Applied Intelligence
https://doi.org/10.1007/s10489-022-03890-w

Taylor-based optimized recursive extended exponential smoothed neural networks forecasting method

Emna Krichene[1] · Wael Ouarda[2] · Habib Chabchoub[3] · Ajith Abraham[4] · Abdulrahman M. Qahtani[5] · Omar Almutiry[6] · Habib Dhahri[6] · Adel M. Alimi[1,7]

Accepted: 11 June 2022

Abstract
The development of a Time series Forecasting System is a major concern for Artificial Intelligence researchers. Commonly, existing systems only assess temporal features and analyze the behavior of the data over time, thus, resulting in uncertain forecasting accuracy. Although many forecasting systems were proposed in the literature; they have not yet answered the attending question. Hence, to overcome this problematic, we propose an innovative method called Taylor-based Optimized Recursive Extended Exponential Smoothed Neural Networks Forecasting method, abbreviated as TOREESNN. Briefly explained, the proposed technique introduces three ideas to solve this issue: First, building an innovative framework for forecasting univariate time series based on Exponential Smoothed theory. Second, designing an Elman Classifier model for uncertainty prediction in order to correct the forecasted values. And finally hybrading the two recurrent systems in one framework to obtain the final results. Experimental results demonstrate that the proposed method has a high accuracy both in training and testing data in terms of Mean Squared Error (MSE) and outperforms the state-of-the-art Recurrent Neural Networks models on Mackey-Glass, Nonlinear Auto-Regressive Moving Average time series (NARMA), Lorenz, and Henon map datasets.

Keywords Recurrent Neural Networks · Taylor · Forecasting · Time series · Error estimation · Exponential smoothed method

1 Introduction

Currently, data are increasingly growing. Hence, data computation becomes a major challenge. Companies dealing with Big Data need a sophisticated decision assistance in order to optimize the planning of these tasks and tools which encounter several problems to achieve their goals. First, the heterogeneity of the data induces the complexity of forecasting future values. Second, the diversity of the methods involved in the computation (i.e., in addition to the several inherent constraints) increases this complexity.

Forecasting is an important data analysis field that aims to examine historical data in order to extend and predict its future values. Thus, forecasting activities play an important role in our daily life and involve various fields; for this reason, many researches aim to develop tools for forecasting and decision making. We often forecast weather ([1]), wind speed [2], stock market [3], electricity [4, 5, 6], travel demand [7], traffic speed [8], etc.

✉ Emna Krichene
emna.krichen@enis.tn

[1] Research Groups in Intelligent Machines (REGIM Lab), University of Sfax, National Engineering School of Sfax (ENIS), 1173, 3038 Sfax, BP, Tunisia

[2] Centre de recherche en numérique de Sfax, B.P. 275, Sakiet Ezzit, 3021 Sfax, Tunisia

[3] College of Business, Al Ain University of Science and Technology, Abu Dhabi, United Arab Emirates

[4] Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, Washington 98071-2259, USA

[5] Department of Computer Science, College of Computers and Information Technology, Taif University, P.O.Box. 11099, Taif 21944, Saudi Arabia

[6] College of Applied Computer Science, King Saud University, Riyadh, Saudi Arabia

[7] Department of Electrical and Electronic Engineering Science, Faculty of Engineering and the Built Environment, University of Johannesburg, Johannesburg, South Africa

Published online: 18 July 2022

Meanwhile, forecasting literature contains a wide diversity of techniques that can be classified into two main families [5]:

1. Qualitative methods [9] where judgmental forecasts are principally based on experts' opinions. Furthermore, these techniques are especially used when sufficient information and data are not available. Hence, when the case study is vague and short of data, quantitative methods cannot be used. However, in case of the appearance of a new product or a new event, this type of forecasting techniques seems very adequate. The most known qualitative methods are: Consumer Survey [10], Consumer Survey-Sample Survey Method [10], Delphi Method [11, 12] and Past Analogies [13]. However, in spite of their robustness, these methods display some disadvantages. As they are totally based on the intuitions of experts, these techniques are not only expensive but also not adequate for most cases especially in big data case; i.e., human capability is not able to predict future values from a wide range of observations. For these reasons and others, another type of forecasting techniques appears. It is totally based on mathematical fundamentals which are later called quantitative methods.
2. Quantitative methods (also called statistical methods) [14] do not relate to experts' intuitions but they mostly rely on quantitative historical data that can be extrapolated to make our forecasts. This type of technique is used when the case study is stable and historical data exist. The most known statistical techniques are: Naive method [15], Moving Average (MA) [16], Weighted Moving Average (WMA) [16], Exponential Smoothing (ES) [17] and Linear Regression Method [18]. These techniques are commonly used when they handle products or phenomena that already exist and when the historical data are obviously available for study.

Compared to qualitative methods, these techniques are less expensive and faster. Yet, their effectiveness is limited to some applications as they are still inappropriate when observations are very different and chaotic i.e, non linear and unorganized. This point motivated researchers to go even further and to move toward Artificial Intelligence as a prediction tool.

Recently, different methods have been presented to the task of forecasting as intelligent techniques; Artificial Intelligence (AI) based systems are conceived to take advantage of the two traditional forecasting approaches: quantitative and qualitative. In fact, Neural Networks (NN) are the most-used techniques as they are nonparametric and nonlinear methods [19]. NNs are able to model and to map complex problems by training them in order to determine parameters and explore relationships between data [20]. Accordingly, NNs have been considered as a promising and useful method applied to a variety of forecasting problems.

From an architectural point of view, there are various types of neural networks that share the same basic concept. Each problem needs to be molded by admitting its own neural network architecture and network configuration since there is no single learning model that can be suitably used for all fields. However, in NN, we distinguish two main categories: (i) Multi-Layer perceptron (MLP) which is the most popular method applied in time series forecasting task, and (ii) Recurrent NN (RNN) [21] where its context-layer serves to store previous states within a dynamic memory. Authors in [20] believe that the importance of this supplementary layer lies in its ability to better underline the relationships between past observations and future values, and therefore an enhanced generalization is achieved.

A brief comparison of these techniques is summarized and presented in Table 1.

Although the forecasting methods proved to be successful and diverse, there are neither guarantees nor expectations that the predictive models would be

optimal or perfect. In fact, the residual error, which is the result of an unpredictable variability in the time series data, should be as small as possible. However, this error cannot be predicted in such optimal case. This fact is the core of this paper, the residual error should be monitored in a time series to enhance the exactitude of the initial pattern. The research's ultimate intent is the use of RNNs variant to forecast time series data. Therefore, this variant reaches an approximate disposition with an estimable residual error. Thus, a corrected current predictive model can be achieved.

Yet, to build the recurrent architecture, we drew inspiration from the Expo-nential Smoothed method (ES) [17]. Thereby, Recursive Extended Exponential Smoothed Neural Networks (REESNN) represents the first part of our proposed method conception. The primary objective of this work is an assessment of this method as a tool of forecasting. Furthermore, the second part of our approach consists in optimizing the results already obtained in the first step. The task of optimization is performed by predicting the estimated error to be then tailored with forecasting results by applying Taylor expansion principle. This paper is organized as follows: In Section 2, we start by a review in which we will discuss recent researches using Recurrent Neural network architectures. Then, Section 3 gives an extensive description of the propounded approach based on Taylor theorem to optimize forecasting future values. Then, we will illustrate the robustness of our approach by giving an exhaustive experimental result on four different datasets: Mackey-Glass times series, NARMA, Lorenz and Henon-Attractor in Section 4. Finally, Section 5 presents the conclusion of the paper.

Table 1 Forecasting Techniques Comparison

Family of the technique	Description (main concept and advantages)
Qualitative methods [9–12, 13]	- Depend on experience of individual experts. - Suitable for new phenomena, products, events and data shortage. - Not able to model complex data especially in case of big data.
Quantitative methods [14, 17, 15, 16, 18]	- Rely on mathematical formula - Suitable when historical data exist. - Not able to model complex data.
Intelligent methods [19]	- Rely on intelligent theories. - Suitable for complex and chaotic data. - Capability to map the relationship between data.

2 Related works

At the present time, significant feature related to RNN is the subject of numerous forecast researches. The literature is vast and growing and it is difficult for a researcher to cover all works contributing to forecasting literature done until now. In this section, we will present some works done over the last five years that have applied RNN as a forecasting technique. Hence, different architectures of RNN have been studied. However it seems that the most influential models are the Elman RNN (ERNN) [21] and Jordan RNN (JRNN) [22] as they are the most used models. This section is divided into three parts: In the first part, we will explain the fundamental principle of ERNN and present some works contributing with ERNN architecture. In the second part, we will briefly describe the architecture of Jordan RNN and introduce a brief survey of research activities contributing to forecasting literature using Jordan RNN over the last 5 years. Lastly, we will list different works that have been presented to the literature with other variants of RNN such as LSTM, BLSTM, GRU, etc..

2.1 Elman recurrent neural networks

The simplest and the most known Recurrent Neural Network is ERNN architecture [21]. ERNN is a three layered design expanded by a context-layer which receives inputs from the hidden units (see Fig. 1). The role of this supplementary layer is to store previous states of the hidden neurons within a dynamic memory.

ERNN is the most popular and widely-used technique on Artificial Intelligence applications. Nevertheless, many researchers use ERNN as a forecasting tool. Among them, authors in [23] proposed a new hybrid approach based on ERNN and Empirical Mode Decomposition EMD as a forecasting technique. Compared to different state-of-the-art methods such as single ERNN and ARIMA. Experiment results show that the proposed technique gives better performance.

In [24], two different architectures of NN were applied as predictor methods to forecast the flood water level earlier so that precaution steps can be taken. To do so, ERNN and NARX were compared to specify which technique gives more accurate results. Both architectures were fed with the same recorded water level. Based on Experiment results, ERNN accuracy is better than NARX's not only during the training step but also during the testing one. Yet, the work of [25] consists in predicting electricity consumption. The main purpose is to allocate suitable power resources and to help electric power companies to present reasonable sales plans. Elman recurrent neural network is applied in this work by using extra inputs which affect the electricity consumption such as gross domestic product, temperature, and Spring Festival. By comparing the results with and without considering effect factors, it can be shown that the proposed architecture with the extra inputs is more accurate than the simple ERNN.

Another application of Elman recurrent neural network contributing to the literature of forecasting was applied by [26]. The proposed technique aimed to forecast load. Authors used Elman architecture taking consideration of some related factors which enhance the performance of the model.

In the same context, researchers in [27] designed an architecture that merged two Elman NN structures to forecast wind power. The architecture was designed as follows: The first Elman NN part was realized to predict wind power component. Then, a second part was designed to forecast the guidance of wind direction. As a final step, the fusion of the two parts gave accurate results about wind power.

In [28], Elman RNN was also applied to forecast the Photovoltaic power. The gradient descent back propagation algorithm was used during the learning phase in order to set the parameters and build the optimized architecture. To prove the efficiency of the system, three tests have been carried out and evaluated through Mean Absolute Error and Root Mean Square Error.

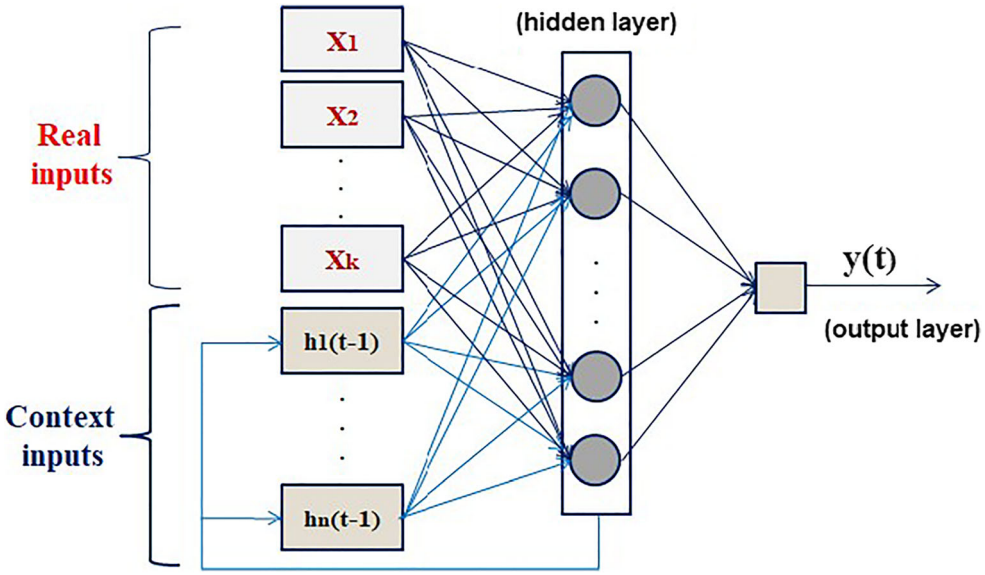


Fig. 1 The architecture of Elman RNN model

Elman RNN was also explored in reference [29] where researchers proposed a new approach based on a Modified version of ERNN Optimized through the Genetic algorithm to predict short-term traffic. 14 forecasting techniques were compared to the proposed approach within the experimental section, and empirical results showed the outperformance of the presented work.

However, based on the Jordan recurrent neural network architecture, many other papers contribute to the literature. In the following sections, we will shed light on the Jordan RNN model and its variants.

2.2 Jordan recurrent neural networks

The major difference between Jordan RNN's architecture and those of other RNNs is that the context-layer is designed as a copy of the output-layer as shown in Fig. 2.

The Jordan RNN [22] models are used in diverse prediction tasks especially in time series forecasting problems since they display inherent competence to map input-output problems.

In [30], the proposed technique aimed to forecast Stock Market Price. To enhance the performance of the model, an unsupervised method was used in the first step to reduce inputs dimension. As a second step, JRNN was used to predict the closing price of a given day based on the information of the day before. Empirical results illustrated the efficiency of the proposed technique; Evaluation was made on the basis of two metrics: MSE and mean absolute percentage error.

In [31], researchers proposed to combine two methods in one architecture to predict the plant output. Their system is composed of Balanced Truncation (BT) cascaded with Jordan network. A comparative study was then performed to different

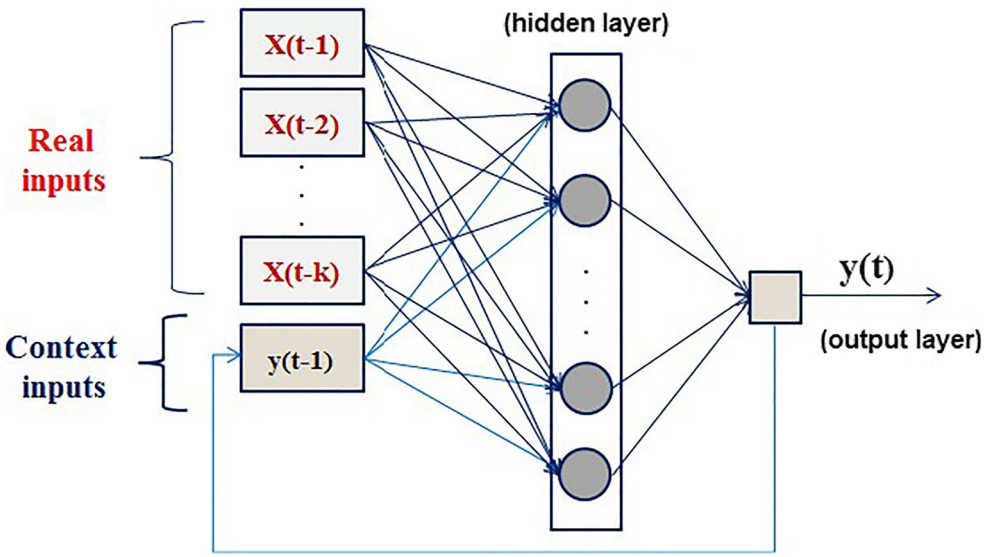


Fig. 2 The architecture of Jordan RNN model

Model Order Reduction (MOR) and showed that the proposed system was remarkably more accurate.

Besides the above mentioned works, reference [32] extended the application of JRNN to inflation forecasting. Authors first defined which variables influenced most inflation forecast, then built the optimal JRNN model after testing several parameters. Hence, authors manipulated over 250 JRNNs architectures, which varied in selected input and model's parameters. The optimal JRNN structure was the one that led to a low error during both the learning and the testing phases, and also used a low number of parameters to approximate them.

More recently, researchers in [33] have implemented the JRNN structure combined with Generalized Space Time Autoregressive with Exogenous Variable (later called GSTARX) to forecast Space-Time Data influenced by Calendar Variation Effect. Experimental results showed that hybrid models are more accurate than individual techniques.

2.3 Other variants of recurrent neural networks

Although our focus is on Elman and Jordan RNN as they represent an important part of our methodology, it should be mentioned that Long-Short Term Memory Network abbreviated as LSTM, as well as its derivatives also lead to good results in prediction tasks.

However, predicting Stock market is considered as one of the most challenging issues to resolve. In this context, authors in [34] present to the literature an evaluation of Bi-LSTM for stock market forecasting. They compare different variants of LSTM to each others: Unidirectional, bidirectional, shallow neural network and stacked LSTM. Experiment results showed that BLSTM and SLSTM perform better than other compared techniques.

Another work was presented to the literature using LSTM [35], where authors applied Deep LSTM to forecast petroleum production. The suggested architecture is optimally configured through the use of the genetic algorithm. Yet, [36] suggest using an LSTM methodology in order to forecast real time data. The proposed architecture is an LSTM-based regression structure extended by an additional gate to predict real time series data.

More recently in 2021, authors in [37] proposed a novel deep learning framework based on the hybridization of two techniques: autoencoders (AE) and LSTM network. The presented method is designed to forecast non linear system and time series with high accuracy. To validate their work, four real-world datasets were applied and tested: Santa Fe dataset, Australian energy market operator dataset, Oilfield production dataset and Gas furnace dataset. Compared to other forecasting methods, the suggested technique performs better in terms of Root Mean Squared Error (RMSE) value.

LSTM is also explored in the work of Chiu et al. [38] where authors presented a novel CNN-LSTM architecture to forecast estate price. During the process of analyzing data, the CNN-LSTM work has taken into consideration the spatiotemporal data structure and extracted the most important features influencing the estate price. Real estate data from Taiwan were used to verify the effectiveness of the proposed model.

Lately, another variant of RNN named Gated Recurrent Units (GRU) has increased its popularity on time series applications. Researchers in [39] empower the usefulness of these different variants of RNN by applying them on Turkish electricity load prediction. The comparison between obtained results and those obtained by existing researches based on Auto Regressive integrated moving average (ARIMA) and other Artificial Neural Networks (ANN) architectures prove the efficiency of RNN as well as its derivatives.

Another work added to the literature [40] is applied to Agriculture as it is considered as one of the highest energy consuming sector. RNN, LSTM and GRU are used to predict hourly short term Agriculture load. Empirical results show that these techniques are more feasible than Moving Average model, Auto Regressive model and ARIMA.

To conclude this section, a brief recapitulation of these contributing papers is summarized and presented in Table 2.

Nevertheless, thanks to their specific design, Jordan RNN is supposed to be a perfect alternative model to forecast time series despite the hard predictability of their elements. Its structure is supposed to have a more powerful competence than other typical RNN architectures to train and predict the given data [42]. In this paper, to address the aforementioned problem, we propose a newly Recurrent NN architecture inspired from Taylor theory as a forecasting technique. Specifically, we have aligned our contribution by focusing on two points: (1) Function approximation using the Recursive Extended Exponential Smoothed method and (2) error approximation in order to optimize the forecasted values using the Taylor expansion [43].

3 Proposed approach

The proposed TOREESNN architecture for time series prediction is illustrated in Fig. 3. The system is built with a recurrent architecture, a classifier learning model, and a Taylor-based mathematical system. The working methodology of TOREESNN is divided into three main steps summarized as follows:

- **Step 1: Function approximation and forecasting values:** Building a system inspired from Jordan RNN & ES technique, called Recursive Extended Exponential Smoothed Neural Networks (REESNN). The introduced method REESNN is designed in order to investigate data behavior and forecast its future values.

Table 2 Summary of Different Contributing Papers Over Last Five Years

	Reference	Data	Brief Description
ELMAN RNN	[23]	Wind Speed	Hybrid model based on ERNN with EMD.
	[24]	Flood Water Level	Comparison between the performance of ERNN and NARX.
	[25]	Monthly Electricity Data	ERNN is applied in this work using extra inputs that affect electricity consuming.
	[26]	Guangdong Electric Power Grids	Authors employ Elman architecture taking consideration of some related factors which enhance the performance of the model.
	[27]	Wind Power	The fusion of two architectures of Elman NN is applied to get accurate wind speed prediction.
	[28]	Photovoltaic Power	Elman architecture is used to forecast the future power of the photovoltaic station. The system is fed by four inputs: Temperature, Humidity, Wind-speed and Radiation which are all normalized. Three tests were made to prove the efficiency of the system.
	[29]	Short-Term Trafficc	Modified Elman RNN architectureis proposed and optimized through the genetic algorithm to predict accurate short-term Traffic.
JORDAN RNN	[30]	Stock Market Price	The Unsupervised method is used in the first step to reduce the dimension of inputs, then based on the information of the previous day, JRNN is applied to predict the closing price of each day.
	[31]	Plant Output	BT method and Jordan network are combined in one architecture to predict plant production. The system's response is then compared to the original signal as well as the output of different other techniques such as the Instrument Variable estimation approach, JRNN and Yang et al. [41]. The proposed method illustrates better results over competing methods named above.
	[32]	Macroeconomic Time Series	250 JRNN structures are implemented and compared. The optimal architecture is the one that gave a low error and used a low number of parameters.
	[33]	M4 forecasting Competition	A Hybrid GSTARX-JRNN is presented to predict Time Data affected by Calendar Variation. Empirical results show that the combined model performs better than individual techniques.
Other variants of RNN	[35]	Petroleum Production	A deep LSTM optimally configured by the use of genetic algorithm is adopted to predict the petroleum production.
	[36]	Real Time Data	Forecasting real time-series data using a LSTM-based regression structures extended by an extra gate.
	[34]	Stock Market	A literature comparison and evaluation of different variants of LSTM: Unidirectional, bidirectional, shallow neural network and stacked LSTM. Experiments prove that BLSTM and SLSTM perform better than other variants of LSTM.
	[39]	Turkish Electricity Load	A Comparison between the efficiency of GRU and others forecasting methods in the case of Turkish electricity load prediction.
	[40]	Hourly Short Term Agriculture Load	Comparison between RNN, LSTM and GRU to other forecasting techniques such as MA, AR and ARIMA. Experiments prove the effectiveness of the different variants of RNN.
	[37]	Four Realworld Datasets	Proposing a Deep Learning Framework that takes advantages from both AE and LSTM in order to predict non linear system and time series datasets. Experimental results show that the presented method is more accurate than other state-of-the art forecasting techniques.
	[38]	Estate Data from Taiwan	Authors present a novel frameworkthat leverages the advantages of both CNN and LSTM architectures. This technique considers the spatiotemporal data structure and extracts the most important features influencing the estate price before analyzing data.

- **Step 2: Forecasting error estimation:** A Elman RNN Classifier is trained to approximate the uncertainty value associated with each forecasted value.
- **Step 3: Taylor-based forecasting results optimization:** The two previous results are fed to a mathematical system based on Taylor expansion, in order to output the final optimized forecasting values.
- These aforementioned steps are more detailed in next sections. The forecasting flowchart of the proposed methodology is shown in Fig. 3.

3.1 Function approximation and forecasting values based on REESNN

A Jordan RNN (JRNN) is a neural network that uses the received previous network output as a new input to be later processed [44]. Hence, a JRNN is a simple recurrent network so that a specific group of neurons (later called context neurons) receives feedback signals from the previous time step. Thereby, input layer is composed of two parts: true input neurons and context neurons which are a duplicate of

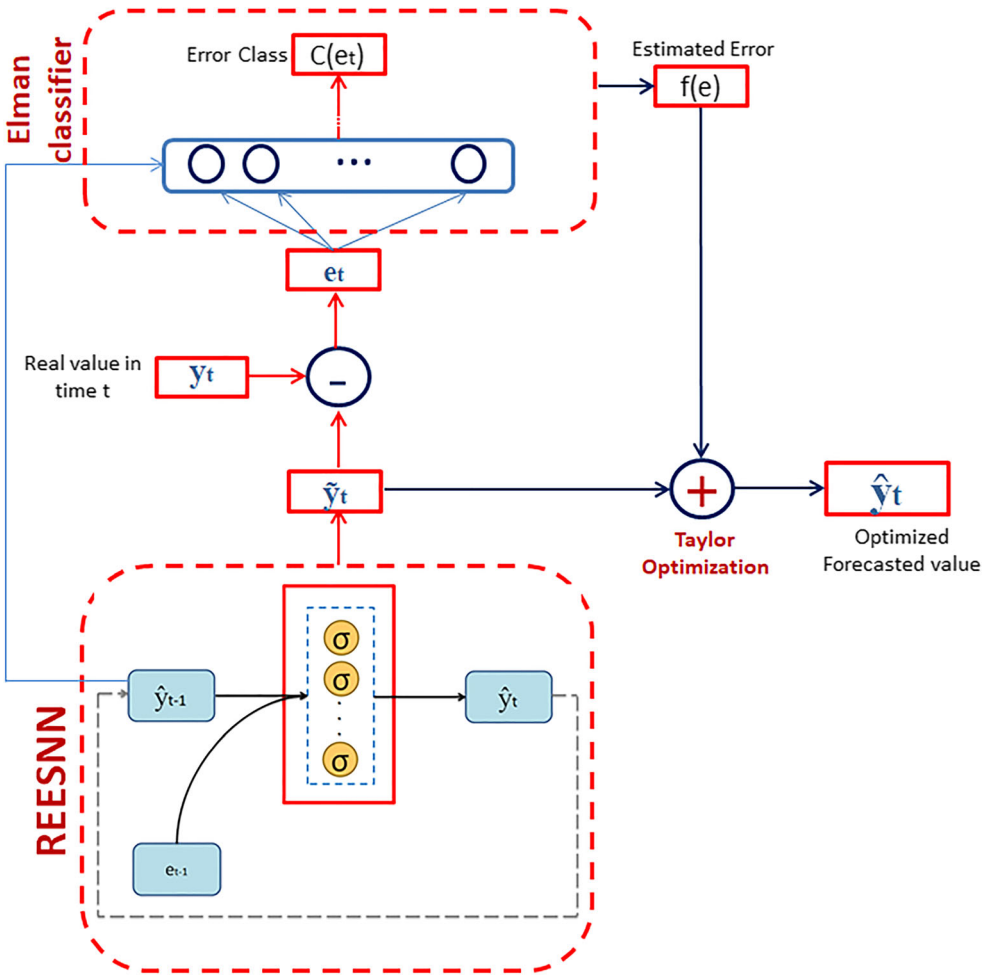


Fig. 3 The flowchart of a general TOREESNN architecture

the outputs provided from the previous time step. Hence, the structure of the JRNN model can be illustrated as in Fig. 2.

Basically, each unit in a particular layer is connected with all neurons in the next layer and then summed and multiplied by the appropriate importance degree called w_{ij} which characterizes the connection and the degree of importance between the i^{th} and the j^{th} neuron. Hence, the nonlinear mapping function F masters the amplitude of the resulting output $\widehat{y}$ which is given by Eq. 1.

$$\widehat{y}_i = F\left(\sum_{j=1}^{i} w_{ij} h_j\right) \tag{1}$$

where F is called the activation function. In general, the transfer function accurately reflects the nonlinearity degree of most preprocessed data by NN. We note that in theory, the activation function can be any differentiable function but in practice only four transfer functions are defined and can be computed according to Eqs 2–5.

– The linear function:

$$F(x) = x \tag{2}$$

– The sigmoid bipolar function (hyperbolic)

$$F(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3}$$

– The cos-sin function:

$$F(x) = sin(x) \text{ or } F(x) = cos(x) \tag{4}$$

– The sigmoid function:

$$F(x) = \frac{1}{1 + e^{-cx}} \tag{5}$$

where c is the adaptive gain parameter of sigmoid function.

Meanwhile, in JRNN structure, the activation of the j^{th} hidden neuron h_j is computed by summing the activation of context units in addition to those of true inputs. It holds that:

$$h_j(t) = F\left(\sum_{i=1}^{k} w_{ij}\, x_i(t) + \sum_{m=1}^{n} v_{mj}\, \hat{y}_i(t-1)\right) \tag{6}$$

where w_{ij} is the connection strength between the j^{th} hidden unit h_j and the i^{th} real input x_i , and v_{mj} is the connection's weight between the j^{th} hidden unit h_j and the m^{th} output past value at time *t-1*.

Based on the modeled architecture in Fig. 2, the JRNN structure can be expressed by Eq. 7.

$$\hat{y} = \sum_{j=1}^{H} u_j h_j(t) \tag{7}$$

where $\hat{y}$ is the predicted output and u_j are the connections weights on output layer.

Subsequently, all connection weights in the network are merely calculated and updated by applying the back-propagation learning algorithm, as expressed in Eq. 8.

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}} \tag{8}$$

where α is the learning rate and E is the associated error with the i^{th} neuron in layer j.

Hence, our goal is to properly adjust the network's connection weights and update them according to (9) in order to understate the value of error.

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij} \tag{9}$$

In the same way, all connection strengths in JRNN architecture are updated. In our study, to approximate the value of a function, we have adopted the recursive architecture inspired from the Exponential Smoothed (ES) method [17] which is a statistical forecasting technique. Its main principle assumes that each observation at time t relies on the previous observation and the variation between that forecast and the actual value of the series at that point. The principle of the ES can be formulated by the equation:

$$\hat{y}(t+1) = \hat{y}(t) + \alpha\left(x(t) - \hat{y}(t)\right) \tag{10}$$

where $\hat{y}(t)$ is the previous forecasted value , x is the actual one and $\alpha \in [0,\ 1]$ is the smoothing factor.

Our experiments aim to convert the ES method into an Extended ES designed as a Jordan Recurrent neural architecture. This task is accomplished through two steps: First, the ES method is modified and extended so that each observation does not only depend on solely one past value as ES assumes, but on many past values. Hence, we consider Extended ES (EES) as a generalization case of ES. Mathematically, the equation of EES can be formulated as:

$$\hat{y}(\mathrm{t}+1) = \sum_{k=0}^{n}\left(\hat{y}(\mathrm{t-k}) + \alpha\left(x(t-k) - \hat{y}(\mathrm{t-k})\right)\right) \tag{11}$$

The second step consists in conceiving the EES as a recurrent architecture since it adopts properly the same principle; i.e the prediction of the future forecasted value depends on the previous estimated output as the JRNN supposes. Moreover, the smoothing factor is supposed to be replaced by the weight connection. Hence, the simplest method for this purpose is to project the EES dynamic equation into a mapping function that can be depicted as:

$$\hat{y}(t+1) = F\left[\hat{y}(t), \ldots, \hat{y}(t-k); x(t) - \hat{y}(t), \ldots, x(t-k) - \hat{y}(t-k)\right] \tag{12}$$

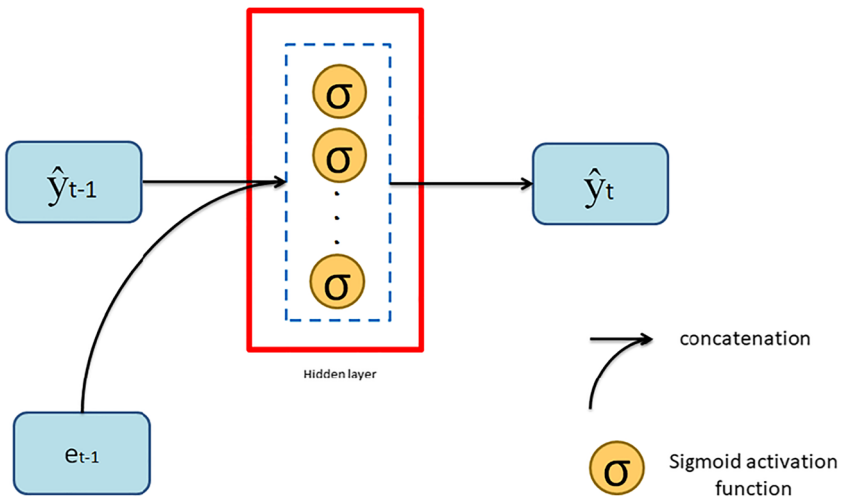


Fig. 4 Architecture of long-short exponential smoothed unit (LSES)

Graphically, the Recursive Extended Exponential Smoothed Neural Networks (REESNN) is considered as a set of Long-Short Exponential Smoothed units (LSES) where the input layer is a set of two types of inputs: The previous forecasted values $\hat{y}$ at time $t-1$, $t-2$,.., $t-k$ and, the estimated error e outlined as the variance between the actual real value x and the forecasted one $\hat{y}$ at time $t-1, t-2,.., t-k$ as illustrated in Fig. 4.

After being processed by sigmoid activation functions, the information passes through many such LSES units as Fig. 5 depicts.

Particularly, our proposed pseudo-code method REESSN is detailed in algorithm 1.

Algorithm 1 REESNN algorithm

Input: $\hat{y}(t\_k)$: previous forecasted value, $e(t\_k)$: the estimated error between actual value and predicted one; $k = 0, 1, .., n$

Output: $\hat{y}(t+1)$: predicted results

1: $\hat{y}(t+1) = \sum_{k=0}^{n}(\hat{y}(t-k) + \alpha(x(t-k) - \hat{y}(t-k)))$ //Modifying and extending ES method to EES depending on data behaviour

2: $\hat{y}(t+1) = F[\hat{y}(t),\ldots,\hat{y}(t-k)\,; x(t)\text{-}\hat{y}(t),\ldots,x(t-k)\text{-}\hat{y}(t-k)]$ //Conceiving the obtained EES as a JRNN

3: dataset={F[$\hat{y}(t),\ldots,\hat{y}(t\text{-}k)$; x(t)-$\hat{y}(t),\ldots$, x(t-k)-$\hat{y}(t\text{-}k)$], $\hat{y}(t+1)$}

4: trainset,validationset,testset= train_Test_Split(dataset)

5: **do**

6: REESNN = train(trainset)

7: predValue= REESNN.predict(validationset)

8: MSE= Mean_Squared_Error(predValue, targetValue)

9: **if** (MSE ≤ Stopping_Criterion) **then**

10: $\hat{y}(t+1)$=REESNN.predict(testset)

11: return $\hat{y}(t+1)$

12: **end if**

13: **while** (MSE == Stopping_Criterion OR NBR_ITER == MAX_ITER)

3.2 Forecasting error estimation

A powerful part of this approach is the prediction of error estimation in order to improve the forecasting precision. Forecasting error estimation is defined as the prediction of the difference between the actual value x and the predicted one $\hat{y}$.

However, the uncertainty associated with forecasting is one of the most powerful factors influencing the resulting values. This point inspires us to examine in advance the task of forecasting by predicting not only the future values but also the uncertainty associated with each one. In our study, Elman RNN Classifier (ERNNC) [21] model is used to model and approximate the error estimation associated to each forecasted value. This choice is justified by two points: First, when examining the error distribution, we notice that the behavior of the resulting errors is chaotic and oscillating around close values which makes the task of prediction hard to model. This point motivates us to solve the problem as a classification task, i.e a set of error classes is defined and associated to each residual value to be then trained. Second, compared to other existing functions in classification forecasting problems, Elman has one of the best learning rates.

The suggested model is a three layered NN in addition to a hidden context layer and a classification function as shown in Fig. 6.

Firstly, to train the Elman RNN as a Classifier method, it is necessary to preprocess the data and transform it on the adequate form. Hence, the transformation of the training Dataset S_c for the Elman RNN Classifier is introduced as follows:

$$S_c = \{(Y_t, C(e))\} \in [0,1] \times \{-1, 0, 1\} \tag{13}$$

where Y_t is an input vector and $C(e)$ is a classification function of the estimated error e defined by Eqs (14) and (15).

$$Y_t = \left[\hat{y}(t-\tau), \hat{y}(t-2\tau), \ldots, \hat{y}(t-d\tau)\right] \tag{14}$$

$$C(e) = \begin{cases} -1 & e < -\varepsilon \\ 0 & -\varepsilon \le e \le \varepsilon \\ +1 & e \ge \varepsilon \end{cases} \tag{15}$$

where τ is the time-delay and ϵ is an appropriate threshold that divides estimated errors into three scales: positive values, negative values and values near to zero. We note that ϵ is chosen, graphically, from the estimated error histogram as Fig. 7 depicts.

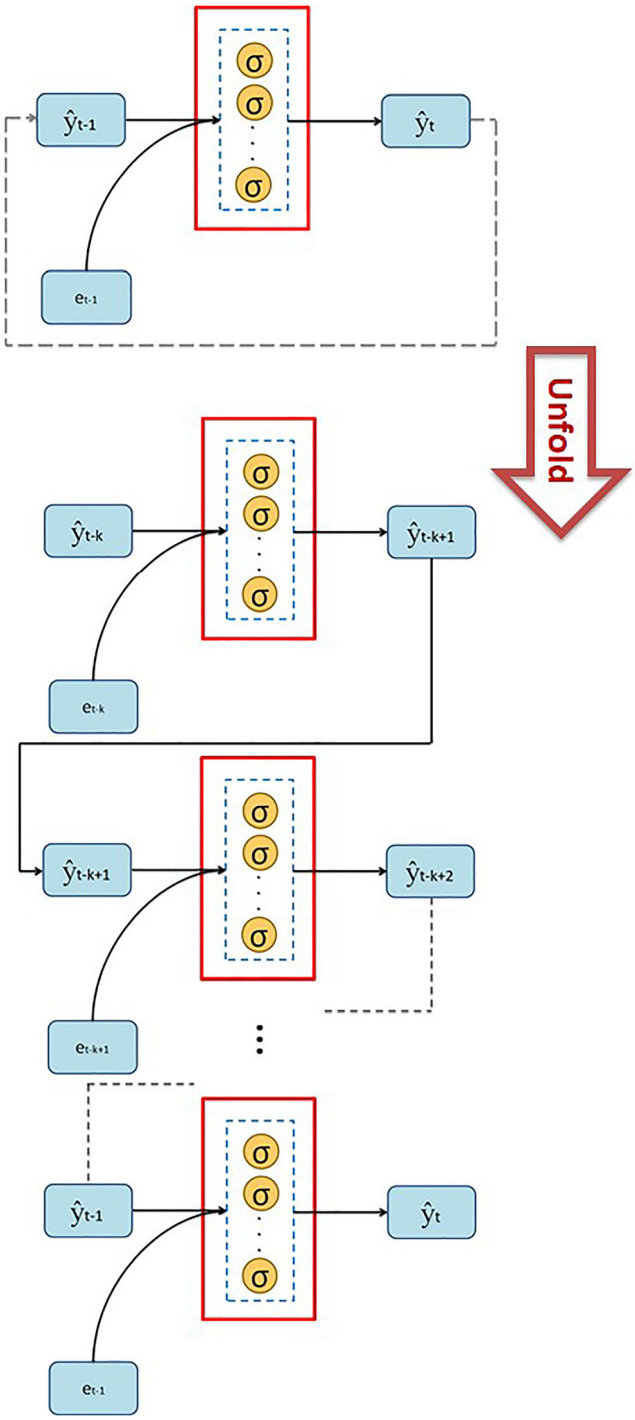


Fig. 5 Architecture of Recursive Extended Exponential Smoothed Neural Networks (REESNN)

After the histogram of the estimated error has been established, the Elman RNN Classifier is trained using S_c to predict the error class associated to each sample in the vector X_t. Then, to estimate the forecasting error, IF-THEN rules are adopted.

$$\begin{cases} IF\ C(e) = 1\ THEN\ f(e) = +\omega \\ IF\ C(e) = 0\ THEN\ f(e) = 0 \\ IF\ C(e) = -1\ THEN\ f(e) = -\omega \end{cases} \tag{16}$$

where $f(e)$ is the estimated forecasting error and ω is an experimental value measured during the learning phase where different values near to ϵ were tested; the one which gives minimal error is adopted for the rest of the algorithm. During the optimization phase, different values of ω were tested; the value giving the best result was memorized to be then applied during the final step.

3.3 Taylor-based forecasting results optimization

After predicting the estimated error, a step of optimization results was done.

The optimized forecasted future value $\widehat{y}_{t+1}$ is calculated according to an approximation function expressed between the first forecasted value $\widehat{y}_t$ resulting from the first part of our architecture denoted as REESNN and the estimated forecasted error $f(e)$ resulting from the second part denoted as Elman Classifier (see Fig. 3). In fact, a Taylor series approximation represents a number as a polynomial that has a very similar value to the number in a neighborhood around a specified value. Hence, the key idea in our method is to optimize the forecasted results relying on the Taylor expansion [43].

Motivation: The Taylor expansion The main principle of the Taylor expansion [43] aims to approximate a function that is many times differentiable in the neighborhood of a point x. Its equation is expressed as:

$$f(x) = f(x_0) + \frac{x-x_0}{1!} f'(x_0) + \frac{(x-x_0)^2}{2!} f^2(x_0) + .. + \frac{(x-x_0)^n}{n!} f^n(x_0) + (x-x_0)^n \delta(x-x_0) \tag{17}$$

where $\delta(x - x_0)$ is a function that gets closer and closer to 0 as $(x - x_0)$ tends to 0.

From the above definition, this formula is similar to the optimization formulation, i.e, the key idea behind optimization is to approximate a number by representing it by its nearest accurate neighborhood. Hence, we formalize a new problem that translates the Taylor expansion formula into a combination of optimization task.

In fact, to resolve y_{t+1} and y_t by Eq. 17 a step of integration of the error estimation function $f(e)$ is needed before time $t + 1$. This integration expression shows that the proposed

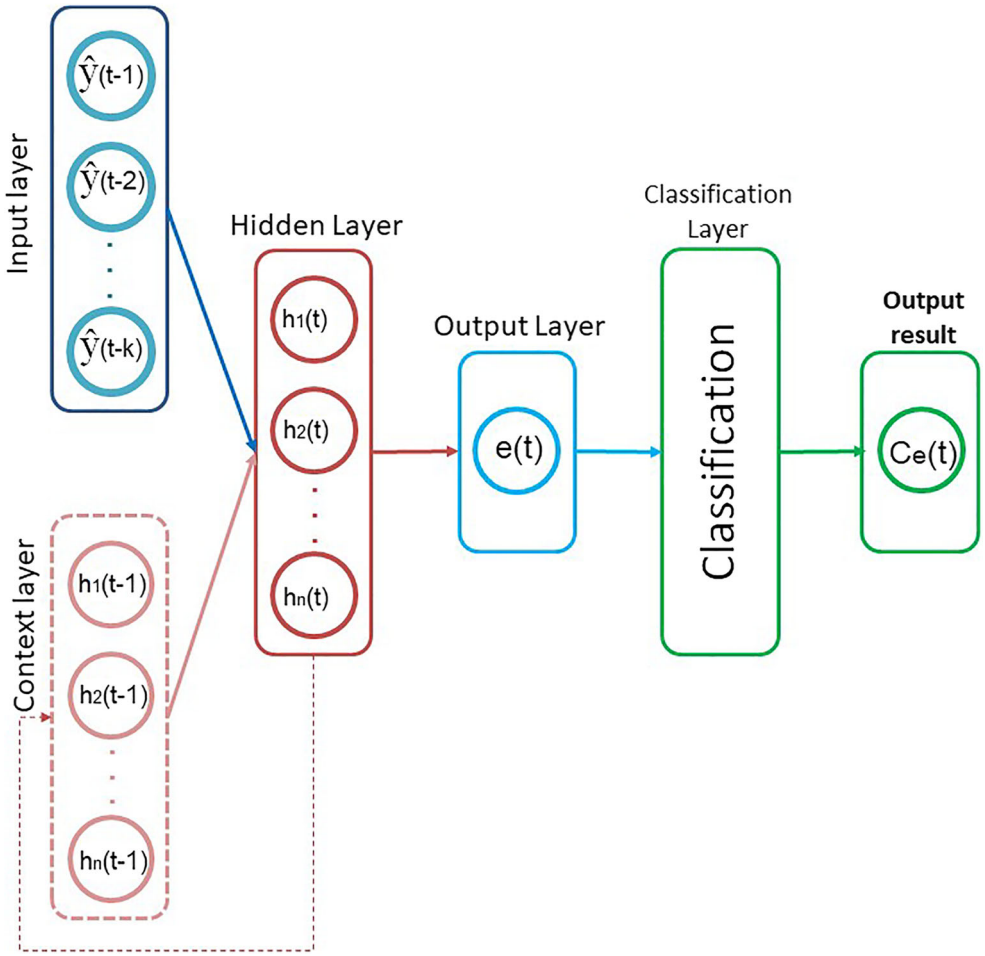


Fig. 6 Elman Recurrent Neural Network Classifier architecture

optimization model can perform the dynamic backward optimization to better exploit the historical knowledge in the studied time series.

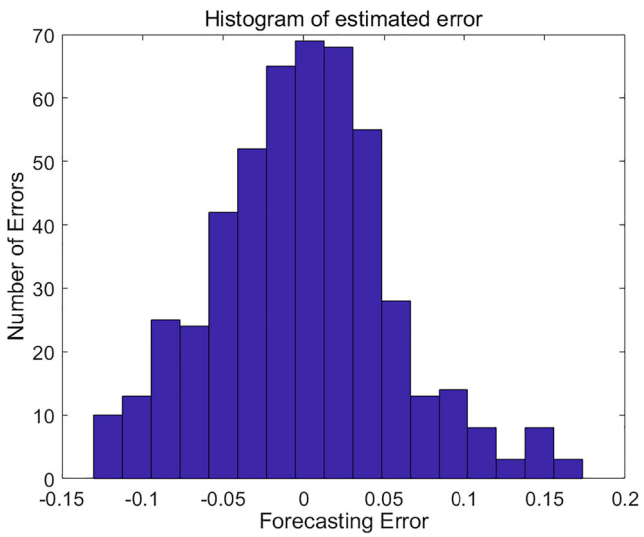


Fig. 7 Histogram of estimated error

Inspired from Taylor principle, and provided the ϵ is small enough, the previous equation denotes that this combinational optimization problem can be approximately formulated as:

$$\widehat{y}_{t+1} = \widehat{y}_t + \frac{(t+1)-t}{1!} y'(t) + 0(t) \tag{18}$$

where $\widehat{y}_{t+1}$ denotes the optimized forecasted value and $\widehat{y}_t$ represents the predicted value before optimization. It is worth mentioning that our study will be restricted to the first stage of Taylor principle. We hypothesize that y'_t is equal to the error approximation $f(e)$ as given in the following equation:

$$y'_t = f(e) = \frac{\widehat{y}_{t+1} - \widehat{y}_t}{(t+1)-t} \tag{19}$$

The step-by-step instructions that describe the TOREESNN algorithm is presented as follows (see algorithm 2).

Algorithm 2 TOREESNN algorithm

Input: $\hat{y}(t\_k)$: previous forecasted value, $e(t\_k)$: the estimated error between actual value and predicted one; $k = 0, 1, .., n$

Output: $\hat{y}'(t+1)$: optimized predicted results

```
Divide data into k-folds
for each k_i ∈ k-folds do
    do
        REESNN = train(trainset_i)
        predValue_i = REESNN.predict(validationset_i)
        MSE= Mean_Squared_Error(predValue_i, y(t+1))
        if (MSE ≤ Stopping_Criterion) then
            ŷ(t+1)=REESNN.predict(testset_i)
            return ŷ(t+1)
        end if
        for t=0 to n do
            e(t+1)=ŷ(t+1)- targetValue(t+1)
        end for
        Histogram(e)
        Select +ϵ and -ϵ // Choosing the values that divide estimated errors into three scales: positive values, negative values and values near to zero.
        S_c = {(Y_t, C(e))} ∈ [0, 1] × {−1, 0, 1}
        ElmanClassifier=train(S_c)
        PredError= ElmanClassifier.predict(testset_i)
        Rp=[+ϵ max(e)]
        Rn=[min(e)  -ϵ]
        while TRUE do
            +ω= random(Rp)
            _ω= random(Rn)
            for t=0 to n do
                if C(e(t+1))==1 then f(e)=+ω
                else if C(e(t+1))==0 then f(e)=0
                else if C(e(t+1))==-1 then f(e)=- ω
                end if
            end for
            ŷ'(t+1)=ŷ(t)+f(e)
            MSE= Mean_Squared_Error(ŷ(t+1),targetValue)
            if (MSE ≤ Stopping_Criterion) then
                storeValues(+ω,- ω,+ϵ,- ϵ, RESSNNmodel, ElmanClassifierModel)
                return ŷ'(t+1)
            end if
        end while
        Evaluate_Model_Performance on fold k_i
    while (MSE == Stopping_Criterion OR NBR_ITER == MAX_ITER)
end for
Calculate average performance over K folds
```

4 Results and discussion

4.1 Basic setting

All experiments in this study were carried out in Matlab using an HP computer running Windows 10 pro64-bit operating system, an Intel Core i7–5500 CPU at 2.40GHz, 8 GB of memory, and graphics card of Intel R HD 5500 with 4 GB RAM. Furthermore, our model's performance is estimated after a 5-fold cross validation. The values of all metrics reported in the study are the average of the results obtained at each fold. The values

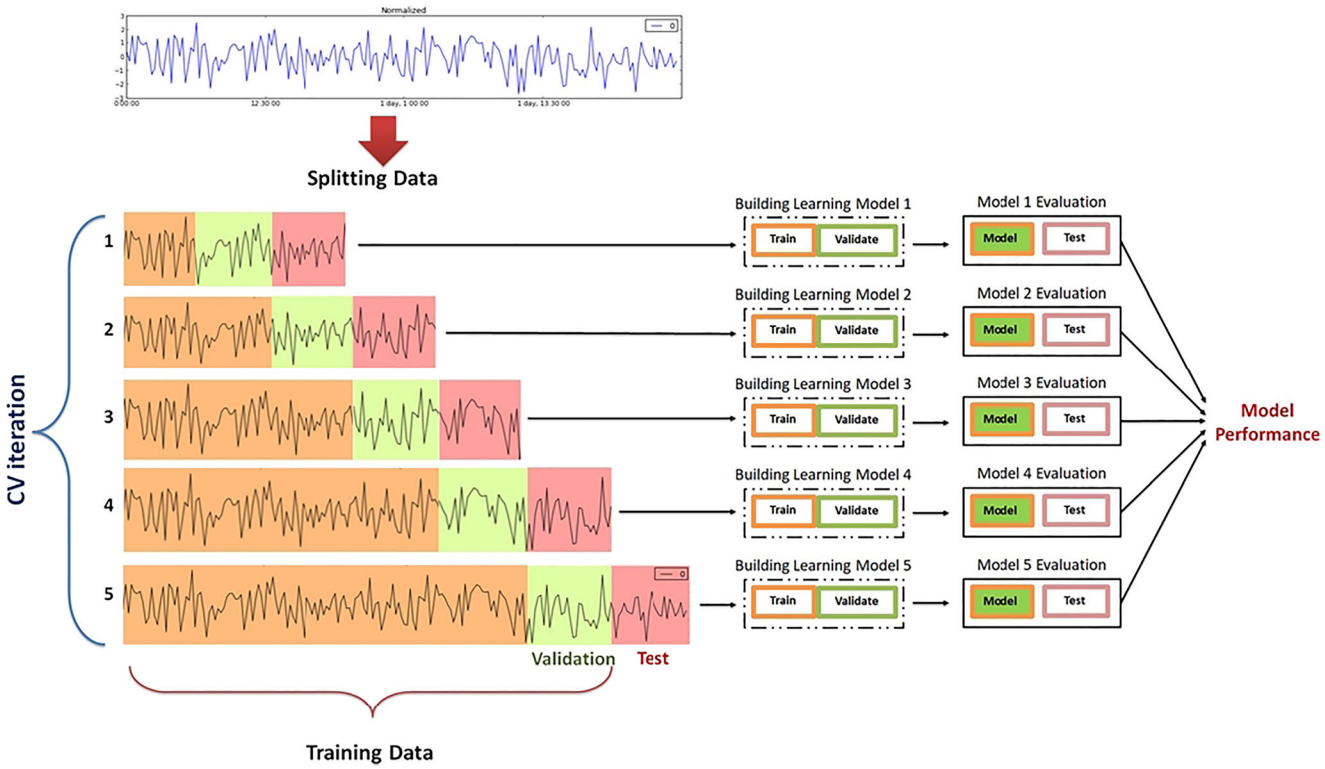


Fig. 8 Time Series-based Cross Validation

of all metrics reported in the study are the average of the results obtained at each fold.

To validate the efficiency of our proposed method, we selected four types of acclaimed Time Series: Mackey-Glass, the Nonlinear Auto-Regressive Moving Average, Lorenz Attractor and Henon Attractor. Each type of data is segmented into three data sets. The first one represents the training data. The second part represents the validation data. It is applied to train the estimated error (training the estimated error is the second part of our architecture). Finally, 1500 new samples are used as testing data in order to improve the performance of our proposed scheme with new cases.

In our quest to prevent over-fitting and to build a robust model, we utilized K-Fold-Cross-Validation method. We chose to partition our data into five folds in order to train and validate it thoroughly. This would generate the optimal learning model. Hence, the final metric is then the average score obtained from all tested folds. We note that the main principle of Cross-Validation is to randomly divide the data into training and testing data. However, in the case of Time Series Data, the process of dividing should be on a rolling basis, i.e, for each fold, respectively, we selected the training data, then the testing part. The learning model is built, forecasting values are generated and evaluation metric is calculated at the end.

Figure 8 illustrates the process of Time Series-based K-fold Cross validation. The orange color represents the training data, the green color represents the validation data where the process of learning estimated error is made and finally pink color symbolizes the testing data.

Finally, to facilitate the learning process, all the data used in our experiments were subject to normalization process; it is then displayed in $[0, 1]$.

4.2 Analysis of experimental results

In order to evaluate the performance of the proposed approach, four well-known benchmarks were studied in this

Table 3 Comparison of Different Forecasting RNN Model to Our Proposed Method (MG Benchmark $\tau = 17$)

Methods	MSE test
Canonical Echo State Network (ESN) [46]	$4.03e$ - 02
LSTM [47]	$8.04e$ - 03
Elman [20]	$3.00e-$ 03
Elman & Elman Classifier	$2.30e-$ 03
ESN-LARS [46]	$7.80e$ - 04
ESN-FSR [46]	*inf*[*]
REESNN	**2.29e-04**
TOREESNN	**2.20e-04**

[*] According to authors, the results are unstable and the error is very high.

In order to highlight our results (given by both REESNN and TOREESNN methods), we noted the corresponding values in bold.

Table 4 Comparison of Different Forecasting RNN Model to Our Proposed Method (MG Benchmark $\tau = 30$)

Methods	MSE test
Canonical ESN [46]	$8.12e-03$
ESN-RR [46]	$8.25e-03$
ESN-LARS [46]	$3.20e-02$
ESN-FSR [46]	$7.72e-03$
ESN-LASSO [46]	$4.31e-03$
ESN-EN [46]	$4.31e-03$
REESNN	**3.74e-04**
TOREESNN	**3.48e-04**

In order to highlight our results (given by both REESNN and TOREESNN methods), we noted the corresponding values in bold.

paper. For each benchmark, we measured the accuracy of our model by first calculating an evaluation metric, then, comparing it to other state-of-the-art works. In our study, Mean Squared Error (MSE) is considered as an evaluation score. The formula of MSE is given in 20.

$$MSE = \frac{\sum_{t=1}^{N}\left(y_t - \widehat{y}_t\right)}{N} \tag{20}$$

where N , y_t, $\widehat{y}_t$ indicate respectively the length of data, the target value and the output system.

4.2.1 Comparison of prediction results

Mackey-glass time series The performance of the proposed methodology is demonstrated via the well-known benchmark time series called the Mackey-Glass time series (MG) [45]. Mathematically, the MG formula is established by a simple dynamic equation that can be noted as:

$$\frac{dx(t)}{dt} = \frac{ax(t-t)}{1 + x^c(t-\tau)}bx(t) \tag{21}$$

where a, b and c are constants, and τ represents the time delay.

In this paper, the used parameters a, b and c are set to 0.2, 0.1 and 10 respectively, whereas the most used τ in the literature are 17 and 30. The aim of this paper is to forecast the actual value $x(t)$ based on some specific historical past values. We note that a step of normalization was done to optimize the training phase.

In order to prove the efficiency of our proposed architecture, different forecasting RNN models were compared and testing results are outlined in Tables 3 and 4.

According to Tables 3 and 4, it is revealed that the proposed (TOREESNN) performs better than other RNN methods. Moreover, it can be noticed that optimizing forecasting task by predicting estimated errors influences greatly the testing results.

As it is the case with supervised learning methods, we aim to make the system output signal closer to the target one. We plotted these two signals after performing the proposed approach. Figures 9 and 10 illustrate the superposition between

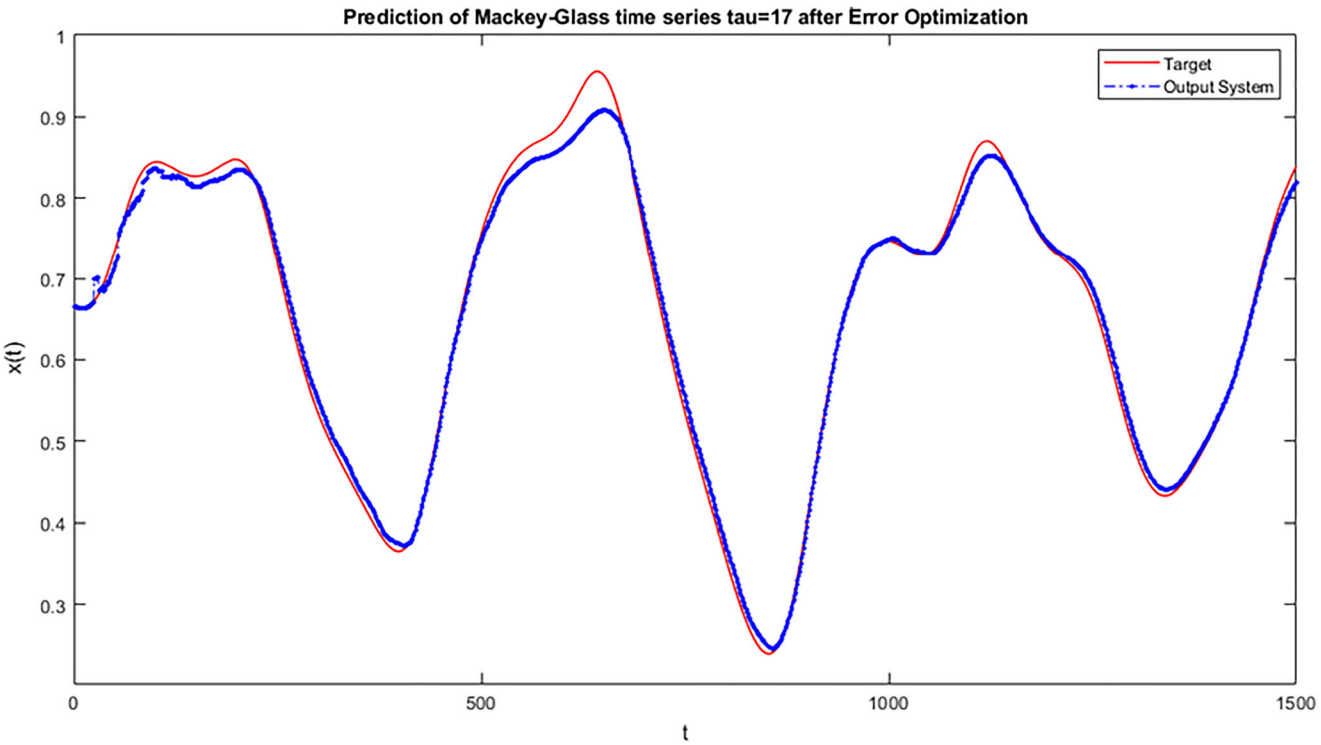


Fig. 9 Mackey-Glass Network output after the error estimation optimization *tau* = 17

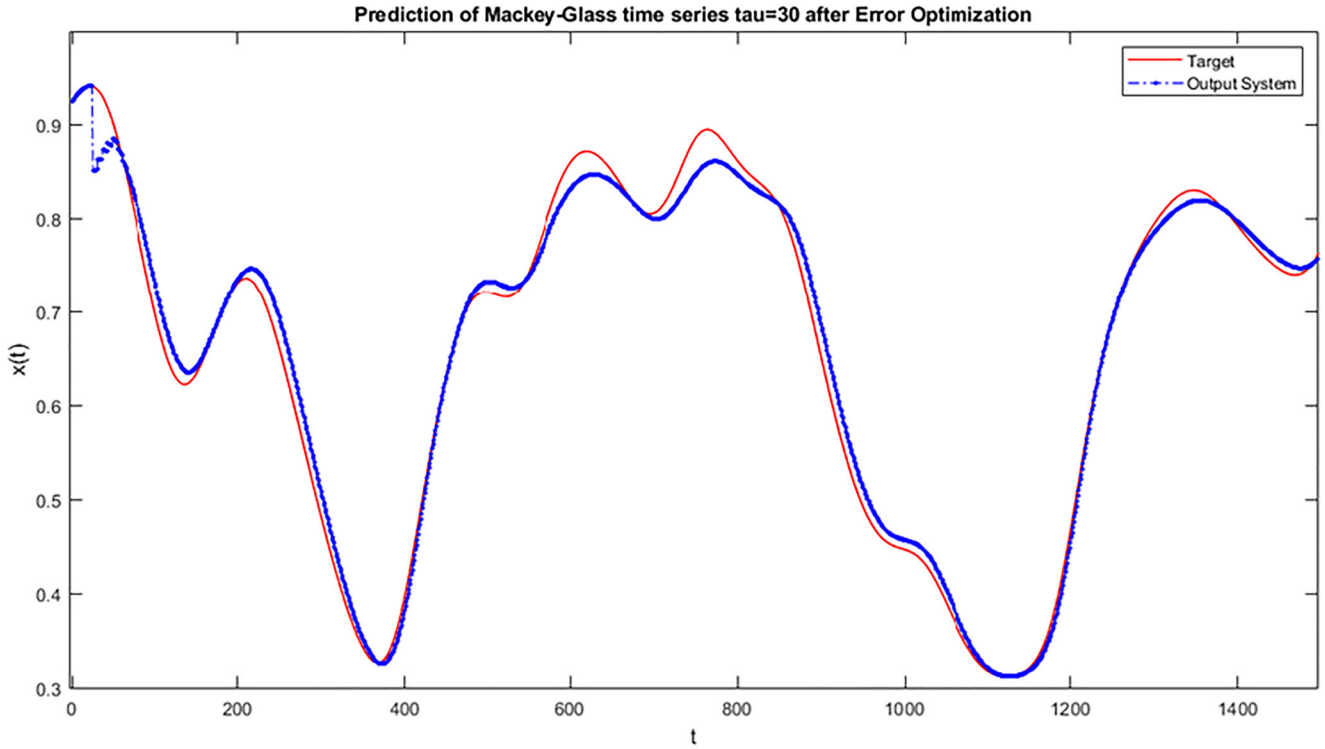


Fig. 10 Mackey-Glass Network output after the error estimation optimization $tau = 30$

our results and the corresponding desired outputs. The superposition between the two curves reflects that the network response is close to target, which is mathematically justified in Tables 3 and 4.

The nonlinear auto-regressive moving average time series The Nonlinear Auto-Regressive Moving Average later called NARMA [48] is a renowned benchmark which is characterized by its chaotic behavior. Furthermore, the non-correlation between its values makes the task of learning more difficult to achieve. Moreover, the dynamic equation of NARMA is mathematically dependent on many parameters which makes it hard to model. In light of these facts, NARMA is considered among the most complex studied benchmarks; this point motivates us to further improve the efficiency of our method by testing NARMA patterns. Mathematically, the NARMA formula is expressed as:

$$y(t+1) = c_1 y(t) + c_2 y(t) \sum_{i=1}^{k} y(t-i) + c_3 x(t-k-1) x(t) + c_4 \tag{22}$$

where $x(t)$ represents the input, k characterizes the order of the time series and the parameters c_i are set to 0.3, 0.05, 1.5 and 0.1 respectively.

The calculated errors between the output values and the desired one are visualized in Fig. 11. As the error signal is displayed around ±0.02, it clearly reflects that the network response is mimicking the required one. In the same way as in the foregoing benchmark, a MSE based comparison study with other existent forecasting methods is applied for NARMA series prediction with order $k = 10$.

As illustrated by the preceding test, the output system and the target values are plotted in the same figure (see Fig. 12). The proportionality between the two curves further justifies the obtained results.

Table 5 presents the comparison of testing results given by the already existent approaches in the literature for NARMA Benchmark forecasting.

Lorenz attractor Lorenz can be defined as a system of three nonlinear differential equations characterizing a fluid motion between a hot and a cool surface.(23–25) [49].

$$\frac{dx}{dt} = a(y - x) \tag{23}$$

$$\frac{dy}{dt} = -y - xz - rx \tag{24}$$

$$\frac{dz}{dt} = xy - bz \tag{25}$$

where $\alpha = 10$, $r = 28$ and $b = 8/3$.

As in the previous time series, our task is to forecast $x(t)$ according to a set of its own historical past values.

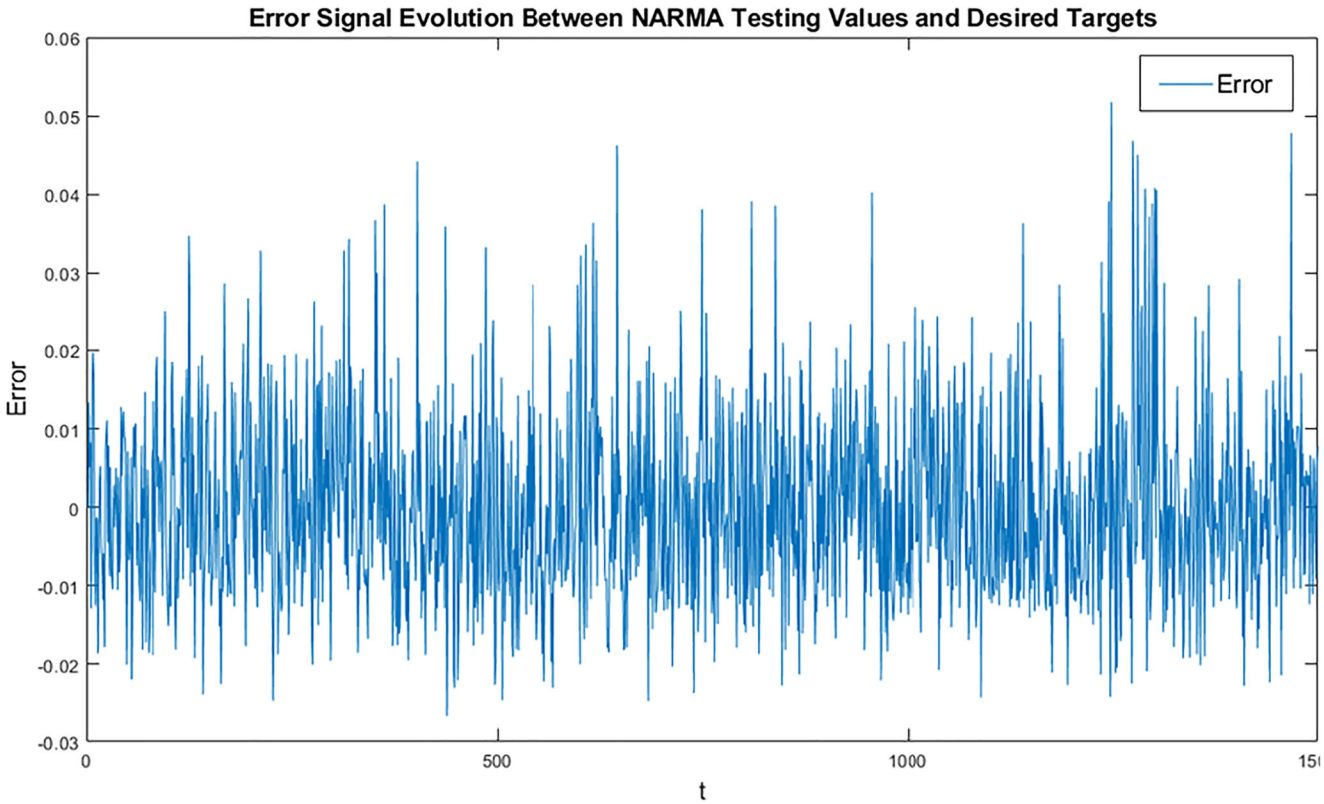


Fig. 11 Evolution of Error signal between NARMA testing values and the desired targets

In the same way as the MG test, the resulting tests are displayed in the same figure with the target values to be later compared. The superposition of the two curves is depicted in Fig. 13. It is noticed that the network outputs mimic the output line of the target values.

As revealed by the previous benchmarks, a MSE-based comparison with some existent literature RNN methods is applied for Lorenz attractor.

Table 6 recapitulates the testing results and proves that our proposed method represents a strong competitive forecaster in terms of accuracy.

Henon attractor The Henon map is a discrete-time dynamic attractor. It is considered among the most common studied dynamical systems. It is typified by its chaotic behavior defined concretely by the following equations in 26 and 27 [49].

$$x(t+1) = y(t) - ax2(t) + 1 \tag{26}$$

$$y(t+1) = bx(t) \tag{27}$$

where $a = 1.4$ and $b = 0.3$. The dataset was normalized in the interval $[0, 1]$ to get more accurate results during the implementation.

Figure 14 presents a depiction of the error signal calculated between desired values and system outputs. According to the figure, the signal of calculated errors is displayed around 0.04 which extremely proves that the network response is following the desired one.

Table 7 presents the comparison of testing results given by the already existent approaches in the literature for Henon attractor forecasting. According to Table 7, we deduce that TOREESNN outperforms the other works in terms of accuracy.

Based on results presented in Tables 3, 4, 5, 6 and 7, TOREESNN achieved the lowest MSE value in all tested time series data.

We deduce in Table 8 the obtained results given by REESNN and TOREESNN with all tested data in term of MSE (presented on the top), the Standard Deviation value denoted as STD (presented at the bottom), in addition to the improvement value.

Based on Table 8, we notice the out-performance of TOREESNN in all tests. This fact further justifies the importance of the proposed process of error estimation-based optimization.

4.2.2 Hyperparameter selection

A system cannot solve any forecasting task unless a learning process is applied. Fundamentally, learning is the mechanism of adjusting a specific set of network's parameters, allowing for obtaining the optimal accuracy. However, for every

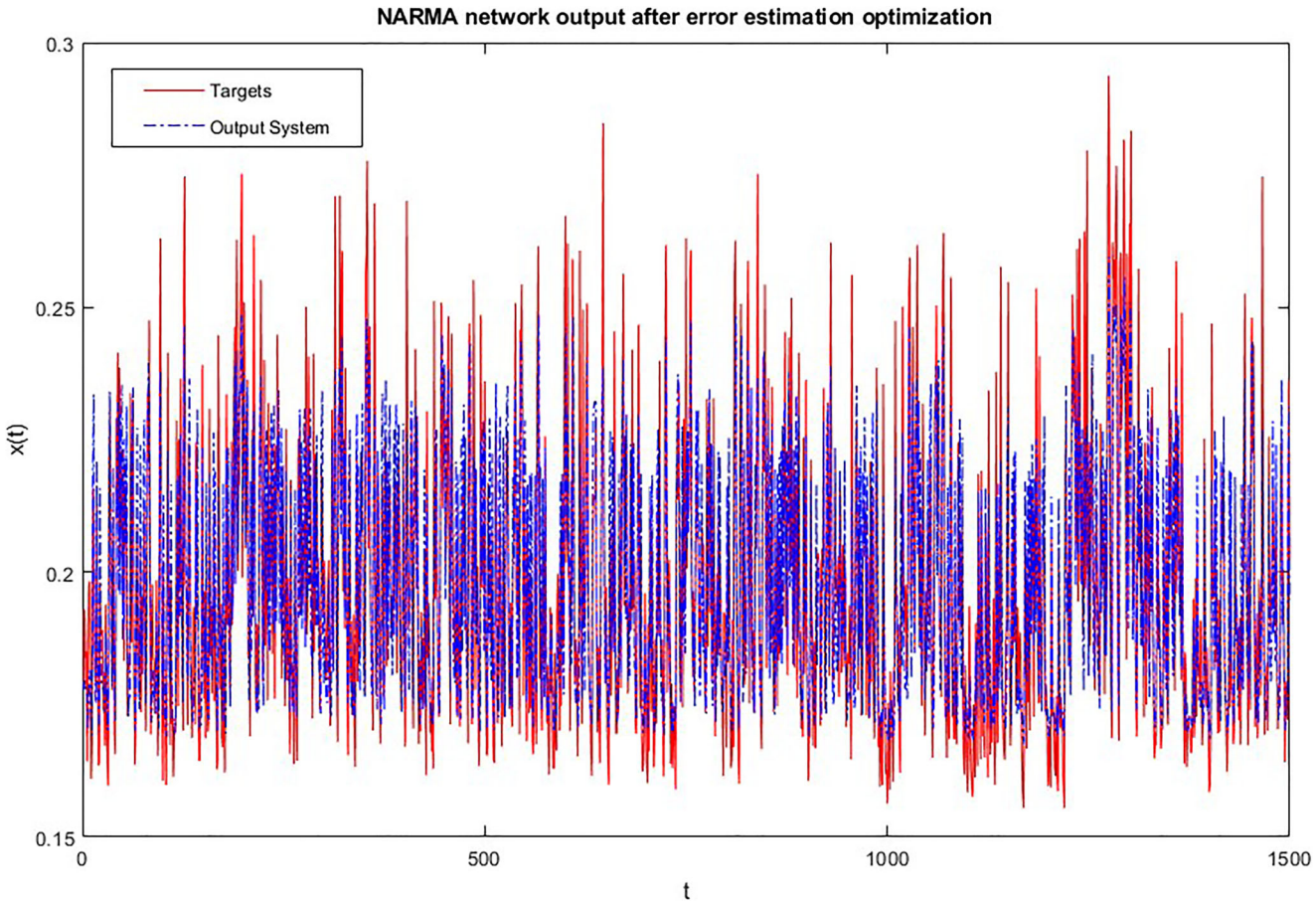


Fig. 12 NARMA Network output after the error estimation optimization

forecasting task, a set of specifications should be selected. There is no approach that can define the optimal network from the first test. Hence, building an optimal learning system requires testing a set of parameters' combination in order to choose the one that gives the best accuracy. Since our architecture is a hybrid system, we need to set the parameters of each part. First, we start by defining the different parameters of REESNN: The first step lies in defining the output and input layer. In this work, as we intend to predict the next future value, we have defined only one unit in the output layer. Whereas the number of inputs differs from one dataset to another, depending on the behavior of the data over time. In time series, the number of inputs refers to the chosen time delay.

After defining inputs and outputs, we initialize the random values of weights' connections in the interval [-0.01, 0.01]. Then, weights will be updated during the learning process. Another important parameter that influences the process of learning is known as the learning rate and denoted as α. α forms generally a small positive value set in the range between 0 and 1. In our work, we select different values of α between 0.001 and 0.1, but we store only the ones that give the lowest error. The number of hidden units is fixed after performing different tests (from 5 to 100 units). Another parameter that plays an important role in a neural architecture is the activation function. In fact, we need to define two activation functions: the first one is within the hidden layer. It defines how the model learns the data. Whereas the second within the output layer defines the type of prediction. Since we are working on chaotic TS, our system needs a function that

Table 5 MSE Based Comparison Between Our Proposed Approach and Other Literature Methods (NARMA Benchmark)

Methods	MSE test
Canonical ESN [46]	$1.71e-04$
ESN-FSR [46]	$7.29e-03$
ESN-LARS [46]	$3.79e-03$
ESN-RR [46]	$3.03e-03$
ESN-LASSO [46]	$2.89e-03$
REESNN	**5.88e-04**
TOREESNN	**1.43e-04**

In order to highlight our results (given by both REESNN and TOREESNN methods), we noted the corresponding values in bold.

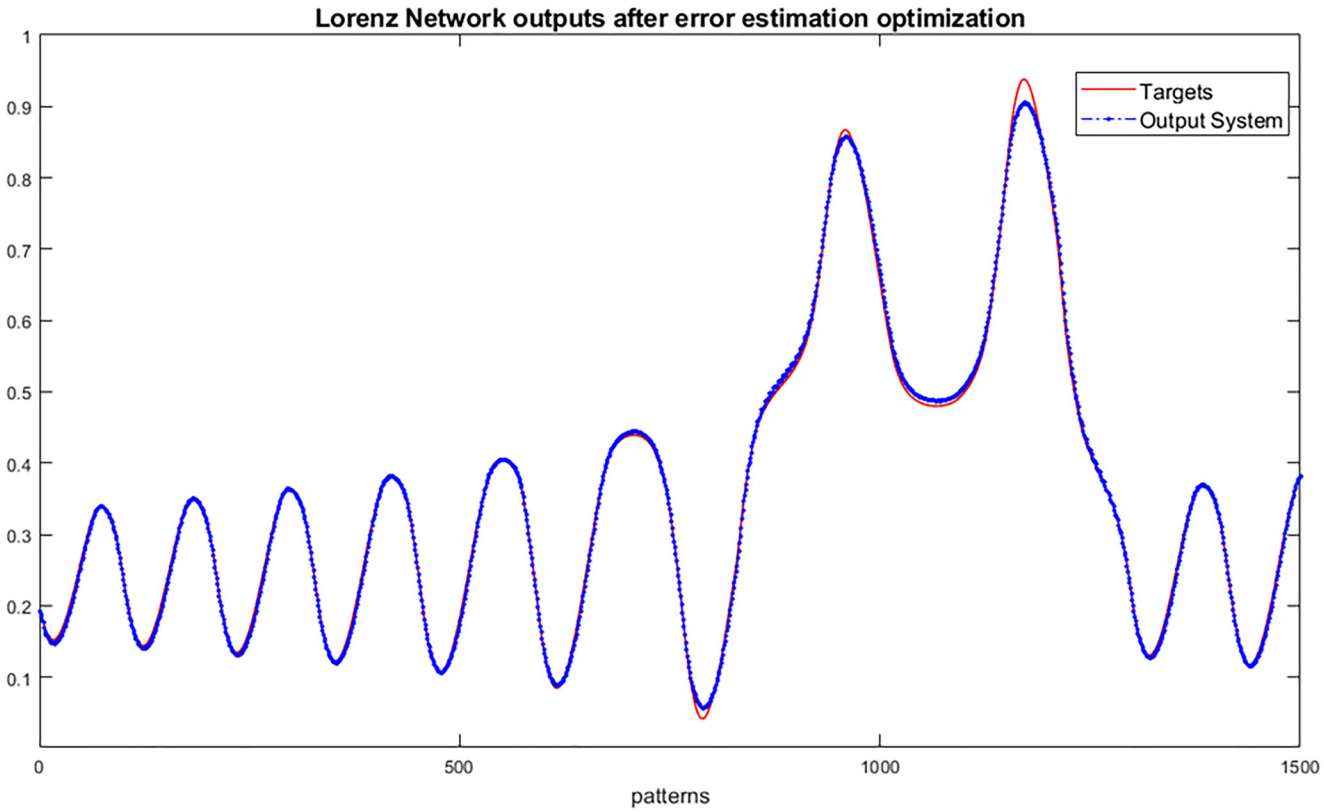


Fig. 13 Lorenz Network output after the error estimation optimization

can deal with the data nonlinearity. Moreover, the sigmoid function is the most appropriate one that can respond to the task. However, as our system has provided relevant features during the train, a simple output transfer function can generate good results, therefore the linear function is defined within the output layer.

After building the REESNN architecture, the same process is applied to the second part of our system, namely, the Elman Classifier. Finally, the combination of different parameters that gave the most accurate results is stored and displayed in Table 9.

where N_R, α_R, F_{Rh} and F_{Ro} denote respectively the number of hidden units, the learning rate, the activation function in hidden layer and the activation function in output layer in REESNN architecture; and N_E, α_E, F_{Eh} and F_{Eo} indicate respectively the number of hidden neurons, the learning rate, the transfer function in hidden layer and the transfer function in output layer in Elman Classifier architecture, whereas ET represents the execution time that takes each run for each architecture and dataset (the unit used is in second).

Table 6 MSE Based Comparison Between our Proposed Approach and Other Literature Methods (LORENZ Benchmark)

Methods	MSE test
Canonical ESN [46]	$7.23e-04$
LSTM [47]	$6.45e-04$
PSO-ESN [46]	$4.04e-06$
SVR	$2.80e-01$
REESNN	**2.06e-06**
TOREESNN	**2.03e-06**

In order to highlight our results (given by both REESNN and TOREESNN methods), we noted the corresponding values in bold.

4.2.3 Friedman test

As mentioned above, TOREESNN method performance is judged basically through the use of particular experiments. In this part of the paper, the proposed algorithm will be compared to different state-of-the-art methods using the Friedman test. In order to compute the Friedman values of the prevalent algorithms reviewed for the five datasets, the MSE indicator in Tables 3, 4, 5, 6 and 7 is conducted as an example. Hence, we provide a ranking of the technique in terms of performance accuracy, i.e, the lower rank denotes the lowest MSE and therefore better performance. However, we have selected

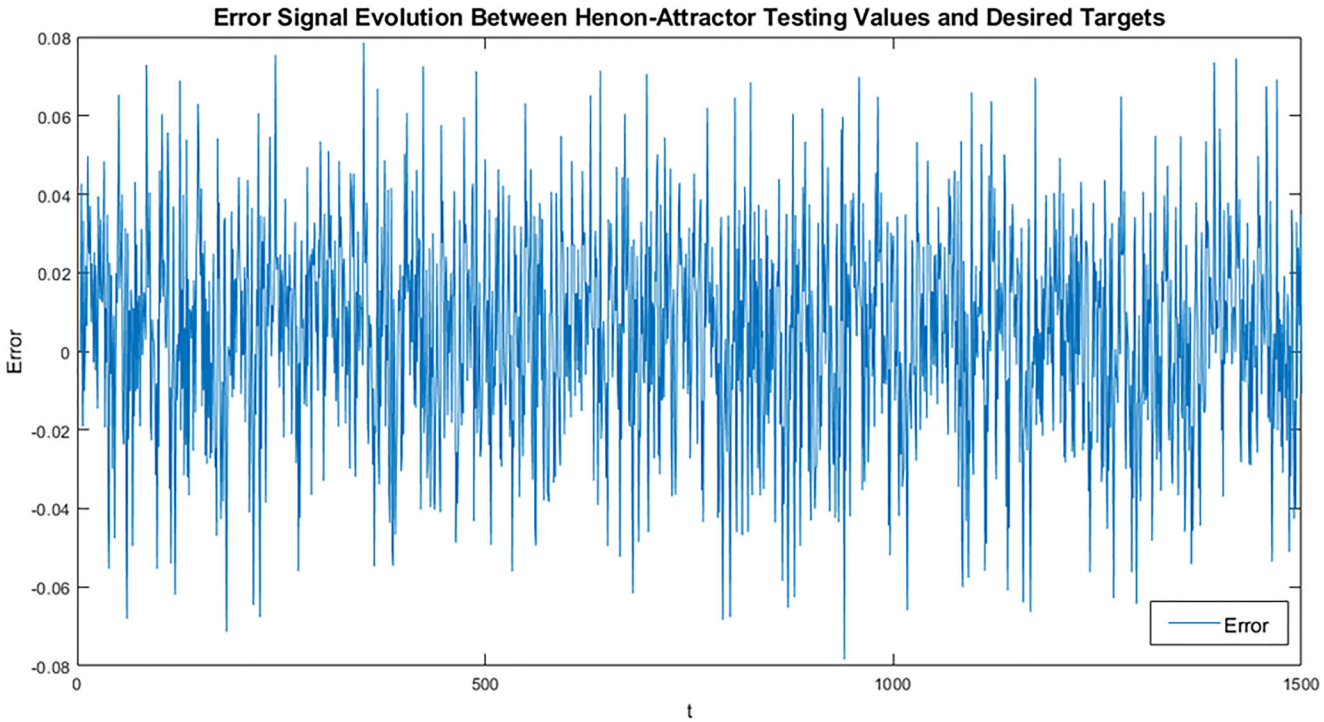


Fig. 14 Evolution of Error signal between Henon-Attractor testing values and the desired targets

only the five common algorithms tested for the dataset namely in Tables 3, 4, 5, 6 and 7. It is worth mentioning that the symbol "-" is used when the corresponding value is not stated in the literature study. The Friedman values of the five algorithms on the five datasets differentiate on the basis of the results of Table 10. Notably, TOREESNN is ranked first, REESNN comes second. While Canonical ESN ranks third and both LSTM and ESN-LARS come fourth. Once more, statistically speaking, this inference elucidates the great advantage our proposed model has over the other comparison algorithms.

Table 7 MSE Based Comparison Between our Proposed Approach and Other Literature Methods (HENON Benchmark)

Methods	MSE test
Canonical ESN [46]	$3.40e-03$
SGD-ESN [46]	$3.67e-02$
RLDDE [46]	$4.70e-03$
LSTM [47]	$2.04e-01$
REESNN	**8.46e-04**
TOREESNN	**7.48e-04**

In order to highlight our results (given by both REESNN and TOREESNN methods), we noted the corresponding values in bold.

4.3 Discussion

In this part, we will discuss the main findings and limitations of our approach.

- Firstly, the discussed approach has the potential to tackle a number of time series data including MG, NARMA, Lorenz and Henon Attractor.
- Secondly, TOREESNN is capable of modeling temporal pattern, therefore it surpasses the state-of-the-art forecasting techniques.
- For all tested datasets, TOREESNN led to the best outcomes in terms of performance measures. Based on Table 8, we notice the decrease of MSE values for the suggested method for all series in comparison with the model without optimization. For this purpose, we discuss that TOREESNN advantages from the knowledge obtained by the refinement step, i.e., the step of forecasting error estimation which permits to rectify the initial forecasting values generated by REESNN.

All along the various illustrations and tables, we endeavored to provide a sequential overview of TOREESNN. Moreover, the tests carried out with the five datasets have resulted in considerable performances and various terms. However, there are some limitations of the application of our approach:

Table 8 Comparison of the proposed model before and after error estimation optimization (top: MSE, bottom: STD)

	MG 17	MG 30	NARMA	Lorenz	Henon
REESNN	**2.29e-04**	**3.74e-04**	**5.88e-04**	**2.06e-06**	**8.46e-04**
	$\pm 2.46e-04$	$\pm 1.17e-04$	$\pm 8.74e-05$	$\pm 1.07e-04$	$\pm 7.17e-05$
TOREESNN	**2.20e-04**	**3.48e-04**	**1.43e-04**	**2.03e-06**	**7.48e-04**
	$\pm 2.35e-04$	$\pm 9.86e-05$	$\pm 2.84e-04$	$\pm 7.66e-05$	$\pm 4.04e-05$
Improvement	**0.09e-04**	**0.26e-04**	**4.45e-04**	**0.03e-06**	**0.98e-04**

In order to highlight our results (given by both REESNN and TOREESNN methods), we noted the corresponding values in bold.

Table 9 The parameters setting of TOREESNN in all experiments

	N_R	α_R	*FRh*	*FRo*	N_E	α_E	*FEh*	*FEo*	*ET* (*sec*)
MG 17	5	0.001	Sigmoid	Linear	4	0.01	Sigmoid	Tanh	1.59
MG 30	5	0.001	Sigmoid	Linear	4	0.01	Sigmoid	Tanh	2.19
NARMA	5	0.1	Sigmoid	Linear	4	0.01	Sigmoid	Tanh	1.78
LORENZ	4	0.01	Sigmoid	Linear	4	0.01	Sigmoid	Tanh	1.79
Henon-Attractor	4	0.1	Sigmoid	Linear	4	0.01	Sigmoid	Tanh	1.98

Table 10 Friedman Test

Method DataSet	LSTM	ESN-LARS	Canonical ESN	REESNN	TOREESNN
MG 17	4	3	5	2	**1**
MG 30	–	4	3	2	**1**
NARMA	–	4	2	3	**1**
LORENZ	3	–	4	2	**1**
Henon-Attractor	4	–	3	2	**1**

– While the empirical outcomes ascertain the theoretical study carried out in the previous sections, the outlined Fig. 15 presents some suspicious observations, i.e., despite the accuracy of our approach in learning the data behavior, it is still not proficient to sort out the anomalies found within a time series. These questionable observations are considered unpredictable noise. Therefore, we are developing a way to address it, so that our system will be able to recognize and generalize temporal patterns including their abnormalities.
– Moreover, the typology of our method is only evaluated to solve univariate time series data. Nevertheless, providing an analogous evaluation for multivariate time series prediction problems is worth elaborating.

5 Conclusion

In literature, the Forecasting task was approached with numerous Artificial Neural Network methods inter alia Recurrent Neural Networks architectures. In this study, we utilized an innovative JRNN architecture (named TOREESNN) inspired from the ES method extended by error approximation to predict and optimize one of the most outstanding time series namely Mackey-Glass time series, Henon Attractor, Lorenz and NARMA. We set a comparison between our method and other RNN architectures in terms of MSE and it is obvious that our approach presents a promising technique to deal with chaotic time series.

While current forecasting systems are working on bettering their accuracy by building a robust architecture, this paper approves the assumption that optimizing the task of forecasting is achieved by predicting the error associated to each forecast value. Precisely, the optimization step offers the possibility to make full use of historical knowledge in past data as it is ideally suitable for memorizing not only past forecasted values but also the error made during the validation phase. Hence, if forecasting systems aim to generalize the temporal patterns within a time series, then the goal of our approach is to degrade

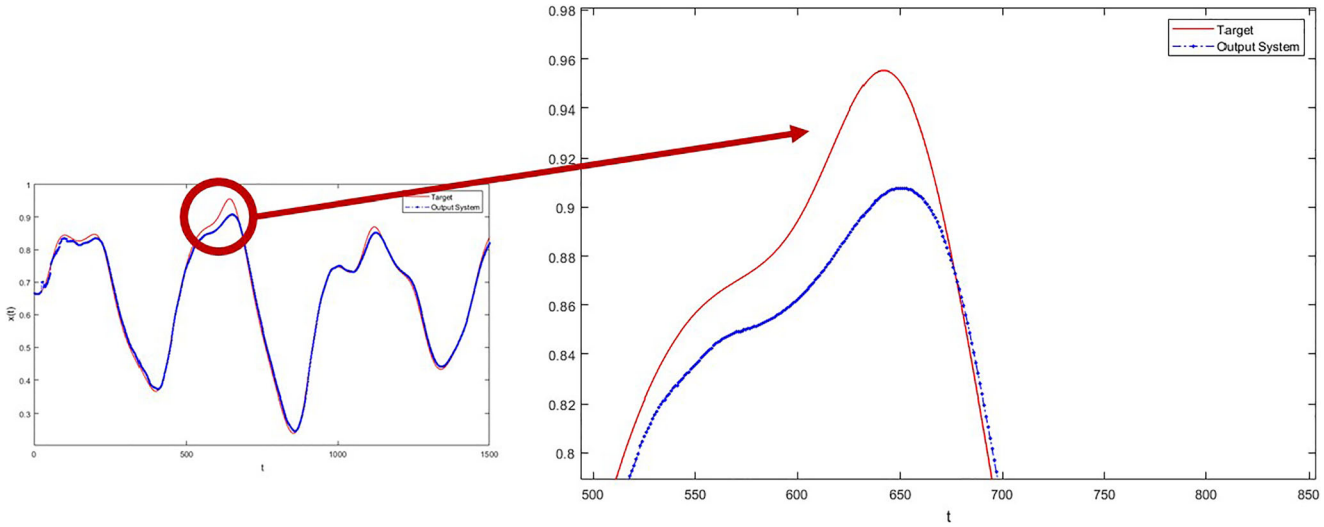


Fig. 15 The unpredictability of Mackey-Glass time series after error optimization

the error associated to each forecast by estimating it then correcting the first predictions.

As for our future work, TOREESNN will be enhanced by extending the optimization system to contain many stages within the given approach. This broadening will boost its capability to detect the anomalies found in time Series. Researchers are also able to try to merge other machine learning methods in order to better refine the learning model. Last but not least, an application in which TOREESNN is introduced in real world is still under elaboration.

Acknowledgments We deeply acknowledge Taif University for Supporting this study through Taif University Researchers Supporting Project number (TURSP-2020/327), Taif University, Taif, Saudi Arabia.

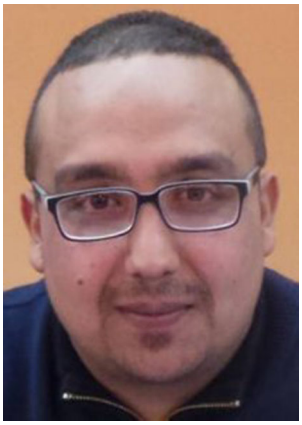
The research leading to these results has received funding from the Ministry of Higher Education and Scientific Research of Tunisia under the grant agreement number LR11ES48.

References

1. Ni Q, Wang Y, Fang Y (2021) GE-STDGN: a novel spatio-temporal weather prediction model based on graph evolution. Appl Intell 52:7638–7652. https://doi.org/10.1007/s10489-021-02824-2
2. Hui H, Jia R, Shi X, Liang J, Dang J (2021) Feature selection and hyper parameters optimization for short-term wind power forecast. Appl Intell 51:10
3. Muller-Navarra M, Lessmann S, Voss S (2015) Sales Forecasting with Partial Recurrent Neural Networks: Empirical Insights and Benchmarking Results. *2015 48th Hawaii International Conference On System Sciences*. pp 1108–1116. https://doi.org/10.1109/HICSS.2015.135
4. Khan G, Khattak A, Zafari F, Mahmud S (2013) Electrical load forecasting using fast learning recurrent neural networks. *The 2013 International Joint Conference On Neural Networks (IJCNN)*. pp 1–6 (8). https://doi.org/10.1109/IJCNN.2013.6706998
5. Cerjan M, Krzelj I, Vidak M, Delimar M (2013) A literature review with statistical analysis of electricity price forecasting methods. IEEE EuroCon 2013 7:756–763
6. Yang Y, Tao Z, Qian C, Gao Y, Zhou H, Ding Z, Wu J (2022) A hybrid robust system considering outliers for electric load series forecasting. Appl Intell 52:1630–1652. https://doi.org/10.1007/s10489-021-02473-5
7. Alghamdi D, Basulaiman K, Rajgopal J (2022) Multi-stage deep probabilistic prediction for travel demand. Appl Intell (1)
8. Xu C, Zhang A, Xu C, Chen Y (2022) Traffic speed prediction: spatiotemporal convolution network based on long-term, short-term and spatial features. Appl Intell 52:1
9. Liu J (1988) Hawaii tourism to the year 2000: a Delphi forecast. Tour Manag 9:279–290
10. Brown B (2003) Survey Methods and Practices. (Canada)
11. Brown B (1968) Delphi process: a methodology used for the elicitation of opinions of experts. RAND Corporation, Santa Monica
12. Sackman H (1974) Delphi Critique; Expert Opinion, Forecasting, and Group Process. (Lexington Books)
13. Green K, Armstrong J (2007) Structured analogies for forecasting. International Journal Of Forecasting 23:365–376
14. Song H, Witt S, Li G (2003) Modelling and forecasting the demand for Thai tourism. Tour Econ 9:363–387
15. Hyndman R, Athanasopoulos G (2014) Forecasting: principles and practice. (OTexts.com [Heathmont?, Victoria])
16. Chou Y (1975) Statistical Analysis. (Holt International)
17. Brown R (1956) Exponential Smoothing for Predicting Demand
18. Abraham, B. & Ledolter, J. Statistical forecasting methods. (New Jersey, 1983)
19. Tealab A (2018) Time series forecasting using artificial neural networks methodologies: a systematic review. Future Computing And Informatics Journal 3:334–340. http://www.sciencedirect.com/science/article/pii/S2314728817300715
20. Krichene E, Masmoudi Y, Alimi A, Abraham A, Chabchoub H (2017) Forecasting using elman recurrent neural network. Intell Sys Des Appl. pp 488–497. https://doi.org/10.1007/978-3-319-53480-0_48
21. Elman J (1990) Finding structure in time. Cogn Sci 14:179–211
22. Jordan M (1990) Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. IEEE Computer Society Neural Networks Technology Series, pp 112–127
23. Yu C, Li Y, Zhang M (2017) Comparative study on three new hybrid models using Elman neural network and empirical mode decomposition based technologies improved by singular Spectrum analysis for hour-ahead wind speed forecasting. Energy Convers Manag 147:75–85

24. Zainorzuli S, Afzal Che Abdullah S, Adnan R, Ruslan F (2019) Comparative Study of Elman Neural Network (ENN) and Neural Network Autoregressive With Exogenous Input (NARX) For Flood Forecasting. *2019 IEEE 9th Symposium On Computer Applications Industrial Electronics (ISCAIE)*. pp 11–15
25. Mengying H, Jiandong D, Zequan H, Peng W, Shuai F, Peijia H, Chaoyuan F (2019) Monthly Electricity Forecast Based on Electricity Consumption Characteristics Analysis and Multiple Effect Factors. *2019 IEEE 8th International Conference On Advanced Power System Automation And Protection (APAP)*. pp 1814–1818
26. Di P, Dong K, Du J, Dong C, He X, Guan Y, Gao H, Li J, Liang Y (2019) Ultra-Short Term Load Forecasting Based on Elman Neural Network. *2019 IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia)*. pp. 911–915
27. Su Y, Wang S, Xiao Z, Tan M, Wang M (2018) An ultra-short-term wind power forecasting approach based on wind speed decomposition, wind direction and elman neural networks. *2018 2nd IEEE Conference On Energy Internet And Energy System Integration (EI2)*. pp 1–9. https://doi.org/10.1109/EI2.2018.8582514
28. Khan I, Zhu H, Yao J, Khan D (2017) Photovoltaic power forecasting based on Elman Neural Network software engineering method. *2017 8th IEEE International Conference On Software Engineering And Service Science (ICSESS)*. pp 747–750. https://doi.org/10.1109/ICSESS.2017.8343021
29. Sadeghi-Niaraki A, Mirshafiei P, Shakeri M, Choi S (2020) Short-term traffic flow prediction using the modified Elman recurrent neural network optimized through a genetic algorithm. IEEE Access 8:217526–217540
30. Gozalpour N, Teshnehlab M (2019) Forecasting Stock Market Price Using Deep Neural Networks. *2019 7th Iranian Joint Congress On Fuzzy And Intelligent Systems (CFIS)*. pp. 1–4
31. Baziyad M, Jarndal A, Bettayeb M (2019) A Model Order Reduction Technique Based on Balanced Truncation Method and Artificial Neural Networks. *2019 8th International Conference On Modeling Simulation And Applied Optimization (ICMSAO)*. pp. 1–5
32. Šestanović T (2019) Jordan neural network for inflation forecasting. Croatian Operation Res Rev 10(7):23–33
33. Hikmawati F, Suhartono S, Prastyo D (2021) A Hybrid GSTARX-Jordan RNN Model for Forecasting Space-Time Data with Calendar Variation Effect. J Phys Conf Ser 1752(2):012013
34. Althelaya K, El-Alfy E, Mohammed S (2018) Evaluation of bidirectional LSTM for short-and long-term stock market prediction. *2018 9th International Conference On Information And Communication Systems (ICICS)*. pp 151–156. https://doi.org/10.1109/IACS.2018.8355458
35. Sagheer A, Kotb M (2019) Time series forecasting of petroleum production using deep LSTM recurrent networks. Neurocomputing 323:203–213. http://www.sciencedirect.com/science/article/pii/S0925231218311639
36. Ergen T, Kozat S (2018) Efficient online learning algorithms based on LSTM neural networks. IEEE Transactions On Neural Networks And Learning Systems 29:3772–3783
37. Tuna T, Beke A, Kumbasar T (2022) Deep learning frameworks to learn prediction and simulation focused control system models. Appl Intell 52:1
38. Chiu S, Chen Y, Lee C (2022) Estate price prediction system based on temporal and spatial features and lightweight deep learning model. Appl Intell 52:1
39. Tokgöz A, Ünal G (2018) A RNN based time series approach for forecasting turkish electricity load. *2018 26th Signal Processing And Communications Applications Conference (SIU)*. pp. 1–4
40. Saini U, Kumar R, Jain V, Krishnajith M (2020) Univariant Time Series forecasting of Agriculture load by using LSTM and GRU RNNs. *2020 IEEE Students Conference On Engineering Systems (SCES)*. pp. 1–6. https://doi.org/10.1109/SCES50439.2020.9236695
41. Tsuji T (1996) Model reduction with time delay combining the least-squares method with the genetic algorithm. IEE Proceedings Control Theory Appl 143(7):247–254
42. Wutsqa D, Kusumawati R, Subekti R (2014) The application of Elman recurrent neural network model for forecasting consumer price index of education, recreation and sports in Yogyakarta. *2014 10th International Conference On Natural Computation (ICNC)*. pp. 192–196 (8)
43. Brook T (1715) Direct and Reverse Methods of Incrementation. (London)
44. Lin C, Lee C (1996) Neural Fuzzy Systems: A Neuro-fuzzy Synergism to Intelligent Systems. (Prentice-Hall, Inc.,)
45. Lapedes A, Farber R (1987) Nonlinear Signal Processing Using Neural Networks: Prediction and System Modeling. (Los Alamos National Laboratory)
46. Chouikhi N, Ammar B, Rokbani N, Alimi A (2017) PSO-based analysis of Echo state network parameters for time series forecasting. Appl Soft Comput 55:211–225
47. Chandra R, Goyal S, Gupta R (2021) Evaluation of Deep Learning Models for Multi-Step Ahead Time Series Prediction. IEEE Access 9:83105–83123. https://doi.org/10.1109/ACCESS.2021.3085085
48. Basterrech S, Alba E, Snásel V (2015) An Experimental Analysis of the Echo State Network Initialization Using the Particle Swarm Optimization. *CoRR*. abs/1501.00436
49. Dhahri H, Alimi A, Abraham A (2014) Designing of Beta Basis Func- tion Neural Network for optimization using cuckoo search (CS). *2014 14th International Conference On Hybrid Intelligent Systems*. pp 110–116 (12)

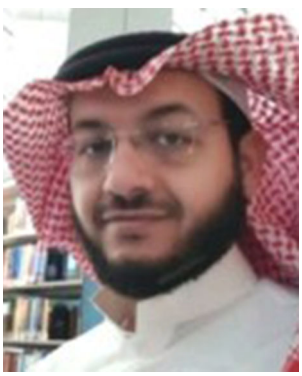
Emna Krichene is a Ph.D. student at REGIM-Lab. Her topic of interest till the fulfillment of her Ph.D. is about timeseries learning serving by recurrent neural networks. She was a master student at Higher institute of computer science and multimedia of Sfax-Tunisia (ISIMS). Her master graduation took place by June, 2013. She was born on February 18th, 1988 in Sfax-Tunisia. Her research interest includes artificial intelligence: neural network, forecasting and prediction optimization.



Dr. Wael Ouarda obtained his PhD in Computer Science from the Research Groups in Intelligent Machines (ReGIM-Lab) at the National School of Engineers of Sfax (ENIS) in 2017. He is currently a Professor-Researcher in Computer Science at the University of Sfax and quality manager at the ReGIM-Lab since 2016. Dr. Wael OUARDA has more than 30 publications in prestigious conferences and journals in the field of Artificial Intelligence. His research focuses on Image Analysis (biometric, medical and social) with deep learning techniques of neural networks for the representation of features and for classification. Dr. Wael OUARDA has been an active IEEE member since 2012.

Dr. Habib Chabchoub is a full professor of Management Science and director of the MBA program at the College of Business, Abu Dhabi campus, in Al Ain University (UAE). He has initiated and led different Master programs in Management and a PhD program in Quantitative Methods at University of Sfax (Tunisia). He has supervised more than 20 PhD theses, co-worn several international academic and research projects (TEMPUS "Aqi-Umed", CMCU, bilateral projects, and others) and been involved in several international conferences (MOPGP, META, ICALT, LOGISTIQUA, etc.). He co-authored more than 100 refereed publications, several of which are in high impact forums. He received a BSc in Mathematics and a MSc in Management Science from University of Tunis (Tunisia) and a PhD in Operations and Decision Systems from Laval University (Canada). His research interest encompasses supply chain and logistics management, multiple objective programming, multi criteria decision making and meta-heuristics.

Ajith Abraham Dr. Ajith Abraham received the M.Sc. degree from Nanyang Technological University, Singapore, in 1998, and the Ph.D. degree in computer science from Monash University, Melbourne, Australia, in 2001. He is currently the Director of the Machine Intelligence Research Labs (MIR Labs), a Not-for-Profit Scientific Network for Innovation and Research Excellence connecting Industry and Academia. The Network with HQ in Seattle, USA, has currently more than 1000 scientific members from more than 100 countries. As an Investigator/Co-Investigator, he has won research grants worth more than U.S.$100 million from Australia, USA, EU, Italy, Czech Republic, France, Malaysia, and China. He works in a multi-disciplinary environment involving machine intelligence, cyber-physical systems, the Internet of Things, network security, sensor networks, web intelligence, web services, data mining, and applied to various real-world problems. In these areas, he has authored/coauthored more than 1400 research publications out of which there are more than 100 books covering various aspects of computer science. He has more than 46,000 academic citations (H-index of 100 as per Google Scholar). He is also the Editor-in-Chief of Engineering Applications of Artificial Intelligence (EAAI) and serves/served the editorial board of more than 15 international journals indexed by Thomson ISI. More information can be available at http://www.softcomputing.net.



Dr. Abdulrahman Qahtani is currently Assistant professor in computer science department at Taif University. Dr. Qahtani received his PhD from Southampton University in 2015. He has extensive experience in software engineering and development process in a distributed domain. His research focused on customization process across organizational boundaries. Recently, he applied machine learning algorithms on software engineering data to predict time and cost estimation for multi-clients projects.

Dr. Omar Almutiry received the PhD degree from the University of Southampton, UK, in 2017. Since 2018, he has been with the Applied Computer Science College, King Saud University, Saudi Arabia, where he is currently working as an Assistant Professor with the Department of Applied Computer Sciences. He is also assistant to the General Director of the Almuzahmiyah branch for academic affairs and development. His research areas include data science, health informatics, deep learning applications in healthcare and medical fields.

Dr. Habib Dhahri was born in Sidi Bouzid, Tunisia in 1975. He graduated in Computer Science in 2001, obtained his PhD in Computer Science in 2013 from the National Engineering School of Sfax (ENIS). He is now an assistant professor in Computer Science at the King Saud University. His area of interest includes computational intelligence, soft computing techniques and Machine Learning for Healthcare Data. He has authored and co-authored more than 30 publications in journals and conferences. He also serves as a reviewer for several international scientific journals.

Dr. Adel M. Alimi was born in Sfax (Tunisia) in 1966. He graduated in Electrical Engineering 1990, obtained a Ph.D. and then an HDR both in Electrical and Computer Engineering in 1995 and 2000 respectively. He is now a Professor in Electrical and Computer Engineering at the University of Sfax. His research interest includes applications of intelligent methods (neural networks, fuzzy logic, evolutionary algorithms) to pattern recognition, robotic systems, vision systems, and industrial processes. He focuses his research on intelligent pattern recognition, learning, analysis and intelligent control of large scale complex systems. He is an Associate Editor and Member of the editorial board of many international scientific journals (e.g. "Pattern Recognition Letters", "Neurocomputing", "Neural Processing Letters", "International Journal of Image and Graphics", "Neural Computing and Applications", "International Journal of Robotics and Automation", "International Journal of Systems Science", etc.). He was a Guest Editor of several special issues of international journals (e.g. Fuzzy Sets and Systems, Soft Computing, Journal of Decision Systems,Integrated Computer Aided Engineering, Systems Analysis Modelling and Simulations). He is an IEEE senior member and member of IAPR, INNS and PRS. He is the 2009-2010 IEEE Tunisia Section Treasurer, the 2009-2010 IEEE Computational Intelligence Society Tunisia Chapter Chair, the 2011 IEEE Sfax Subsection, the 2010-2011 IEEE Computer Society Tunisia Chair, the 2011 IEEE Systems, Man, and Cybernetics Tunisia Chapter, the SMCS corresponding member of the IEEE Committee on Earth Observation, and the IEEE Counselor of the ENIS Student Branch.

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com