



Article

PLS-CNN-BiLSTM: An End-to-End Algorithm-Based Savitzky–Golay Smoothing and Evolution Strategy for Load Forecasting

Mohamed Massaoudi ^{1,2,*} , Shady S. Refaat ¹ , Haitham Abu-Rub ¹, Ines Chihi ³
and Fakhreddine S. Oueslati ²

¹ Department of Electrical and Computer Engineering, Texas A and M University at Qatar, Doha 3263, Qatar; shady.khalil@qatar.tamu.edu (S.S.R.); haitham.abu-rub@qatar.tamu.edu (H.A.-R.)

² Unité de Recherche de Physique des Semi-Conducteurs et Capteurs, Carthage University, Tunis 2070, Tunisia; Segnioueslati@enicarthage.rnu.tn

³ Laboratory of Energy Applications and Renewable Energy Efficiency (LAPER), El Manar University, Tunis 1002, Tunisia; Ines.Chihi@enib.rnu.tn

* Correspondence: mohamed.massaoudi@qatar.tamu.edu

Received: 2 September 2020; Accepted: 14 October 2020; Published: 19 October 2020



Abstract: This paper proposes an effective deep learning framework for Short-Term Load Forecasting (STLF) of multivariate time series. The proposed model consists of a hybrid Convolutional neural network-Bidirectional Long Short-Term Memory (CBiLSTM) based on the Evolution Strategy (ES) method and the Savitzky–Golay (SG) filter (SG-CBiLSTM). The adopted methodology incorporates the virtue of different preprocessing blocks to enhance the performance of the CBiLSTM model. In particular, a data-augmentation strategy is employed to synthetically improve the feature representation of the CBiLSTM model. The augmented data is forwarded to the Partial Least Square (PLS) method to select the most informative features above the predefined threshold. Next, the SG algorithm is computed for smoothing the load to enhance the learning capabilities of the underlying system. The structure of the SG-CBiLSTM for the ISO New England dataset is optimized using the ES technique. Finally, the CBiLSTM model generates output forecasts. The proposed approach demonstrates a remarkable improvement in the performance of the original CBiLSTM model. Furthermore, the experimental results strongly confirm the high effectiveness of the proposed SG-CBiLSTM model compared to the state-of-the-art techniques.

Keywords: Bidirectional Long Short-Term Memory (BiLSTM); Convolutional Neural Network (CNN); evolution strategy; Partial Least Square (PLS) method; Savitzky–Golay; Short-Term Load Forecasting (STLF)

1. Introduction

For power systems, grid stability is becoming very sensible to the bulk integration of renewable energy sources due to the irregularity and seasonality of numerous external factors. These factors fall into two major aspects: (i) Meteorological, such as temperature, irradiation, and wind speed, and (ii) social, such as working days, lifestyles, and number of consumers [1]. Thus, grid sustainability is threatened by the high fluctuations in the load profiles. These fluctuations cause financial penalties and serious technical issues related to the electrical grid. Load forecasting (LF) is a central solution to effectively cope with the dynamic operational environment. The LF is mainly classified into three categories based on the time horizon:

long-term, medium-term, and short-term. The Long-Term Load Forecasting (LTLF) ranges from months until years ahead. The LTLF is used for project planning and long-term capital investment return tasks. While the medium-term load forecasting results, ranging from few days until weeks ahead, are employed for asset maintenance schedule, power interchange with other utilities, and security of the grid. Finally, the Short-Term Load Forecasting (STLF) is valid for minutes until days ahead. STLF achieves higher unit commitment, sustainable operations, and safe economic dispatch [2].

STLF is vitally beneficial for the optimization of the operational generation capacity between the generation units and demand-side units. Furthermore, STLF plays an extremely important role in enabling the bulk penetration of grid-connected wind and solar energy in a reliable manner. The energy balance between the utility grid and load demand accentuates the need for STLF techniques to comprehensively manage electrical asset utilization and operating efficiency. However, precise STLF persists a fairly challenging problem due to the greater complexity of underlying load patterns and the high volatility of the meteorological parameters.

Many techniques have been employed to improve the forecasting accuracy of the STLF [3–5]. Most efforts in the area of LF have been focused on the application of Machine learning (ML) models due to their mature development, large popularity, and ease of implementation [6]. Nevertheless, ML techniques in many case studies were unable to satisfactorily meet the actual load demand in terms of prediction accuracy [7]. Moreover, manpower and human expertise are still required to tune the hyperparameters such as learning rate and number of iterations. Since the load patterns have greater complexity, the traditional ML-based STLF models lack accuracy and robustness due to the narrow variations of the load patterns [8]. Alternatively, Deep Learning (DL) presents cutting-edge technology to model complex and nonlinear systems with higher generalization capabilities and better reliability with the increase in depth [8]. Some recent advances of DL methods could be found in [9–13]. Nevertheless, DL techniques remain less explored compared to the large deployment of ML techniques. This is due to the fact that DL requires a high understanding of the model construction and deployment [2]. Many researchers paid attention to DL deployment with the progress of hardware and software resources. DL techniques, especially Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) achieve accurate and defensible forecasts than conventional forecasting-based modeling [14]. Authors in [15] used LSTM based ensemble learning. The proposed model is employed for multi-step industrial power demand forecasting. In [16], a multi-scale CNN with time-cognition is developed for STLF. A state-of-the-art time coding to emphasize the uniqueness of the moment in one period is implemented [15]. This gives more information about the samples periodicity [15]. To improve cognition for time series data, the authors in [17] coupled LSTM with Singular Spectral analysis and variational mode decomposition. However, most of CNN and LSTM deployment is applied to classification problems such as natural language recognition and image classification. The applications of LSTM and CNN networks on regression tasks are relatively limited according to the recently published research papers.

In [18], the authors proposed a deep CNN method based on modified Residual Networks (ResNet) to perform the STLF. The proposed model implements the knowledge expertise based on Modified ResNet model to cope with the nonlinear variation of the load demand. However, the computational burden causes a low and heavy training process that remains to over 8 h. Further, the proposed Modified ResNet does not have a higher generalization capability such as the original ResNet [18]. The authors in [19] implemented LSTM model to forecast the load of individual residential households. The proposed method in [19] presents one of the most efficient architectures in the realm of sequential learning. The forecasting framework is coupled with Density-Based Spatial Clustering of Application with Noise (BSCAN) clustering algorithm for the consistency analysis of daily power profile. The proposed LSTM model provided accurate results compared to the back-propagation neural network and empirical Mean Absolute Percentage Error (MAPE) minimization [19]. However, even with a deep architecture, the proposed framework-based LSTM

was found very sensible to the inconsistency of residents' lifestyles which significantly diminishes its extrapolation ability [19].

Authors in [14] introduce a hybrid CNN-LSTM model for residential energy consumption forecasting. While the proposed model outperforms several benchmarks, the visualization graphs and the score metrics clearly show that the experimental results are not satisfactory with a MAPE = 34.84% due to the irregular trend of power consumption data. To the authors' best knowledge, the research works focus to date on the development of the forecasting engine rather than the deep investigation of the feature representations related to load data itself. The DL architectures can automatically carry out the data processing. However, there are very few available research works oriented towards the hybridization of DL models along with different features engineering techniques to achieve higher prediction accuracy [20,21]. Therefore, this paper aims to tackle the non-linearity and volatility of STLF by coupling data augmentation, variable importance analysis, and feature smoothing alongside with hybrid CBiLSTM model with the aim to provide a highly reliable hourly STLF system. The major contributions of this paper are the following:

- A novel forecasting framework is comprehensively presented for STLF. The proposed SG-CBiLSTM model is recommended for highly nonlinear time series, whose patterns are difficult to capture by simple models.
- A novel method for STLF is introduced and verified based on real-world dataset and simulation results. Regardless the numerous benefits of STLF, the actual forecasting methods remain less effective yielding non-negligible forecast errors due to the non-linearity of the load patterns. The proposed methodology greatly enhances the overall accuracy of the prediction engine for the STLF application.
- The performance of the hybrid model is investigated using point forecast assessment, ten-fold cross-validation, score metrics, reference DL models, and simulation graphs.

The structure of this paper is organized as follows: Section 2 introduces the existing state-of-the-art techniques. Section 3 presents the preliminaries relative to the current study and describes the proposed model. Section 4 evaluates the feasibility of the model with the numerical analysis. Moreover, the comparison of the proposed method with some relevant ML techniques considered as benchmarks for this study is conducted in Section 4. Section 5 concludes the study and ends the paper.

2. Related Works

The need for very precise load forecasts requires the use of well-defined techniques in the feature engineering procedure. In this research study, the proposed techniques are divided into three folds: data smoothing, features selection, and Data Augmentation (DA). A brief introduction of these techniques is given as follows:

2.1. Data Augmentation (DA)

The DA technique is employed to reduce the underfitting problem of DL architectures by describing the data in a meaningful way [19]. DA improves the size and quality of the load information. This improvement lies in generating more information from the data representation to give a better insight about the load profile for a specific forecasting horizon. However, the application of DA is not very common on time series contrary to its vast applicability to image recognition and natural language processing. For instance, the authors in [22] derive from the datetime data type additional information to the celebration days and holidays such as Christmas Day, Thanksgiving Day, and Independence Day. Authors in [23] proposed an augmented CNN method to provide better learning potential covering unexplored input space for the CNN training. The key idea of the proposed DA procedure consists of concatenating heterogeneous residual load data generated from the electrical load of a single household. A technical paper in [24] provides a complete survey on the latest taxonomy of time series DL technique.

In our work, the hourly temperature is used as indicator feature for the load state, the proposed DA technique consists of two stages: firstly apply the time decomposition to reveal the behavior of consumers and then lagged load generation to follow the chains of load variation in time.

2.2. Feature Importance and Dimensionality Reduction

Feature importance is highly required to identify the most relevant features of the data representation. Feature dimension optimization is essential to improve the forecasting accuracy comprehensibility, and avoid overfitting problems. For a high dimensional data analysis, The removal of irrelevant and repetitive feature labels is the simplest technique to perform Dimensionality Reduction (DR). In [23], the authors studied the effect of DR on the load demand. Their study consists of generating 67 vectors regarded as possible potential inputs of the forecasting system. Next, they demonstrated that DR-based Principal Components Analysis (PCA) and Canonical Variate Analysis (CVA) succeed to enhance the accuracy of the forecasting results with the performance superiority of the PCA method. PCA is a nonparametric classical method frequently used to extract a smaller representative data set from high-dimensional data while conserving its spatial characteristics. The mechanism of PCA lies in using orthogonal linear transformation to transfer a dataset to another coordinate system, In the new coordination, the first parameter has the greatest variance followed by the remaining coordinates, respectively. The number of coordinates is fixed based on the data variability.

However, Feed Forward Neural Network (FFNN) is not very effective for high dimensional space. Thus, the FFNN-based PCA produces non-negligible forecast errors. On the other hand, PCA presents a high sensitivity to the data distribution in the search space. Another study in [25] employed Echo State Networks with PCA decomposition for STLF application. While PCA achieves an acceptable results for correlated and accelerates the forecasting system, it has been noticed that the selection of the number of Principal Components (PC) requires a level of expertise to achieve the desired results. A wrong PC parameter causes a significant decrease in the performance on the model performance since some relevant features are omitted. Further, the interpretability of the features of PCA decomposition is relatively a difficult task.

2.3. Data Smoothing

To improve temporal cognition of data representation, the removal of the noisy samples is mandatory to boost the forecasting system accuracy by allowing important patterns to stand out. Furthermore, smoothing techniques are very important to track the seasonality of the data and thus improve the performance of ML techniques. Most of the commonly used techniques implement moving average and seasonal exponential smoothing due to their simplicity and good performance in time series forecasting [26]. For instance, the authors in [27] reviewed exponential smoothing techniques, specifically, single exponential smoothing, double exponential smoothing, holt's method, and adaptive response rate exponential smoothing. The cited methods are advantageous in terms of ease of interpretability and integration of the changing pattern effect of the data series into the prediction model. However, most of those approaches face problems with seasonal data and parameter tuning difficulties [28]. To overcome these problems, a Stavisky Golay (SG) filter is proposed in this paper to tackle the rapid variations from signals. In [29], the authors used an SG filter to remove the noise from phonocardiogram signals in the preprocessing stage. Then, an automatic classification CNN model receives the smoothed results to generate forecasts. In [30], an LSTM SG model was proposed to predict the Large-Scale Water Quality Time Series. However, the SG smoothing receives little attention in the field of smart grid applications despite its considerable importance. From our paper, an SG filter is implemented for the noise reduction of the load demand to further supplement the existing smoothing techniques for STLF in generating more accurate results.

3. The Proposed Methodology

The various parts of the hybrid SG-CBiLSTM, particularly, SG, PLS, and CBiLSTM have been presented. Furthermore, the workflow of the proposed technique is explicitly explained.

3.1. Bidirectional Lstm (BiLSTM) Model

The major success of LSTM model is accorded to its excellent ability in solving the vanishing problem of vanilla Recurrent Neural Network (RNN). The LSTM mechanism consists of the deployment of three memory gates, namely, input gate (i_t), forget (f_t) gate, and output gate (o_t). Therefore, LSTM network carries out long-term dependencies of sequential data with high efficiency. The mathematical equations of LSTM gates are given as follows [31]:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (2)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (3)$$

$$\tilde{c} = f_t \odot c_{t-1} \quad (4)$$

$$h_t = o_t \odot \tanh(c_t) \quad (5)$$

where x_t , σ , and c_t denote the input sample at time t , the sigmoid activation function, and the memory unit, respectively. (b_f, b_i, b_o) and (W_f, W_i, W_o) stands for the bias and weight matrix for each gate, respectively. The symbol \odot is the corresponding multiplication of the elements. Firstly, the h_{t-1} , c_{t-1} , and x_t pass the input information to the LSTM unit. Then, the LSTM gates interact with the inputs to generate an action based on a logic function. After passing by f_t , a new cell state c_t is built. x_t and h_{t-1} move to the forget layer to quantify the importance of the information between 0 and 1. At this stage, the f_t gate takes a decision whether if the information has to be stored, maintained, or removed. Then, the forget gate will update the cell state c_t with the new important information based on the proportion of the information occupied by the actual and the previous cell state. The final hidden layer of the LSTM is computed to obtain the remaining state value [31].

To enhance the learning capabilities of the traditional LSTM model, the temporal structure considers two-way relationship of the input data. In other words, instead of using one direction of input processing through the LSTM gates, the Bi-directional LSTM (BiLSTM) model considers the next information when dealing with the current time series data as shown in Figure 1.

This bidirectional processing enhances the information-wise fashion manner with a door mechanism by capturing more structural information. The BiLSTM model encodes the information on back to front. This leads to acquiring additional context information that leads to better generalization ability. First, the LSTM cells are forwarded from the input sequence. Then, the reverse form of the input sequence is integrated into the LSTM network. The output of the forward layer (h_t^f) and backward layer (h_t^b) of the BiLSTM model are generated as [32]:

$$h_t = \alpha h_t^f + \beta h_t^b \quad (6)$$

$$y_t = \sigma(h_t) \quad (7)$$

where α and σ are the numerical factors respecting the equality $\alpha + \sigma = 1$ [32]. The experimental results verify that the BiLSTM significantly improves the accuracy of the original LSTM using the spatial correlation mechanism [32].

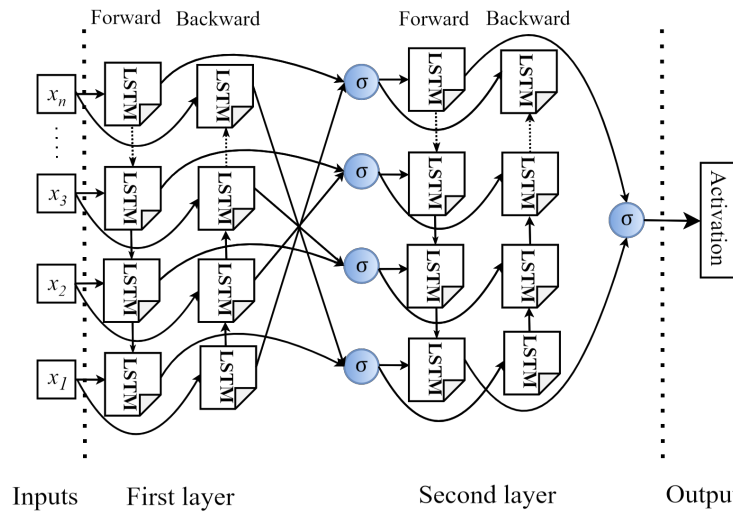


Figure 1. Bidirectional Long Short-Term Memory (BiLSTM) model structure with $\sigma(x) = \frac{1}{1+e^{-x}}$.

3.2. Convolutional Neural Network (CNN)

The CNN model is a DL network that maps local features from the input at higher layers and combines them into more complex features at lower layers. The general architecture of this model consists of convolutional layers, activation function, pooling layers, fully connected layer, and dropout layer. The convolutional layers apply a set of filters to the input to generate a set of feature maps. The convolutional layers are advantageous for preventing the model from overfitting, accelerating the training process, decreasing the input volume. The max-pooling layers present the commonly used type of pooling layers. The max-pooling consists of selecting the maximum value of the field covered by the pooling kernels with its parametric equation given as [33]:

$$P^{l(i,j)} = \max_{(j-1)W+1 \leq t \leq jW} \{a^{l(i,j)}\} \tag{8}$$

where $P^{l(i,j)}$ and $a^{l(i,j)}$ denote the width of the pooling layer and the activation value of t neuron. The activation function permits changing the linearly of an inseparable problem into a separable one to enhance the adaptability of the model. The Rectified Linear Unit (ReLU) function is used as an activation function do to its inherent ability to derive values between 0 and 1. The equation of ReLU function is given by [33]:

$$a^{l(i,j)} = f(y^{l(i,j)}) = \max\{0, y^{l(i,j)}\} \tag{9}$$

where $a^{l(i,j)}$ denotes the activation value of the layer output $y^{l(i,j)}$. The fully connected layers combine the previously extracted information to generate the final prediction as shown in the following function [33]:

$$z^{l+1(j)} = \sum_{i=1}^n w_{ij}^l a^{l(i)} + b_j^l \tag{10}$$

where w_{ij}^l , $a^{l(i)}$ and b_j^l denote the weight of the i th neuron, the i th neuron of the l layer, and the bias values. n is the length of the input data. Finally, the dropout function is employed as a regularization technique to avoid overfitting at the end of iterations by dropping randomly selected neurons during the training process. Much of the research work has emphasized CNN due to its better ability to approximate complex functions.

3.3. Convolutional Bi-Directional Lstm Network (CBiLSTM)

The CBiLSTM model consists of initial convolution layers that receive feature embeddings as input. Its output is pooled to a smaller dimension, then fed into BiLSTM layers. The association of the CNN layers is beneficial for its strong capabilities in local feature extraction. The proposed mechanism consists of extracting the inherent characteristics of the load demand by CNN model and extracting the temporal dependencies by BiLSTM layer. The motivation behind the fusion between CNN and BiLSTM is explained by (1) the excellent feature-extracting ability of CNN model in capturing short-term trends in the time series data. (2) the BiLSTM network is efficient in saving the temporal order between the data in both directions and avoiding the gradient disappearance. Owing to the association of CNN with BiLSTM, the pattern recognition for time series data is tracked spatially and temporally. Figure 2 presents a flowchart of CBiLSTM model.

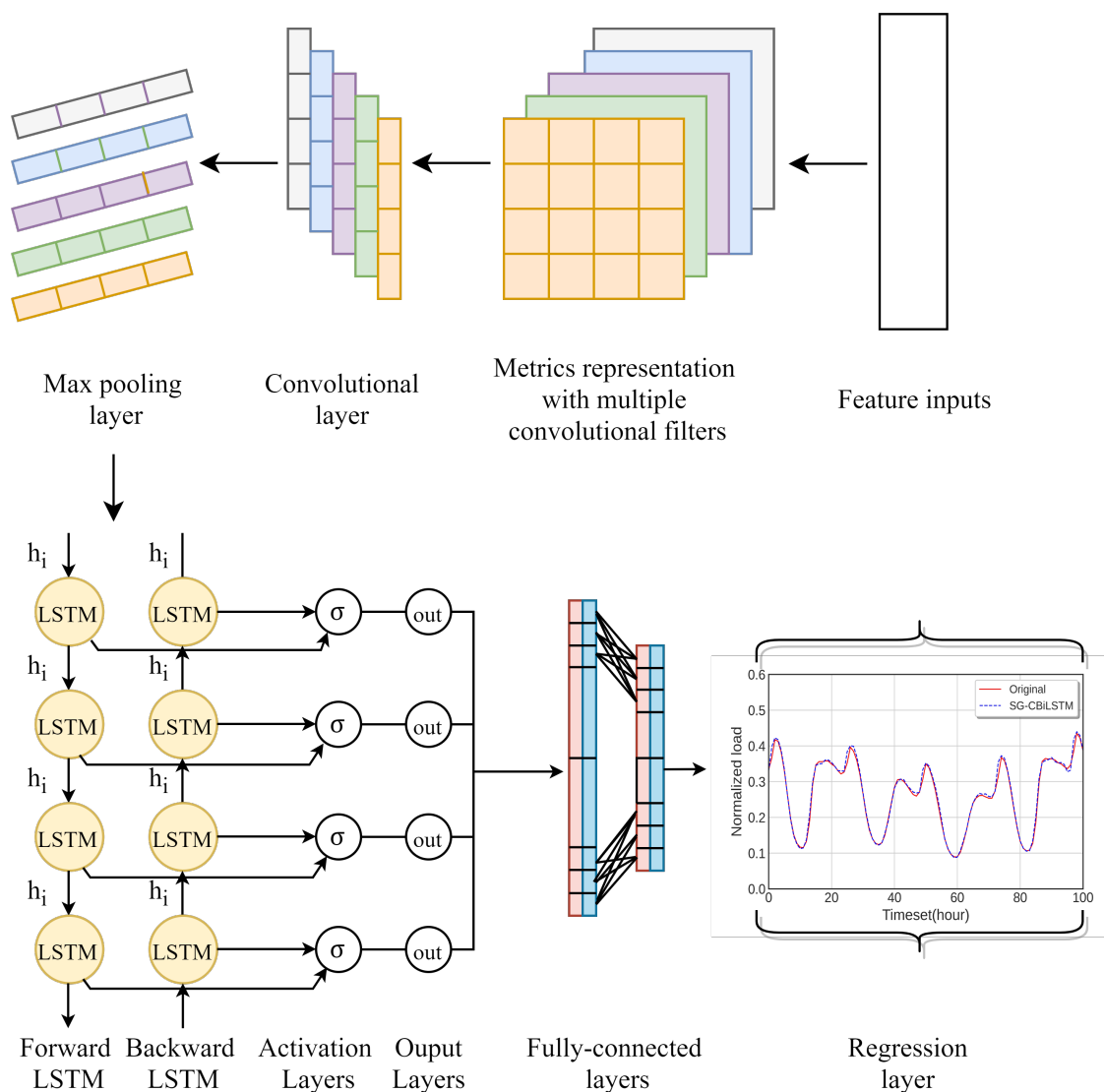


Figure 2. Flowchart of the proposed Convolutional neural network-Bidirectional Long Short-Term Memory (CBiLSTM) model.

3.4. Savitsky Golay Filter (SG)

The SG filter is a popular smoothing technique for removing the noise from signals of abruptly fluctuating and non-stationary data [34]. This technique lies in moving average the least square polynomial approximation with conserving the main characteristics of the signal (width, peak, and high). The core concept of SG consists of generating $2n + 1$ equidistant points (centered at $n = 0$) to represent a polynomial of degree p . Thus, SG filter uses three elements, specifically, the input signal, the order of the polynomial, and the frame size. These elements require a specific tuning to achieve the desired results. The polynomial order value attenuates the width of the signals. SG filter computes the value of the least square polynomial at point $i = 0$, over the entire sample space. Let $x(n)$ be a time series of n elements. We consider $y(n) = x(n) + w(n)$ as the observed time series where $w(n)$ denote the noise of $y(n)$ signal. The output of SG filter denoted by \hat{x} is equal [35]:

$$\hat{x}(n) = \sum_{k=-M}^M h(n)y(n - k) \tag{11}$$

where $h(n)$ and M denote the filter impulse response and the SG filter parameter, respectively. The polynomial $p(n)$ and the squared error between the smoothed and noisy signal E are given as follows [35]:

$$p(n) = \sum_{k=0}^N c_k n^k \tag{12}$$

$$E = \sum_{n=-M}^M (x(n) - p(n))^2 = \sum_{n=-M}^M \left(\sum_{k=0}^N c_k n^k - x[n] \right)^2 \tag{13}$$

where k denotes the polynomial order that ranges between 0 and $2M$ inclusive. $C(0)$ is the best fitting polynomial coefficient satisfying $p(0) = c(0)$ as [36]:

$$\frac{\partial E}{\partial c_i} = \sum_{n=-M}^M 2n^i \left(\sum_{k=0}^N c_k n^k - x[n] \right) = 0 \tag{14}$$

Analytically, the optimal coefficients c_k could be calculated by interchanging the order of the summations. In order to find the optimal polynomial coefficients, we differentiate E as follows [36]:

$$\sum_{n=0}^N \left(\sum_{n=-M}^M n^{i+k} c_k \right) = \sum_{n=-M}^M n^i x[n] \tag{15}$$

where $i \in (0, 1, \dots, N)$. Equation (15) could be reformulated as $(C^T C)c = C^T x$. $c = [c_0, c_1, \dots, c_n]^T$ denotes the polynomial coefficient vectors and $x = [x_{-M}, \dots, x_{-1}, x_0, x_1, \dots, x_M]^T$ denotes the input samples vector. Thus, the coefficient vector is conducted as $c = (A^T A)^{-1} A^T x$. Solving Equation (15) passes by setting the derivatives equal to zero [36]. Typically, this digital filter uses the technique of linear least squares for data smoothing, which helps to obtain a high signal-to-noise ratio and retains the original shape of the signal. The SG filter has been employed in this paper for two major merits: (1) the SG filter retains the width and height of waveform peaks in noisy load which allows the system to achieve higher performance; (2) the SG method transforms the load demand without destroying its fundamental properties.

3.5. Partial Least Square Method (PLS)

The PLS is a widely used dimensionality reduction technique. Unlike PCA dimension reduction method, which maximizes the variance-based objective function, PLS maximizes the covariance-objective

function [37]. The covariance-objective function maximization is conducted by the optimization of the square of the L_2 norm. Assuming we have two sets of dependent variables $Y = \{y_1, y_2, \dots, y_m\}$ and $X = \{x_1, x_2, \dots, x_n\}$ representing the input variables and the response variables, respectively. For the STLF application, we have s samples. So, X_0 and Y_0 could be written as:

$$X_0 = \begin{pmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{s,1} & \cdots & x_{s,n} \end{pmatrix}, Y_0 = \begin{pmatrix} y_1 \\ \vdots \\ y_s \end{pmatrix} \tag{16}$$

The PLS method consists of extracting two components (t_1, u_1) from X and Y , respectively. The vectors t_1 and u_1 acquire the variation information from the initial dataset where the correlation is set to its maximum. Thus, the regression of $y_k, k \in N$ and t_1 is conducted. In case the regression result provides the best accuracy, the PLS stop improving. Otherwise, the PLS algorithm passes to the next pair of components (t_r, u_r) . with r presents the pair number until the PLS algorithm achieves a satisfactory result. Mathematically, the first component could be written as [38]:

$$t_1 = w_1^T X \tag{17}$$

where w_1 is the eigenvector corresponding to the maximum eigenvalue of the matrix $X_0^T Y_0 Y_0^T X_0$. The component score vector \hat{t}_1 and the residual matrix X_1 are calculated as [38]:

$$\hat{t}_1 = X_0 w_1 \tag{18}$$

$$X_1 = X_0 - \hat{t}_1 \alpha_1^T, \alpha_1 = \frac{X_0^T \hat{t}_1}{\|\hat{t}_1\|^2} \tag{19}$$

This process is repeated until the construction of the r components for (X, Y) . The optimization of the objective function is based on the covariance $cov(Xw, Y)$ between the output vector Y and the input matrix Xw of a size $(N \times P)$. Therefore, the K th PLS component is identified by searching the weight vector w_k as [37]:

$$w_k = \underset{w'w=1}{argmax} cov(Xw, Y) = \underset{w'w=1}{argmax} cov((N - 1)^{-1} w' X' Y) \tag{20}$$

where the equation $Xw_k = w'_k X' X w_j = 0$ is valid for all $1 \leq j < k$. The PLS method is advantageous for its ease of implementation and unique biased solutions.

3.6. Evolution Strategy for Hyperparameter Optimization

Hyperparameter Optimization (HO) is one of the most difficult problems for DL models due to two major points: large number of parameters involved and expansive computational training. These points dramatically complicate testing all the possible values of the search space. In this paper, ES meta-heuristic method is considered the key solution to tackle the HO problem with higher efficiency. The ES is a meta-heuristic search technique inspired by the natural biological evolution. It consists of mutations and combinations of the best individuals (σ) of a certain population of solutions (λ) [39]. According to fitness values, natural selection empowers the best candidate solutions for mutation and reproduction [40]. These solutions dictate the distribution of future generations. Reciprocally, the weak candidates are removed from the population. The mechanism of the ES is presented in Figure 3.

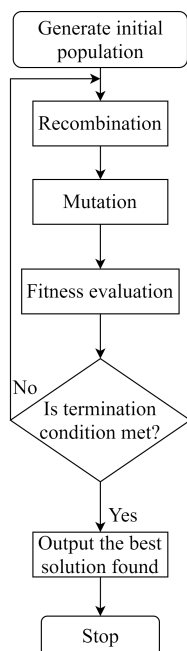


Figure 3. Flow chart of the evolution strategy algorithm.

Regarding the flowchart in Figure 3, the ES algorithm starts with a random population of σ parents. Successive recombination and mutation take place to compute the fitness values. An assessment of the fitness values to quantify the solution effectiveness. In case there is an area of improvement, the ES algorithm generates a λ children to improve the objective function. This process is done by the Gaussian distributed random modifications $G(x_i^P, \lambda_i^S)$, where x_i^P and λ_i^S are the mean and derivation parameters, respectively. $i(1 \leq i \leq n)$, $P(1 \leq i \leq \sigma)$, and $S(1 \leq S \leq \mu)$ are the parameter index predecessor index and successor index, respectively. The derivation parameter is calculated as follows [40]:

$$\sigma_i^S = \sigma_i^P \exp(G(0, \tau_1) + G(0, \tau_2)) \tag{21}$$

with $\tau_1 = 0.1$ and $\tau_2 = 0.2$ are the standard derivations [40]. The whole multiplicative σ^S mutation operation determine vector of strategy parameters $\sigma^S = (\sigma_1^S, \dots, \sigma_N^S)$ [41]. Each vector component of σ^S is mutated individually. The final σ^S is mutatively scaled in order to learn axes-parallel mutation ellipsoids [41]. The evolutionary dynamics could lead to undesirable fluctuations which require the removal to keep the high optimization performance of ES method [41]. The individual parameters are calculated as [40]:

$$x_i^S = G(x_i^P, \sigma_i^S) \tag{22}$$

From Equation (22), the individual parameters x_i are modified in order to create a λ solutions. The mutation operation of ES is conducted through the Gaussian distributed random modifications between x_i^P and σ_i^S . The best individuals are selected to be used as parents for the next iterations. The iterative process stops when the new individuals have minimal objective function value from the whole population. In this paper, the Evolution strategy is used as a key solution for the global optimization of the complex CBiLSTM black-box function. After fixing the search space, ES uses the batch-sequential process to find the most suitable hyperparameters of a specific model [42]. The ES presents a robust algorithm with a highly parallelizable architecture. Hence, the ES could face the risk of being trapped in a local minimum in a large search space environment.

3.7. Proposed Architecture

The proposed SG-CBiLSTM is a cupola of models defined in three major stages: data engineering and preprocessing, object determination, model construction and model evaluation as illustrated in Figure 4.

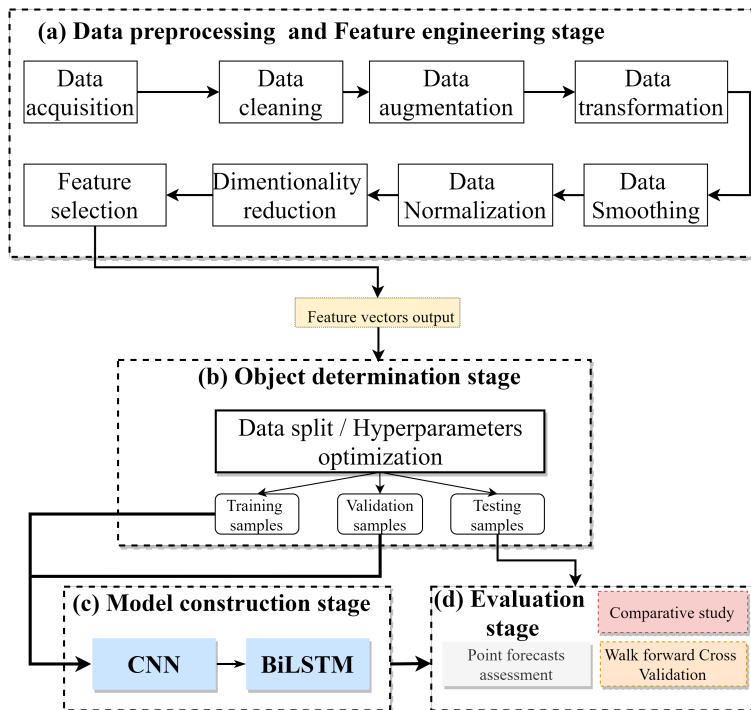


Figure 4. The adopted methodology for the proposed Savitzky–Golay (SG)-CBiLSTM model.

According to Figure 4, the proposed methodology is conducted using four main stages: (1) Data processing and feature engineering, object determination, model construction and evaluation stages, for further explanation of the proposed methodology, a complete flow chart is illustrated in Figure 5.

According to Figure 5, the proposed methodology is presented as follows: The data is acquired from an open-sourced portal and cleaned from misleading values in the data preprocessing and feature engineering stage. Then, a data augmentation strategy is adopted to enhance the significance of the data and add value to the learning ability of the whole system. Further, data transformation is conducted to convert all the data to its numeric form using on-hot encoder function. Next, data smoothing-based SG filter and data normalization are conducted to remove the noise from the load demand and standardize the data under a defined interval between 0 and 1. At this stage, SG filter is able to reject the noise efficiently along with the least distortion from the original load data. All the feature inputs are given probability values to assess their relevance to the load demand determination. Above a specific threshold, the features are kept and the rest are neglected. Afterwards, the object determination stage split the data into training, validation, and testing intervals. The hyperparameter tuning is mandatory for CBiLSTM model to generate more accurate results. The final stage is the evaluation of the model. In this stage, the point forecast assessment is conducted using score metrics. Moreover, K-fold cross validation technique is used in this paper as a reliable means to evaluate the proposed model accuracy to an unknown testing data [43]. K-fold cross validation schemes is widely used in regression problems [44,45]. The methodology of K-fold cross validation is illustrated in Figure 6.

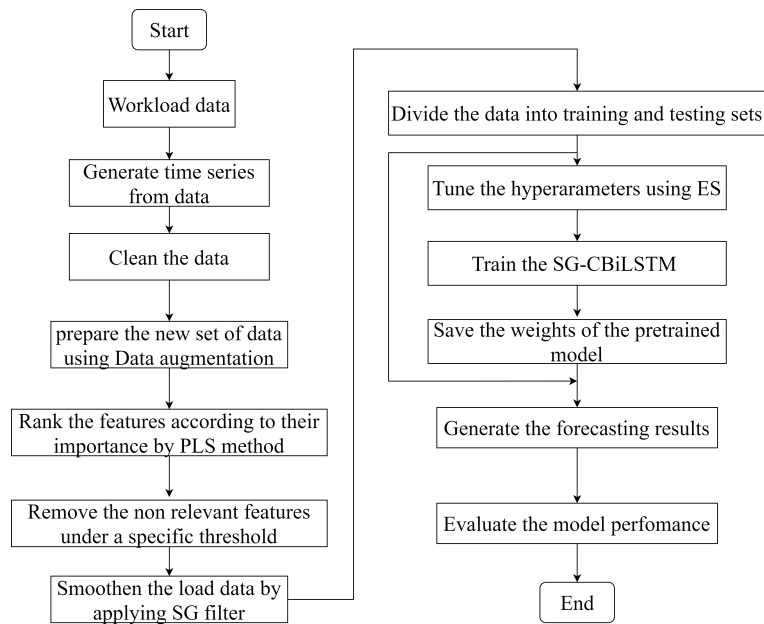


Figure 5. Flow chart of the proposed architecture.

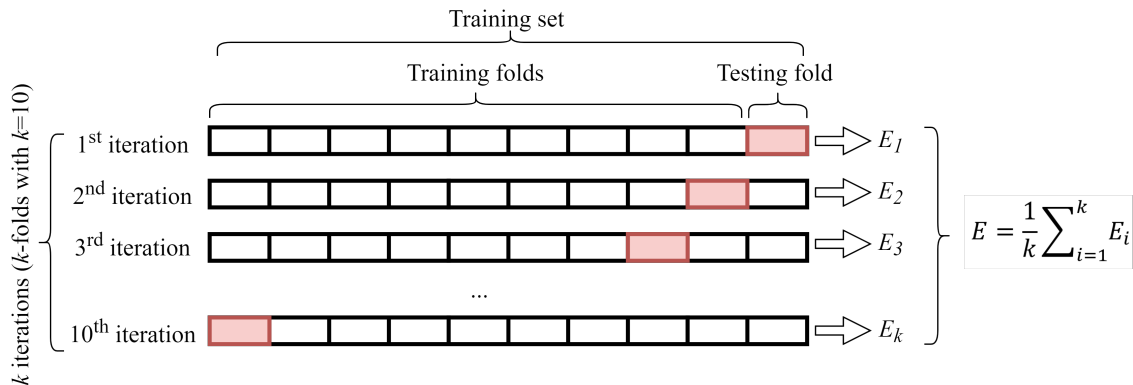


Figure 6. Ten-fold cross validation procedure where E denote the mean cross-validation error (E_i) of the i th iteration. So, the total effectiveness of the model can be evaluated.

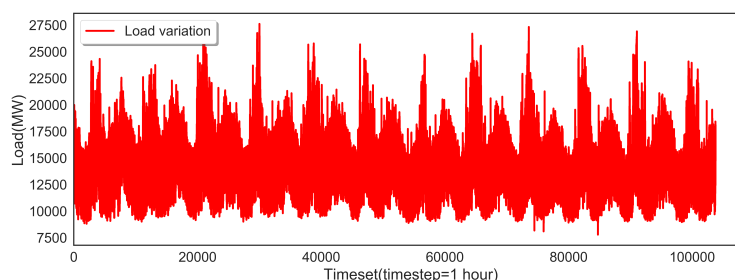
Regarding Figure 6, the procedure of K -fold cross validation is conducted as follows. Firstly, k equal sized sub-samples are partitioned from the original dataset. Ten-fold cross-validation (10-CV) is preferred to verify the model robustness with cross-validated data. The ten parts evaluate the model generalization to an independent set. For each 10-CV round, a random data separation into k folds $\{D_1, D_2, \dots, D_k\}$ is constructed. The proposed model is repeatedly trained by nine training set folds and tested against the remaining testing $k - 1$ set fold. The holdout method is repeated 10 times. Consequently, the bias and variance are remarkably reduced due to the use of the original data set in the fitting and validation procedure in the same time. For checking the model competitiveness, the proposed SG-CBiLSTM is compared to a bench of DL models to validate its performance superiority. The comprehensive model design and the implementation details of each part are given in the following section.

4. Simulation Results

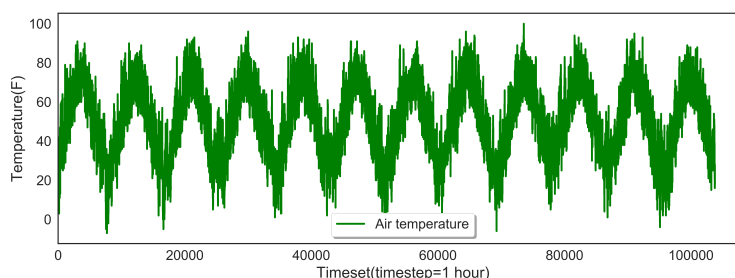
In this section, the validation of the proposed SG-CBiLSTM for STLF is demonstrated through extensive experimental studies with the an open source dataset. The evaluation procedure includes dataset description, model construction stages, score metrics, and benchmarks comparison analysis.

4.1. Data Pre-Processing and Feature Engineering

The experimental settings mainly uses the ISO New England incorporation (ISO-NE) dataset [46]. The ISO-NE is non-profit incorporation to manage day-to-day operations with more than 200 market participants in New England [46]. The publicly available data covers the hourly temperature and load data for the public and private electricity market of New England from 2003–2014 as plotted in Figure 7.



(a) Electric load (MW) between 2003 and 2014.



(b) Hourly temperature ($^{\circ}\text{C}$) between 2003 and 2014.

Figure 7. The overall system level load and temperature between 2003 and 2011 (103,753 samples) for the ISO-NE dataset.

The total data samples collected consists of 103,753 rows. The used data is cleaned from erroneous and Not a Number (NaN) samples. The cleaning stage is mandatory to feed the model with complete data samples and avoid any outliers or misleading values. Nevertheless, the hourly temperature for the ISO-NE data is the only indicator of the load demand. This data is insufficient for accurate prediction and may lead to underfitting. Therefore, data augmentation is conducted to identify and generate artificial features using Microsoft Excel software as shown in Table 1.

According to Table 1, the total features are the DateTime components (hour, day, month, year, workday, week number, weekend), season of the year, and hourly lagged load for the last 23 h. The Boolean features are converted to numerical values using hot encoder function. Then, the load demand is smoothed via SG filter to improve the pattern recognition of the forecasting system. The cleaned data is forwarded to the feature engineering phase. Four case scenarios were investigated according to the value of the polynomial function specifically 3, 5, 7, and 10. The load demand and its smoothed derivations are illustrated in Figure 8.

Table 1. Generated feature vectors.

Data Fold	Augmented Data	Formula	Unit/Range
Load forecast	Load($t - 1$), ..., Load($t - 23$)	$L_{(t-1)}, \dots, L_{(t-23)}$	MW
	Load demand	L_t	MW
Datetime	Year	$Y_t = \text{YEAR}(\text{DateTime})$	[2003, 2014]
	Month	$M_t = \text{MONTH}(\text{DateTime})$	[1, 12]
	Day	$D = \text{DAY}(\text{DateTime})$	[1, 31]
	Weekday	$W_d = \text{WORKDAY}(\text{Datetime})$	[1, 7]
	Hour	$H_t = \text{HOUR}(\text{DateTime})$	[1, 24]
	Season of the year	$S = \text{SEASON}(\text{Datetime})$	[0, 3]
	Week number	$W_n = \text{WEEKNUMBER}(\text{DateTime})$	[1, 52]
	Weekend	$W_{nd} = \text{IF}(W_d > 5 \text{ Then Yes , Else, No})$	Boolean
Weather data	Temperature	T_t	°C

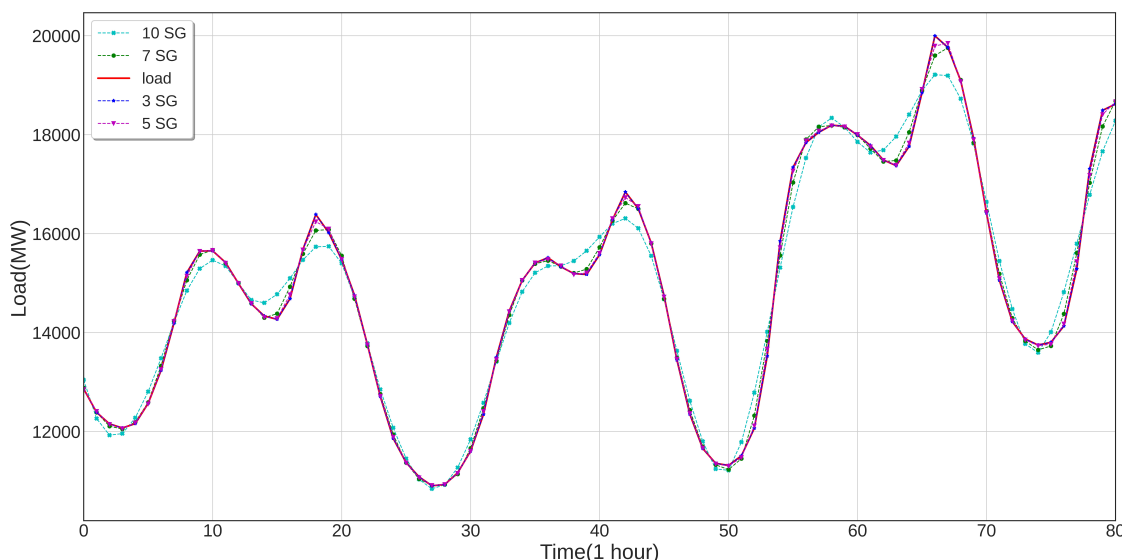


Figure 8. Comparative results of SG-CBiLSTM model with different polynomial functions.

According to Figure 8, the increase of the probability values (P -values) results in a more smoothed signal. The generated signals are assessed together in the same conditions with the aim select the optimum SG polynomial value based on the less error produced. Further, the data normalization is conducted using MinMaxScaler function form Scikit-Learn library [47]. The normalized data from 0 to 1 to keep the values within limits of the activation function and to reduce the influence of different units on the forecasting model.

The selection of the most informative features in the dataset is an essential step of the forecasting system performance. This technique is exploited the P -values for each feature to map the domain representation, and assess the weights for each feature input. Therefore, the irrelevant and redundant inputs are discarded from the feature vectors. These artificial features are associated in tandem with the temperature to make a prediction. The data augmentation technique is commonly used for low dimensional systems to track the seasonality and trends of the target variation and acquire more information about domain knowledge. The ranking of the feature is essential at this stage to make the system more efficient with less computational time due to the high number of features. The PLS technique is used to reduce the dimension of the data. PLS is a straightforward dimensionality reduction

technique that maps the variables in a new feature space with lower dimensions. The Variable Importance of load Patterns (VIP) for 32 features is shown in Figure 9.

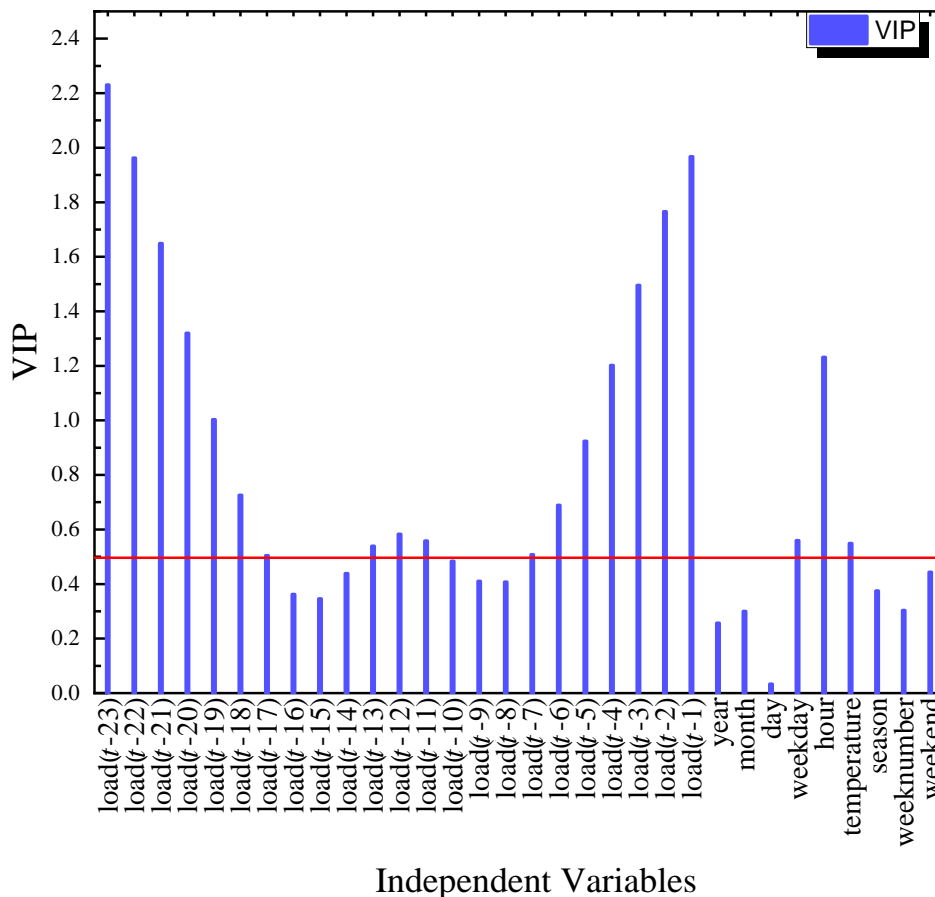


Figure 9. Variable importance of load patterns with a threshold value (red line) equal to Variable Importance of load Patterns (VIP) = 0.5.

Regarding Figure 9, the most important features are hour, workday, temperature and lagged load ($t - x$) with $x \in [1, 2, 3, 4, 5, 6, 7, 11, 12, 13, 17, 18, 19, 20, 21, 22, 23]$. Thus, the selected threshold is $VIP = 0.5$. The rest of the simulation results adopted a feature representation including a 20 feature inputs. The data is split into three sets: training, validation, and testing sets with a proportion of 75%, 15%, and 10%, respectively. More precisely, the training, validation, and testing instances comprise 77,814, 15,562, and 10,376 samples, respectively.

4.2. Score Metrics

The assessment of the model performance is conducted using Percentage Error Measure (PEM). The PEM method is a percentage of the error regardless of the units of the measurements (scale-independent). In order to standardize the SDEM measures, particularly, the Rooted Mean Squared Error (RMSE) and Mean Absolute Error (MAE) measures, the Min-Max normalization method was kept for the evaluation process. In other words, the data is normalized with a magnitude range of [0, 1]. The data normalization is calculated as follows:

$$x(t) = \frac{x_r - x_{min}}{x_{max} - x_{min}} \tag{23}$$

where x_t denote the normalized value, x_r is the real value. Here, x_{min} and x_{max} are the minimum and maximum values. The error measures in this work include the coefficient of determination (R^2), normalized RMSE (nRMSE), normalized MAE (nMAE), Mean Squared Logarithmic Error (MSLE) and MAPE. The selection of the aforementioned scores is based on their popularity for regression-based techniques evaluation. The parametric equations of the score metrics are given as follows [48]:

$$nMAE = \frac{1}{N} \sum_{i=0}^{N-1} |y_i - \hat{y}_i| \tag{24}$$

$$nRMSE = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2} \tag{25}$$

$$MAPE = \frac{100(\%)}{N} \sum_{i=0}^{N-1} \frac{|\hat{y}_i - y_i|}{y_i} \tag{26}$$

$$R^2 = 1 - \frac{\sum_{i=0}^{N-1} (\hat{y}_i - y_i)^2}{\sum_{i=0}^{N-1} (\bar{y}_i - y_i)^2}, \bar{y} = \sum_{i=0}^{N-1} y_i \tag{27}$$

$$MSLE = \frac{1}{N} \sum_{i=0}^{N-1} (\log_e(y_i + 1) - \log_e(\hat{y}_i + 1))^2 \tag{28}$$

where \hat{y}_i and y_i present the i th forecasted values and the actual values. Here N denotes the total number of samples.

4.3. Model Construction and Parameter Settings

In this part, the assessment of the proposed SG-CBiLSTM model is reported. The simulation results were conducted using PYTHON programming and Keras library [49]. Keras is a high-level framework specialized in DL. All simulations were run on a Lenovo Intel®i7®Nvidia Geforce GTX 1650@ 2.30 GHz. The characteristics of the experiments in this study are resumed in Table 2.

Table 2. Experimental environment.

Experimental Environment	Proprieties
Operating system	Windows 10
Processor	Intel Core i7 9th Gen
Graphics card	NVIDIA GeForce GTX 1650
Programming language	Python 3.6.7
Deep learning framework	Keras 2.4.3
Platform	Tensorflow 2.3.0
Supplementary tools	Sklearn 0.23.2

The simulation analysis-based on trial and error method is firstly adopted to limit the search space as much as possible for the optimization process. This method is conducted by selecting arbitrary extreme values of BiLSTM and CNN layers and measuring the accuracy of the forecasting engine in terms of R^2 score. Next, a new random values are defined to further decrease the possibilities of hyperparameters values. The final results of trial and error method shows that three convolutional layers with a kernel size equal to 1×1 associated with three successive BiLSTM layers achieve better results according the highest the R^2 criteria. For the rest of optimization work, ES is employed to tune the critical hyperparameters of the proposed computing framework, specifically, the batch size, the number of units, the number

of training epochs [49,50]. To tune these hyperparameters, Hyperactive library is employed on open source [39]. It worth noting that the search space of the ES method only considers the key hyperparameters of the proposed C-BiLSTM to reduce the number of combinations and thus accelerate the computational time. In the model construction stage, the search space of the proposed C-BiLSTM model is resumed in Table 3.

Table 3. The search space and Evolution Strategy (ES) results of the proposed SG-CBiLSTM.

Model	Configuration/Neurons/Filters	Search Space	ES
CNN	CNN Layer 1	[32, 64, 72, 128]	32
	CNN layer 2	[32, 64, 128]	64
	CNN layer 3	[32, 64, 128]	64
BiLSTM	LSTM layer 1	[100, 150, 250]	150
	LSTM layer 2	[32,50, 150, 250]	50
	LSTM layer 3	[32, 50, 150, 250]	32
	Batch size	[32, 64, 72, 256, 512]	512
	Iterations number	[20, 30, 70, 100]	100
Time (min)			131.64
Time/iteration (min)			8.77

The best hyperparameters are selected as follows. Firstly, the population of ES is randomly initialized. Each candidate solution represents a possible configuration of the optimized C-BiLSTM model. From the data training, the C-BiLSTM model is trained with a specific structure and parameter sittings included in Table 3. Finally, the testing set is used to evaluate each population performance based on MAPE. Regarding the ES performance in Table 3, the ES algorithm is characterised by a fast convergence speed with 8.77 min for every iteration. From Table 3, the input layer consists of a 3 successive one-dimensional Convolutional layer (Conv1D), with 32, 64 and 64 filters of the convolution, respectively. The Conv1D layers are associated with a ReLU activation function. The ReLU function is employed to solve the vanishing gradient with fast convergence speed. The convolution layers are followed by a maximum one-dimensional sampling (MaxPooling1D) layer, three BiLSTM hidden layers, a dropout layer, and a dense layer. The one-dimensional CNN layer is mainly adopted for the sequential data processing of the load demand. The sampling size of MaxPooling1D is 1. The number of features of the BiLSTM hidden layers is 150, 50, and 32, respectively. The proposed C-BiLSTM takes advantage of the nonlinear fitting ability form BiLSTM and the feature extraction ability from CNN. The final structure of the SG-CBiLSTM model is resumed in Table 4.

Table 4. Layer description of C-BiLSTM.

Layer	Configuration/Neurons/Filters	Activation Function
Input layer	Sequential	-
Convolution 1-D	32, $k1 \times k1$	ReLU
Convolution 1-D	64, $k1 \times k1$	ReLU
Convolution 1-D	64, $k1 \times k1$	ReLU
MaxPooling 1-D	$k1 \times k1$	ReLU
Flatten	64	-
BiLSTM	150	ReLU
BiLSTM	50	ELU
BiLSTM	32	ELU
Fully connected	1	Softmax

The general architecture of reference models is fixed through repeated training. While, the units' number and the batch size and the activation function is selected using the ES method. The settings of reference models are given in Table 5.

Table 5. Hyperparameter settings of nine methods.

Model	Hyperparameter Settings
SG-CBiLSTM	The convolution layers are 3 with a number of filters of 32, 64, and 64, respectively; the activation function is ReLU; the BiLSTM layers are 3 with a number of neurons 150, 50, and 32, respectively; the batch size is set to 512; the dropout function between layers is fixed at 0.5.
BiGRU	The layers number is 4 with a neurons number of 128, 32, and 4 ended by a fully connected layer, respectively; the dropout function between layers is fixed at 0.4, The optimizer function is Adam; the activation function of the BiGRU layers is ReLU; the batch size is set to 512
BiLSTM	The layers number is 7 with a neurons number of 256, 128, 64, 32, 16, 4 finished by a fully connected layer, respectively; the dropout function between layers is fixed at 0.2, The optimizer is Adam; the activation function of the BiLSTM layers is ReLU; the batch size is set to 512.
CBiLSTM	The convolution layers are 3 with a number of filters of 32, 64, and 64; the activation function is ReLU; the BiLSTM layers are 3 with a number of neurons 150, 50, and 32, respectively; the batch size is set to 512; the dropout function between layers is fixed at 0.5.
BiLSTM-AE	The biLSTM layers are three with a number of neurons of 500, 400, and 300, respectively; the repeat vector layers are two between the located between the BiLSTM layers; the optimizer function is Adam.
ConvLSTM-AE	The convLSTM2D is the initial layer with a filters number of 64; the kernel size is with a dimation of 1×1 ; the activation function is ReLU; the two next layers are a flatten layers and repeat vector layers; the LSTM layers has a number of neurons equal to 200; the number of time distributed layers is equal to 2; the optimization function is Adam optimizer.
CNN	The Conv1D layer has a filter units of 64; The kernel size is 2×2 ; The activation function is ReLU; The max pooling layer has a pooling size of 2; The third layer is a flatten layer; The fully connected layers are two with a 50 units and 1 units; the optimizer function is Adam.
RNN	The RNN layers are two with a number of neurons of 500 and 100, respectively; the dropout function between layers is fixed at 0.2; the Adam optimizer function is used; the fully connected layer uses a tanh activation function.
ELM	The neuron number of the hidden layer is 560; the activation function is tanh; the alpha is set to 0.5.

4.4. Experimental Results

The SG-CBiLSTM evaluation is conducted in this section. The empirical results at this level are firstly conducted by presenting the performance of the proposed framework individually and with several case studies of different SG filter characteristics. Second, a comparative study is computed and presented where the simulation settings are unified for the sake of fair assessment. The conclusive remarks and interpretations are based on the actual results simulated in this work. During the validation experiments of this paper, the selection of the most appropriate Polynomial (P) degree is based on the hit-and-trial method of 4 p values. For the consistency and coherence of this section, the validation of our proposed SG-CBiLSTM model is firstly conducted compared to the original CBiLSTM using the proposed framework explained in the previous section. Secondly, the supremacy of the proposed method was verified compared to different benchmarks. In order to yield credible results, each case scenario is performed 10 times and the results are averaged. The simulation results as shown in Figure 10.

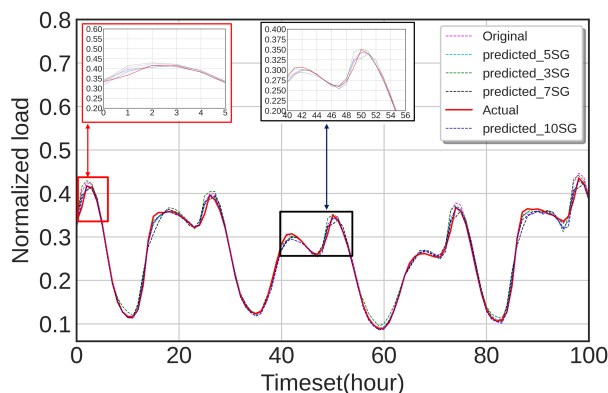


Figure 10. Performance comparison of SG-CBiLSTM model with a variety of polynomial filters $p = \{3,5,7,10\}$.

According to Figure 10, the proposed SG-CBiLSTM is generating a good result and follows the actual load demand with high precision. Further, the SG filter demonstrated its ability to improve the performance of the forecasting system. As can be seen, the proposed model performed best for $p = 10$ where the gap between the prediction curve and the original curve is the smallest compared to the original CBiLSTM and the other models for different p values. However, the polynomial degree variates the errors from a point to another. It could be concluded that the higher the signal is smoothed the better the model will be performing. A quantitative comparison of the proposed SG-CBiLSTM model performance under different p values is given in Table 6.

Table 6. Performance variation of CBiLSTM and different SG filters.

Models	MSLE	MAPE(%)	nRMSE(%)
3 SG-CBiLSTM	9.97	59.85	1.36
5 SG-CBiLSTM	5.82	60.14	1.04
7 SG-CBiLSTM	6.87	59.51	1.13
10 SG-CBiLSTM	4.92	56.66	0.93
Original CBiLSTM	8.38	61.17	1.23

According to Table 6, the best performance has been attributed to 10 SG-CBiLSTM which achieves the highest results. The registered MSLE and MAPE for 10 SG-BiLSTM is equal to 0.93% and 56.66%.

Compared to 3SG-CbiLSTM which reaches to RMSE = 1.36% and MAPE = 59.85%. This leads to conclude that the SG filters are a key element for achieving higher performance. Furthermore, the selection of the most suitable parameters is mandatory to achieve the desired results. For the rest of the work, it will be considered the 10 SG-CBiLSTM as a reference for the proposed forecasting system. For notation simplicity and flow of reading, the 10 polynomial SG-CBiLSTM is referred to as the SG-CBiLSTM hereinafter. The loss function is the training and validation of the model simulation is an essential criterion to follow the learning accumulation in the model construction. The parametric equation of the loss function is calculated as follows [51]:

$$Loss(\hat{y}, y) = \sqrt{\frac{1}{B} \frac{1}{O} \sum_{i=1}^B \sum_{j=1}^O (\hat{y} - y)^2} \tag{29}$$

where B, O are the output matrix size. Here, the proposed model is trained with a training set stating from 2 March 2003 with the first 75% portion of the whole set. In order to track the loss values when the proposed SG-BiLSTM fetches the next batch of data. it has been used the MAPE as a loss function with an Adam optimizer [52]. The training process was fixed at 100 epochs with an early stopping function. This function is adopted to automatically track the best-trained model without searching for the most suitable epoch number. Figure 11 displays the variation of the training loss of the first 40 epochs for the training and testing sets.

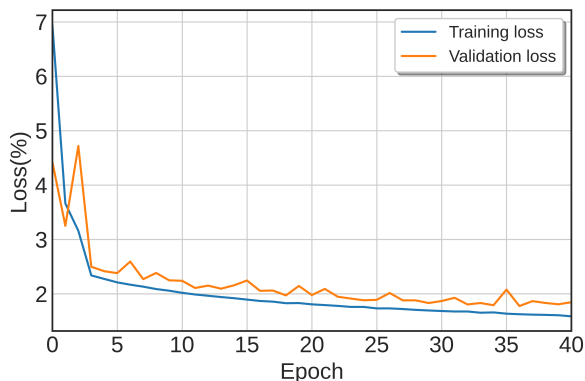


Figure 11. Training and testing loss variation for epochs.

As seen in Figure 11, the training loss value starts at 6.93% and continuously decreases until the 10th epoch starts to stabilize. At this epoch, the loss value reaches to 2.08%. At this level, the training loss value decreases slowly to achieve 1.59% at the 40th epoch. On the other side, the validation loss started at 4.415% to reach 2.407% in the 10th epoch where it decreases slowly aside with the training loss. This indicates the training and validation loss reaches their equilibrium at the same iteration. Moreover, it is verified that the convergence speed is quite fast. The training and validation loss are following the same behavior. The simulation results of the proposed method in tandem with its single components (CNN, BiLSTM, and CbiLSTM) with 10-CV have been computed and shown in Table 7.

Table 7. The error measures of 10-fold cross validation. The best result is marked in bold.

Model		BiLSTM						CNN					
Fold	R^2 (%)		nMAE (%)		nRMSE (%)		R^2 (%)		nMAE (%)		nRMSE (%)		
Sets	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	
1	95.47	95.23	3.11	3.22	2.07	2.13	90.56	90.62	3.11	3.22	3.21	3.21	
2	95.40	95.82	3.14	2.99	2.08	2.02	86.38	87.03	3.14	2.99	3.93	3.88	
3	95.67	95.89	3.05	2.95	1.99	1.98	89.67	89.76	3.05	2.95	3.34	3.31	
4	95.54	95.19	3.09	3.19	2.06	2.12	88.58	88.10	3.09	3.19	3.55	3.59	
5	95.34	95.23	3.16	3.20	2.12	2.12	89.27	88.86	3.16	3.20	3.47	3.51	
6	95.58	95.24	3.07	3.22	2.02	2.10	89.57	89.66	3.07	3.22	3.40	3.42	
7	95.31	95.47	3.17	3.11	2.12	2.10	90.10	90.11	3.17	3.11	3.24	3.25	
8	95.24	95.17	3.19	3.24	2.13	2.15	90.51	90.10	3.19	3.24	3.30	3.42	
9	95.51	95.43	3.11	3.10	2.08	2.09	90.27	90.24	3.11	3.10	3.26	3.26	
10	95.37	95.36	3.15	3.14	2.14	2.18	89.77	89.79	3.15	3.14	3.30	3.30	
Mean	95.45	95.41	3.13	3.14	2.09	2.10	89.47	89.43	3.13	3.14	3.41	3.42	
SD	0.13	0.25	0.04	0.10	0.05	0.06	1.18	1.05	0.04	0.10	0.20	0.19	
Time(s)	3238.38						3643.24						
Model		CBiLSTM						SG-CBiLSTM					
Fold	R^2 (%)		nMAE (%)		nRMSE (%)		R^2 (%)		nMAE (%)		nRMSE(%)		
Sets	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	
1	88.54	87.79	4.96	5.16	2.84	2.95	99.18	99.05	0.78	1.43	0.89	0.90	
2	88.11	88.46	5.05	4.97	3.00	3.96	99.27	99.09	0.73	1.38	0.65	0.66	
3	85.84	85.52	5.52	5.54	3.17	3.18	99.02	99.28	0.91	1.22	0.70	0.71	
4	87.63	87.79	5.16	5.09	2.92	2.90	99.12	99.05	0.77	1.43	0.70	0.70	
5	86.55	85.84	5.37	5.53	3.12	3.18	99.22	98.97	0.79	1.48	0.62	0.64	
6	86.55	85.02	5.71	5.73	3.24	3.29	99.16	99.33	0.73	1.20	0.68	0.69	
7	84.80	89.16	4.76	4.81	2.74	2.77	99.21	99.16	0.78	1.33	0.73	0.73	
8	89.47	86.9	5.23	5.34	2.95	3.04	99.13	99.29	0.73	1.23	0.67	0.67	
9	87.26	84.91	5.68	5.64	3.31	3.33	99.29	99.02	0.76	1.45	0.74	0.75	
10	85.13	85.49	5.65	5.56	3.26	3.26	99.37	99.55	0.64	0.98	0.70	0.70	
Mean	86.84	86.96	5.31	5.34	3.06	3.09	99.20	99.18	0.77	1.32	0.71	0.72	
SD	1.54	1.45	0.31	0.29	0.18	0.18	0.09	0.17	0.07	0.15	0.07	0.07	
Time(s)	2416.56						1808.4						

According to Table 7, the 10-CV results clearly demonstrate that our proposed framework significantly outperform single models in terms of the error measures. The SG-CBiLSTM achieves a high mean testing result of $R^2 = 99.18\%$ compared to 86.96%, 89.43%, and 95.41% for CBiLSTM, CNN, BiLSTM, respectively. The overall increase of the performance for the DL models reported in Table 7 is explained by adopting the data augmentation strategy for the original dataset. Further, employing the S-G filter with the 10th polynomial fitting empowers the CBiLSTM the accuracy with a 12.22% in terms of R^2 . An illustration of the 10-CV results in terms of R^2 is shown in Figure 12 to highlight the performance improvement of the SG-BiLSTM model.

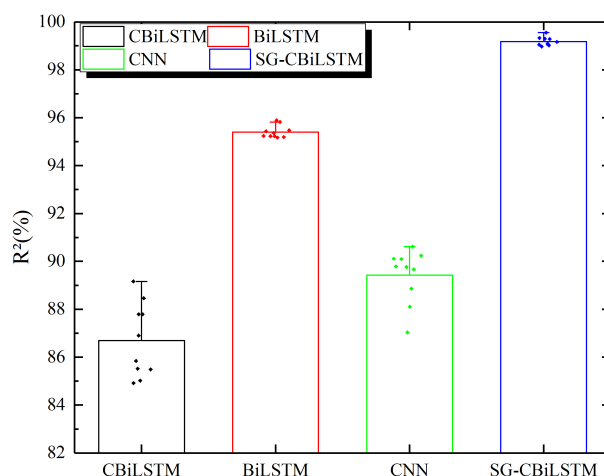


Figure 12. Box plot of ten-fold cross-validation (CV) of four models in terms of $R^2(\%)$.

According to Figure 12, the SG-BiLSTM model clearly outperforms the single models in terms of R^2 . The SG smoothing technique helps to better see patterns and detects trends. According to the CV variation results, the proposed model is characterized by a high location independency. The achieved results indicate that the SG-BiLSTM could be applied for STLF at different weather and load conditions. The experimental results of the SG-BiLSTM compared to the original load values are shown in Figure 13.

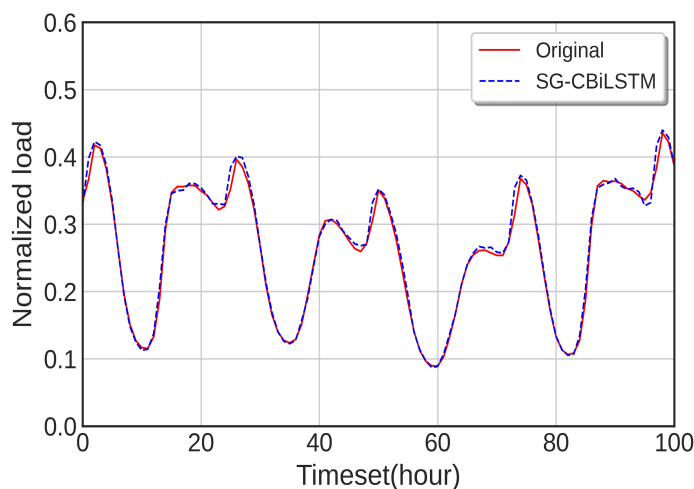


Figure 13. Hourly predictions of the forecasting system.

According to Figure 13, the proposed system achieves high accuracy and follows the real curve with great precision. The proposed model is compared to nine other DL models trained at the same conditions for the integrity of the evaluation. The reference models assessed here include Extreme Learning Machines (ELM), BiLSTM, BiGRU, ConvLSTM, CNN, BiLSTM-Autoencoders (BiLSTM-AE), RNN, CNN, CBiLSTM to verify its competitiveness with the existing benchmarks. It should be noted that the used dataset and extracted features from the proposed model construction process are conserved for the comparison of all the models and the rest of the validation analysis. The comparative simulation results is shown in Figure 14.

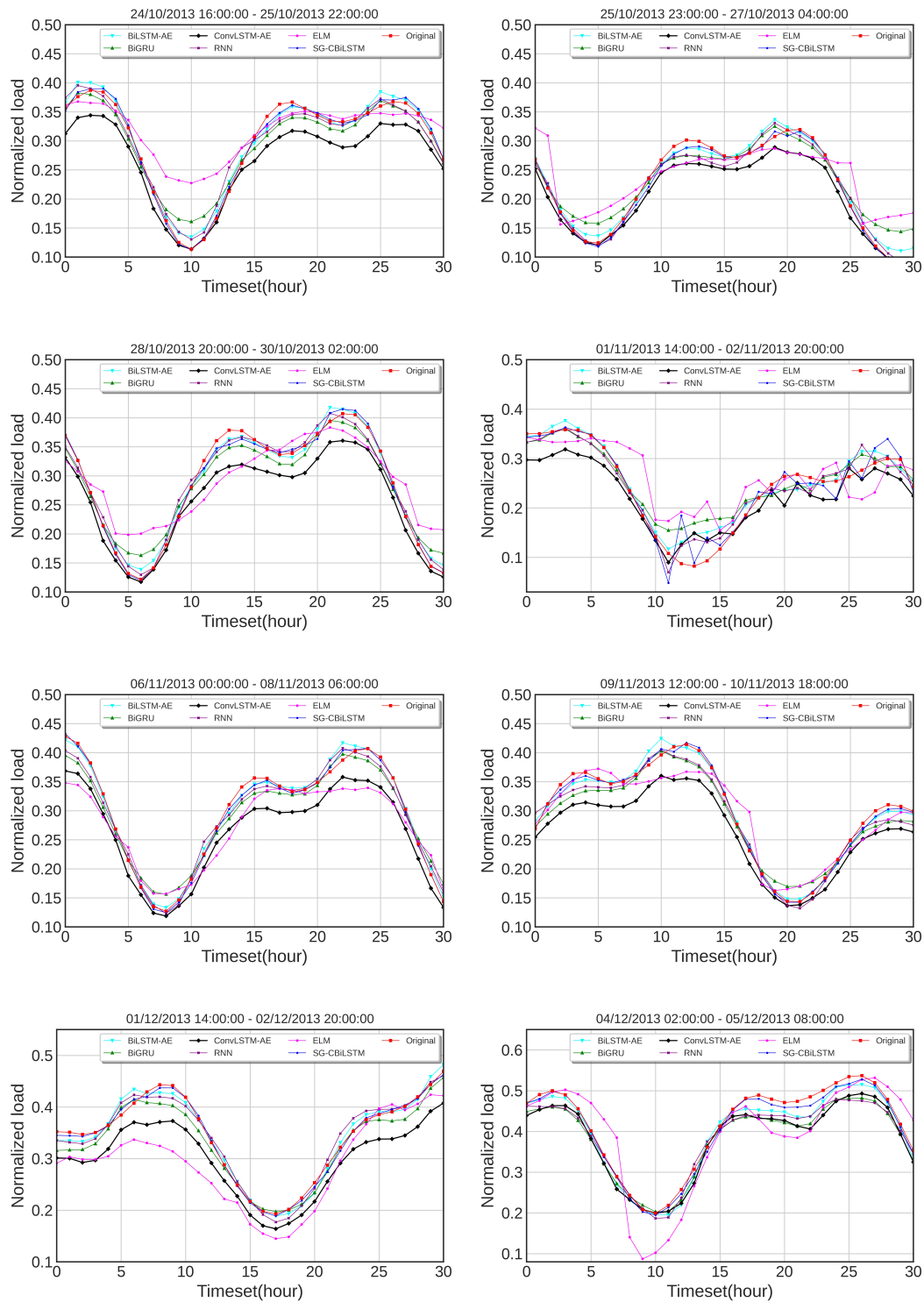


Figure 14. Multiple tests of different Deep Learning (DL) models using variant timesets. Despite the high competitiveness of BiLSTM-autoencoders (AE) and the proposed model for their close performance, the proposed SG-CBiLSTM model outperforms BiLSTM-AE model in most cases.

Regarding Figure 14, the experimental results clearly show that the proposed SG-CBiLSTM outperforms the rest of the models for hourly STLTF due to the low gap between the actual and generated forecasts. In order to quantify the quality of the forecasts, Table 8 presents the score errors of the proposed SG-CBiLSTM and reference models.

Table 8. Comparison of the proposed SG-CBiLSTM with the reference models on the ISO New England (ISO-NE) dataset.

Mean Values of 10 Times	nRMSE (%) \pm SD	nMAE (%) \pm SD	R^2 (%) \pm SD
SG-CBiLSTM	1.22 \pm 0.20	0.89 \pm 0.23	99.22 \pm 0.29
BiGRU	2.71 \pm 0.27	2.94 \pm 0.37	95.93 \pm 1.10
BiLSTM-AE	1.76 \pm 0.11	1.16 \pm 0.05	98.56 \pm 0.18
ConvLSTM-AE	4.13 \pm 0.31	3.13 \pm 0.11	85.21 \pm 0.23
RNN	3.77 \pm 0.32	2.82 \pm 0.25	93.35 \pm 1.06
ELM	5.86 \pm 0.30	4.45 \pm 0.24	83.98 \pm 1.74

According to Table 8, the SG-CBiLSTM hybrid model is performing best comparing to the rest of his counterparts on the scale of the nRMSE, nMAE, MAPE, and R^2 performance measures. The R^2 reaches 99.22% due to SG filtering. Furthermore, the model performance was slightly close to BiLSTM-AE model which achieves $R^2 = 98.56\%$. Due to the complex nonlinear relationship between the load demand and its drivers, the ELM model achieves the lowest forecasting performance accuracy with an $R^2 = 83.98\%$. Moreover, it is evident that the proposed model is preferable compared to numerous DL baseline and top-of-the-line models. The proposed SG-CBiLSTM is found more efficient to enhance the profitability form the electrical load operations.

4.5. Multi-Step Validation

In this study, the model performance is assessed for multi-step forecasting using multiple case scenarios. The multi-step validation is crucial to evaluate the model robustness under an extended time horizon. The investigated forecasting horizons include one hour, twelve hours, twenty four hours, and thirty-six hours ahead. In the simulations results, we conserve the same repartition of the data set with 75%, 10%, and 24% for the training, validation, and testing sets of the whole data. For multi-step-ahead forecasting, the structure of the proposed SG-CBiLSTM model conserves the shape of CNN layers compared to one-step forecasting. For the on-step-ahead, the n forecasted outputs $\hat{x}_{i+1} = f(x_i, x_{i-1}, \dots, x_{i-n})$ takes into consideration the real values of the load data. However, in multi-step-ahead forecasting, the previously generated forecasts are fed as inputs as $\hat{x}_{i+2} = f(\hat{x}_{i+1}, x_i, x_{i-1}, \dots, x_{i-n})$. The experimental results of multi-step forecasting for ISO-NE data were repeated ten times and the results are averaged in Table 9. Furthermore, the error between the predicted and actual load demand according to the forecasting horizon are shown in Figure 15.

Table 9. Results of Multistep forecasting on ISO-NE dataset.

Mean Values of 10 Times	One-Step	Twelve-Step	Twenty Four-Step	Thirty-Six-Step
nRMSE (%)	0.54	3.3	2.26	3.59
nMAE (%)	1.23	2.14	1.34	2.02
R^2 (%)	99.11	87.39	94.26	87.22

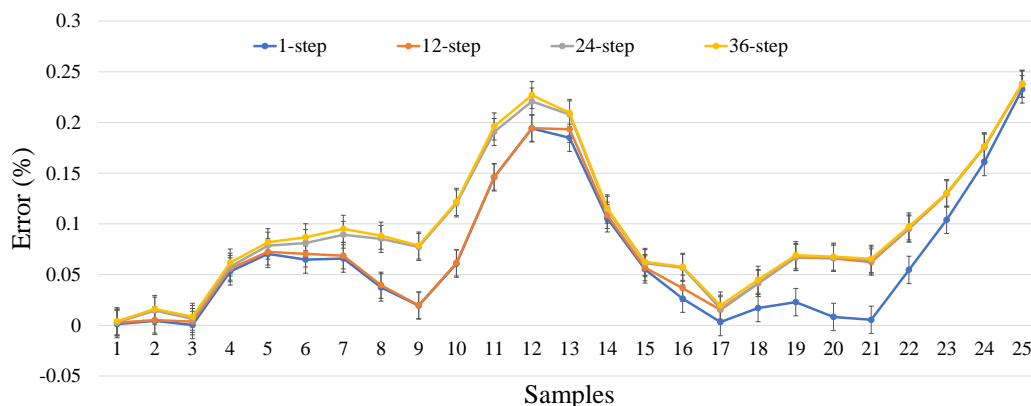


Figure 15. Multi-step Short-Term Load Forecasting (STLF) error variation with the Error (%) = $100 \times (\hat{y}_{norm} - y_{norm})$.

According to Table 9, it can be deduced that the most suitable scale is the hourly LF. Considering the scores indicators, the achieved results for one hour ahead reports an $R^2 = 99.11\%$ and $nRMSE = 0.54\%$ compared to $R^2 = 87.22\%$ and $nRMSE = 3.59\%$ for 36 h ahead. Thus, the lowest error measure is produced by single-step forecasting and the highest is generated by thirty-six step forecasting ahead as shown in Figure 15. In order to increase the model effectiveness, a state-of-the-art optimization approaches such as Coyote Optimization Algorithm (COA) and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) could be employed as alternatives to ES for hyper parameter optimization procedure [53,54]. CMA-ES is an enhanced version of ES where the search individuals are generated according to multivariate normal random distribution. Consequently, the corresponding covariance matrix is updated and assessed in each iterations [53]. Similarly to ES, COA is a population-based algorithm inspired by the coyotes adaptation the environmental conditions [54]. In our work, the lowest performance is attributed to a sequence of thirty-six-steps ahead with a score $R^2 = 87.22\%$. This demonstrates the relative weak stability of the SG-CBiLSTM for longer forecasting horizon which needs further investigations.

5. Conclusions

The major contribution of this paper is to propose an end-to-end forecasting framework called SG-CBiLSTM model for hourly STLF. The proposed system implements a smoothing technique called Savitsky Golay (SG) filter to enhance the extrapolation capability for convolutional neural network-bidirectional long short-term memory. The experimental results strongly reveal that the SG-CBiLSTM model is advantageous for coping with the sudden variation of the load demand and improving the overall accuracy to reach a mean $R = 99.18\%$. The simulation results have demonstrated that the proposed methodology has the following characteristics: (1) The ten polynomial levels provide the most suitable form after the variation of the levels of the SG filter; (2) the proposed SG-CBiLSTM model acquires a higher extrapolation performance to achieve better results than the single CBiLSTM-AE; (3) the proposed model outperforms nine deep learning models used as benchmarks. The SG-CBiLSTM-AE is perfectly tailored to accurately predict the hourly load data. However, its advantages actually hide some limitations related to the poor performance under an extended forecasting horizon. Future work will broaden the scope to include an additional evaluation of the proposed SG-CBiLSTM model for other power systems and a variety of forecasting applications such as the electricity price and renewable energy prediction.

Author Contributions: Conceptualization, Methodology, Software, original draft preparation, M.M.; validation, writing—review & editing, project administration, funding acquisition, S.S.R.; validation, formal analysis, Data curation, Writing—review & editing, H.A.-R.; validation, writing—review & editing, I.C.; supervision, project administration, F.S.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This publication was made possible by NPRP grant [NPRP10-0101-170082] from the Qatar National Research Fund (a member of Qatar Foundation), the co-funding by IBERDROLA QSTPLLC. The statements made herein are solely the responsibility of the authors.

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

Abbreviations

The following abbreviations are used in this manuscript:

STLF	Short-Term Load Forecasting
ES	Evolution Strategy
PLS	Partial Least Square
BiLSTM	Bidirectional Long-Short Term Memory
DL	Deep Learning
RNN	Recurrent Neural Network
σ	softmax function
SDEM	Scale-Dependent Error Measure
CNN	Convolutional Neural Network
k1	Kernel size = 1
PCA	Principal Components Analysis
CVA	Canonical Variate Analysis
SD	Standard derivation
CBiLSTM	Convolutional Neural Network-Bidirectional Long Short-Term Memory
PEM	Percentage Error Measure

References

1. Mamun, A.A.; Sohel, M.; Mohammad, N.; Sunny, M.S.H.; Dipta, D.R.; Hossain, E. A Comprehensive Review of the Load Forecasting Techniques Using Single and Hybrid Predictive Models. *IEEE Access* **2020**, *8*, 134911–134939. [[CrossRef](#)]
2. Hernandez, L.; Baladron, C.; Aguiar, J.M.; Carro, B.; Sanchez-Esguevillas, A.J.; Lloret, J.; Massana, J. A survey on electric power demand forecasting: Future trends in smart grids, microgrids and smart buildings. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1460–1495. [[CrossRef](#)]
3. Caro, E.; Juan, J. Short-Term Load Forecasting for Spanish Insular Electric Systems. *Energies* **2020**, *13*, 3645. [[CrossRef](#)]
4. Park, R.J.; Song, K.B.; Kwon, B.S. Short-Term Load Forecasting Algorithm Using a Similar Day Selection Method Based on Reinforcement Learning. *Energies* **2020**, *13*, 2640. [[CrossRef](#)]
5. Trierweiler Ribeiro, G.; Guilherme Sauer, J.; Fraccanabbia, N.; Cocco Mariani, V.; dos Santos Coelho, L. Bayesian Optimized Echo State Network Applied to Short-Term Load Forecasting. *Energies* **2020**, *13*, 2390. [[CrossRef](#)]
6. Dong, Y.; Zhang, Z.; Hong, W.C. A hybrid seasonal mechanism with a chaotic cuckoo search algorithm with a support vector regression model for electric load forecasting. *Energies* **2018**, *11*, 1009. [[CrossRef](#)]
7. Kuster, C.; Rezgui, Y.; Mourshed, M. Electrical load forecasting models: A critical systematic review. *Sustain. Cities Soc.* **2017**, *35*, 257–270. [[CrossRef](#)]
8. Lecun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [[CrossRef](#)]
9. Lago, J.; De Ridder, F.; De Schutter, B. Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms. *Appl. Energy* **2018**, *221*, 386–405. [[CrossRef](#)]

10. Khwaja, A.; Anpalagan, A.; Naeem, M.; Venkatesh, B. Joint bagged-boosted artificial neural networks: Using ensemble machine learning to improve short-term electricity load forecasting. *Electr. Power Syst. Res.* **2020**, *179*, 106080. [[CrossRef](#)]
11. Heydari, A.; Nezhad, M.M.; Pirshayan, E.; Garcia, D.A.; Keynia, F.; De Santoli, L. Short-term electricity price and load forecasting in isolated power grids based on composite neural network and gravitational search optimization algorithm. *Appl. Energy* **2020**, *277*, 115503. [[CrossRef](#)]
12. Haq, M.R.; Ni, Z. A new hybrid model for short-term electricity load forecasting. *IEEE Access* **2019**, *7*, 125413–125423. [[CrossRef](#)]
13. Massaoudi, M.; Refaat, S.S.; Chihi, I.; Trabelsi, M.; Oueslati, F.; Abu-Rub, H. A Novel Stacked Generalization Ensemble-Based Hybrid LGBM-XGB-MLP Model for Short-Term Load Forecasting. *Energy* **2020**, *214*, 118874. [[CrossRef](#)]
14. Kim, T.Y.; Cho, S.B. Predicting residential energy consumption using CNN-LSTM neural networks. *Energy* **2019**, *182*, 72–81. [[CrossRef](#)]
15. Tan, M.; Yuan, S.; Li, S.; Su, Y.; Li, H.; He, F. Ultra-Short-Term Industrial Power Demand Forecasting Using LSTM Based Hybrid Ensemble Learning. *IEEE Trans. Power Syst.* **2020**, *35*, 2937–2948. [[CrossRef](#)]
16. Deng, Z.; Wang, B.; Xu, Y.; Xu, T.; Liu, C.; Zhu, Z. Multi-scale convolutional neural network with time-cognition for multi-step short-term load forecasting. *IEEE Access* **2019**, *7*, 88058–88071. [[CrossRef](#)]
17. Moreno, S.R.; da Silva, R.G.; Mariani, V.C.; dos Santos Coelho, L. Multi-step wind speed forecasting based on hybrid multi-stage decomposition model and long short-term memory neural network. *Energy Convers. Manag.* **2020**, *213*, 112869. [[CrossRef](#)]
18. Chen, K.; Chen, K.; Wang, Q.; He, Z.; Hu, J.; He, J. Short-Term Load Forecasting with Deep Residual Networks. *IEEE Trans. Smart Grid* **2019**, *10*, 3943–3952. [[CrossRef](#)]
19. Kong, W.; Dong, Z.Y.; Jia, Y.; Hill, D.J.; Xu, Y.; Zhang, Y. Short-term residential load forecasting based on LSTM recurrent neural network. *IEEE Trans. Smart Grid* **2017**, *10*, 841–851. [[CrossRef](#)]
20. Yang, Y.; Shang, Z.; Chen, Y.; Chen, Y. Multi-Objective Particle Swarm Optimization Algorithm for Multi-Step Electric Load Forecasting. *Energies* **2020**, *13*, 532. [[CrossRef](#)]
21. Aprillia, H.; Yang, H.T.; Huang, C.M. Optimal decomposition and reconstruction of discrete wavelet transformation for short-term load forecasting. *Energies* **2019**, *12*, 4654. [[CrossRef](#)]
22. Hong, T.; Wilson, J.; Xie, J. Long term probabilistic load forecasting and normalization with hourly information. *IEEE Trans. Smart Grid* **2014**, *5*, 456–462. [[CrossRef](#)]
23. Acharya, S.K.; Wi, Y.M.; Lee, J. Short-term load forecasting for a single household based on convolution neural networks using data augmentation. *Energies* **2019**, *12*, 3560. [[CrossRef](#)]
24. Wen, Q.; Sun, L.; Song, X.; Gao, J.; Wang, X.; Xu, H. Time Series Data Augmentation for Deep Learning: A Survey *arXiv* **2020**, arXiv:2002.12478.
25. Bianchi, F.M.; De Santis, E.; Rizzi, A.; Sadeghian, A. Short-term electric load forecasting using echo state networks and PCA decomposition. *IEEE Access* **2015**, *3*, 1931–1943. [[CrossRef](#)]
26. Liu, L.; Ding, F.; Xu, L.; Pan, J.; Alsaedi, A.; Hayat, T. Maximum likelihood recursive identification for the multivariate equation-error autoregressive moving average systems using the data filtering. *IEEE Access* **2019**, *7*, 41154–41163. [[CrossRef](#)]
27. Rahim, R.A.; Jamaluddin, F.; Ibrahim, H.; Khan, S.K.N. A review on smoothing techniques in Markov chains methods. *AIP Conf. Proc.* **2014**, *1635*, 195–200.
28. Ryu, S.; Noh, J.; Kim, H. Deep neural network based demand side short term load forecasting. *Energies* **2017**, *10*, 3. [[CrossRef](#)]
29. Wu, J.M.T.; Tsai, M.H.; Huang, Y.Z.; Islam, S.H.; Hassan, M.M.; Alelaiwi, A.; Fortino, G. Applying an ensemble convolutional neural network with Savitzky–Golay filter to construct a phonocardiogram prediction model. *Appl. Soft Comput.* **2019**, *78*, 29–40. [[CrossRef](#)]
30. Dong, Q.; Lin, Y.; Bi, J.; Yuan, H. An Integrated Deep Neural Network Approach for Large-Scale Water Quality Time Series Prediction. In Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 6–9 October 2019; pp. 3537–3542.

31. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [[CrossRef](#)]
32. Liu, Z.X.; Zhang, D.G.; Luo, G.Z.; Lian, M.; Liu, B. A new method of emotional analysis based on CNN–BiLSTM hybrid neural network. *Clust. Comput.* **2020**, *122*, 1–13.
33. Ju, Y.; Sun, G.; Chen, Q.; Zhang, M.; Zhu, H.; Rehman, M.U. A model combining convolutional neural network and LightGBM algorithm for ultra-short-term wind power forecasting. *IEEE Access* **2019**, *7*, 28309–28318. [[CrossRef](#)]
34. Acharya, D.; Rani, A.; Agarwal, S.; Singh, V. Application of adaptive Savitzky–Golay filter for EEG signal processing. *Perspect. Sci.* **2016**, *8*, 677–679. [[CrossRef](#)]
35. Atif, A.; Khalid, M. Savitzky–Golay Filtering for Solar Power Smoothing and Ramp Rate Reduction Based on Controlled Battery Energy Storage. *IEEE Access* **2020**, *8*, 33806–33817. [[CrossRef](#)]
36. Seo, J.; Ma, H.; Saha, T.K. On Savitzky–Golay filtering for online condition monitoring of transformer on-load tap changer. *IEEE Trans. Power Deliv.* **2017**, *33*, 1689–1698. [[CrossRef](#)]
37. Nguyen, D.V.; Rocke, D.M. On partial least squares dimension reduction for microarray-based classification: A simulation study. *Comput. Stat. Data Anal.* **2004**, *46*, 407–425. [[CrossRef](#)]
38. Sun, W.; Tang, J.; Bai, C. Evaluation of university project based on partial least squares and dynamic back propagation neural network group. *IEEE Access* **2019**, *7*, 69494–69503. [[CrossRef](#)]
39. Simon Blanke. Hyperactive: A Hyperparameter Optimization and Meta-Learning Toolbox for Machine-/Deep-Learning Models. Since 2019. Available online: <https://github.com/SimonBlanke> (accessed on 20 August 2020).
40. Eichardt, R.; Haueisen, J.; Knösche, T.R.; Schukat-Talamazzini, E.G. Reconstruction of multiple neuromagnetic sources using augmented evolution strategies—A comparative study. *IEEE Trans. Biomed. Eng.* **2008**, *55*, 703–712. [[CrossRef](#)]
41. Beyer, H.G.; Schwefel, H.P. Evolution strategies—A comprehensive introduction. *Nat. Comput.* **2002**, *1*, 3–52. [[CrossRef](#)]
42. Loshchilov, I.; Hutter, F. CMA-ES for Hyperparameter Optimization of Deep Neural Networks. *arXiv* **2016**, arXiv:1604.07269.
43. Wong, T.T.; Yeh, P.Y. Reliable accuracy estimates from k-fold cross validation. *IEEE Trans. Knowl. Data Eng.* **2019**, *32*, 1586–1594. [[CrossRef](#)]
44. Massaoudi, M.; Refaat, S.S.; Abu-Rub, H.; Chihi, I.; Wesleti, F.S. A Hybrid Bayesian Ridge Regression-CWT-Catboost Model For PV Power Forecasting. In Proceedings of the 2020 IEEE Kansas Power and Energy Conference (KPEC), Manhattan, KS, USA, 13–14 July 2020; pp. 1–5.
45. Massaoudi, M.; Chihi, I.; Sidhom, L.; Trabelsi, M.; Oueslati, F.S. Medium and Long-Term Parametric Temperature Forecasting using Real Meteorological Data. In Proceedings of the IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society, Lisbon, Portugal, 14–17 September 2019; Volume 1, pp. 2402–2407.
46. ISO New England Web Page of Pricing Reports, Ancillary Services—Final Hourly Regulation Clearing Prices—Historical Data Section. Available online: <https://www.iso-ne.com/isoexpress/web/reports/pricing/-/tree/day-ahead-energy-offer-data> (accessed on 30 July 2020).
47. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
48. Botchkarev, A. Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. *arXiv* **2018**, arXiv:1809.03006.
49. Chollet, F.; Keras. Available online: <https://github.com/fchollet/keras> (accessed on 15 June 2020).
50. Xie, H.; Zhang, L.; Lim, C.P. Evolving CNN-LSTM Models for Time Series Prediction Using Enhanced Grey Wolf Optimizer. *IEEE Access* **2020**, *8*, 161519–161541. [[CrossRef](#)]
51. Shi, H.; Xu, M.; Li, R. Deep Learning for Household Load Forecasting—A Novel Pooling Deep RNN. *IEEE Trans. Smart Grid* **2018**, *9*, 5271–5280. [[CrossRef](#)]
52. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.

53. Hansen, N.; Müller, S.D.; Koumoutsakos, P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evol. Comput.* **2003**, *11*, 1–18. [[CrossRef](#)]
54. Pierezan, J.; Coelho, L.D.S. Coyote optimization algorithm: A new metaheuristic for global optimization problems. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).