Eds.) *Ensemble Machine Learning: Methods and Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012.
22. Gollapudi, S. *Practical Machine Learning*; Packt Publishing Ltd.: Birmingham, UK, 2016.
23. Paluszek, M.; Thomas, S. *MATLAB Machine Learning*; Apress: New York, NY, USA, 2016.
24. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*; MIT Press: Cambridge, MA, USA, 2012.
25. Adeli, H.; Hung, S.L. *Machine Learning: Neural Networks, Genetic Algorithms, and Fuzzy Systems*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 1994.
26. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2006.
27. Hutter, F.; Kotthoff, L.; Vanschoren, J. *Automated Machine Learning: Methods, Systems, Challenges*; Springer Nature: Berlin/Heidelberg, Germany, 2019; p. 219.
28. Ciaburro, G.; Iannace, G.; Passaro, J.; Bifulco, A.; Marano, D.; Guida, M.; Branda, F. Artificial neural network-based models for predicting the sound absorption coefficient of electrospun poly (vinyl pyrrolidone)/silica composite. *Appl. Acoust.* **2020**, *169*, 107472. [CrossRef]
29. Lantz, B. *Machine Learning with R: Expert Techniques for Predictive Modeling*; Packt Publishing Ltd.: Birmingham, UK, 2019.
30. Dangeti, P. *Statistics for Machine Learning*; Packt Publishing Ltd.: Birmingham, UK, 2017.
31. Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; Volume 1, No. 2.
32. Koustas, Z.; Veloce, W. Unemployment hysteresis in Canada: An approach based on long-memory time series models. *Appl. Econ.* **1996**, *28*, 823–831. [CrossRef]
33. Teyssière, G.; Kirman, A.P. (Eds.) *Long Memory in Economics*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2006.
34. Gradišek, J.; Siegert, S.; Friedrich, R.; Grabec, I. Analysis of time series from stochastic processes. *Phys. Rev. E* **2000**, *62*, 3146. [CrossRef] [PubMed]
35. Grenander, U.; Rosenblatt, M. Statistical spectral analysis of time series arising from stationary stochastic processes. *Ann. Math. Stat.* **1953**, *24*, 537–558. [CrossRef]

36. Alessio, E.; Carbone, A.; Castelli, G.; Frappietro, V. Second-order moving average and scaling of stochastic time series. *Eur. Phys. J. B Condens. Matter Complex Syst.* **2002**, *27*, 197–200. [CrossRef]
37. Klöckl, B.; Papaefthymiou, G. Multivariate time series models for studies on stochastic generators in power systems. *Electr. Power Syst. Res.* **2010**, *80*, 265–276. [CrossRef]
38. Harvey, A.; Ruiz, E.; Sentana, E. Unobserved component time series models with ARCH disturbances. *J. Econom.* **1992**, *52*, 129–157. [CrossRef]
39. Nelson, C.R.; Plosser, C.R. Trends and random walks in macroeconmic time series: Some evidence and implications. *J. Monet. Econ.* **1982**, *10*, 139–162. [CrossRef]
40. Shephard, N.G.; Harvey, A.C. On the probability of estimating a deterministic component in the local level model. *J. Time Ser. Anal.* **1990**, *11*, 339–347. [CrossRef]
41. Duarte, F.S.; Rios, R.A.; Hruschka, E.R.; de Mello, R.F. Decomposing time series into deterministic and stochastic influences: A survey. *Digit. Signal Process.* **2019**, *95*, 102582. [CrossRef]
42. Rios, R.A.; De Mello, R.F. Improving time series modeling by decomposing and analyzing stochastic and deterministic influences. *Signal Process.* **2013**, *93*, 3001–3013. [CrossRef]
43. Franzini, L.; Harvey, A.C. Testing for deterministic trend and seasonal components in time series models. *Biometrika* **1983**, *70*, 673–682. [CrossRef]
44. Time Series Data Library. Available online: https://pkg.yangzhuoranyang.com/tsdl/ (accessed on 24 March 2021).
45. Granger, C.W.J.; Hatanaka, M. *Spectral Analysis of Economic Time Series. (PSME-1)*; Princeton University Press: Princeton, NJ, USA, 2015.
46. Durbin, J.; Koopman, S.J. *Time Series Analysis by State Space Methods*; Oxford University Press: Oxford, UK, 2012.
47. Gourieroux, C.; Wickens, M.; Ghysels, E.; Smith, R.J. *Applied Time Series Econometrics*; Cambridge University Press: Cambridge, UK, 2004.
48. Longobardi, A.; Villani, P. Trend analysis of annual and seasonal rainfall time series in the Mediterranean area. *Int. J. Climatol.* **2010**, *30*, 1538–1546. [CrossRef]
49. Hylleberg, S. *Modelling Seasonality*; Oxford University Press: Oxford, UK, 1992.
50. Beveridge, S.; Nelson, C.R. A new approach to decomposition of economic time series into permanent and transitory components with particular attention to measurement of the 'business cycle'. *J. Monet. Econ.* **1981**, *7*, 151–174. [CrossRef]
51. Adhikari, R.; Agrawal, R.K. An introductory study on time series modeling and forecasting. *arXiv* **2013**, arXiv:1302.6613.
52. Maçaira, P.M.; Thomé, A.M.T.; Oliveira, F.L.C.; Ferrer, A.L.C. Time series analysis with explanatory variables: A systematic literature review. *Environ. Model. Softw.* **2018**, *107*, 199–209. [CrossRef]
53. Box-Steffensmeier, J.M.; Freeman, J.R.; Hitt, M.P.; Pevehouse, J.C. *Time Series Analysis for the Social Sciences*; Cambridge University Press: Cambridge, UK, 2014.
54. Box, G. Box, G. Box and Jenkins: Time series analysis, forecasting and control. In *A Very British Affair*; Palgrave Macmillan: London, UK, 2013; pp. 161–215.
55. Hagan, M.T.; Behr, S.M. The time series approach to short term load forecasting. *IEEE Trans. Power Syst.* **1987**, *2*, 785–791. [CrossRef]
56. Velasco, C. Gaussian semiparametric estimation of non-stationary time series. *J. Time Ser. Anal.* **1999**, *20*, 87–127. [CrossRef]
57. Dau, H.A.; Bagnall, A.; Kamgar, K.; Yeh, C.C.M.; Zhu, Y.; Gharghabi, S.; Keogh, E. The UCR time series archive. *IEEE CAA J. Autom. Sin.* **2019**, *6*, 1293–1305. [CrossRef]
58. Dau, H.A.; Keogh, E.; Kamgar, K.; Yeh, C.M.; Zhu, Y.; Gharghabi, S.; Ratanamahatana, C.A.; Chen, Y.; Hu, B.; Begum, N.; et al. The UCR Time Series Classification Archive. 2019. Available online: https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ (accessed on 24 March 2021).
59. Kleiber, C.; Zeileis, A. *Applied Econometrics with R*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2008.
60. Kleiber, C.; Zeileis, A.; Zeileis, M.A. *Package 'AER'*, R package version 1.2, 9; R Foundation for Statistical Computing: Vienna, Austria, 2020; Available online: http://www.R-project.org/ (accessed on 24 May 2021).
61. Graves, S.; Boshnakov, G.N. *'FinTS' Package*, R package version 0.4-6; R Foundation for Statistical Computing: Vienna, Austria, 2019; Available online: http://www.R-project.org/ (accessed on 24 May 2021).
62. Croissant, Y.; Graves, M.S. *'Ecdat' Package*, R package version 0.3-9; R Foundation for Statistical Computing: Vienna, Austria, 2020; Available online: http://www.R-project.org/ (accessed on 24 May 2021).
63. ANES Time Series Study. Available online: https://electionstudies.org/data-center/ (accessed on 23 March 2021).
64. Schlittgen, R.; Sattarhoff, C. 9 Regressionsmodelle für Zeitreihen. In *Angewandte Zeitreihenanalyse mit R*; De Gruyter Oldenbourg: Berlin, Germany, 2020; pp. 225–246.
65. Harvard Dataverse. Available online: https://dataverse.harvard.edu/ (accessed on 24 March 2021).
66. Data.gov. Available online: https://www.data.gov/ (accessed on 24 March 2021).
67. Dua, D.; Graff, C. *UCI Machine Learning Repository*; University of California, School of Information and Computer Science: Irvine, CA, USA, 2019; Available online: http://archive.ics.uci.edu/ml (accessed on 24 March 2021).
68. Cryer, J.D.; Chan, K. *Time Series Analysis with Applications in R*; Springer: Berlin/Heidelberg, Germany, 2008.
69. Chan, K.S.; Ripley, B.; Chan, M.K.S.; Chan, S. *Package 'TSA'*, R package version 1.3; R Foundation for Statistical Computing: Vienna, Austria, 2020; Available online: http://www.R-project.org/ (accessed on 24 May 2021).

70. Google Dataset Search. Available online: https://datasetsearch.research.google.com/ (accessed on 24 March 2021).
71. Shumway, R.H.; Stoffer, D.S. *Time Series Analysis and Its Applications: With R Examples*; Springer: New York, NY, USA, 2017.
72. Stoffer, D. *Astsa: Applied Statistical Time Series Analysis*, R package version 1.7; R Foundation for Statistical Computing: Vienna, Austria, 2016. Available online: http://www.R-project.org/ (accessed on 24 May 2021).
73. Kaggle Dataset. Available online: https://www.kaggle.com/datasets (accessed on 23 March 2021).
74. Hyndman, M.R.J.; Akram, M.; Bergmeir, C.; O'Hara-Wild, M.; Hyndman, M.R. *Package 'Mcomp'*, R package version 2.8; R Foundation for Statistical Computing: Vienna, Austria, 2018; Available online: http://www.R-project.org/ (accessed on 24 May 2021).
75. Makridakis, S.; Andersen, A.; Carbone, R.; Fildes, R.; Hibon, M.; Lewandowski, R.; Winkler, R. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *J. Forecast.* **1982**, *1*, 111–153. [CrossRef]
76. Makridakis, S.; Hibon, M. The M3-Competition: Results, conclusions and implications. *Int. J. Forecast.* **2000**, *16*, 451–476. [CrossRef]
77. BenTaieb, S. *Package 'M4comp'*, R package version 0.0.1; R Foundation for Statistical Computing: Vienna, Austria, 2016; Available online: http://www.R-project.org/ (accessed on 24 May 2021).
78. Ciaburro, G. Sound event detection in underground parking garage using convolutional neural network. *Big Data Cogn. Comput.* **2020**, *4*, 20. [CrossRef]
79. Mohri, M.; Rostamizadeh, A.; Talwalkar, A. *Foundations of Machine Learning*; MIT Press: Cambridge, MA, USA, 2018.
80. Caruana, R.; Niculescu-Mizil, A. An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 161–168.
81. Celebi, M.E.; Aydin, K. (Eds.) *Unsupervised Learning Algorithms*; Springer International Publishing: Berlin, Germany, 2016.
82. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
83. Abiodun, O.I.; Jantan, A.; Omolara, A.E.; Dada, K.V.; Mohamed, N.A.; Arshad, H. State-of-the-art in artificial neural network applications: A survey. *Heliyon* **2018**, *4*, e00938. [CrossRef] [PubMed]
84. Ciaburro, G.; Iannace, G.; Ali, M.; Alabdulkarem, A.; Nuhait, A. An Artificial neural network approach to modelling absorbent asphalts acoustic properties. *J. King Saud Univ. Eng. Sci.* **2021**, *33*, 213–220. [CrossRef]
85. Da Silva, I.N.; Spatti, D.H.; Flauzino, R.A.; Liboni, L.H.B.; dos Reis Alves, S.F. Artificial neural network architectures and training processes. In *Artificial Works*; Springer: Cham, Switzerland, 2017; pp. 21–28.
86. Fabio, S.; Giovanni, D.N.; Mariano, P. Airborne sound insulation prediction of masonry walls using artificial neural networks. *Build. Acoust.* **2021**. [CrossRef]
87. Alanis, A.Y.; Arana-Daniel, N.; Lopez-Franco, C. (Eds.) *Artificial Neural Networks for Engineering Applications*; Academic Press: Cambridge, MA, USA, 2019.
88. Romero, V.P.; Maffei, L.; Brambilla, G.; Ciaburro, G. Modelling the soundscape quality of urban waterfronts by artificial neural networks. *Appl. Acoust.* **2016**, *111*, 121–128. [CrossRef]
89. Walczak, S. Artificial neural networks. In *Advanced Methodologies and Technologies in Artificial Intelligence, Computer Simulation, and Human-Computer Interaction*; IGI Global: Hershey, PA, USA, 2019; pp. 40–53.
90. Al-Massri, R.; Al-Astel, Y.; Ziadia, H.; Mousa, D.K.; Abu-Naser, S.S. Classification Prediction of SBRCTs Cancers Using Artificial Neural Network. *Int. J. Acad. Eng. Res.* **2018**, *2*, 1–7.
91. Wang, L.; Wang, Z.; Qu, H.; Liu, S. Optimal forecast combination based on neural networks for time series forecasting. *Appl. Soft Comput.* **2018**, *66*, 1–17. [CrossRef]
92. Gholami, V.; Torkaman, J.; Dalir, P. Simulation of precipitation time series using tree-rings, earlywood vessel features, and artificial neural network. *Theor. Appl. Climatol.* **2019**, *137*, 1939–1948. [CrossRef]
93. Vochozka, M.; Horák, J.; Šuleř, P. Equalizing seasonal time series using artificial neural networks in predicting the Euro–Yuan exchange rate. *J. Risk Financ. Manag.* **2019**, *12*, 76. [CrossRef]
94. Olawoyin, A.; Chen, Y. Predicting the future with artificial neural network. *Procedia Comput. Sci.* **2018**, *140*, 383–392. [CrossRef]
95. Adeyinka, D.A.; Muhajarine, N. Time series prediction of under-five mortality rates for Nigeria: Comparative analysis of artificial neural networks, Holt-Winters exponential smoothing and autoregressive integrated moving average models. *BMC Med. Res. Methodol.* **2020**, *20*, 1–11. [CrossRef] [PubMed]
96. Azadeh, A.; Ghaderi, S.F.; Sohrabkhani, S. Forecasting electrical consumption by integration of neural network, time series and ANOVA. *Appl. Math. Comput.* **2007**, *186*, 1753–1761. [CrossRef]
97. Miller, R.G., Jr. *Beyond ANOVA: Basics of Applied Statistics*; CRC Press: Boca Raton, FL, USA, 1997.
98. Hill, T.; O'Connor, M.; Remus, W. Neural network models for time series forecasts. *Manag. Sci.* **1996**, *42*, 1082–1092. [CrossRef]
99. Zhang, G.P. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing* **2003**, *50*, 159–175. [CrossRef]
100. Contreras, J.; Espinola, R.; Nogales, F.J.; Conejo, A.J. ARIMA models to predict next-day electricity prices. *IEEE Trans. Power Syst.* **2003**, *18*, 1014–1020. [CrossRef]
101. Jain, A.; Kumar, A.M. Hybrid neural network models for hydrologic time series forecasting. *Appl. Soft Comput.* **2007**, *7*, 585–592. [CrossRef]
102. Tseng, F.M.; Yu, H.C.; Tzeng, G.H. Combining neural network model with seasonal time series ARIMA model. *Technol. Forecast. Soc. Chang.* **2002**, *69*, 71–87. [CrossRef]

103. Chen, C.F.; Chang, Y.H.; Chang, Y.W. Seasonal ARIMA forecasting of inbound air travel arrivals to Taiwan. *Transportmetrica* **2009**, *5*, 125–140. [CrossRef]
104. Khashei, M.; Bijari, M. An artificial neural network (p, d, q) model for timeseries forecasting. *Expert Syst. Appl.* **2010**, *37*, 479–489. [CrossRef]
105. Chaudhuri, T.D.; Ghosh, I. Artificial neural network and time series modeling based approach to forecasting the exchange rate in a multivariate framework. *arXiv* **2016**, arXiv:1607.02093.
106. Aras, S.; Kocakoç, İ.D. A new model selection strategy in time series forecasting with artificial neural networks: IHTS. *Neurocomputing* **2016**, *174*, 974–987. [CrossRef]
107. Doucoure, B.; Agbossou, K.; Cardenas, A. Time series prediction using artificial wavelet neural network and multi-resolution analysis: Application to wind speed data. *Renew. Energy* **2016**, *92*, 202–211. [CrossRef]
108. Lohani, A.K.; Kumar, R.; Singh, R.D. Hydrological time series modeling: A comparison between adaptive neuro-fuzzy, neural network and autoregressive techniques. *J. Hydrol.* **2012**, *442*, 23–35. [CrossRef]
109. Chicea, D.; Rei, S.M. A fast artificial neural network approach for dynamic light scattering time series processing. *Meas. Sci. Technol.* **2018**, *29*, 105201. [CrossRef]
110. Horák, J.; Krulický, T. Comparison of exponential time series alignment and time series alignment using artificial neural networks by example of prediction of future development of stock prices of a specific company. In *SHS Web of Conferences*; EDP Sciences: Les Ulis, France, 2019; Volume 61, p. 01006.
111. Liu, H.; Tian, H.Q.; Pan, D.F.; Li, Y.F. Forecasting models for wind speed using wavelet, wavelet packet, time series and Artificial Neural Networks. *Appl. Energy* **2013**, *107*, 191–208. [CrossRef]
112. Wang, C.C.; Kang, Y.; Shen, P.C.; Chang, Y.P.; Chung, Y.L. Applications of fault diagnosis in rotating machinery by using time series analysis with neural network. *Expert Syst. Appl.* **2010**, *37*, 1696–1702. [CrossRef]
113. Xu, R.; Wunsch, D. *Clustering*; John Wiley & Sons: Hoboken, NJ, USA, 2008; Volume 10.
114. Rokach, L.; Maimon, O. Clustering methods. In *Data Mining and Knowledge Discovery Handbook*; Springer: Boston, MA, USA, 2005; pp. 321–352.
115. Gaertler, M. Clustering. In *Network Analysis*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 178–215.
116. Gionis, A.; Mannila, H.; Tsaparas, P. Clustering aggregation. *Acm Trans. Knowl. Discov. Data* **2007**, *1*, 4-es. [CrossRef]
117. Vesanto, J.; Alhoniemi, E. Clustering of the self-organizing map. *IEEE Trans. Neural Netw.* **2000**, *11*, 586–600. [CrossRef]
118. Jain, A.K. Data clustering: 50 years beyond K-means. *Pattern Recognit. Lett.* **2010**, *31*, 651–666. [CrossRef]
119. Mirkin, B. *Clustering: A Data Recovery Approach*; CRC Press: Boca Raton, FL, USA, 2012.
120. Forina, M.; Armanino, C.; Raggio, V. Clustering with dendrograms on interpretation variables. *Anal. Chim. Acta* **2002**, *454*, 13–19. [CrossRef]
121. Hirano, S.; Tsumoto, S. Cluster analysis of time-series medical data based on the trajectory representation and multiscale comparison techniques. In Proceedings of the Sixth International Conference on Data Mining (ICDM'06), Hong Kong, China, 18–22 December 2006; IEEE: Piscataway, NJ, USA, 2006; pp. 896–901.
122. Caraway, N.M.; McCreight, J.L.; Rajagopalan, B. Multisite stochastic weather generation using cluster analysis and k-nearest neighbor time series resampling. *J. Hydrol.* **2014**, *508*, 197–213. [CrossRef]
123. Balslev, D.; Nielsen, F.Å.; Frutiger, S.A.; Sidtis, J.J.; Christiansen, T.B.; Svarer, C.; Law, I. Cluster analysis of activity-time series in motor learning. *Hum. Brain Mapp.* **2002**, *15*, 135–145. [CrossRef] [PubMed]
124. Mikalsen, K.Ø.; Bianchi, F.M.; Soguero-Ruiz, C.; Jenssen, R. Time series cluster kernel for learning similarities between multivariate time series with missing data. *Pattern Recognit.* **2018**, *76*, 569–581. [CrossRef]
125. Corduas, M.; Piccolo, D. Time series clustering and classification by the autoregressive metric. *Comput. Stat. Data Anal.* **2008**, *52*, 1860–1872. [CrossRef]
126. Otranto, E.; Trudda, A. Classifying Italian pension funds via GARCH distance. In *Mathematical and Statistical Methods in Insurance and Finance*; Springer: Milano, Italy, 2008; pp. 189–197.
127. Gupta, S.K.; Gupta, N.; Singh, V.P. Variable-Sized Cluster Analysis for 3D Pattern Characterization of Trends in Precipitation and Change-Point Detection. *J. Hydrol. Eng.* **2021**, *26*, 04020056. [CrossRef]
128. Iglesias, F.; Kastner, W. Analysis of similarity measures in times series clustering for the discovery of building energy patterns. *Energies* **2013**, *6*, 579–597. [CrossRef]
129. Gopalapillai, R.; Gupta, D.; Sudarshan, T.S.B. Experimentation and analysis of time series data for rescue robotics. In *Recent Advances in Intelligent Informatics*; Springer: Cham, Switzerland, 2014; pp. 443–453.
130. Wismüller, A.; Lange, O.; Dersch, D.R.; Leinsinger, G.L.; Hahn, K.; Pütz, B.; Auer, D. Cluster analysis of biomedical image time-series. *Int. J. Comput. Vis.* **2002**, *46*, 103–128. [CrossRef]
131. Guo, C.; Jia, H.; Zhang, N. Time series clustering based on ICA for stock data analysis. In Proceedings of the 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing, Dalian, China, 12–14 October 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 1–4.
132. Stone, J.V. Independent component analysis: An introduction. *Trends Cogn. Sci.* **2002**, *6*, 59–64. [CrossRef]
133. Lee, D.; Baek, S.; Sung, K. Modified k-means algorithm for vector quantizer design. *IEEE Signal Process. Lett.* **1997**, *4*, 2–4.
134. Shumway, R.H. Time-frequency clustering and discriminant analysis. *Stat. Probab. Lett.* **2003**, *63*, 307–314. [CrossRef]

135. Elangasinghe, M.A.; Singhal, N.; Dirks, K.N.; Salmond, J.A.; Samarasinghe, S. Complex time series analysis of PM10 and PM2. 5 for a coastal site using artificial neural network modelling and k-means clustering. *Atmos. Environ.* **2014**, *94*, 106–116. [CrossRef]
136. Möller-Levet, C.S.; Klawonn, F.; Cho, K.H.; Wolkenhauer, O. Fuzzy clustering of short time-series and unevenly distributed sampling points. In *International Symposium on Intelligent Data Analysis*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 330–340.
137. Rebbapragada, U.; Protopapas, P.; Brodley, C.E.; Alcock, C. Finding anomalous periodic time series. *Mach. Learn.* **2009**, *74*, 281–313. [CrossRef]
138. Paparrizos, J.; Gravano, L. Fast and accurate time-series clustering. *ACM Trans. Database Syst.* **2017**, *42*, 1–49. [CrossRef]
139. Paparrizos, J.; Gravano, L. K-shape: Efficient and accurate clustering of time series. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Australia, 31 May–4 June 2015; pp. 1855–1870.
140. Sardá-Espinosa, A. Comparing time-series clustering algorithms in r using the dtwclust package. *R Package Vignette* **2017**, *12*, 41.
141. Zhang, Q.; Wu, J.; Zhang, P.; Long, G.; Zhang, C. Salient subsequence learning for time series clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *41*, 2193–2207. [CrossRef]
142. Chen, T.; Shi, X.; Wong, Y.D. A lane-changing risk profile analysis method based on time-series clustering. *Phys. A Stat. Mech. Appl.* **2021**, *565*, 125567. [CrossRef]
143. Steinmann, P.; Auping, W.L.; Kwakkel, J.H. Behavior-based scenario discovery using time series clustering. *Technol. Forecast. Soc. Chang.* **2020**, *156*, 120052. [CrossRef]
144. Kuschnerus, M.; Lindenbergh, R.; Vos, S. Coastal change patterns from time series clustering of permanent laser scan data. *Earth Surf. Dyn.* **2021**, *9*, 89–103. [CrossRef]
145. Motlagh, O.; Berry, A.; O'Neil, L. Clustering of residential electricity customers using load time series. *Appl. Energy* **2019**, *237*, 11–24. [CrossRef]
146. Hallac, D.; Vare, S.; Boyd, S.; Leskovec, J. Toeplitz inverse covariance-based clustering of multivariate time series data. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 215–223.
147. McDowell, I.C.; Manandhar, D.; Vockley, C.M.; Schmid, A.K.; Reddy, T.E.; Engelhardt, B.E. Clustering gene expression time series data using an infinite Gaussian process mixture model. *PLoS Comput. Biol.* **2018**, *14*, e1005896. [CrossRef] [PubMed]
148. LeCun, Y.; Boser, B.E.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.E.; Jackel, L.D. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems (NIPS 1989)*; Morgan Kaufmann: Denver, CO, USA, 1990; Volume 2, pp. 396–404.
149. Han, J.; Shi, L.; Yang, Q.; Huang, K.; Zha, Y.; Yu, J. Real-time detection of rice phenology through convolutional neural network using handheld camera images. *Precis. Agric.* **2021**, *22*, 154–178. [CrossRef]
150. Chen, T.; Sun, Y.; Li, T.H. A semi-parametric estimation method for the quantile spectrum with an application to earthquake classification using convolutional neural network. *Comput. Stat. Data Anal.* **2021**, *154*, 107069. [CrossRef]
151. Ciaburro, G.; Iannace, G.; Puyana-Romero, V.; Trematerra, A. A Comparison between Numerical Simulation Models for the Prediction of Acoustic Behavior of Giant Reeds Shredded. *Appl. Sci.* **2020**, *10*, 6881. [CrossRef]
152. Han, J.; Miao, S.; Li, Y.; Yang, W.; Yin, H. Faulted-Phase classification for transmission lines using gradient similarity visualization and cross-domain adaption-based convolutional neural network. *Electr. Power Syst. Res.* **2021**, *191*, 106876. [CrossRef]
153. Yildiz, C.; Acikgoz, H.; Korkmaz, D.; Budak, U. An improved residual-based convolutional neural network for very short-term wind power forecasting. *Energy Convers. Manag.* **2021**, *228*, 113731. [CrossRef]
154. Ye, R.; Dai, Q. Implementing transfer learning across different datasets for time series forecasting. *Pattern Recognit.* **2021**, *109*, 107617. [CrossRef]
155. Perla, F.; Richman, R.; Scognamiglio, S.; Wüthrich, M.V. Time-series forecasting of mortality rates using deep learning. *Scand. Actuar. J.* **2021**, 1–27. [CrossRef]
156. Ciaburro, G.; Iannace, G. Improving Smart Cities Safety Using Sound Events Detection Based on Deep Neural Network Algorithms. *Informatics* **2020**, *7*, 23. [CrossRef]
157. Yang, C.L.; Yang, C.Y.; Chen, Z.X.; Lo, N.W. Multivariate time series data transformation for convolutional neural network. In Proceedings of the 2019 IEEE/SICE International Symposium on System Integration (SII), Paris, France, 14–16 January 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 188–192.
158. Stoian, A.; Poulain, V.; Inglada, J.; Poughon, V.; Derksen, D. Land cover maps production with high resolution satellite image time series and convolutional neural networks: Adaptations and limits for operational systems. *Remote Sens.* **2019**, *11*, 1986. [CrossRef]
159. Anantrasirichai, N.; Biggs, J.; Albino, F.; Bull, D. The application of convolutional neural networks to detect slow, sustained deformation in InSAR time series. *Geophys. Res. Lett.* **2019**, *46*, 11850–11858. [CrossRef]
160. Wan, R.; Mei, S.; Wang, J.; Liu, M.; Yang, F. Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. *Electronics* **2019**, *8*, 876. [CrossRef]
161. Ni, L.; Li, Y.; Wang, X.; Zhang, J.; Yu, J.; Qi, C. Forecasting of forex time series data based on deep learning. *Procedia Comput. Sci.* **2019**, *147*, 647–652. [CrossRef]
162. LeCun, Y.; Bengio, Y. Convolutional networks for images, speech, and time series. In *The Handbook of Brain Theory and Neural Networks*; MIT Press: Cambridge, MA, USA, 1995.
163. Zhao, B.; Lu, H.; Chen, S.; Liu, J.; Wu, D. Convolutional neural networks for time series classification. *J. Syst. Eng. Electron.* **2017**, *28*, 162–169. [CrossRef]

164. Chen, Y.; Keogh, E.; Hu, B.; Begum, N.; Bagnall, A.; Mueen, A.; Batista, G. The Ucr Time Series Classification Archive. 2015. Available online: https://www.cs.ucr.edu/~eamonn/time_series_data/ (accessed on 4 April 2021).
165. Liu, C.L.; Hsaio, W.H.; Tu, Y.C. Time series classification with multivariate convolutional neural network. *IEEE Trans. Ind. Electron.* **2018**, *66*, 4788–4797. [CrossRef]
166. PHM Data Challenge. 2015. Available online: https://www.phmsociety.org/events/conference/phm/15/data-challenge (accessed on 4 April 2021).
167. Cui, Z.; Chen, W.; Chen, Y. Multi-scale convolutional neural networks for time series classification. *arXiv* **2016**, arXiv:1603.06995.
168. Borovykh, A.; Bohte, S.; Oosterlee, C.W. Conditional time series forecasting with convolutional neural networks. *arXiv* **2017**, arXiv:1703.04691.
169. Oord, A.V.D.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv* **2016**, arXiv:1609.03499.
170. Yang, J.; Nguyen, M.N.; San, P.P.; Li, X.; Krishnaswamy, S. Deep convolutional neural networks on multichannel time series for human activity recognition. *IJCAI* **2015**, *15*, 3995–4001.
171. Roggen, D.; Calatroni, A.; Rossi, M.; Holleczek, T.; Förster, K.; Tröster, G.; Millan, J.D.R. Collecting complex activity datasets in highly rich networked sensor environments. In Proceedings of the 2010 Seventh international conference on networked sensing systems (INSS), Kassel, Germany, 15–18 June 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 233–240.
172. Bulling, A.; Blanke, U.; Schiele, B. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Comput. Surv.* **2014**, *46*, 1–33. [CrossRef]
173. Le Guennec, A.; Malinowski, S.; Tavenard, R. Data Augmentation for Time Series Classification Using Convolutional Neural Networks. *Ecml/Pkdd Workshop on Advanced Analytics and Learning on Temporal Data*. 2016. Available online: https://halshs.archives-ouvertes.fr/halshs-01357973 (accessed on 24 May 2021).
174. Hatami, N.; Gavet, Y.; Debayle, J. Classification of time-series images using deep convolutional neural networks. In Proceedings of the Tenth international conference on machine vision (ICMV 2017), Vienna, Austria, 13–15 November 2017; International Society for Optics and Photonics: Bellingham, WA, USA, 2018; Volume 10696, p. 106960Y.
175. Marwan, N.; Romano, M.C.; Thiel, M.; Kurths, J. Recurrence plots for the analysis of complex systems. *Phys. Rep.* **2007**, *438*, 237–329. [CrossRef]
176. Sezer, O.B.; Ozbayoglu, A.M. Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. *Appl. Soft Comput.* **2018**, *70*, 525–538. [CrossRef]
177. Hong, S.; Wang, C.; Fu, Z. Gated temporal convolutional neural network and expert features for diagnosing and explaining physiological time series: A case study on heart rates. *Comput. Methods Programs Biomed.* **2021**, *200*, 105847. [CrossRef]
178. Lu, X.; Lin, P.; Cheng, S.; Lin, Y.; Chen, Z.; Wu, L.; Zheng, Q. Fault diagnosis for photovoltaic array based on convolutional neural network and electrical time series graph. *Energy Convers. Manag.* **2019**, *196*, 950–965. [CrossRef]
179. Han, L.; Yu, C.; Xiao, K.; Zhao, X. A new method of mixed gas identification based on a convolutional neural network for time series classification. *Sensors* **2019**, *19*, 1960. [CrossRef]
180. Gao, J.; Song, X.; Wen, Q.; Wang, P.; Sun, L.; Xu, H. RobustTAD: Robust time series anomaly detection via decomposition and convolutional neural networks. *arXiv* **2020**, arXiv:2002.09545.
181. Kashiparekh, K.; Narwariya, J.; Malhotra, P.; Vig, L.; Shroff, G. ConvTimeNet: A pre-trained deep convolutional neural network for time series classification. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–8.
182. Tang, W.H.; Röllin, A. Model identification for ARMA time series through convolutional neural networks. *Decis. Support Syst.* **2021**, *146*, 113544. [CrossRef]
183. Mikolov, T.; Karafiát, M.; Burget, L.; Černocký, J.; Khudanpur, S. Recurrent neural network based language model. In Proceedings of the Eleventh Annual Conference of the International Speech Communication Association, Chiba, Japan, 26–30 September 2010.
184. Zaremba, W.; Sutskever, I.; Vinyals, O. Recurrent neural network regularization. *arXiv* **2014**, arXiv:1409.2329.
185. Mikolov, T.; Kombrink, S.; Burget, L.; Černocký, J.; Khudanpur, S. Extensions of recurrent neural network language model. In Proceedings of the 2011 IEEE international conference on acoustics, speech and signal processing (ICASSP), Prague, Czech Republic, 22–27 May 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 5528–5531.
186. Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* **1982**, *79*, 2554–2558. [CrossRef]
187. Gregor, K.; Danihelka, I.; Graves, A.; Rezende, D.; Wierstra, D. Draw: A recurrent neural network for image generation. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 1462–1471.
188. Saon, G.; Soltau, H.; Emami, A.; Picheny, M. Unfolded recurrent neural networks for speech recognition. In Proceedings of the Fifteenth Annual Conference of the International Speech Communication Association, Singapore, 14–18 September 2014.
189. Goodfellow, I.; Bengio, Y. *Courville, Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
190. Kag, A.; Zhang, Z.; Saligrama, V. Rnns incrementally evolving on an equilibrium manifold: A panacea for vanishing and exploding gradients? In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
191. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]

192. Fischer, T.; Krauss, C. Deep learning with long short-term memory networks for financial market predictions. *Eur. J. Oper. Res.* **2018**, *270*, 654–669. [CrossRef]
193. Bao, W.; Yue, J.; Rao, Y. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLoS ONE* **2017**, *12*, e0180944. [CrossRef] [PubMed]
194. Soni, B.; Patel, D.K.; Lopez-Benitez, M. Long short-term memory based spectrum sensing scheme for cognitive radio using primary activity statistics. *IEEE Access* **2020**, *8*, 97437–97451. [CrossRef]
195. Connor, J.T.; Martin, R.D.; Atlas, L.E. Recurrent neural networks and robust time series prediction. *IEEE Trans. Neural Netw.* **1994**, *5*, 240–254. [CrossRef]
196. Qin, Y.; Song, D.; Chen, H.; Cheng, W.; Jiang, G.; Cottrell, G. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv* **2017**, arXiv:1704.02971.
197. Che, Z.; Purushotham, S.; Cho, K.; Sontag, D.; Liu, Y. Recurrent neural networks for multivariate time series with missing values. *Sci. Rep.* **2018**, *8*, 1–12. [CrossRef]
198. Chandra, R.; Zhang, M. Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction. *Neurocomputing* **2012**, *86*, 116–123. [CrossRef]
199. Elman, J.L. Finding structure in time. *Cogn. Sci.* **1990**, *14*, 179–211. [CrossRef]
200. Hüsken, M.; Stagge, P. Recurrent neural networks for time series classification. *Neurocomputing* **2003**, *50*, 223–235. [CrossRef]
201. Hermans, M.; Schrauwen, B. Training and analysing deep recurrent neural networks. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 190–198.
202. Hua, Y.; Zhao, Z.; Li, R.; Chen, X.; Liu, Z.; Zhang, H. Deep learning with long short-term memory for time series prediction. *IEEE Commun. Mag.* **2019**, *57*, 114–119. [CrossRef]
203. Song, X.; Liu, Y.; Xue, L.; Wang, J.; Zhang, J.; Wang, J.; Cheng, Z. Time-series well performance prediction based on Long Short-Term Memory (LSTM) neural network model. *J. Pet. Sci. Eng.* **2020**, *186*, 106682. [CrossRef]
204. Yang, B.; Yin, K.; Lacasse, S.; Liu, Z. Time series analysis and long short-term memory neural network to predict landslide displacement. *Landslides* **2019**, *16*, 677–694. [CrossRef]
205. Sahoo, B.B.; Jha, R.; Singh, A.; Kumar, D. Long short-term memory (LSTM) recurrent neural network for low-flow hydrological time series forecasting. *Acta Geophys.* **2019**, *67*, 1471–1481. [CrossRef]
206. Benhaddi, M.; Ouarzazi, J. Multivariate Time Series Forecasting with Dilated Residual Convolutional Neural Networks for Urban Air Quality Prediction. *Arab. J. Sci. Eng.* **2021**, *46*, 3423–3442. [CrossRef]
207. Kong, Y.L.; Huang, Q.; Wang, C.; Chen, J.; Chen, J.; He, D. Long short-term memory neural networks for online disturbance detection in satellite image time series. *Remote Sens.* **2018**, *10*, 452. [CrossRef]
208. Lei, J.; Liu, C.; Jiang, D. Fault diagnosis of wind turbine based on Long Short-term memory networks. *Renew. Energy* **2019**, *133*, 422–432. [CrossRef]
209. Tschannen, M.; Bachem, O.; Lucic, M. Recent advances in autoencoder-based representation learning. *arXiv* **2018**, arXiv:1812.05069.
210. Myronenko, A. 3D MRI brain tumor segmentation using autoencoder regularization. In *International MICCAI Brainlesion Workshop*; Springer: Cham, Switzerland, 2018; pp. 311–320.
211. Yang, X.; Deng, C.; Zheng, F.; Yan, J.; Liu, W. Deep spectral clustering using dual autoencoder network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4066–4075.
212. Ashfahani, A.; Pratama, M.; Lughofer, E.; Ong, Y.S. DEVDAN: Deep evolving denoising autoencoder. *Neurocomputing* **2020**, *390*, 297–314. [CrossRef]
213. Semeniuta, S.; Severyn, A.; Barth, E. A hybrid convolutional variational autoencoder for text generation. *arXiv* **2017**, arXiv:1702.02390.
214. Mehdiyev, N.; Lahann, J.; Emrich, A.; Enke, D.; Fettke, P.; Loos, P. Time series classification using deep learning for process planning: A case from the process industry. *Procedia Comput. Sci.* **2017**, *114*, 242–249. [CrossRef]
215. Corizzo, R.; Ceci, M.; Zdravevski, E.; Japkowicz, N. Scalable auto-encoders for gravitational waves detection from time series data. *Expert Syst. Appl.* **2020**, *151*, 113378. [CrossRef]
216. Yang, J.; Bai, Y.; Lin, F.; Liu, M.; Hou, Z.; Liu, X. A novel electrocardiogram arrhythmia classification method based on stacked sparse auto-encoders and softmax regression. *Int. J. Mach. Learn. Cybern.* **2018**, *9*, 1733–1740. [CrossRef]
217. Rußwurm, M.; Körner, M. Multi-temporal land cover classification with sequential recurrent encoders. *ISPRS Int. J. Geo-Inf.* **2018**, *7*, 129. [CrossRef]
218. Zdravevski, E.; Lameski, P.; Trajkovik, V.; Kulakov, A.; Chorbev, I.; Goleva, R.; Garcia, N. Improving activity recognition accuracy in ambient-assisted living systems by automated feature engineering. *IEEE Access* **2017**, *5*, 5262–5280. [CrossRef]
219. Christ, M.; Braun, N.; Neuffer, J.; Kempa-Liehr, A.W. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing* **2018**, *307*, 72–77. [CrossRef]
220. Caesarendra, W.; Pratama, M.; Kosasih, B.; Tjahjowidodo, T.; Glowacz, A. Parsimonious network based on a fuzzy inference system (PANFIS) for time series feature prediction of low speed slew bearing prognosis. *Appl. Sci.* **2018**, *8*, 2656. [CrossRef]