Big Data
Volume 00, Number 00, 2020
© Mary Ann Liebert, Inc.
DOI: 10.1089/big.2020.0159

Deep Learning for Time Series Forecasting: A Survey

José F. Torres,[1,†] Dalil Hadjout,[2,†] Abderrazak Sebaa,[3,4] Francisco Martínez-Álvarez,[1] and Alicia Troncoso[1,*]

Abstract

Time series forecasting has become a very intensive field of research, which is even increasing in recent years. Deep neural networks have proved to be powerful and are achieving high accuracy in many application fields. For these reasons, they are one of the most widely used methods of machine learning to solve problems dealing with big data nowadays. In this work, the time series forecasting problem is initially formulated along with its mathematical fundamentals. Then, the most common deep learning architectures that are currently being successfully applied to predict time series are described, highlighting their advantages and limitations. Particular attention is given to feed forward networks, recurrent neural networks (including Elman, long-short term memory, gated recurrent units, and bidirectional networks), and convolutional neural networks. Practical aspects, such as the setting of values for hyper-parameters and the choice of the most suitable frameworks, for the successful application of deep learning to time series are also provided and discussed. Several fruitful research fields in which the architectures analyzed have obtained a good performance are reviewed. As a result, research gaps have been identified in the literature for several domains of application, thus expecting to inspire new and better forms of knowledge.

Keywords: big data; deep learning; time series forecasting

Introduction

The interest in processing huge amounts of data has experienced a rapid increase during the past decade due to the massive deployment of smart sensors[1] or the social media platforms,[2] which generate data on a continuous basis.[3] However, this situation poses new challenges, such as storing these data in disks or making available the required computational resources.

Big data analytics emerges, in this context, as an essential process focused on efficiently collecting, organizing, and analyzing big data with the aim of discovering patterns and extracting valuable information.[4] In most organizations, this helps to identify new opportunities and making smarter moves, which leads to more efficient operations and higher profits.[5]

From all the learning paradigms that are currently being used in big data, deep learning highlights because of its outstanding performance as the scale of data increases.[6] Most of the layer computations in deep learning can be done in parallel by, for instance, powerful graphic processing units (GPUs). That way, scalable distributed models are easier to be built and they provide better accuracy at a much higher speed. Higher depth allows for more complex non-linear functions but, in turn, with higher computational costs.[7]

Deep learning can be applied to numerous research fields. Applications to both supervised and unsupervised problems can be abundantly found in the literature.[8] Pattern recognition and classification were the first and most relevant uses of deep learning, achieving great success in speech recognition, text mining, or image analysis. Nevertheless, the application to regression problems is becoming quite popular nowadays mainly due to the development of deep-learning architectures particularly conceived to deal with data indexed over time. Such is the case of time series and, more specifically, time series forecasting.[9]

[1]Data Science and Big Data Lab, Pablo de Olavide University, Seville, Spain.
[2]Department of Commerce, SADEG Company (Sonelgaz Group), Bejaia, Algeria.
[3]LIMED Laboratory, Faculty of Exact Sciences, University of Bejaia, Bejaia, Algeria.
[4]Higher School of Sciences and Technologies of Computing and Digital, Bejaia, Algeria.
[†]Equally contributing authors.

*Address correspondence to: *Alicia Troncoso, Data Science and Big Data Lab, Pablo de Olavide University, Seville ES-41013, Spain, E-mail:* atrolor@upo.es

A time series is a set of measures collected at even intervals of time and ordered chronologically.[10] Given this definition, it is hard to find physical or chemical phenomena without variables that evolve over time. For this reason, the proposal of time series forecasting approaches is fruitful and can be found in almost all scientific disciplines.

Statistical approaches have been used from the 1970s onward, especially those based on the Box-Jenkins methodology.[11] With the appearance of machine learning and its powerful regression methods,[12] many models were proposed as outperforming the former, which have remained as baseline methods in most research works. However, methods based on deep learning are currently achieving superior results and much effort is being put into developing new architecture.

For all that has been mentioned earlier, the primary motivation behind this survey is to provide a comprehensive understanding of deep-learning fundamentals for researchers interested in the field of time series forecasting. Further, it overviews several applications in which these techniques have been proven successful and, as a result, research gaps have been identified in the literature and are expected to inspire new and better forms of knowledge.

Although other surveys discussing deep-learning properties have been published during the past years, the majority of them provided a general overview of both theory and applications to time series forecasting. Thus, Zhang et al.[13] reviewed emerging researches of deep-learning models, including their mathematical formulation, for big data feature learning. Another remarkable work can be found in Ref.,[14] in which the authors introduced the time series classification problem and provided an open-source framework with implemented algorithms and the University of East Anglia/University of California in Riverside repository.[15] Recently, Mayer and Jacobsen published a survey about scalable deep learning on distributed infrastructures, in which the focus was placed on techniques and tools, along with a smart discussion about the existing challenges in this field.[16]

The rest of the article is structured as follows. The forecasting problem and mathematical formulation for time series can be found in the Problem Definition section. Deep-Learning Architectures section introduces the deep-learning architectures typically used in the context of time series forecasting. Practical Aspects section provides information about several practical aspects (including implementation, hyper-parameter tuning, or hardware resources) that must be considered when applying deep learning to forecast time series. Applications section overviews the most relevant papers, sorted by fields, in which deep learning has been applied to forecast time series. Finally, the lessons learned and the conclusions drawn are discussed in the Conclusions section.

Problem Definition

This section provides the time series definition (Time Series Definition section), along with a description of the main time series components (Time Series Components section). The mathematical formulation for the time series forecasting problem is introduced in the Mathematical Formulation section. Final remarks about the length of the time series can be found in Short- and Long-Time Series Forecasting section.

Time series definition

A time series is defined as a sequence of values, chronologically ordered, and observed over time. Although the time is a variable measured on a continuous basis, the values in a time series are sampled at constant intervals (fixed sampling frequency).

This definition holds true for many applications, but not every time series can be modeled in this way, due to some of the following reasons:

1. Missing data in time series is a very common problem due to the reliability of data collection. To deal with these values, there are a lot of strategies but those based on imputing the missing information and on omitting the entire record are the most widely used.[17]
2. Outlying data is also an issue that appears very frequently in time series. Methods based on robust statistics must be chosen to remove these values or, simply, to incorporate them into the model.[18]
3. When data are collected at irregular time periods, they can be called either unevenly spaced time series or, if big enough, data streams.[3]

Some of these issues can be handled natively by the used model, but if the data are collected irregularly, this should be accounted for in the model. In this survey, the time series preprocessing is out of scope, but please refer to this work for detailed information.[19]

Time series components

Time series are usually characterized by three components: trend, seasonality, and irregular components, also known as residuals.[20] Such components are described later:

1. Trend. It is the general movement that the time series exhibits during the observation period, without considering seasonality and irregularities. In some texts, this component is also known as long-term variation. Although there are different kinds of trends in time series, the most popular are linear, exponential, or parabolic ones.
2. Seasonality. This component identifies variations that occur at specific regular intervals and may provide useful information when time periods exhibit similar patterns. It integrates the effects reasonably stable along with the time, magnitude, and direction. Seasonality can be caused by several factors such as climate or economical cycles, or even festivities.
3. Residuals. Once the trend and cyclic oscillations have been calculated and removed, some residual values remain. These values can be, sometimes, high enough to mask the trend and the seasonality. In this case, the term *outlier* is used to refer these residuals, and robust statistics are usually applied to cope with them.[20] These fluctuations can be of diverse origin, which makes the prediction almost impossible. However, if by any chance, this origin can be detected or modeled, they can be thought of precursors in trend changes.

A time series is an aggregate of these three components. Real-world time series present a meaningful irregular component and are not stationary (mean and variance are not constant over time), turning this component into the most challenging one to model. For this reason, to make accurate predictions for them is extremely difficult, and many forecasting classical methods try to decompose the target time series into these three components and make predictions for all of them separately.

The effectiveness of one technique or another is assessed according to its capability of forecasting this particular component. It is for the analysis of this component where data mining-based techniques have been shown to be particularly powerful.

Time series can be graphically represented. In particular, the x-axis identifies the time, whereas the y-axis identifies the values recorded at punctual time stamps (x_t). This representation allows the visual detection of the most highlighting features of a series, such as oscillations amplitude, existing seasons, and cycles or the existence of anomalous data or outliers. Figure 1 depicts an time series, x_t, using an additive model with linear seasonality with constant frequency and amplitude over time, represented by the function $sin(x)$; linear trend where changes over time are consistently made by the same amount, represented by the function $0.0213x$; and residuals, represented by random numbers in the interval $[0, 0.1]$.

Mathematical formulation

Time series models can be either univariate (one time-dependent variable) or multivariate (more than one time-dependent variables). Although models may dramatically differ between a univariate and a multivariate system, the majority of the deep-learning models can handle indistinctly with both of them.

On the one hand, let $y = y(t-L), \ldots, y(t-1), y(t), y(t+1), \ldots, y(t+h)$ be a given univariate time series with L values in the historical data, where each $y(t-i)$, for $i = 0, \ldots, L$, represents the recorded value of the variable y at time $t-i$. The forecasting process consists of estimating the value of $y(t+1)$, denoted by $\hat{y}(t+1)$, with the aim of minimizing the error, which is typically represented as a function of $y(t+1) - \hat{y}(t+1)$. This prediction can be made also when the horizon of prediction, h, is greater than one, that is, when the objective is to predict the h next values after $y(t)$, that is, $y(t+i)$, with $i = 1, \ldots, h$. In this situation, the best prediction is reached when the function $\sum_{i=1}^{h}(y(t+i) - \hat{y}(t+i))$ is minimized.

On the other hand, multivariate time series can be expressed as follows, in the matrix form:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ y_T \end{pmatrix} = \begin{pmatrix} y_1(t-L) & \ldots & y_1(t-1) & y_1(t) & y_1(t+1) & \ldots & y_1(t+h) \\ y_2(t-L) & \ldots & y_2(t-1) & y_2(t) & y_2(t+1) & \ldots & y_2(t+h) \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ y_n(t-L) & \ldots & y_n(t-1) & y_n(t) & y_n(t+1) & \ldots & y_n(t+h) \end{pmatrix}, \qquad (1)$$

where $y_i(t-m)$ identifies the set of time series, with $i = \{1, 2, \ldots, n\}$, being $m = \{0, 1, \ldots, L\}$ the historical data and current sample and $m = \{-1, -2, \ldots, -h\}$

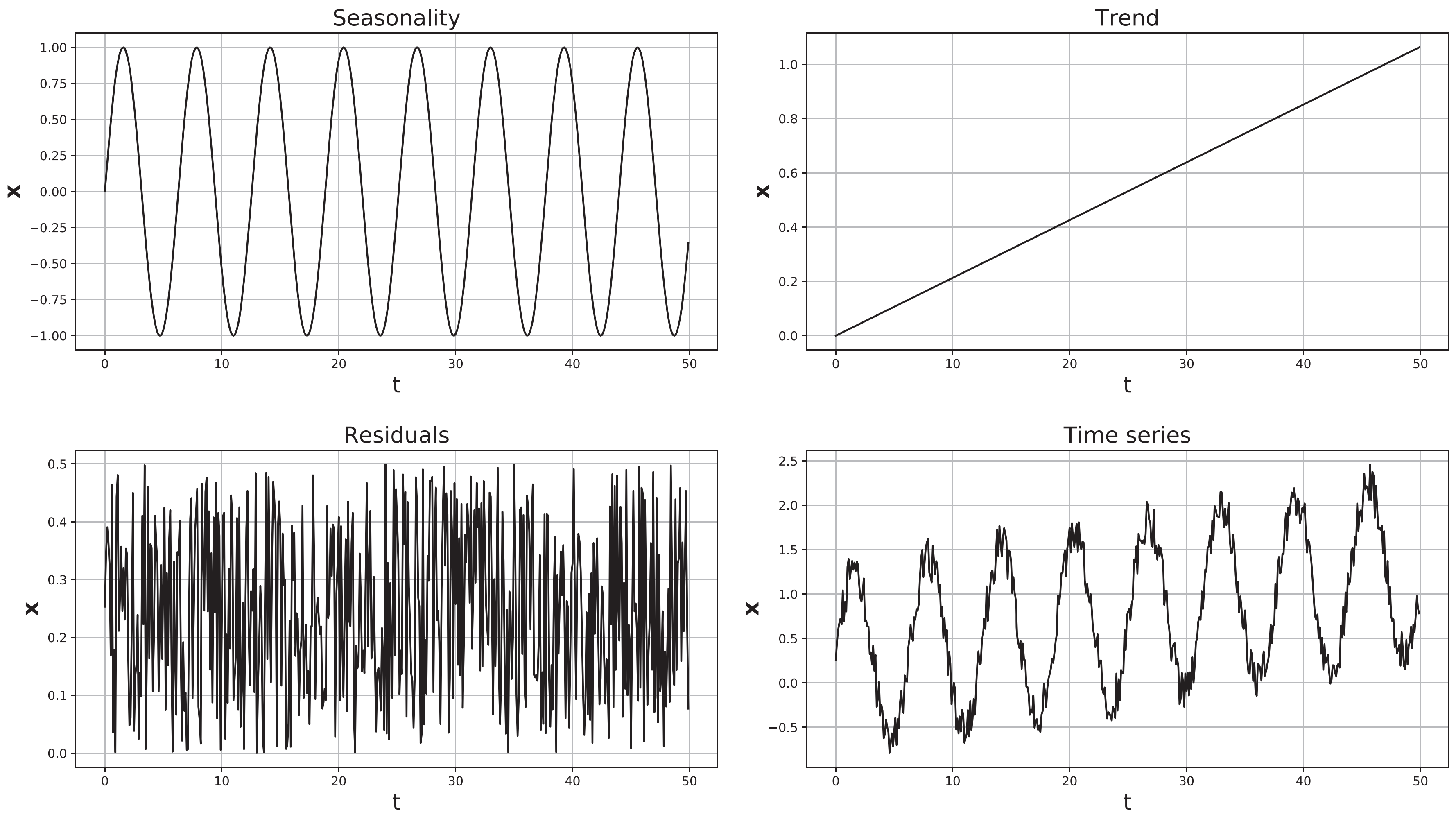

FIG. 1. Illustrative time series, showing seasonality, trend, and residuals.

the future *h* values. Usually, there is one target time series (the one to be predicted) and the remaining ones are denoted as independent time series.

Short- and long time series forecasting

Another key issue is the length of the time series. Depending on the number of samples, long- or short time series can be defined. It is well known that the Box-Jenkins' models do not work well for long time series mainly due to the time-consuming process of parameters optimization and to the inclusion of information, which is no longer useful to model the current samples.[9]

How to deal with these issues is highly related to the purpose of the model. Flexible nonparametric models could be used, but this still assumes that the model structure will work over the whole period of the data, which is not always true. A better approach consists of allowing the model to vary over time. This can be done by either adjusting a parametric model with time-varying parameters or adjusting a nonparametric model with a time-based kernel. But if the goal is only to forecast a few observations, it is simpler to fit a model with the most recent samples and transforming the long time series into a short one.[21]

Although a preliminary approach to use a distributed ARIMA model has been recently published,[22] it remains challenging to deal with such time series with classical forecasting methods. However, a number of machine-learning algorithms adapted to deal with ultra-long time series, or big data time series, have been published in recent years.[23] These models make use of clusters of machines or GPUs to overcome the limitations described in the previous paragraphs.

Deep-learning models can deal with time series in a scalable way and provide accurate forecasts.[24] Ensemble learning can also be useful to forecast big data time series[25] or even methods based on well-established methods such as nearest neighbours[26,27] or pattern sequence similarity.[28]

Deep-Learning Architectures

This section provides a theoretical tour of deep learning for time series prediction in big data environments. First, a description of the most used architectures in the literature to predict time series is made. Then, a state-of-the-art analysis is carried out, where the deep-learning works and frameworks to deal with big data are described.

Deep feed forward neural network

Deep feed forward neural networks (DFFNN), also called multi-layer perceptron, arose due to the inability of single-layer neural networks to learn certain functions. The architecture of a DFFNN is composed of an input layer, an output layer, and different hidden layers, as shown in Figure 2. In addition, each hidden layer has a certain number of neurons to be determined.

The relationships between the neurons of two consecutive layers are modeled by weights, which are calculated during the training phase of the network. In

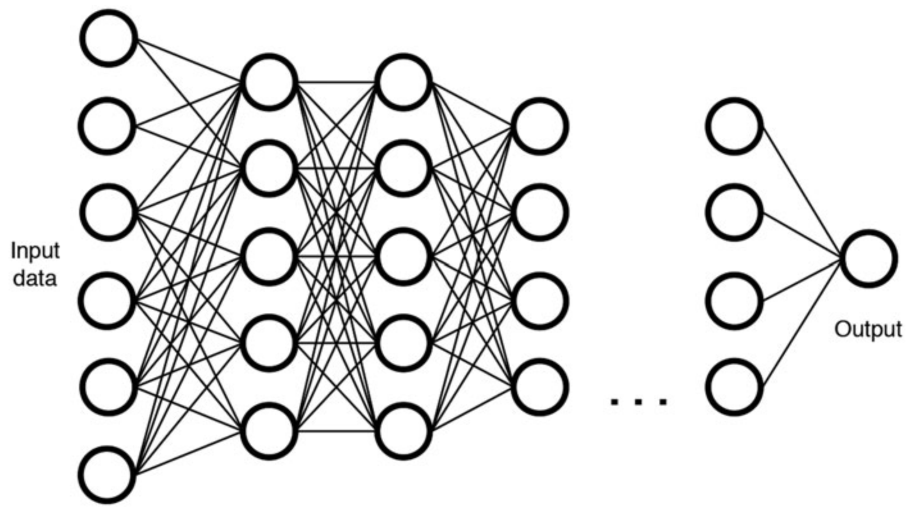

FIG. 2. Basic architecture of a DFFNN for time series forecasting. DFFNN, deep feed-forward neural network.

particular, the weights are computed by minimizing a cost function by means of gradient descent optimization methods. Then, the back-propagation algorithm is used to calculate the gradient of the cost function. Once the weights are computed, the values of the output neurons of the network are obtained by using a feed-forward process defined by the following equation:

$$a^l = g(W_a^l a^{l-1} + b_a^l), \tag{2}$$

where a^l are the activation values in the l-th layer, that is, a vector composed of the values of the neurons of the l-th layer, W_a^l and b_a^l are the weights and bias corresponding to the l-th layer, and g is the activation function. Therefore, the a^l values are computed by using the activation values of the $l-1$ layer, a^{l-1}, as input. In time series forecasting, the rectified linear unit function is commonly used as activation function for all layers, except for the output layer to obtain the predicted values, which generally uses the hyperbolic tangent function (tanh).

For all network architectures, the values of some hyper-parameters have to be chosen in advance. These hyper-parameters, such as the number of layers and the number of neurons, define the network architecture, and other hyper-parameters, such as the learning rate, the momentum, and number of iterations or mini-batch size, among others, have a great influence on the convergence of the gradient descend methods. The optimal choice of these hyper-parameters is important, as these values greatly influence the prediction results obtained by the network. The hyper-parameters will be discussed in more detail in the Hyper-Parameter Optimization section.

Recurrent neural network

Recurrent neural networks (RNNs) are specifically designed to deal with sequential data such as sequences of words in problems related to machine translation, audio data in speech recognition, or time series in forecasting problems. All these problems present a common characteristic, which is that the data have a temporal dependency between them. Traditional feed-forward neural networks cannot take into account these dependencies, and RNNs arise precisely to address this problem.[29] Therefore, the input data in the architecture of a RNN are both past and current data. There are different types of architectures, depending on the number of data inputs and outputs in the network, such as one to one (one input and one output), one to many (one input and many outputs), many to one (many inputs and one output), and many to many (many inputs and outputs). The most common RNNs are many to one for classification problems or many to many for machine translation or time series forecasting for instance. In addition, for the case of a time series, the length of the input data sequence is usually different from the size of the output data sequence that usually is the number of samples to be predicted. A basic RNN architecture to address the forecasting of time series is shown in Figure 3. x_i and $\hat{x}_i$ are the actual and predicted values of the time series at time i, and h is the number of samples to be predicted, called prediction horizon.

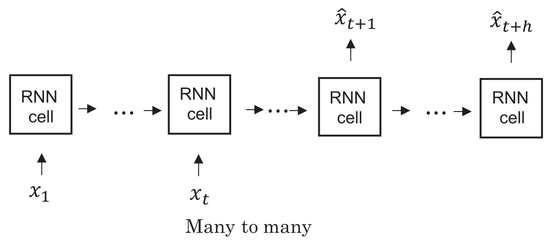

FIG. 3. Basic architecture of an RNN for time series forecasting. RNN, recurrent neural network.

The most widely used RNNs for time series forecasting are briefly described later.

Elman RNN. The Elman network (ENN) was the first RNN and it incorporated the t state of a hidden unit to make predictions in data sequences.[30] The ENN consists of a classical one-layer feed-forward network but the hidden layer is connected to a new layer, called context layer, using fixed weights equal to one, as shown in Figure 4. The main function of the neurons of this

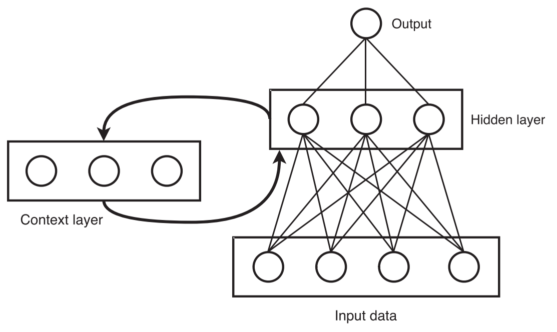

FIG. 4. Architecture of an ENN for time series forecasting. ENN, Elman network.

context layer is to save a copy of the values of activation of the neurons of the hidden layer. Then, the model is defined by:

$$a_t = g(W_a x_t + U_a a_{t-1} + b_a), \tag{3}$$

where a_t are the values of the neurons in the t state in the hidden layer, x_t is the current input, a_{t-1} is the information saved in the context hidden units, W_a, U_a, and b_a are the weights and the bias, and g is the activation function.

Long short-term memory. Standard basic RNNs suffer the vanishing gradient problem, which consists of the gradient decreasing as the number of layers increases. Indeed, for deep RNNs with a high number of layers, the gradient practically becomes null, preventing the learning of the network. For this reason, these networks have a short-term memory and do not obtain good results when dealing with long sequences that require memorizing all the information contained in the complete sequence. Long short-term memory (LSTM) recurrent networks emerge to solve the vanishing gradient problem.[31] For this purpose, LSTM uses three gates to keep longstanding relevant information and discard irrelevant information. These gates are Γ^f forget gate, Γ^u update gate, and Γ^o output gate. Γ^f decides what information should be thrown away or saved. A value close to 0 means that the past information is forgotten whereas a value close to 1 means that it remains. Γ^u decides what new information $\tilde{c}_t$ to use to update the c_t memory state. Thus, c_t is updated by using both Γ^f and Γ^u. Finally, Γ^o decides which is the output value that will be the input of the next hidden unit.

The information of the a_{t-1} previous hidden unit and the information of the x_t current input is passed through the σ sigmoid activation function to compute all the gate values and through the tanh activation function to compute the $\tilde{c}_t$ new information, which will be used to update. The equations defining an LSTM unit are:

$$\tilde{c}_t = \tanh(W_c[a_{t-1}, x_t] + b_c), \tag{4}$$

$$\Gamma^u = \sigma(W_u[a_{t-1}, x_t] + b_u), \tag{5}$$

$$\Gamma^f = \sigma(W_f[a_{t-1}, x_t] + b_f), \tag{6}$$

$$\Gamma^o = \sigma(W_o[a_{t-1}, x_t] + b_o), \tag{7}$$

$$c_t = \Gamma^u \times \tilde{c}_t + \Gamma^f \times c_{t-1}, \tag{8}$$

$$a_t = \Gamma^o \times \tanh(c_t), \tag{9}$$

where W_u, W_f, and W_o, and b_u, b_f, and b_o are the weights and biases that govern the behavior of the Γ^u, Γ^f, and Γ^o gates, respectively, and W_c and b_c are the weights and bias of the $\tilde{c}_t$ memory cell candidate.

Figure 5 shows a picture of how a hidden unit works in an LSTM recurrent network. The $\times$ and + operators mean an element-wise vectors multiplication and sum.

Gated recurrent units. Recurrent networks with gated recurrent units (GRU) are long-term memory networks such as LSTMs but they emerged in 2014[32,33] as a simplification of LSTMs due to the high computational cost of the LSTM networks. GRU is one of the most commonly used versions that researchers have converged on and found to be robust and useful for many different problems. The use of gates in RNNs has made it possible to improve capturing of very long-range dependencies, making RNNs much more effective. The LSTM is more powerful and more effective since it has three gates instead of two, but the GRU is a simpler model and it is computationally faster as it only has two gates, Γ^u update gate and Γ^r relevance gate as shown in Figure 6. The Γ^u gate will decide whether the c_t memory state is or is not updated by using the $\tilde{c}_t$ memory state candidate. The Γ^r gate determines how relevant c_{t-1} is to compute the next candidate for c_t, that is, $\tilde{c}_t$. A GRU is defined by the following equations:

$$\Gamma^u = \sigma(W_u[c_{t-1}, x_t] + b_u, \tag{10}$$

$$\Gamma^r = \sigma(W_r[c_{t-1}, x_t] + b_r), \tag{11}$$

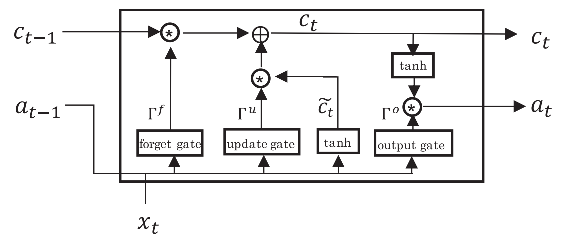

FIG. 5. Hidden unit in an LSTM. LSTM, long short-term memory.

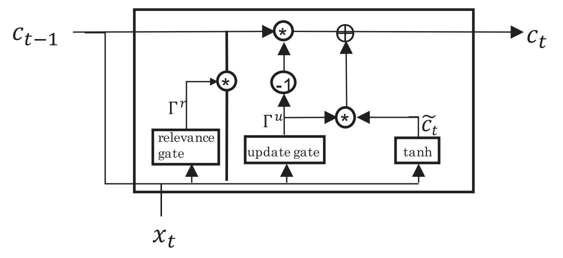

FIG. 6. Hidden unit in a GRU. GRU, gated recurrent units.

$$\tilde{c}_t = \tanh(W_c[\Gamma^r \times c_{t-1}, x_t] + b_c), \tag{12}$$

$$c_t = \Gamma^u \times \tilde{c}_t + (1 - \Gamma^u) \times c_{t-1}, \tag{13}$$

$$a_t = c_t, \tag{14}$$

where W_u and W_r, and b_u and b_r are the weights and the bias that govern the behavior of the Γ_u and Γ_r gates, respectively, and W_c and b_c are the weights and bias of the $\tilde{c}_t$ memory cell candidate.

Bidirectional RNN. There are some problems, in the field of natural language processing (NLP) for instance, where to predict a value of a data sequence in a given instant of time, information from the sequence both before and after that instant is needed. Bidirectional recurrent neural networks (BRNNs) address this issue to solve this kind of problems. The main disadvantage of the BRNNs is that the entire data sequence is needed before the prediction can be made.

Standard networks compute the activation values for hidden units by using a unidirectional feed-forward process. However, in a BRNN, the prediction uses information from the past as well as information from the present and the future as input, using both forward and backward processing.

Thus, the prediction at time t, $\hat{x}_t$, is obtained by using a g activation function applied to the corresponding weights with both the forward and backward activation at time t. That is:

$$\hat{x}_t = g(W_x[a_t^f, a_t^b] + b_x), \tag{15}$$

where W_x and b_x are the weights and bias and a_t^f and a_t^b are the activation values of the hidden units computed by forward and backward processing, respectively, and g is an activation function.

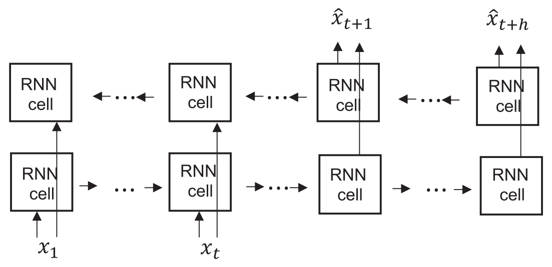

FIG. 7. Basic architecture of a BRNN. BRNN, bidirectional recurrent neural network,

Figure 7 presents the basic architecture of a BRNN. A BRNN can be seen as two RNNs together, where the different hidden units have two values, one computed by forward and another one by backward. In addition, the BRNN units can be standard RNN units or GRU or LSTM units. In fact, a BRNN with LSTM units is commonly used for a lot of NLP problems.

Deep recurrent neural network. A deep recurrent neural network (DRNN) can be considered as an RNN with more than one layer, also called stacked RNN. The hidden units can be standard RNN, GRU or LSTM units, and it can be unidirectional or bidirectional as described in previous sections. Figure 8 illustrates the architecture of a DRNN with three layers.

In general, a DRNN works quite well for time series forecasting, but its performance deteriorates when using very long data sequences as input. To address this issue, attention mechanisms can be incorporated into the model, being one of the most powerful ideas in deep learning.[34] An attention model allows a neural

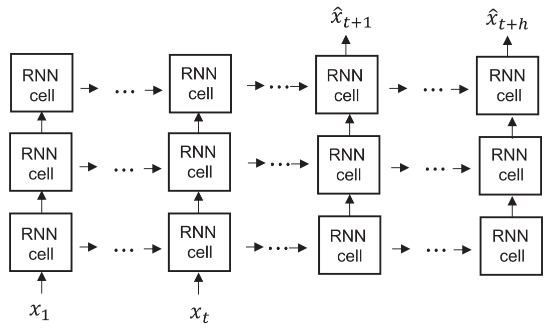

FIG. 8. Basic architecture of a DRNN. DRNN, deep recurrent neural network.

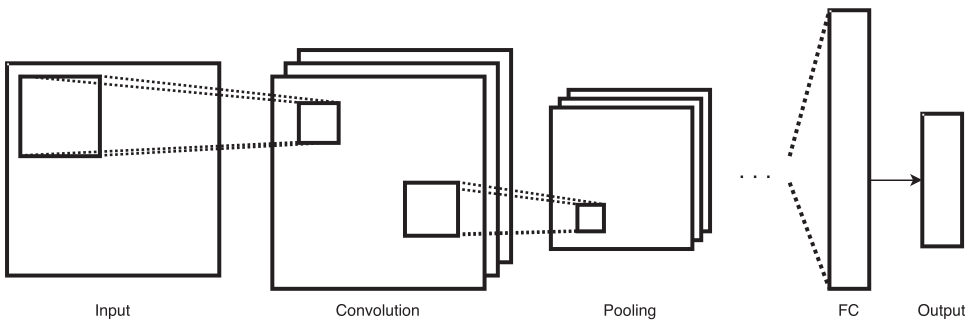

FIG. 9. Architecture of a CNN. CNN, convolutional neural networks.

network to pay attention to only part of an input data sequence while it is generating the output. This attention is modeled by using weights, which are computed by a single-layer feed-forward neural network.[35]

Convolutional neural networks

Convolutional neural networks (CNN) were presented in Ref.[36] by Fukushima and are one of the most common architectures in image processing and computer vision.[37] The CNNs have three kinds of layers: convolution, pooling, and fully connected. The main task of the convolution layers is the learning of the features from data input. For that, filters of a predefined size are applied to the data by using the convolution operation between matrices. The convolution is the sum of all element-wise products. The pooling reduces the size of input, speeding up the computing and preventing overfitting. The most popular pooling methods are average and max pooling, which summarize the values by using the mean or maximum value, respectively. Once the features have been extracted by the convolutional layers, the forecasting is carried out by using fully connected layers, also called dense layers, as in DFFNN. The input data for these last fully connected layers are the flattened features resulting of the convolutional and pooling layers. Figure 9 depicts the overall architecture of a CNN.

Recently, a variant of CNN, called temporal convolutional networks (TCNs),[38] has emerged for data sequence, competing directly with DRNNs in terms of execution times and memory requirements.

The TCNs have the same architecture as a DFFNN but the values of activations for each layer are computed by using earlier values from the previous layer. Dilated convolution is used to select which values of the neurons from the previous layer will contribute to the values of the neurons in the next layer. Thus, this dilated convolution operation captures both local and temporal information.

The dilated convolution, F_d, is a function defined as follows:

$$F_d(x) = \sum_{i=0}^{K-1} f(i) \cdot x_{t-d \cdot i}, \tag{16}$$

where d is the dilation factor parameter, and f is a filter of size K.

Figure 10 shows the architecture of a TCNN when applying a dilated convolution by using a filter of size 3 and dilation factors of 1, 2, and 4 for each layer, respectively.

Moreover, it is necessary to use generic residual modules in addition to convolutional layers when deeper and larger TCN are used to achieve further stabilization. These generic residual blocks consist of adding the input of data to the output before applying the activation function. Then, the TCN model can be defined as follows:

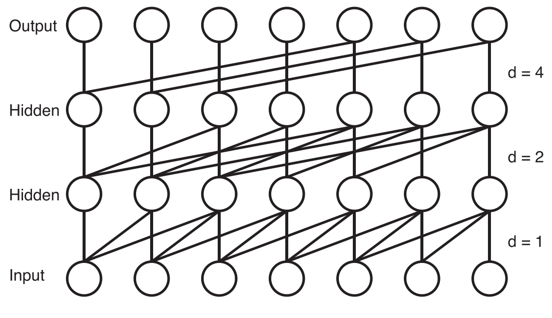

FIG. 10. Architecture of a TCN using a filter of size 3. TCN, temporal convolutional network.

$$a_t^l = g(W_a^l F_d(a_t^{l-1}) + b_a^l + a_t^{l-1}), \quad (17)$$

where $F_d(\cdot)$ is the dilated convolution of d factor defined in Equation (16), a_t^l is the value of the neuron of the l-th layer at time t, W_a^l and b_a^l are the weights and bias corresponding to the l-th layer, and g is the activation function.

Practical aspects

Implementation

The implementation of a multilayer perceptron is relatively simple. However, deep-learning models are more complex, and their implementation requires a high level of technical expertise and a considerable time investment to implement. For this reason, the profile of the deep-learning expert has become one of the most demanded nowadays. To make easier implementations and reduce the time needed to design and train a model, some companies have focused their work on developing frameworks that allow for the implementation, training and use of deep learning models.

The main idea of the deep-learning frameworks is to provide an interface that allows for the implementation of models without having to pay too much attention to the mathematical complexity behind them. There are several frameworks available in the literature. The choice of one or another will depend on several important factors, such as the type of architecture that can be implemented, support for distributed programming environments, or whether it can run on GPUs. In this sense, Table 1 summarizes the most widely used frameworks in the literature, where the term *all* includes the DFFNN, CNN, TCN, RNN, LSTM, GRU, or BRNN architectures, and CPU is a central processing unit.

Table 1 shows that the predominant programming language for developing deep-learning models is Python. In addition, most of the frameworks support distributed execution and the use of GPU's. Although the described frameworks facilitate the development of the models, some of them require too many lines of code to obtain a complete implementation. For this reason, high-level libraries based on the core of the frameworks have been developed, making programming even easier. Some examples of high-level libraries can be Keras,[50] Sonnet,[51] Swift, or Gluon,[52] among others. The main advantage of using a high-level-library is that the syntax can be reused for another base framework, in addition to facilitating its implementation. However, the lack of flexibility is the main disadvantage.

Hyper-parameter optimization

The combination of frameworks and high-level-libraries greatly facilitates the implementation of models. However, there is an important study gap: the model optimization. This optimization will determine the quality of the model, and it must be performed based on the adjustment of its hyper-parameters. In deep learning, there are two types of hyper-parameters: model parameters and optimization parameters. The model parameters must be adjusted in the model definition to obtain optimal performance. The optimization parameters are adjusted during the training phase of the model by using the dataset. Some of the most relevant hyper-parameters are described and categorized by network architecture in Table 2.

The number of hyper-parameters will depend on the network architecture to be used. In addition, the value of each one will be influenced by the characteristics of

Table 1. Deep-learning frameworks

Framework	*Core language*	*Available interfaces*	*Architecture*	*Distributed*	*CPU | GPU*
TensorFlow[39]	C++	Python, JavaScript, C++, Java, Go, C#, Julia	All	✓	✓|✓
H2O[40]	Java	Python, R, Scala, REST	DFFNN	✓	✓|✓
Dl4j[41]	Java	Python, Scala, Clojure, Kotlin, C, C++	All	✓	✓|✓
PyTorch[42]	Lua	Python, C, C++	All	✓	✓|✓
Caffe[43]	C++	Python, MATLAB	CNN	✗	✓|✓
Neon[44]	Python	Python	All	✗	✓|✓
Chainer[45]	Python	Python	All	✓	✓|✓
Theano[46]	Python	Python	All	✗	✓|✓
MXNet[47]	Python	Python, Scala, Julia, Clojure, Java, C++, R, Perl	All	✓	✓|✓
ONNX[48]	Python	Python	CNN, DFFNN	✗	✓|✗
PaddlePaddle	Python	Python	CNN	✓	✓|✓
CNTK[49]	C++	Python, C++, C#	DFFNN, CNN, RNN	✓	✓|✓

CNN, convolutional neural networks; CPU, central processing unit; DFFNN, deep feed-forward neural networks; GPU, graphic processing unit; RNN, recurrent neural network.

Table 2. Relevant hyper-parameters

Hyper-parameter	*Architectures*	*Description*
Optimizer	All	Algorithm used to update the weights of each layer after each iteration.[53]
Learning rate	All	It determines the size of the step at each iteration of the optimization method.[54]
Number of epochs	All	Number of passes made in the whole training set.[55]
Batch size	All	Number of sub-samples that the network uses to update the weights.[56]
Hidden layers	All	It determines the depth of the neural network.[57]
Activation function	All	Introduces nonlinearity in the model, which allows the extraction of more complex knowledge.[58]
Momentum	All	It prevents oscillations in the convergence of the method.[59]
Weight initialization	All	It prevents the explosion or vanishing of the activation in the layers.[60]
Dropout	All	It eliminates certain connections between neurons in each iteration. It is used to prevent over-fitting.[61]
L1/L2 Regularization	All	It prevents over-fitting, stopping weights that are too high so that the model does not depend on a single feature.[62]
Units	RNN, DFFNN	It determines the level of knowledge that is extracted by each layer. It is highly dependent on the size of the data used.[57]
Kernel/filter	CNN	Matrix that moves over the input data. It allows the extraction of characteristics.[63]
Stride	CNN	The number of pixels that move over the input matrix for each filter.[64]
Padding	CNN	Number of null samples added to a dataset when it is processed by the kernel.[65]
Number of channels	CNN	Depth of the matrices involved in the convolutions.[66]
Pooling	CNN	It allows to reduce the number of parameters and calculations in the network.[67]
nb_stacks	TCN	Number of stacks of residual blocks.
Dilations	TCN	A deep stack of dilated convolutions to capture long-range temporal patterns.

TCN, temporal convolutional network.

the problem and the data. This makes the task of optimizing a model a challenge for the research community. Moreover, and taking into account the parameters described in Table 2, an immense number of possible combinations can be deduced. For this reason, various metaheuristics and optimization strategies are used. According to the literature, there are several strategies to optimize a set of hyper-parameters for deep-learning models, as shown in Table 3.

Thus, the hyper-parameter optimization methods can be classified into four major blocks:

1. Trial-error: This optimization method is based on varying each of the hyper-parameters manually. Therefore, this method implies a high time investment, having a relatively low computational cost and a low search space, because it requires the action of a user to modify the values manually each time a run is finished. Since in deep learning there are a large number of hyper-parameters and the values they can set are infinite, it is not advisable to use this optimization method.
2. Grid: The grid method explores the different possible combinations for a set of established hyper-parameters. This method covers a high search space, although it has a high computational cost associated with it, which makes this method unviable to apply in deep learning, let alone in big data environments.
3. Random: Random search allows to cover a high search space, because infinite combinations of hyper-parameters can be generated. Within this group we can differentiate between totally random or guided search strategies, such as those based on metaheuristics. Examples of this type of searches are the genetic algorithms,[68,69] particle swarm optimization,[70] or neuroevolution of augmenting topologies[71] algorithms, among others. The wide search range, added to the medium cost involved in this search strategy, makes it one of the best methods for optimizing deep-learning models. In addition, new hyper-parameters optimization metaheuristics are being published, such as the bioinspired model in the propagation of COVID-19 presented by the authors in Ref.[72]
4. Probabilistic: This optimization method tracks each of the evaluations. These evaluations are used to generate a probabilistic model that assigns values to the different hyper-parameters. The most common algorithms to optimize hyper-parameters by using probabilistic methods are those based on Bayesian approaches.[73]

Table 3. Search strategies

Strategy	*Deep learning*	*Cost*	*Search space*
Trial-error	✗	Low	Low
Grid	✗	High	High
Random	✓	Medium	High
Probabilistic	✓	Medium	Medium-driven

There are many libraries for the optimization of hyper-parameters in an automated way. However, very

Table 4. Hyper-parameters optimization libraries

Library	*Search strategy*	*Distributed*	*Language*	*Framework*
Elephas	Random, Probabilistic	Yes	Python	Keras
Hyperas	Random, Probabilistic	Yes	Python	Keras
Hyperopt[74]	Random, Probabilistic	Yes	Python	—
Dlopt[75]	Random	No	Python	Keras
Talos[76]	Grid, Random	Yes	Python	keras
Keras-tuner	Random	Yes	Python	Keras
H_2O[40]	Grid, Random	Yes	Python, R	H_2O
BoTorch[77]	Probabilistic	Yes	Python	PyTorch
HPOLib[78]	Probabilistic	—	Python	—

few are designed specifically for the optimization of deep-learning model hyper-parameters, being also compatible with the frameworks and high-level libraries described in Table 1. Table 4 summarizes a set of libraries for the optimization of hyper-parameters in deep-learning models, classifying them by search strategies, support to distributed computing, programming language, and compatible framework from Table 1. Note that it is not known whether HPOLib supports distributed computing or in which frameworks it works.

Hardware performance

One of the most important decisions a researcher must make is to determine the physical resources needed to ensure that deep-learning algorithms will find accurate models. Hence, this section overviews different hardware infrastructures typically used for deep-learning contexts, given its increasing demand for better and more sophisticated hardware.

Although a CPU can be used to execute deep-learning algorithms, the intensive computational requirements usually make the CPU physical resources insufficient (scalar architecture). For this reason, three different hardware architectures are typically used for mining information with deep-learning frameworks: GPU, tensor processing unit (TPU), and intelligence processing unit (IPU).

A GPU is a co-processor allocated in a CPU that is specifically designed to handle graphics in computing environments. The GPUs can have hundreds or even thousands of more cores than a CPU, but running at lower speeds. The GPUs achieve high data parallelism with single instructions, multiple data architecture and play an important role in the current artificial intelligence domain, with a wide variety of applications.

The first generation of TPUs was introduced in 2016, at the Google I/O Conference and they were specifically designed to run already trained neural networks. The TPUs are custom application-specific integrated circuits built specifically for machine learning. Compared with GPUs (frequently used for the same tasks since 2016), TPUs are implicitly designed for a larger volume of reduced precision calculation (e.g., from 8 bits of precision) and lack of hardware for rasterization/texture mapping. The term was coined for a specific chip designed for Google's TensorFlow framework. Generally speaking, TPUs have less accuracy compared with the computations performed on a normal CPU or GPU, but it is sufficient for the calculations they have to perform (an individual TPU can process more than 100 millions of pictures per day). Moreover, TPUs are highly optimized for large batches and CNNs and have the highest training throughput.[79]

The IPU is completely different from today's CPU and GPU processors. It is a highly flexible, easy-to-use, parallel processor that has been designed from the ground up to deliver state-of-the-art performance on current machine-learning models. But more importantly, the IPU has been designed to allow new and emerging machine intelligence workloads to be realized. The IPU delivers much better arithmetic efficiency on small batch sizes for both training and inference, which results in faster model convergence in training, models that generalize better, the ability to parallelize over many more IPU processors to reduce training time for a given batch size, and also delivers much higher throughput at lower latencies for inference. Another interesting feature is its lower power consumption compared with GPUs or TPUs (up to 20% less).

Table 5 summarizes the properties of the processing units explored in this section. Note that the performance is measured in flops and the cost in USD.

Table 5. Processing units properties

Units	*Architecture*	*Batch size*	*Performance*	*Cost*
CPU	Scalar	Small	$\sim 10^9$	$\sim 10^2$
GPU	Vector	Large	$\sim 10^{12}$	$\sim 10^3$
TPU	ASIC	Large	$\sim 10^{12}$	—
IPU	Graph	Small	$\sim 10^{15}$	$\sim 10^5$

Note that for TPUs cloud services are available for a price starting at 4.50 USD per hour (retrieved in March 2020).

Applications

To motivate the relevance of the time series prediction problem, an analysis of the state of the art has been carried out by classifying the deep-learning research works by application domain (such as energy and fuels, image and video, finance, environment, industry, or health) and the most widespread network architectures used (ENN, LSTM, GRU, BRNN, DFFNN, CNN, or TCN). A summary on the works reviewed can be found in Table 6.

An overview of the items for each application domain is made in the following paragraphs, to highlight the goals reached for each method and field:

1. Energy and fuels: With the increasing use of renewable energies, accurate estimates are needed to improve power system planning and operating. Many techniques have been used to make predictions, including deep learning.[195] Reviewing the literature in the past few years, it can be concluded that the vast majority of deep-learning architectures are suitable to this application area. For example, architectures based on LSTM,[90] ENN,[85,86] GRU,[94] BRNN,[95] and TCN[100] have been used to predict electricity demand consumption. LSTM[89] and CNN[99] have also been used to forecast photo-voltaic energy load. A GRU has been used to forecast soot emission in diesel engines in Ref.[93] An ensemble of DFFNN was developed by the authors in Ref.[98] to forecast time series of general purpose. After that, this strategy has been also used to forecast load demand time series.[97] In Ref.[91] the authors proposed an application of LSTM to forecast oil production. Hybrid architectures have been also used in this research field, for example, to forecast the price of carbon,[102] the price of energy in electricity markets,[101] energy consumption,[103] or solar power generation.[105]
2. Image and video: Image and video analysis is a very broad area of research, and it works related to any application domain. For example, Hu et al. conducted a wide study of deep learning for image-based cancer detection and diagnosis.[196] In Ref.[197] the authors summarized some techniques and studies used to recognize video sequence actions from timed images. The authors presented in Ref.[107] an application of an ENN to forecast and monitor the slopes displacement over photogrammetry performed by unmanned aerial vehicles. In Ref.[117] the authors combined GRU, RNN, and CNN to classify satellite image time series. Although all these works offer highly competitive results, the use of convolution-based networks predominates in the literature to solve forecasting problems using image or video time series data. On the one hand, CNNs have been used to forecast the combustion instability,[108] temporal dependencies in satellite images,[111] the speed of large-scale traffic[109] or to detect coronary artery stenosis,[113] among others. On the other hand, TCN are booming when it comes to analyzing images and videos. For example, Miao et al. used a TCN to estimate density maps from videos.[114] The authors in Ref.[116] also applied a TCN to summarize generic videos. Another interesting work in which images were used can be found in Ref.[115] In this work, they used a TCN model to dynamically detect stress through facial photographs.

Table 6. Summary of the works reviewed and classified into network architecture and application domain

	RNN				*DFFNN*	*CNN*		*Hybrid/others*
	ENN	*LSTM*	*GRU*	*BRNN*		*CNN*	*TCN*	
Energy and fuels	80–86	87–92	93,94	95	96–98	99	100	101–106
Image and video	107	—	—	—	—	108–113	114–116	117
Financial	—	118–121	120–122	121	123	120,124–126	—	120,127–133
Environmental	134–142	143–146	145,147–149	150	151	152	153	150,154,155
Industry	156,157	158–160	161,162	163,164	164,165	166	167,168	166,169,170
Health	—	171	—	172	173	113,174,175	—	176–181
Misc	—	—	182	183	184	185–187	188–192	193,194

BRNN, bidirectional recurrent neural network; CNN, convolutional neural networks; ENN, Elman network; LSTM, long-short term memory; GRU, gated recurrent units.

3. Financial: Financial analysis has been a challenging issue for decades. Therefore, there are many research works related to this application area, as described in Ref.[198] In addition, various architectures such as CNN,[124–126] DNN,[123] GRU,[122] or LSTM[118,119] have been used. Some authors make a comparison between some of these architectures, analyzing which one offers better results.[121] Although these studies are widespread, the complexity of the problem requires the search for new methodologies and architectures.[120,128–133]
4. Environmental: Environmental data analysis is one of the most popular areas for the scientific community. Many of these works are also based on the application of deep-learning techniques to forecast time series. The authors in Ref.[150] applied CNN and LSTM to forecast wind speed or temperature by using meteorological data from Beijing, China. Other authors focused on a single specific variable. For instance, the authors used TCN, GRU, ENN, BRNN, and LSTM architectures to forecast information related to wind in.[134,136,137,146,147,149,155] Water quality and demand were also predicted by using TCN and ENN in Refs.[140,153] An application of LSTM-based neural networks for correlated time series prediction was also proposed by Wan et al.[143] Further, carbon dioxide emissions,[139] flood,[143] or NH_3 concentration for swine house[199] were also predicted by using deep-learning techniques, in particular ENN.
5. Industry: In the industry sector, deep-learning techniques are also being used to carry out tasks of different kinds.[200] For instance, TCN and BRNN can be used to traffic flow forecasting.[163,167] The LSTM can be used for multiple purposes, such as process planning,[160] construction equipment recognition[158] or to improve the performance of organizations.[159,161] The authors in Ref.[164] used a DFFNN to forecast bath and metal height features in the electrolysis process. The ENN and GRU networks have been also used, for example, to forecast the useful life or degradation of the materials.[156,157,195] Deep-learning techniques are also widely applied to architecture, as can be seen in the in-depth study conducted by the authors in Ref.[201] It can be concluded that almost all network architectures have been used, given the wide variety of problems existing in this area.
6. Health: The use of deep-learning architectures in the area of health is common in the past years.[196,202] However, time series prediction using deep-learning models is not very widespread as time series are generally short in this field, along with the high computational cost involved in recurrent network training. The authors of Ref.[173] conducted a comprehensive study of time series prediction models in health care diagnosis and prognosis with a focus on cardiovascular disease. Instead, it is usual to apply convolution-based architectures or implement hybrid models. For example, the authors used CNN to accelerate the computation for magnetic resonance fingerprinting in Ref.[175] CNN was also used to monitoring the sleep stage in Ref.[176] for detecting premature problems as ventricular contractions[174] or to forecast the Sepsis.[180] In Ref.[179] the authors used a backpropagation network to forecast the incidence rate of pneumonia. Other network architecture such as LSTM can be used to forecast the status of critical patients according to their vital functions.[171] A recent study conducted by the authors in Ref.[181] uses some deep-learning architectures to forecast COVID-19 cases.
7. Miscellaneous: In recent years, the TCN has been one of the most widely checked general purpose architectures for time series forecasting.[182,189,190,192] However, any of the other network architectures can be applied to time series of miscellaneous application domains not classified in Table 6. For example, CNN and RNN can be used to detect human activity[186] or hybrid models to detect anomalies.[194] Namely, readers interested in cybersecurity can find a detailed description in Refs.[184,203]

From the previous analysis of Table 6, two main conclusions can be drawn. First, there exist several methods that have not been applied yet to particular application fields. Second, the existence of these gaps encourages the conduction of research in such lines.

Conclusions

Deep learning has proven to be one of the most powerful machine-learning techniques for solving complex problems dealing with big data. Most of the data mainly generated through smart devices are time series nowadays, and their prediction is one of the most frequent and current problems in almost all research areas. Thus, these two topics have been jointly analyzed in this survey to provide an overview of deep-learning techniques applied to time series forecasting. First, the most used deep-learning architectures for time series data in the past years have been described, with special emphasis on important practical aspects that can have a great influence on the reported results. In particular, it has placed focus on the search for hyper-parameters, the frameworks for deployment of the different architectures, and the existing hardware to lighten the hard training of the proposed network architectures. Second, a study of the deep neural networks used to predict time series in different application domains has been carried out in this survey, with the aim of providing a good comparative framework to be used in future works and to show which architectures have not been sufficiently tested in some applications.

Author Disclosure Statement

No competing financial interests exist.

Funding Information

This work was supported by the Spanish Ministry of Science, Innovation and Universities under project TIN2017-88209-C2-1-R. Also, this work has been partially supported by the General Directorate of Scientific Research and Technological Development (DGRSDT, Algeria), under the PRFU project (ref: C00 L07UN060120200003).

References

1. Plageras AP, Psannis KE, Stergiou C, et al. Efficient IoT-based sensor big data collection-processing and analysis in smart buildings. Future Gener Comput Syst. 2018;82:349–357.
2. Patil HP, Atique M. CDNB: CAVIAR-dragonfly optimization with naive bayes for the sentiment and affect analysis in social media. Big Data. 2020;8:107–124.
3. Gama J. Knowledge discovery from data streams. UK: Chapman & Hall/CRC, 2010.
4. Al-Jarrah OY, Yoo PD, Muhaidat S, et al. Efficient machine learning for big data: A review. Big Data Res. 2015;2:87–93.
5. Dhar V, Sun C, Batra P. Transforming finance into vision: Concurrent financial time series as convolutional net. Big Data. 2019; 7:276–285.
6. Nguyen G, Dlugolinsky S, Bobák M, et al. Machine learning and deep learning frameworks and libraries for large-scale data mining: A survey. Artif Intell Rev. 2019;52:77–124.
7. Maji P, Mullins R. On the reduction of computational complexity of deep convolutional neural networks. Entropy. 2018;20:305.
8. Schmidhuber J. Deep learning in neural networks: An overview. Neural Netw. 2015;61:85–117.
9. Makridakis S, Wheelwright SC, Hyndman RJ. Forecasting methods and applications. USA: John Wiley and Sons, 2008.
10. Chatfield C. The analysis of time series: An introduction. UK: Chapman & Hall/CRC, 2003.
11. Box GEP, Jenkins GM. Time series analysis: forecasting and control. USA: John Wiley and Sons, 2008.
12. Martínez-Álvarez F, Troncoso A, Asencio-Cortés G, Riquelme JC. A survey on data mining techniques applied to electricity-related time series forecasting. Energies 2015;8:13162–13193.
13. Zhang Q, Yang LT, Chen Z, et al. A survey on deep learning for big data. Inf Fusion 2018;42:146–157.
14. Fawaz HI, Forestier G, Weber J, et al. Deep learning for time series classification: A review. Data Min Knowl Discov 2019;33:917–963.
15. Bagnall A, Lines J, Vickers W, et al. 2017. The UEA & UCR time series classification repository. Available online at www.timeseriesclassification .com. (last accessed on April 30, 2020).
16. Mayer R, Jacobsen HA. Scalable deep learning on distributed infrastructures: challenges, techniques, and tools. ACM Comput Surv 2020; 53:Article 3.
17. Buuren S. Flexible imputation of missing data. UK: Chapman & Hall/CRC, 2012.
18. Maronna RA, Martin RD, Yohai VJ. Robust statistics: theory and methods. USA: Wiley, 2006.
19. Fu TC. A review on time series data mining. Eng Appl Artif Intell. 2011;24: 164–181.
20. Shumway RH, Stoffer DS. Time series analysis and its applications (with R examples). USA: Springer, 2011.
21. Hyndman RJ, Athanasopoulos G. Forecasting: principles and practice. Australia: Otexts, 2018.
22. Wang X, Kang Y, Hyndman RJ, et al. Distributed ARIMA models for ultra-long time series. *arXiv e-prints*, arXiv:2007.09577, 2020.
23. Rakthanmanon T, Campana B, Mueen A, et al. Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. ACM Trans Knowl Discov Data 2013;7:10.
24. Torres JF, Galicia A, Troncoso A, et al. A scalable approach based on deep learning for big data time series forecasting. Integr Comput Aided Eng 2018;25:335–348.
25. Galicia A, Talavera-Llames RL, Troncoso A, et al. Multi-step forecasting for big data time series based on ensemble learning. Knowl Based Syst 2019;163:830–841.
26. Talavera-Llames R, Pérez-Chacón R, Troncoso A, et al. Big data time series forecasting based on nearest neighbors distributed computing with spark. Knowl Based Syst 2018;161:12–25.
27. Talavera-Llames R, Pérez-Chacón R, Troncoso A, Martínez-Álvarez F. MV-kWNN: A novel multivariate and multi-output weighted nearest neighbors algorithm for big data time series forecasting. Neurocomputing 2019;353:56–73.
28. Pérez-Chacón R, Asencio-Cortés G, Martínez Álvarez F, et al. Big data time series forecasting based on pattern sequence similarity and its application to the electricity demand. Inf Sci 2020;540: 160–174.
29. Rumelhart D, Hinton G, Williams R. Long short-term memory. Nature 1986;323:533–536.
30. Elman JL. Finding structure in time. Energy Rep 1990;14:179–211.
31. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9:1735–1780.
32. Chung J, Gulcehre C, Cho K, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. In: Proceedings of the Neural Information Processing Systems, Canada, 2014. pp. 1–12.
33. Cho K, Merrienboer BV, Bahdanau D, et al. On the properties of neural machine translation: encoder-decoder approaches. In: Proceedings of SSST-8. Qatar, 2014. pp. 103–111.
34. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. In: Proceedings of the International Conference on Learning Representations, 2015, pp. 149–158.
35. Xu K, Ba J, Kiros R, et al. Show, attend and tell: Neural image caption generation with visual attention. In: Proceedings of the International Conference on Machine Learning, 2015, pp. 2048–2057.
36. Fukushima K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybern 1980;36:193–202.
37. Zhang W, Hasegawa A, Matoba O, et al. Shift-invariant neural network for image processing: Learning and generalization. Appl Artif Neural Netw III 1992;1709:257–268.
38. Alla S, Adari SK. Beginning anomaly detection using Python-based deep learning. USA: Apress, 2019.
39. Abadi M, Agarwal A, Barham P, et al. 2015. TensorFlow: large-scale machine learning on heterogeneous systems. Available online at http://tensorflow.org. (last accessed on April 30, 2020).
40. Candel A, LeDell E, Arora A, et al. 2015. Deep learning with h2o. Available online at http://h2o.ai/resources. (last accessed on April 30, 2020).

41. Eclipse Deeplearning4j development team. 2016. DL4J: deep learning for Java. Available online at https://github.com/eclipse/deeplearning4j. (last accessed on April 30, 2020).
42. Paszke A, Gross S, Massa F, et al. Pytorch: an imperative style, highperformance deep learning library. In: Proceedings of the Advances in Neural Information Processing Systems. Vol. 32, 2019. pp. 8026–8037.
43. Jia Y, Shelhamer E, Donahue J, et al. Caffe: Convolutional architecture for fast feature embedding. arXiv e-prints, arXiv:1408.5093, 2014.
44. Intel Nervana systems. Neon deep learning framework 2017. Available online at https://github.com/NervanaSystems/neon, 2017. (last accessed on April 30, 2020).
45. Tokui S, Okuta R, Akiba T, et al. Chainer: A deep learning framework for accelerating the research cycle. In: Proceedings of International Conference on Knowledge Discovery and Data Mining 2019, pp. 2002–2011.
46. Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints, arXiv:1605.02688, 2016.
47. Tianqi C, Li M, Li Y, et al. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv e-prints arXiv:1512.01274, 2015.
48. Bai J, Lu F, Zhang K, et al. 2019. Onnx: Open neural network exchange. Available online at https://github.com/onnx/onnx. (last accessed on June 15, 2020).
49. Seide F, Agarwal A. CNTK: Microsoft's open-source deep-learning toolkit. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. USA, 2016, pp. 2135–2135.
50. Chollet F. 2015. Keras. Available online at https://keras.io
51. DeepMind revision. Sonnet, 2019.
52. Guo J, He H, He T, et al. Gluoncv and gluonnlp: Deep learning in computer vision and natural language processing. J Mach Learn Res 2020; 21:1–7.
53. Choi D, Shallue CJ, Nado Z, et al. On empirical comparisons of optimizers for deep learning. arXiv e-prints, arXiv:1910.05446, 2019.
54. You K, Long M, Wang J, Jordan MI. How does learning rate decay help modern neural networks? In Proceedings of the International Conference on Learning Representations, 2019. pp. 1–14.
55. Sinha S, Singh TN, Singh V, Verma A. Epoch determination for neural network by self-organized map (SOM). Comput Geosci 2010 14:199–206.
56. Masters D, Luschi C. Revisiting small batch training for deep neural networks. arXiv e-prints, arXiv:1804.07612, 2018.
57. Shafi I, Ahmad J, Shah SI, et al. Impact of varying neurons and hidden layers in neural network architecture for a time frequency application. In: Proceedings of the IEEE International Multitopic Conference. USA, 2006. pp. 188–193.
58. Ding B, Qian H, Zhou J. Activation functions and their characteristics in deep neural networks. In: Proceedings of the Chinese Control and Decision Conference. USA, 2018. pp. 1836–1841.
59. Sutskever I, Martens J, Dahl G, et al. On the importance of initialization and momentum in deep learning. In: Proceedings of the International Conference on Machine Learning, 2013. pp. 1139–1147.
60. Kumar SK. On weight initialization in deep neural networks. arXiv e-prints, arXiv:1704.08863, 2017.
61. Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: A simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15: 1929–1958.
62. Ng AY. Feature selection, l1 vs. l2 regularization, and rotational invariance. In: Proceedings of the ACM International Conference on Machine Learning. USA, 2004. pp. 78–85.
63. Mairal J, Koniusz P, Harchaoui Z, Schmid C. Convolutional kernel networks. In: Proceedings of the Neural Information Processing Systems. USA, 2014. pp. 1–9.
64. Zaniolo L, Marques O. On the use of variable stride in convolutional neural networks. Multimed Tools Appl. 2020;79:13581–13598.
65. Dwarampudi M, Reddy NVS. Effects of padding on LSTMs and CNNs. arXiv e-prints, arXiv:1903.07288, 2019.
66. Zhu H, An Z, Yang C, et al. Rethinking the number of channels for the convolutional neural network. arXiv e-prints, arXiv:1909.01861, 2019.
67. Scherer F, Müller A, Behnke S. Evaluation of pooling operations in convolutional architectures for object recognition. In: Proceedings of Artificial Neural Networks. USA, 2010. pp. 92–101.
68. Ma B, Li X, Xia Y, et al. Autonomous deep learning: A genetic DCNN designer for image classification. Neurocomputing 2020;379:152–161.
69. Itano F, De-Abreu-De-Sousa MA, Del-Moral-Hernandez E. Extending MLP ANN hyper-parameters optimization by using genetic algorithm. In: Proceedings of the IEEE International Joint Conference on Neural Networks. USA, 2018. pp. 1–8.
70. Kennedy K, Eberhart R. Particle swarm optimization. In: Proceedings of International Conference on Neural Networks. Vol. 4, USA, 1995. pp. 1942–1948.
71. Stanley KO, Miikkulainen R. Evolving neural networks through augmenting topologies. Evol Comput. 2002;10:99–127.
72. Martínez-Álvarez F, Asencio-Cortés G, Torres JF, et al. Coronavirus optimization algorithm: A bioinspired metaheuristic based on the COVID-19 Propagation Model. Big Data 2020;8:308–322.
73. Ranjit MP, Ganapathy G, Sridhar K, et al. Efficient deep learning hyperparameter tuning using cloud infrastructure: intelligent distributed hyperparameter tuning with bayesian optimization in the cloud. In: Proceedings of International Conference on Cloud Computing. USA, 2019. pp. 520–522.
74. Bergstra J, Yamins D, Cox DD. Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In: Proceedings of the International Conference on International Conference on Machine Learning, 2013. pp. 115–123.
75. Camero A, Toutouh J, Alba E. Dlopt: Deep learning optimization library. arXiv e-prints, arXiv:1807.03523, 2018.
76. Autonomio. Talos. Available online at http://github.com/autonomio/talos, 2019.
77. Balandat M, Karrer B, Jiang DR, et al. BoTorch: Programmable Bayesian Optimization in PyTorch. arXiv e-prints, arXiv:1910.06403, 2019.
78. Eggensperger K, Feurer M, Hutter F, et al. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In: Proceedings of the Neural Information Processing Systems. USA, 2013. pp. 1–5.
79. Wang YE, Wei GY, Brooks D. Benchmarking TPU, GPU, and CPU platforms for deep learning. arXiv e-prints, arXiv:1907.10701, 2019.
80. Yu D, Wang Y, Liu H, et al. System identification of PEM fuel cells using an improved Elman neural network and a new hybrid optimization algorithm. Energy Rep. 2019;5:1365–1374.
81. Zheng Y, Yao Z, Zhou H, et al. Power generation forecast of top gas recovery turbine unit based on Elman model. In: Proceedings of the IEEE Chinese Control Conference. USA, 2018. pp. 7498–7501.
82. Ruiz LGB, Rueda R, Cuéllar MP, et al. Energy consumption forecasting based on Elman neural networks with evolutive optimization. Expert Syst Appl. 2018;92:380–389.
83. Wang J, Lv Z, Liang Y, et al. Fouling resistance prediction based on GA–Elman neural network for circulating cooling water with electromagnetic anti-fouling treatment. J Energy Inst. 2019;92:1519–1526.
84. Li W, Jiao Z, Du L, et al. An indirect RUL prognosis for lithium-ion battery under vibration stress using Elman neural network. Int J Hydrog Energy. 2019;44:12270–12276.
85. Yu Y, Wang X, Bründlinger R. Improved Elman neural network short-term residents load forecasting considering human comfort index. J Electr Eng Technol. 2019;14:2315–2322.
86. Li D, Wang H, Zhang Y, et al. Power grid load state information perception forecasting technology for battery energy storage system based on Elman neural network. In: Proceedings of information Technology, Networking, Electronic and Automation Control Conference. USA, 2019. pp. 914–917.
87. Abdel-Nasser M, Mahmoud K. Accurate photovoltaic power forecasting models using deep LSTM-RNN. Neural Comput Appl. 2019;31:2727–2740.
88. Khodabakhsh A, Ari I, Bakır M, et al. Forecasting multivariate time-series data using LSTM and mini-batches. In: Proceedings of Data Engineering and Communications Technologies, 2020. pp. 121–129.
89. Gao M, Li J, Hong F, et al. Day-ahead power forecasting in a large-scale photovoltaic plant based on weather classification using LSTM. Energy. 2019;187:115838.
90. Muzaffar S, Afshari A. Short-term load forecasts using LSTM networks. Proc Energy Proc. 2019;158:2922–2927.
91. Song X, Liu Y, Xue L, et al. Time-series well performance prediction based on long short-term memory (LSTM) neural network model. J Pet Sci Eng. 2020;186:106682.

92. Wang JQ, Du Y, Wang J. LSTM based long-term energy consumption prediction with periodicity. Energy. 2020;197:117197.
93. Gokhan A, Yilmaz E, Unel M, et al. Estimating soot emission in diesel engines using gated recurrent unit networks. IFAC Papers Online. 2019;52:544–549.
94. Wu W, Liao W, Miao J, et al. Using gated recurrent unit network to forecast short-term load considering impact of electricity price. Proc Energy Proc. 2019;158:3369–3374.
95. Tang X, Dai Y, Wang T, et al. Short-term power load forecasting based on multi-layer bidirectional recurrent neural network. IET Gener Transm Distrib. 2019;13:3847–3854.
96. Shao Z, Zheng Q, Yang S, et al. Modeling and forecasting the electricity clearing price: A novel BELM based pattern classification framework and a comparative analytic study on multi-layer BELM and LSTM. Energy Econ. 2020;86:104648.
97. Qiu X, Ren Y, Suganthan PN, et al. Empirical mode decomposition based ensemble deep learning for load demand time series forecasting. Appl Soft Comput J. 2017;54:246–255.
98. Qiu X, Zhang L, Ren Y, et al. Ensemble deep learning for regression and time series forecasting. In: Proceedings of the IEEE Symposium Series on Computational Intelligence in Ensemble Learning. USA, 2014. pp. 1–6.
99. Manohar M, Koley E, Ghosh S, et al. Spatio-temporal information based protection scheme for PV integrated microgrid under solar irradiance intermittency using deep convolutional neural network. Int J Electr Power Energy Syst. 2020;116:105576.
100. Mishra K, Basu S, Maulik U. DaNSe: A dilated causal convolutional network based model for load forecasting. Lect Notes Comput Sci. 2019; 11941:234–241.
101. Qiao W, Yang Z. Forecast the electricity price of U.S. using a wavelet transform-based hybrid model. Energy 2020;193:116704.
102. Ji L, Zou Y, He K, et al. Carbon futures price forecasting based with ARIMA-CNN-LSTM model. Proc Comput Sci. 2019;162:33–38.
103. Kim TY, Cho SB. Predicting residential energy consumption using CNN-LSTM neural networks. Energy. 2019;182:72–81.
104. Shen M, Xu Q, Wang K, et al. Short-term bus load forecasting method based on cnn-gru neural network. Lect Notes Electr Eng 2020;585: 711–722.
105. AlKandari M, Ahmad I. Solar power generation forecasting using ensemble approach based on deep learning and statistical methods. Appl Comput Inf. 2020;6:1–20.
106. Kong Z, Tang B, Deng L, et al. Condition monitoring of wind turbines based on spatio-temporal fusion of SCADA data by convolutional neural networks and gated recurrent units. Renew Energy 2020;146: 760–768.
107. Wang S, Zhang Z, Ren Y, et al. UAV photogrammetry and AFSA-Elman neural network in slopes displacement monitoring and forecasting. KSCE J Civil Eng. 2020;24:19–29.
108. Sarkar S, Lore KG, Sarkar S, et al. Early detection of combustion instability from hi-speed flame images via deep learning and symbolic time series analysis. In: Proceedings of the Annual Conference of the Prognostics and Health Management Society. USA, 2015. pp. 353–362.
109. Ma X, Dai Z, He Z, et al. Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. Sensors 2017;17:818.
110. Chen W, Shi K. A deep learning framework for time series classification using relative position matrix and convolutional neural network. Neurocomputing 2019;359:384–394.
111. Ienco D, Interdonato R, Gaetano R, et al. Combining Sentinel-1 and Sentinel-2 satellite image time series for land cover mapping via a multi-source deep learning architecture. J Photogrammetry Remote Sens. 2019;158:11–22.
112. Martínez-Arellano G, Terrazas G, Ratchev S. Tool wear classification using time series imaging and deep learning. Int J Adv Manuf Technol. 2019; 104:3647–3662.
113. Wu W, Zhang J, Xie H, et al. Automatic detection of coronary artery stenosis by convolutional neural network with temporal constraint. Comput Biol Med 2020;118:103657.
114. Miao Y, Han J, Gao Y, Zhang B. ST-CNN: Spatial-Temporal Convolutional Neural Network for crowd counting in videos. Pattern Recogn Lett. 2019;125:113–118.
115. Feng S. Dynamic facial stress recognition in temporal convolutional network. In: Proceedings of the Communications in Computer and Information Science. USA, 2019. pp. 698–706.
116. Zhang Y, Kampffmeyer M, Liang X, et al. Dilated temporal relational adversarial network for generic video summarization. Multimed Tools Appl. 2019;78:35237–35261.
117. Interdonato R, Ienco D, Gaetano R, et al. DuPLO: A DUal view Point deep Learning architecture for time series classificatiOn. ISPRS J Photogramm Remote Sens. 2019;149:91–104.
118. Yan H, Ouyang H. Financial time series prediction based on deep learning. Wireless Pers Commun. 2018;102:683–700.
119. Sismanoglu G, Onde M, Kocer F, et al. Deep learning based forecasting in stock market with big data analytics. In: Proceedings of the IEEE Scientific Meeting on Electrical-Electronics and Biomedical Engineering and Computer Science. USA, 2019. pp. 10057–10059.
120. Jayanth BA, Harish RDS, Nair BB. Applicability of deep learning models for stock price forecasting an empirical study on bankex data. Proc Comput Sci. 2018;143:947–953.
121. Jiang M, Liu J, Zhang L, et al. An improved stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms. Physica A 2020;541:122272.
122. Wu W, Wang Y, Fu J, et al. Preliminary study on interpreting stock price forecasting based on tree regularization of GRU. In: Proceedings of Communications in Computer and Information Science. Vol. 1059, USA, 2019. pp. 476–487.
123. Orimoloye LO, Sung MC, Ma T, et al. Comparing the effectiveness of deep feedforward neural networks and shallow architectures for predicting stock price indices. Expert Syst Appl. 2020;139:112828.
124. Makarenko AV. Deep learning algorithms for estimating Lyapunov exponents from observed time series in discrete dynamic systems. In: Proceedings of International Conference Stability and Oscillations of Nonlinear Control Systems. USA, 2018. pp. 1–4.
125. Dingli A, Fournier KS. Financial time series forecasting—a deep learning approach. Int J Mach Learn Comput. 2017;7:118–122.
126. Kelotra A, Pandey P. Stock market prediction using Optimized Deep-ConvLSTM Model. Big Data 2020;8:5–24.
127. Ni L, Li Y, Wang X, et al. Forecasting of Forex time series data based on deep learning. Proc Comput Sci. 2019;147:647–652.
128. 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.
129. Munkhdalai L, Li M, Theera-Umpon N, et al. VAR-GRU: A hybrid model for multivariate financial time series prediction. Lect Notes Artif Intell. 2020;12034:322–332.
130. Chen CT, Chiang LK, Huang YC, et al. Forecasting interaction of exchange rates between fiat currencies and cryptocurrencies based on deep relation networks. In: Proceedings of the IEEE International Conference on Agents. USA, 2019. pp. 69–72.
131. Berradi Z, Lazaar M. Integration of principal component analysis and recurrent neural network to forecast the stock price of Casablanca stock exchange. Proc Comput Sci 2019;148:55–61.
132. Wang Q, Xu W, Huang X, et al. Enhancing intraday stock price manipulation detection by leveraging recurrent neural networks with ensemble learning. Neurocomputing 2019;347:46–58.
133. Long W, Lu Z, Cui L. Deep learning-based feature engineering for stock price movement prediction. Knowl Based Syst. 2019;164:163–173.
134. Liu H, Tian HQ, Liang XF, et al. Wind speed forecasting approach using secondary decomposition algorithm and Elman neural networks. Appl Energy. 2015;157:183–194.
135. Yu C, Li Y, Xiang H, et al. Data mining-assisted short-term wind speed forecasting by wavelet packet decomposition and Elman neural network. J Wind Eng Indus Aerodyn. 2018;175:136–143.
136. Liu H, Wei MX, Fei LY. Wind speed forecasting method based on deep learning strategy using empirical wavelet transform, long short term memory neural network and Elman neural network. Energy Conv Manage. 2018;156:498–514.
137. Zhang Y, Pan G. A hybrid prediction model for forecasting wind energy resources. Environ Sci Pollut Res. 2020;27:19428–19446.
138. Zhang L, Xie Y, Chen A, et al. A forecasting model based on enhanced Elman neural network for air quality prediction. Lect Notes Electr Eng. 2019;518:65–74.

139. Huang Y, Shen L. Elman neural network optimized by firefly algorithm for forecasting China's carbon dioxide emissions. Commun Comput Inf Sci. 2018;951:36–47.
140. Xiao D, Hou S, Li WZ, et al. Hourly campus water demand forecasting using a hybrid EEMD-Elman neural network model. In: Sustainable Development of Water Resources and Hydraulic Engineering in China. Environmental Earth Sciences. USA, 2019. pp. 71–80.
141. Shen W, Fu X, Wang R, et al. A prediction model of NH3 concentration for swine house in cold region based on empirical mode decomposition and Elman neural network. Inf Proc Agric. 2019;6:297–305.
142. Wan X, Yang Q, Jiang P, et al. A hybrid model for real-time probabilistic flood forecasting using Elman neural network with heterogeneity of error distributions. Water Resour Manage. 2019;33:4027–4050.
143. Wan H, Guo S, Yin K, et al. CTS-LSTM: LSTM-based neural networks for correlatedtime series prediction. Knowl Based Syst. 2019;191:105239.
144. Freeman BS, Taylor G, Gharabaghi B, et al. Forecasting air quality time series using deep learning. J Air Waste Manage Assoc. 2018;68:866–886.
145. De-Melo GA, Sugimoto DN, Tasinaffo PM, et al. A new approach to river flow forecasting: LSTM and GRU multivariate models. IEEE Latin Am Trans. 2019;17:1978–1986.
146. Chen J, Zeng GQ, Zhou W, et al. Wind speed forecasting using nonlinear-learning ensemble of deep learning time series prediction and extremal optimization. Energy Conv Manage. 2018;165:681–695.
147. Niu Z, Yu Z, Tang W, et al. Wind power forecasting using attention-based gated recurrent unit network. Energy. 2020;196:117081.
148. Li W, Wu H, Zhu N, et al. Prediction of dissolved oxygen in a fishery pond based on gated recurrent unit (GRU). Inf Process Agric. 2020, In press.
149. Peng Z, Peng S, Fu L, et al. A novel deep learning ensemble model with data denoising for short-term wind speed forecasting. Energy Convers Manage. 2020;207:112524.
150. Jin X, Yu X, Wang X, et al. Prediction for time series with CNN and LSTM. Lect Notes Electr Eng 2020;582:631–641.
151. Maqsood H, Mehmood I, Maqsood M, et al. A local and global event sentiment based efficient stock exchange forecasting using deep learning. Int J Inf Manage. 2020;50:432–451.
152. O'Shea TJ, Roy T, Clancy TC. Over-the-air deep learning based radio signal classification. IEEE J Select Topics Signal Process. 2018;12:168–179.
153. Zhang Y, Thorburn PJ, Fitch P. Multi-task temporal convolutional network for predicting water quality sensor data. In: Proceedings of Neural Information Processing Communications in Computer and Information Science. USA, 2019. pp. 122–130.
154. Sun Y, Zhao Z, Ma X, et al. Short-timescale gravitational microlensing events prediction with ARIMA-LSTM and ARIMA-GRU hybrid model. Lect Notes Comput Sci. 2019;11473:224–238.
155. Liu H, Mi X, Li Y, et al. Smart wind speed deep learning based multi-step forecasting model using singular spectrum analysis, convolutional gated recurrent unit network and support vector regression. Renew Energy. 2019;143:842–854.
156. Li X, Zhang L, Wang Z, et al. Remaining useful life prediction for lithium-ion batteries based on a hybrid model combining the long short-term memory and Elman neural networks. J Energy Storage. 2019;21:510–518.
157. Yang L, Wang F, Zhang J, et al. Remaining useful life prediction of ultrasonic motor based on Elman neural network with improved particle swarm optimization. Measurement. 2019;143:27–38.
158. Rashid KM, Louis J. Times-series data augmentation and deep learning for construction equipment activity recognition. Adv Eng Inform. 2019;42:100944.
159. Huang X, Zanni-Merk C, Crémilleux B. Enhancing deep learning with semantics: An application to manufacturing time series analysis. Proc Comput Sci. 2019;159:437–446.
160. Mehdiyev N, Lahann J, Emrich A, et al. Time series classification using deep learning for process planning: A case from the process industry. Proc Comput Sci 2017;114:242–249.
161. Wang S, Chen J, Wang H, et al. Degradation evaluation of slewing bearing using HMM and improved GRU. Measurement. 2019;146:385–395.
162. Wang J, Yan J, Li C, et al. Deep heterogeneous GRU model for predictive analytics in smart manufacturing: Application to tool wear prediction. Comput Indus. 2019;111:1–14.
163. Bohan H, Yun B. Traffic flow prediction based on BRNN. In: Proceedings of the IEEE International Conference on Electronics Information and Emergency Communication. USA, 2019. pp. 320–323.
164. Pasias A, Vafeiadis T, Ioannidis D, et al. Forecasting bath and metal height features in electrolysis process. In: Proceedings of the International Conference on Distributed Computing in Sensor Systems. 2019. pp. 312–317.
165. Jiang P, Chen C, Liu X. Time series prediction for evolutions of complex systems: A deep learning approach. In: Proceedings of rhe IEEE International Conference on Control and Robotics Engineering, 2016. pp. 1–6.
166. Canizo M, Triguero I, A Conde, et al. Multi-head CNN–RNN for multi-time series anomaly detection: An industrial case study. Neurocomputing. 2019;363:246–260.
167. Kuang L, Hua C, Wu J, et al. Traffic volume prediction based on multi-sources GPS trajectory data by temporal convolutional network. Mobile Netw Appl. 2020;25:1–13.
168. Wu P, Sun J, Chang X, et al. Data-driven reduced order model with temporal convolutional neural network. Comput Methods Appl Mech Eng. 2020;360:112766.
169. Varona B, Monteserin A, Teyseyre A. A deep learning approach to automatic road surface monitoring and pothole detection. Pers Ubiquitous Comput. 2020;24:519–534.
170. Cai M, Pipattanasomporn M, Rahman S. Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques. Appl Energy. 2019;236:1078–1088.
171. da Silva DB, Schmidt D, da Costa CA, da Rosa Righi R, Eskofier B. Deepsigns: A predictive model based on deep learning for the early detection of patient health deterioration. Expert Syst Appl. 2021;165: 113905.
172. Yu W, Kim Y, Mechefske C. Remaining useful life estimation using a bi-directional recurrent neural network based autoencoder scheme. Mech Syst Signal Process 2019;129:764–780.
173. Bui C, Pham N, Vo A, Tran A, Nguyen A, Le T. Time series forecasting for healthcare diagnosis and prognostics with the focus on cardiovascular diseases. In: Proceedings of the International Conference on the Development of Biomedical Engineering in Vietnam. USA, 2018. pp. 809–818.
174. Liu Y, Huang Y, Wang J, et al. Detecting premature ventricular contraction in children with deep learning. J Shanghai Jiaotong Univ. 2018;23: 66–73.
175. Hoppe E, Körzdörfer G, Würfl T, et al. Deep learning for magnetic resonance fingerprinting: A new approach for predicting quantitative parameter values from time series. In: Studies in Health Technology and Informatics, Vol. 243. Netherlands: IOS Press, 2017. pp. 202–206.
176. Chambon S, Galtier MN, Arnal PJ, et al. A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series. IEEE Trans Neural Syst and Rehabil Eng. 2018;26: 758–769.
177. Lauritsen SM, Kalør ME, Kongsgaard EL, et al. Early detection of sepsis utilizing deep learning on electronic health record event sequences. Artif Intell Med. 2020;104:101820.
178. Chen X, He J, Wu X, et al. Sleep staging by bidirectional long short-term memory convolution neural network. Fut Gen Comput Syst. 2020;109: 188–196.
179. Liang liang M, Fu peng T. Pneumonia incidence rate predictive model of nonlinear time series based on dynamic learning rate BP neural network. In: Proceedings of the Fuzzy Information and Engineering. USA, 2010. pp. 739–749.
180. Sarafrazi S, Choudhari RS, Mehta C, et al. Cracking the "sepsis" code: Assessing time series nature of ehr data, and using deep learning for early sepsis prediction. In: Proceedings of the Computing in Cardiology. UK, 2019. pp. 1–4.
181. Zeroual A, Harrou F, Dairi A, Sun Y. Deep learning methods for forecasting covid-19 time-series data: A comparative study. Chaos Solit Fract. 2020;140:110121.
182. Zhang X, Shen F, Zhao J, et al. Time series forecasting using GRU neural network with multi-lag after decomposition. Lect Notes Comput Sci. 2017;10638:523–532.
183. Zhao X, Xia L, Zhang J, et al. Artificial neural network based modeling on unidirectional and bidirectional pedestrian flow at straight corridors. Physica A. 2020;547:123825.

184. Imamverdiyev Y, Abdullayeva F. Deep learning method for denial of service attack detection based on restricted boltzmann machine. Big Data. 2018;6:159–169.
185. Munir M, Siddiqui SA, Dengel A, et al. DeepAnT: A deep learning approach for unsupervised anomaly detection in time series. IEEE Access. 2019;7:1991–2005.
186. Zebin T, Scully PJ, Ozanyan KB. Human activity recognition with inertial sensors using a deep learning approach. In: Proceedings of IEEE Sensors. USA, 2017. pp. 1–3.
187. Bendong Z, Huanzhang L, Shangfeng C, et al. Convolutional neural networks for time series classification. J Syst Eng Electr. 2017;28: 162–169.
188. Jiang W, Wang Y, Tang Y. A sequence-to-sequence transformer premised temporal convolutional network for chinese word segmentation. In: Proceedings of Parallel Architectures, Algorithms and Programming. USA, 2020. pp. 541–552.
189. Shao J, Shen H, Cao Q, et al. Temporal convolutional networks for popularity prediction of messages on social medias. Lect Notes Comput Sci. 2019;:135–147.
190. Chen Y, Kan Y, Chen Y, et al. Probabilistic forecasting with temporal convolutional neural network. Neurocomputing. 2020;399:491–501.
191. Xi R, Hou M, Fu M, et al. Deep dilated convolution on multimodality time series for human activity recognition. In: Proceedings of the IEEE International Joint Conference on Neural Networks. USA, 2018. pp. 53381–53396.
192. Wang R, Peng C, Gao J, et al. A dilated convolution network-based LSTM model for multi-step prediction of chaotic time series. Comput Appl Math. 2020;39:1–22.
193. Rodrigues F, Markou I, Pereira FC. Combining time-series and textual data for taxi demand prediction in event areas: A deep learning approach. Inf Fusion. 2019;49:120–129.
194. Kalinin MO, Lavrova DS, Yarmak AV. Detection of threats in cyberphysical systems based on deep learning methods using multidimensional time series. Autom Control Comput Sci. 2018;52:912–917.
195. Wang H, Lei Z, Zhang X, et al. A review of deep learning for renewable energy forecasting. Energy Conv Manage. 2019;198:111799.
196. Hu Z, Tang J, Wang Z, et al. Deep learning for image-based cancer detection and diagnosis—A survey. Pattern Recogn. 2018;83:134–149.
197. Atto AM, Benoit A, Lambert P. Timed-image based deep learning for action recognition in video sequences. Pattern Recogn. 2020;104: 107353.
198. Sezer OB, Gudelek M, Ozbayoglu AM. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. Appl Soft Comput. 2020;90:106181.
199. Shen Z, Zhang Y, Lu J, et al. A novel time series forecasting model with deep learning. Neurocomputing. 2020;396:302–313.
200. Wang Y, Zhang D, Liu Y, et al. Enhancing transportation systems via deep learning: A survey. Transp Res Part C Emerg Technol. 2019;99: 144–163.
201. Kamilaris A, Prenafeta-Boldú FX. Deep learning in agriculture: A survey. Comput Electr Agric. 2018;147:70–90.
202. Rui Z, Ruqiang Y, Zhenghua C, et al. Deep learning and its applications to machine health monitoring. Mech Syst Signal Process. 2019;115: 213–237.
203. Mahdavifar S, Ghorbani AA. Application of deep learning to cybersecurity: A survey. Neurocomputing 2019;347:149–176.

Cite this article as: Torres JF, Hadjout D, Sebaa A, Martínez-Álvarez F, Troncoso A (2020) Deep learning for time series forecasting: a survey. *Big Data* 3:X, 1–19, DOI: 10.1089/big.2020.0159.

Abbreviations Used

BRNN	=	bidirectional recurrent neural network
CNN	=	convolutional neural networks
CPU	=	central processing unit
DFFNN	=	deep feed forward neural networks
DRNN	=	deep recurrent neural network
ENN	=	Elman network
GPU	=	graphic processing unit
GRU	=	gated recurrent units
IPU	=	intelligence processing unit
LSTM	=	long-short term memory
NLP	=	natural language processing
RNN	=	recurrent neural network
TCN	=	temporal convolutional network
TPU	=	tensor processing unit