



Article

Multi-Sequence LSTM-RNN Deep Learning and Metaheuristics for Electric Load Forecasting

Salah Bouktif ^{1,*} , Ali Fiaz ¹, Ali Ouni ² and Mohamed Adel Serhani ³ 

¹ Department of Computer Science and Software Engineering, UAE University, Al Ain 64141, UAE; engr.alifiaz@gmail.com

² Department of Software Engineering and IT, Ecole de Technologie Supérieure, Montréal, QC H3C 1K3, Canada; ali.ouni@etsmtl.ca

³ Department of Information Systems and Security, UAE University, Al Ain 64141, UAE; Serhanim@uaeu.ac.ae

* Correspondence: salahb@uaeu.ac.ae

Received: 14 December 2019; Accepted: 3 January 2020; Published: 13 January 2020



Abstract: Short term electric load forecasting plays a crucial role for utility companies, as it allows for the efficient operation and management of power grid networks, optimal balancing between production and demand, as well as reduced production costs. As the volume and variety of energy data provided by building automation systems, smart meters, and other sources are continuously increasing, long short-term memory (LSTM) deep learning models have become an attractive approach for energy load forecasting. These models are characterized by their capabilities of learning long-term dependencies in collected electric data, which lead to accurate prediction results that outperform several alternative statistical and machine learning approaches. Unfortunately, applying LSTM models may not produce acceptable forecasting results, not only because of the noisy electric data but also due to the naive selection of its hyperparameter values. Therefore, an optimal configuration of an LSTM model is necessary to describe the electric consumption patterns and discover the time-series dynamics in the energy domain. Finding such an optimal configuration is, on the one hand, a combinatorial problem where selection is done from a very large space of choices; on the other hand, it is a learning problem where the hyperparameters should reflect the energy consumption domain knowledge, such as the influential time lags, seasonality, periodicity, and other temporal attributes. To handle this problem, we use in this paper metaheuristic-search-based algorithms, known by their ability to alleviate search complexity as well as their capacity to learn from the domain where they are applied, to find optimal or near-optimal values for the set of tunable LSTM hyperparameters in the electrical energy consumption domain. We tailor both a genetic algorithm (GA) and particle swarm optimization (PSO) to learn hyperparameters for load forecasting in the context of energy consumption of big data. The statistical analysis of the obtained result shows that the multi-sequence deep learning model tuned by the metaheuristic search algorithms provides more accurate results than the benchmark machine learning models and the LSTM model whose inputs and hyperparameters were established through limited experience and a discounted number of experimentations.

Keywords: long short-term memory (LSTM); genetic algorithm (GA); particle swarm optimization (PSO); time series; energy forecasting

1. Introduction

Accurate load forecasting is of vital importance in the planning and management of power plants, production facilities, and cost-effective operation of power grid networks. Electric load consumption is a time-series data involving a sequence of observations over regularly spaced intervals

of time encompassing both linear and nonlinear components. The essential components describing a time-series are (1) trend, which is upward, downward, or absent, (2) seasonality, which is the periodic fluctuation in time series within a certain period, (3) cycles, where rises and falls are not of a fixed period, and finally (4) irregular movement, which is left after explaining the trend, seasonal, and cyclical movements [1]. Electricity consumption usage is typically highly random due to the operation of several electrical appliances and consumer behavior, which makes the identification of these time series consumption patterns more difficult [2].

Traditional linear statistical models such as autoregressive (AR), moving average (MA) and autoregressive integrated moving average (ARIMA) have remained the baseline for time-series predictions, with extensive use in several industrial applications [3,4]. The basic assumptions made to implement these models are based on the fact that time series are considered stationary and linear and follow a particular known statistical distribution. Similarly, several machine learning models have been proposed for time series forecasting as an alternative to statistical models that can address the non-linearity in time series data. The most commonly used models are support vector regression (SVR) and the artificial neural networks (ANNs), which can model complex, mostly nonlinear relationships between electric load and related factors to achieve higher precision in energy consumption [5,6].

Although neural network architectures offer numerous advantages compared to ARIMA models in addressing nonlinear and non-normal-distributed data, mostly encountered in real-world problems [7], they have the drawback of assuming that all inputs and outputs are independent of each other, even when dealing with sequential data. This assumption omits the predictive potential that exists in the dependency relationship among energy-consumption sequential data. Hence, it is imperative to consider the dependency between successive consumptions of electric energy.

Deep learning architectures are suitable in tackling, at the same time, the specific problems of electric load forecasting such as nonlinearity, periodicity, and seasonality, and the sequential dependence among consumption data sequences. In particular, LSTM networks, deep learning variations proposed by Hochreiter and Schmidhuber [8], were designed specially to learn long-term dependencies present in sequential data. Compared to shallow ANN architecture, deep learning models apply non-linear transformations to automatically learn complex temporal patterns via high-level abstractions. LSTM's high performance is achieved by memorizing long-term dependencies, thanks to an internal memory that makes it perfectly suitable for problems involving sequence-dependent behavior such as electricity load and demand [9,10]. In other terms, load forecasting characteristics make their predictions based on past observations since the pattern and the behavior of the energy consumption will likely reappear in the future. Hence, an important step for time series forecasting is to choose the right past observations or lags that can be the best predictors of future electric consumption. For example, the immediate past history of consumption could be meant for short and medium-term load forecasting. These lagged temporal predictors enable capturing causality relationships between the load present and past values, as well as the repeatedness of the consumption patterns. Moreover, the inclusion of lagged temporal load parameters, such as the length of lag sequences and their positions in the history of consumption, can implicitly bring information that is originally contained within implicit variables, such as the weather as well as the occupancy characteristics for different horizons of load forecasts [11].

A univariate model with only past lags can effectively capture the underlying pattern for load forecasting, which has also been validated in our previous research [9]. However, appropriate lag selection also enables the elimination of irrelevant features, thereby reducing the computational effort in the training phase and improving the model prediction performance. Habitually, lag selection in time series methods is performed by visual interpretation of autocorrelation and partial autocorrelation function outputs, followed by an empirical setting of the lag parameters. However, with energy data, which has a complex pattern, the selection may not be obvious, so several options may have to be considered before landing on an optimal choice of the best past periods in the time series [12]. This technique is error-prone and does not consider the energy-domain knowledge that could help in finding the best-correlated lags.

In addition, the proposed deep learning model hyperparameters need to be optimally determined. They are those parameters whose values need to be set before the training process, and their tuning is an integral part of building deep learning models. Parameters can be few in number or can be in the hundreds, such as the number of layers, the number of cells per each layer, batch size, and the type of activation, as well as their parameters. Choosing the values for these hyperparameters can affect the training process, model underfitting, overfitting, and subsequently the final-model accuracy, but it is complex, error-prone, and time-consuming.

To summarize, building an accurate electric load forecasting model requires searching for an optimal configuration that involves lag selection and deep-learning hyperparameter setting. Unfortunately, this is a non-trivial task that can be solved using an exhaustive search and deterministic methods, especially in a big data context. It is rather a hard problem for which an optimal solution cannot be found in a polynomial time. This hardness is accentuated by the complexity of electricity-consumption data patterns. One alternative solution toward an optimal configuration of energy forecasting models is to use metaheuristics approaches, known by their ability to find near-optimal solutions in a very large space. In our case, metaheuristics will select from an infinite number of configurations the appropriate lags as well as the best parameter setting for an accurate time series analysis using deep learning [13]. Beside circumventing the complexity of searching in an infinite space of configuration solutions, we share the belief that metaheuristics and in particular evolutionary algorithms can integrate domain knowledge in their mechanism of individual representation, fitness function, and evolutionary operators [14].

In this paper, we customize the application of two evolutionary metaheuristics, namely a genetic algorithm (GA) and particle swarm optimization (PSO), to the problem of optimizing the performance of an LSTM model for electric load forecasting. Our approach is compared to alternative baseline approaches that use state-of-the-art machine learning techniques as well as a multi-sequence LSTM model with manually tuned parameters through extensive experimentation [9,10]. The comparison shows that the application of metaheuristics for the optimal configuration of electric load forecasting derived a multi-sequence LSTM model that performs significantly better than the benchmark models including other machine learning techniques (i.e., SVR, random forest, and ANN) and manually configured LSTM.

This paper is organized as follows: Section 2 describes the related work where different approaches for hyperparameters tuning are discussed for machine learning and deep learning models. Section 3 describes the background, the LSTM model, a padding operation for a one-dimension (1D) sequence, hyperparameter tuning, and the two metaheuristic algorithms used in this work: a genetic algorithm (GA) and particle swarm optimization (PSO). Section 4 describes the problem formulation and emphasizes the importance of lag selection and hyperparameter tuning. Section 5 presents our metaheuristic search-based solution using the GA and PSO for optimal LSTM configuration. Section 6 presents the experimental study and model validation, while Section 7 presents the conclusions and discusses future work.

2. Related Work

Deep neural architectures have recently shown their capability of modeling complex underlying patterns for electricity forecasting. Zheng et al. proposed the LSTM model for short term forecasting using complex univariate electric load time series data of a school building with strong non-stationarity and non-seasonality [15]. It was found that the LSTM-based forecasting method outperformed other methods, including seasonal autoregressive integrated moving average (SARIMA), a nonlinear autoregressive neural network with exogenous inputs (NARX), support vector regression (SVR), and a traditional feed-forward neural network. Narayan & Hipel used the LSTM model for short-term electric load forecasting using ten years of historical data for the province of Ontario in Canada and reported very reliable and robust results [16]. Similarly, Marino et al. used two different deep architectures for

energy load forecasting, a standard LSTM, and an LSTM with sequence-to-sequence architecture that produced good results [17].

Several works utilizing PSO for optimizing machine learning models in the energy domain are reported in the literature. Wang et al. used hybrid intelligent forecasting models utilizing cuckoo search and particle swarm optimization for tuning parameters of SARIMA and SVR models for improving short-term load forecasting [18]. Results showed that single spectral analysis denoising and swarm intelligence-based optimization effectively improved model performance. Ozerdem et al. used particle swarm technique for optimization of a feedforward neural network for predicting hourly load supplied by an energy company and found that both particle-swarm-optimized neural networks and backpropagation neural networks are relevant for load forecasting [19]. Furthermore, the PSO networks result in faster convergence. Similarly, Ren et al. used an integrated prediction approach for the annual electricity consumption of Beijing, and PSO was used to find the best parameters for the SVM model [20].

Several published works have successfully used GAs for various optimization objectives. Islam et al. implemented a genetic algorithm for optimizing the number of layers and neurons per layer for an ANN trained with a backpropagation algorithm. Mean absolute percentage error (MAPE) and model computational time was significantly reduced after optimizing ANN topology [21]. Defilippo et al. used GAs to select the appropriate architecture and training parameters using evolutionary simulations of a neural network model for the load forecast for a city in Brazil. Two naive time series forecasts, a linear method, and a neural network with grid search parameters were used as benchmarks. It was found that the GA-based method produced much better results, with a lower MAPE error than the other four benchmarks [22].

Recently, time-series modeling using the LSTM technique has gained popularity to model complicated sequences such as electric loads where it is essential to learn long-term dependencies and keep track of what is happening far back in the past. LSTM modifies the recurrent neural network (RNN) architecture using a memory cell and a gating mechanism that allows for the regulation of information flow across the network. Srivastava and Lessmann used LSTM for solar energy forecasting. Neurons in the first and second layers were grid-searched for both LSTM and ANN models, and it was found that a properly configured LSTM model outperforms gradient boosting regression and feed-forward neural networks for day-ahead predictions [23]. Lago et al. predicted day-ahead electricity prices by employing four different deep learning models. For benchmark comparison, 23 different models proposed in the literature for electricity prices forecasting were used. Bayesian optimization methods were used for the selection of optimal parameters for deep neural network (DNN), LSTM, and gated recurrent unit (GRU) models. It was found that the DNN, LSTM, and GRU models achieved a predictive accuracy that is statistically significantly better than all other benchmarks [24]. Bandara et al. built a prediction model using LSTM on subgroups of a similar time series using two benchmark datasets. The approach discovered clusters of a similar series from the overall set of the time series. LSTM hyperparameters tuning such as epoch size, mini-batch size, and regularization weight were performed using grid search on an additional validation set [25]. Results showed that LSTM can outperform univariate forecasting methods, and subgrouping a similar time series augments the accuracy of this baseline LSTM model.

In addition to energy forecasting, LSTM and metaheuristics have been used in several other domains and have demonstrated superior performance with respect to other deep learning models. Xiao and Yin proposed a hybrid LSTM neural network for traffic flow prediction that was optimized for both large and small traffic flow sets, achieving a lower RMSE value compared to other baseline models [26]. Roman et al. used fuzzy logic and a metaheuristic optimizer, namely a Grey Wolf Optimizer, to address a non-linear process control problem [27]. Mahanipour et al. proposed a gravitational search algorithm (GSA), a population-based algorithm to automatically create computer programs and applied it to symbolic regression (SR) and the problem of feature construction (FC) [28].

Liu et al. designed a wind power short-term forecasting method based on discrete wavelet transform and LSTM, and reported an increased prediction accuracy compared to five different benchmarks [29].

Algorithms based on the metaheuristic paradigm have been effectively applied to various complex problems in the energy domain; however, the implementation strategy of a metaheuristic for LSTM model improvement through hyperparameter tuning has hardly been investigated in the energy domain. It is very important to determine an optimal value for several hyperparameters related to both model inputs and model configuration, i.e., the number of input sequences, their lengths, starting points, the number of LSTM cells, batch size, activation function, and optimizers, to achieve the best model performance.

3. Background

This section provides brief backgrounds on the LSTM-RNN model, hyperparameter tuning approaches, GA and PSO metaheuristics, advantages and disadvantages of these approaches, and forecasting evaluation metrics.

3.1. The Long Short-Term Memory (LSTM) Model

LSTM models belong to the family of deep recurrent neural networks that have been widely used across different domains such as language modeling, sentiment analysis, speech recognition, and time-series models [30]. LSTM models and their variants achieve good performance when the temporal dependency in the sequential data is an important implicit feature, and learning from earlier stages is essential to forecast future trends. Compared to a standard feedforward neural network, LSTM contains cycles that feed network activations from a previous time step as inputs to the network to influence predictions at the current time step. Thus, the recurrent connection allows the model to develop a memory of previous events, which is implicitly encoded in its hidden state variables. The network takes three inputs: x_t is the input at the current time step, h_{t-1} is the output from the previous LSTM unit, and c_{t-1} is the memory of the previous unit. As for outputs, h_t is the output of the current network, and c_t is the memory of the current unit. The LSTM model has input i_t , output o_t , and forget f_t learnable gates that modulate the flow of information and maintains an explicit hidden state that is recursively carried forward and updated as each element of the sequential data is passed through the network, as shown in Figure 1. The input gate decides what information to add from the present input to the cell state, the forget gate decides what must be removed from the h_{t-1} state, thus keeping only relevant information, and the output gate decides what information to output from the current cell state.

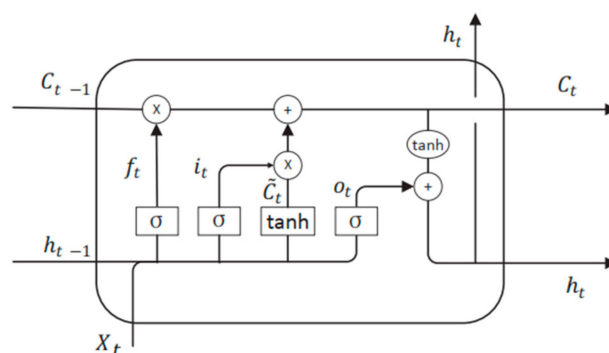


Figure 1. Architecture of a long short-term memory (LSTM) cell.

3.1.1. Equations Controlling the LSTM Cell

The LSTM transition equations for the LSTM are as follows:

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

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (2)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (3)$$

$$\tilde{C} = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (4)$$

$$C_t = f_t \odot C_{t-1} + i_t * \tilde{C}_t \quad (5)$$

$$h_t = o_t \odot \tanh(C_t) \quad (6)$$

In the equations above, c_t is a cell state, h_t is a hidden state, σ denotes the logistic sigmoid function, and the symbol \odot denotes elementwise multiplication. Thus, the output at each time step depends on previous inputs and past computation. LSTM models can also have a multi-sequence input configuration where various lags of the same or different time series can be presented as model inputs. This can allow the LSTM model to capture the valuable information contained with different timescales [31].

3.1.2. Padding for Variable-Length Input Sequences for the LSTM Model

In the case of multiple variable-length input sequences to the LSTM model, the data must be transformed such that each sequence has the same length. In the case of electricity consumption prediction, the multiple sequences of lags can be of different lengths. In this case, zero-padding for one dimension (1D) temporal sequence input can be used, which consists of adding zeros at the beginning or at the end of the padding dimension [32]. The padding can be connected to the beginning or to the end of the sequence. Mostly post padding is used to pad sequences to the preferred length.

3.2. Hyperparameter Tuning approaches for Data-Driven Models

Selecting optimal parameters for a neural network architecture such as LSTMs can make a significant difference in improving the model performance. Popular search strategies that can find these hyperparameters include exhaustive search, random search, and Bayesian optimization approaches [33]. An exhaustive approach such as grid search consists of trying all possible combinations of a manually defined subset of discrete hyperparameters. However, with the increasing number of hyperparameters to be tuned, the search space complexity grows exponentially. An alternative to this type of search is a random search that uses a randomly selected subset of the parameters. Random search is known to find better models in most cases while taking less computational time. In contrast to grid and random searches, Bayesian optimization allows one to make use of the results of previous iterations to improve the sampling method of the next experiment, thus considering the information on the hyperparameter combinations seen so far when choosing the hyperparameter set to evaluate in the next stage. Comparatively, metaheuristic algorithms have shown effectiveness in stochastic optimization for hard optimization problems giving global or near-optimal solutions within a satisfactory search time and computational cost [34].

3.3. Metaheuristic Algorithms

Metaheuristics are a popular scheme for solving hard optimization problems and usually provide a sufficiently good solution, especially with limited computational capacity and time constraints. These algorithms are essentially stochastic and are termed as nature-inspired algorithms since their derivation originates from the observation of natural behavior. While GAs come from biology, Ant Colony and Particle Swarms come from ethology. But simulated annealing is inspired by physics. These approaches explore iterative modifications of the population of individuals, while the quality of individuals of the population is evaluated using a fitness function. Compared to traditional gradient-based methods that can become stuck at the local optima, metaheuristics can find optimal or near-optimal solutions to complex problems made of many peaks and valleys [14].

3.3.1. The Genetic Algorithm (GA)

GAs are a global probabilistic search technique inspired by Darwinian evolution, and they aim at finding optimal solutions by simulating a natural evolutionary process [35]. Initially, the first papers presenting the idea of GAs were published by Alex Fraser [36], and they were then developed by Holland at the University of Michigan [37]. A GA is predominantly suitable for combinatorial optimization where an exhaustive search of all possible solutions necessitates enormous computational power. While a GA has several variants, the fundamental underlying mechanism operates on a population of individuals following a process that is repeated until the system stops improving. In each generation, new individuals are added to the population, and the worst members are removed. Individuals with higher fitness produce more offspring than those with lower fitness. These cycles of survival mimic natural evolution. The flowchart of the GA is shown in Figure 2.

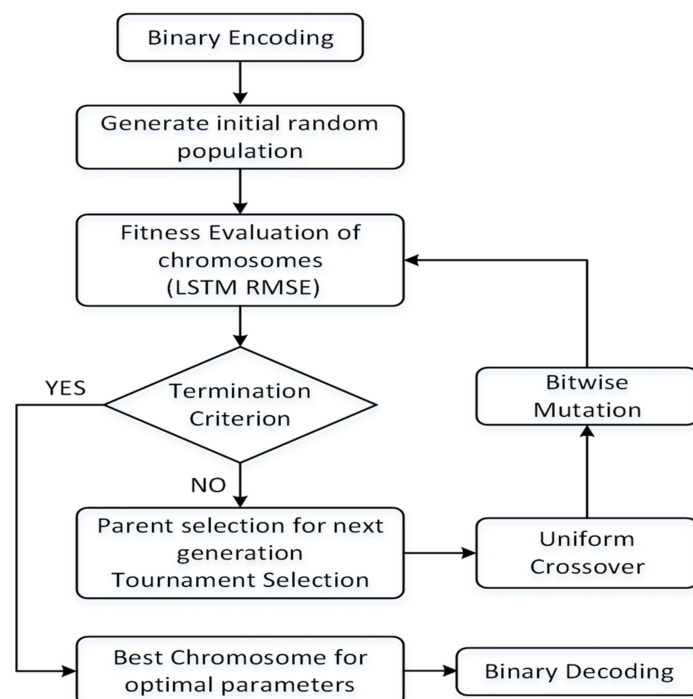


Figure 2. Tailored genetic algorithm (GA) for LSTM model optimization.

Genetic operators such as crossover, mutation, and selection are used to maintain genetic diversity, which is necessary for the evolution process. In general, the selection mechanism facilitates the exploitation of the accumulated information resulting from the GA search, while the crossover and mutation operators account for the exploration of new regions of search space. Selection operation emphasizes fitter individuals in the population, with the aim that their offspring has a higher fitness to survive in the next generation. Crossover generates new individuals by replacing some parts of two parents' individuals, while the mutation operator changes some genes of the individual string. The operators of mutation and uniform crossover are depicted in Figure 3.

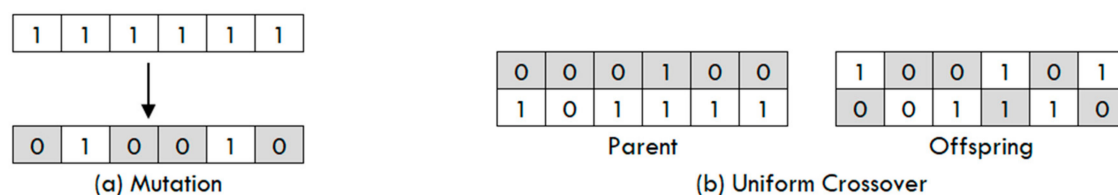


Figure 3. Mutation and crossover operators.

3.3.2. Particle Swarm Optimization (PSO)

Particle swarm optimization developed by Kennedy and Eberhart in 1995 was inspired by the social behavior [38]. It is evolved by imitating the social behavior of a school of fishes and a flock of birds that collectively search for the best locations and are likely to move close to an optimum fitness function [39]. In particle swarm optimization, entities called particles are positioned in the search space of the problem. Each particle calculates the objective function at its current location. The movement of each particle in the search space is determined by its personal best and swarm best, enabling stochastic exploration of solution space. Each particle has a vector representing the speed of the particle in each dimension along with a reference to the best state in which it has been so far. PSO is different from a GA in the sense that it contains a network of communication among particles of the swarm. For each step, the velocity of particles is changed toward local and global best locations until a defined termination condition is reached. The algorithm for PSO is depicted in Figure 4.

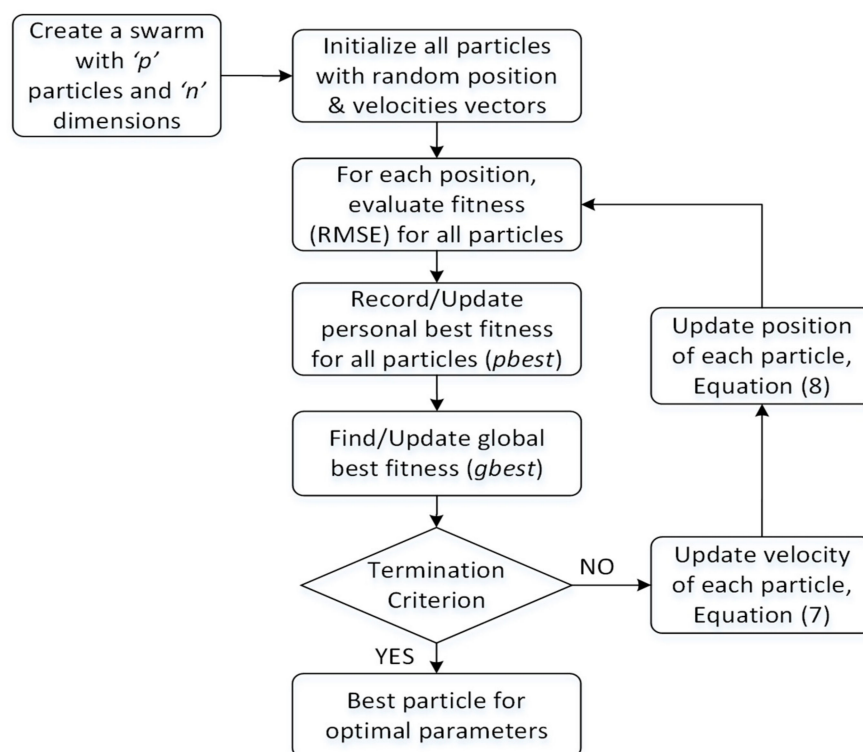


Figure 4. Flowchart algorithm for particle swarm optimization (PSO).

Every particle is a candidate solution to the problem to be solved. These particles are interacting and learning from each other, while maintaining a memory of their personal best and global best position. The new position of the particle is thus a function of three vectors: personal best, global best, and previous velocity vector. Figure 5 shows the concept of changing the position x_i of a particle from the current position following the calculations made based on the current position, personal, and global best.

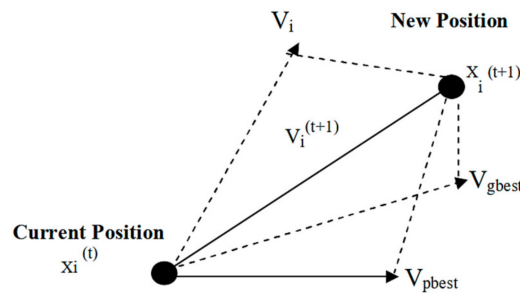


Figure 5. Change of position of PSO particles.

The position vector shows the position of the particle in the n -dimensional search space, while the velocity vector shows the direction and intensity of the movement. The new position is better as it considers the previous decision about the movement of the particle, the personal best, and the whole swarm global best. Thus, the swarm cooperates to find the best location in the search space. A mathematical model of the particle's motion in the search space is defined as follows:

$$v_i(t+1) = wv_i(t) + c_1(p_i(t) - x_i(t)) + c_2(g(t) - x_i(t)) \quad (7)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (8)$$

where x_i is the current position, v_i is the current velocity, $v_i(t+1)$ and $x_i(t+1)$ are the velocity and position at time $t+1$, p_i is the local best for a particle, and $g(t)$ is the global best position. The new velocity in Equation (7) depends on three terms, w , the inertia coefficient, c_1 , the cognitive coefficient, and c_2 , the social coefficient. The first term in Equation (7) of the velocity vector is the inertia term, which maintains the current movement direction, the second term is known as the cognitive component, in which each particle considers the distance between its personal best and current location, and the third term is known as the social component, where the particle calculates the distance between its current position and the best position found by the entire swarm, the global best. A new velocity vector is created by combining these three components. This velocity vector translates the position to a new position in the search space using Equation (8).

3.3.3. Advantages and Disadvantages of GA and PSO Approaches

The GA can perform a global search and explore the search space using their different crossover and mutation operators, which makes its population characteristics more diverse. Furthermore, they do not have any prior knowledge about the structure of the function to be optimized in advance. Some of the challenges of GA include defining a suitable solution representation for a given problem, which can sometimes be complex and leads to inappropriate choice of mutation and cross-over operators [40]. As compared to the GA, the PSO algorithm is easier to implement, having fewer tuning parameters and often found to be more robust, providing repeatable results. However, PSO can also be prone to premature convergence and low accuracy.

3.4. Evaluation Metrics

To evaluate the performance of the models, three metrics, the coefficient of variation (CV-RMSE), the root mean squared error (RMSE), and the mean absolute error (MAE) are used:

$$CV(\%) = \frac{\sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2}}{\bar{y}} \times 100 \quad (9)$$

$$RMSE = \frac{\sqrt{\sum_{i=1}^N (y_i - \hat{y}_i)^2}}{N} \quad (10)$$

$$\text{MAE} = \sum_{i=1}^N \frac{|y_i - \hat{y}_i|}{N} \quad (11)$$

where \hat{y}_i is the predicted energy consumption, y_i is the actual energy consumption, and \bar{y} is the average energy consumption found by averaging y_i , the actual consumption. CV(RMSE) is the root mean square error normalized by the mean of the measured values, with a high score indicating that the model has a high error range. MAE is the mean value of the sum of absolute differences between actual and forecasted. RMSE has the benefit of penalizing the larger error terms and has the same units as the quantity being predicted.

4. Problem Formulation

In this section, we describe the LSTM network configuration, the input representation for the univariate model, the importance of lagged input selection, and the hyperparameters to be optimized for the electric load forecasting problem.

4.1. The LSTM Configuration Problem

We have modeled load forecasting as a regression problem where the task is to utilize n previous data points $\{X_{t_1}, X_{t_2}, \dots, X_{t_n}\}$ to predict the next h data points $\{X_{t_{n+1}}, X_{t_{n+2}}, \dots, X_{t_{n+h}}\}$. The input data for LSTM needs to be reshaped into a 3D array with the following dimensions: [samples, time steps, features]. A train test set split is performed on the dataset while preserving the observations' temporal order. This will allow us to construct a many-to-one model that produces one output $y(t)$ value after receiving the input sequence $X(t), X(t+1), \dots, X(t+n)$. The lags can be presented to the model either as single or multiple input sequences across different timescales.

4.2. Input Selection Problem for the Univariate LSTM Model

A time series containing records of a single variable, here the past electric load consumption, is termed as a univariate model including sequence data that imposes an explicit order on the observations. An important difference in training an LSTM in contrast to an ANN model is that to predict a value y_t at time t , previous n samples, $\{y_{t-1}, y_{t-2}, \dots, y_{t-n}\}$, are propagated through the network. The number of past lags have to be defined for this network. These univariate time-series models require the careful selection of past lags, which defines its underlying dynamics, inherent structure, and non-linear characteristics. Lag window selection is one of the most important tasks in time-series modeling and prediction tasks. If the selected lags are not relevant, this may unnecessarily increase the curse of dimensionality, slow down the training process, and lead to model overfitting [41]. The lag selection must, therefore, be more advanced than the determination of the model order as is done in simple linear autoregressive models.

We aim to find the optimal number of lags, their starting point, and their lag length to achieve the best forecasting performance as described in Table 1. The number of input sequences defines how many lagged windows to use, the starting point defines at which past points in time we should select the load values from, and the size determines how many previous load values are considered in the forecasting model. In the case where the sequences are of different lengths, sequences that are shorter than the number of time steps of the longest proposed sequence by the GA/LSTM solution are padded with values at the end of the sequence. It is assumed that an additional context using multiple lags as inputs from the past consumption data may improve the performance of the predictive model. Thus, the model can have single or multiple lag sequences as inputs, which would be searched upon by the metaheuristics.

Table 1. Input lag parameters for the LSTM model.

No	Parameter	Description
1	number of input sequence	single or multi-sequence of time lags as input to the LSTM model
2	sequence length	length of the lags to predict the target, zero padding in case of a sequence of different length
3	sequence starting points	starting points of lagged inputs

4.3. LSTM Hyperparameter Setting Problem

Selecting good hyperparameters for deep learning models requires many experiments, which is a tedious and time-consuming task. Most researchers rely on their experience in choosing suitable parameters for a deep neural network. Depending on the dataset size, it can take several days to train a single model, so a common approach is a meticulous selection of limited values of the hyperparameters to train several models and then choose the model that performs best on a validation set. However, instead of restraining ourselves to a few selected hyperparameters, the objective is the selection of optimal parameters in this large sub-space using the aforementioned metaheuristics. The parameters to be optimized for the LSTM model are described in Table 2.

Table 2. Hyperparameters for the LSTM model.

No	Parameter	Description
1	LSTM cells	the number of LSTM memory cells that store the temporal information
2	batch Size	number of samples per gradient update
3	activation function	type of nonlinear activation function
4	optimizer	type of optimizer to update weights during training

LSTM cells facilitate learning the long-term dependencies in the data, and the number of LSTM cells to use depends on the application and context in which the model is being used. Batch size is an important hyperparameter that can affect the training efficiency of the neural network that defines the number of samples that are going to be propagated through the network before the model's internal parameters are updated. The data is divided into several smaller batches, and the neural network is trained in multiple stages. Typically, networks train faster with mini-batches; however, with a batch size that is too small, a less accurate estimate of the gradient is obtained, and with a larger batch size, more memory space is required.

Other parameters to be optimized are the activation function and optimizer for the model. Selecting the type of activation function is a critical task. The activation function introduces non-linearity to the network, enabling it to learn more complex patterns. Activation functions that are searched include sigmoid, hyperbolic tangent (*tanh*), exponential linear unit (*elu*), and rectified linear unit (*relu*). Likewise, optimizers that are searched include stochastic gradient descent (*sgd*), root mean square propagation (*RMSprop*), the adaptive gradient algorithm (*Adagrad*), and adaptive moment estimation (*Adam*).

Model hyperparameters that have been predetermined in advanced include the number of layers, epochs, loss function, dropout, and the number of neurons in the dense layers. Optimizing these parameters in conjunction with those defined in Tables 1 and 2 is time-consuming and can be very computationally expensive. Therefore, for a satisfactory search time and computational cost, these parameters were chosen based on our previous research.

5. A Metaheuristic-Based Solution for Optimal LSTM Configuration

This section represents an overview of the metaheuristic framework adopted to discover the optimal tuning of our forecasting model, the used dataset description, the LSTM setting, and the tailoring of the two applied metaheuristics to search for optimal hyperparameters for the LSTM configuration for electric load forecasting. The objective function of our search problem for both

metaheuristics, the GA and PSO, is the RMSE error of the LSTM model. The termination criterion chosen for the GA and PSO is met when a specified number of generations is reached or when no improvement of the fitness value is recorded. Based on the complexity of the electric load problem, the generation size for both metaheuristics is after several experiments set to $N = 30$.

5.1. Framework

The framework we developed to determine the optimal parameters for the time series forecasting of electricity load involved representation, search, and final training, as depicted in Figure 6. The first step is the representation or encoding of a solution, and the searching component then uses two different metaheuristics, PSO and a GA, to find the optimal lag inputs and hyperparameter subset for the LSTM model. These parameters are then used to build and train the final LSTM model. The performance of the optimized model is then compared to the benchmark. For our benchmark, we used random forest and support vector regressor machine learning models as well as a multi-sequence LSTM model from our previous work, whose inputs and hyperparameters have been established through experimentation [10]. The aforementioned multi-sequence model outperformed challenging machine learning models such as the ANN and the ensemble for this complex time series forecasting task. The root mean squared error (RMSE) was used as the fitness function for both optimization algorithms.

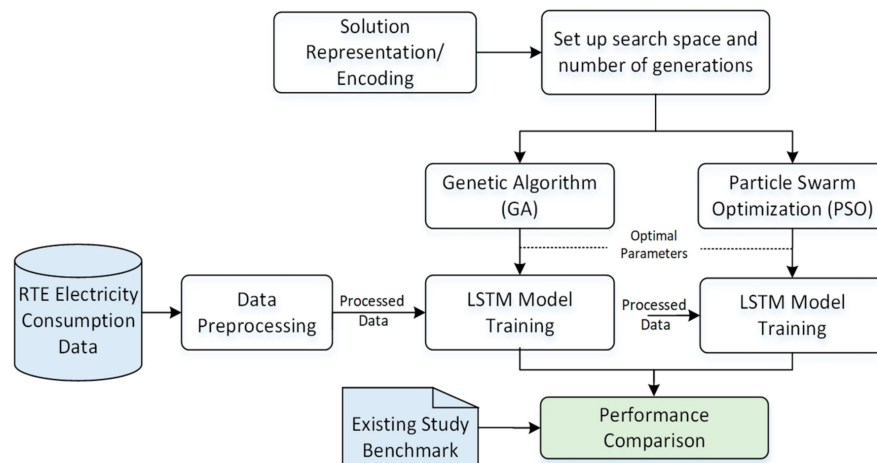


Figure 6. Research methodology framework.

5.2. Dataset Description and Preparation

The dataset was obtained from the RTE Corporation, a public utility company that operates the French electricity transmission network [42]. The data is a univariate time series comprising nine years of half-hourly electrical consumption data from 2008 to 2016 in megawatts (MW) for metropolitan France. Figure 7 represents the time series decomposition of the log-transformed data aggregated to daily values presenting trends as well as seasonal and irregular components over the nine-year period. As seen in the figure below, the data exhibits daily, weekly, and seasonal patterns.

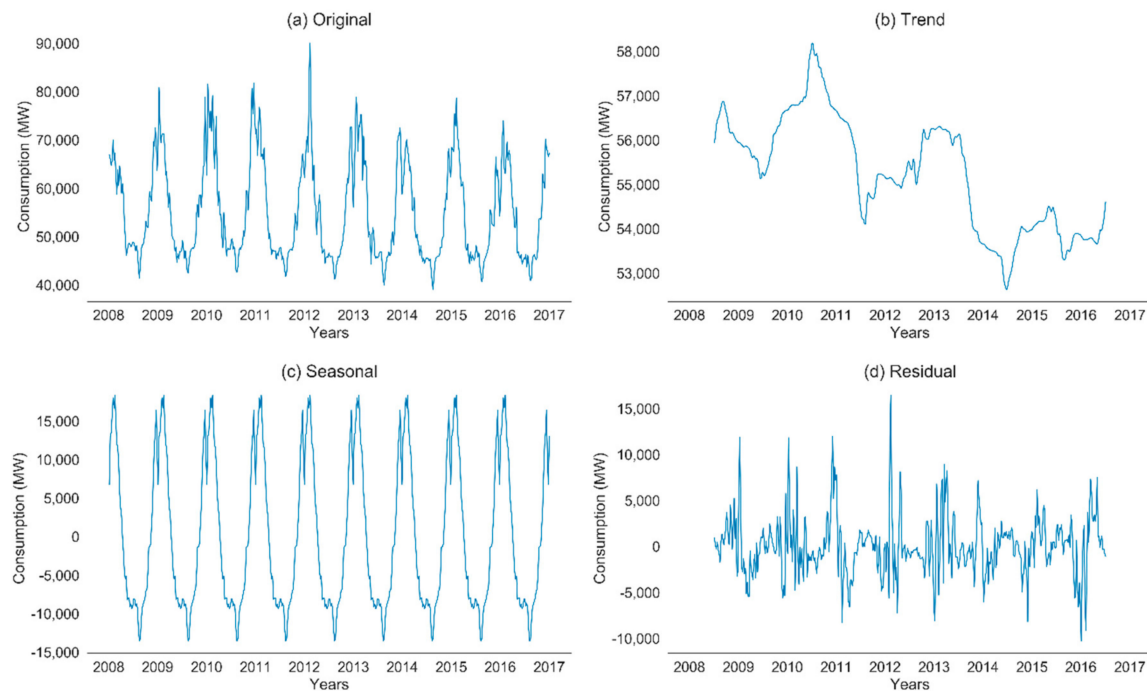


Figure 7. Original and decomposed time series.

The preprocessing of the raw data involved data cleansing, normalization, and structural change. Data was scaled in the range from 0 to 1 using feature scaling, as large values can slow down the learning and convergence of the deep neural network. The autocorrelation function (ACF) for the first difference of the log-transformed consumption data to detect important dependencies in the time series data for the past three months is shown in Figure 8. Spikes that rise above or below the dashed lines are considered to be statistically significant. For this data, spikes are statistically significant for lags up to the past two months. The dataset comprises 48 measured load values per day, so lags from the $t - 1$ to $t - 2880$ time step range were selected to be explored by the metaheuristics. It was also ensured that, if a multi-sequence model was proposed by the metaheuristic search, various time windows did not overlap, as this would result in a replication of information, which may adversely affect model performance.

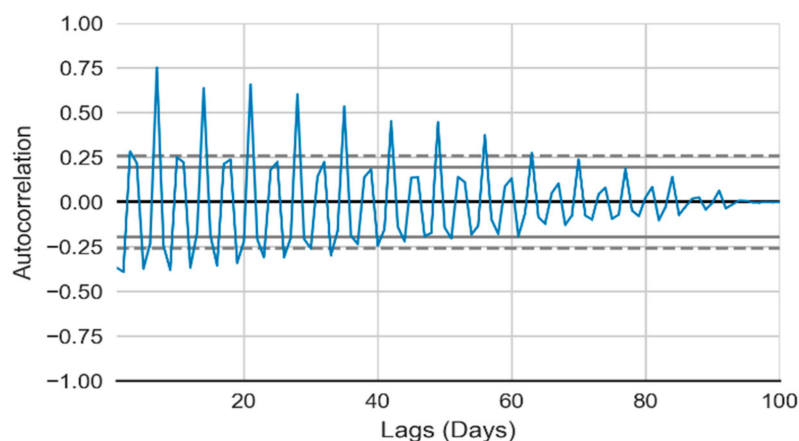


Figure 8. Autocorrelation function (ACF) plot.

5.3. LSTM Settings

A deeper neural network with more layers will have a greater capacity, but more layers can eventually result in a decrease in the performance due to high variance and model overfitting and thus

a reduced generalization power for the electricity dataset [43]. A deep model with one LSTM layer and three hidden layers, with 100, 60, and 50 neurons having carefully selected multi-sequence inputs, was used as the benchmark model. Using an early stopping criterion, the LSTM network training process was terminated before the algorithm's convergence criteria were satisfied. This was done by monitoring the validation loss at each epoch and stopping the training if the validation loss did not decrease for several epochs. The maximum number of epochs was set to 150. Dropout was used for regularization to prevent overfitting by randomly choosing cells in a layer and setting their output to zero according to a chosen probability. The rest of the parameters for the model were identified by the metaheuristics search. The training set was used for updating network weights and biases, while the validation set was used when overfitting started.

5.4. Genetic Algorithms Tailoring

Considering the nature of the electric consumption sequence and the LSTM hyperparameters, the GA was represented as a binary coded model, which is the most common way of encoding, where an individual or combination of bits in the string represents some characteristics of the solution. The role of genetic operators in an evolutionary algorithm is very crucial, as they have a strong effect on the performance of the algorithm. The encoding of the problem and the genetic operators are described in the sub-sections below.

5.4.1. Solution Encoding

Each chromosome is a binary encoded representation of the parameters to be optimized, and they correspond to the lag inputs and LSTM hyperparameters, as stated in Tables 1 and 2. The initial population comprised of 20 chromosomes, represented as a sequence of 69 binary genes, was created, and it represents the parameters to be searched, as depicted in Figure 9.

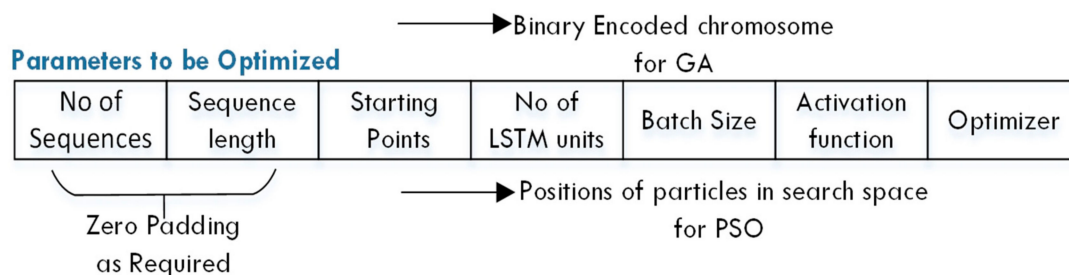


Figure 9. Binary Encoding for the GA.

5.4.2. Crossover Operator

We used uniform crossover with a probability of 0.6, where each bit from the offspring's genome was independently chosen from the two parents. Thus, uniform crossover works by exchanging individual bits and not segments of the bit array, eliminating any potential positional bias. This crossover can provide the majority of the GA search capability by randomly distributing genes from the two original chromosomes, so the mutation rate can be set relatively low.

5.4.3. Mutation Operator

Mutation provides randomness in the search and facilitates local optima avoidance in the search space. Probabilistic bitwise mutation, in which a gene value may be flipped from 0 to 1, or vice versa, was used here, with a mutation probability of 0.1. This small random change inserted into a randomly selected position into the multivariate time series inputs and model configuration introduces diversity to the chromosomes.

5.4.4. Selection

The selection operator chooses chromosomes from the mating pool in which the fittest individuals have a better likelihood of survival compared to the weaker ones. We used tournament selection with a size of 3, and this is a popular selection method in which several tournaments are run between randomly selected chromosomes from the population. The winner of each tournament was selected for crossover. A large tournament value results in a stronger selective pressure, and the population can converge to a faster solution, at the cost of the solution being potentially not as good [44].

5.5. Particle Swarm Optimization Configuration

To implement PSO, we created a swarm of particles with dimensions equal to the number of parameters to be optimized for the LSTM model, stated in Tables 1 and 2. The size of the population was empirically set to 20, based on dimensionality, time constraints, and the perceived difficulty of the problem. The particle's goal is to minimize the residuals—the difference between the actual values and the predicted values of the LSTM model. The setting of the various parameters for the PSO is described below.

5.5.1. Position and Velocity

The position and velocity of the particles were initialized in a range so that any lag in the past two months, with a length of thirty at most, can be selected. Velocity in the PSO algorithm is one of its major features, as this is used to move the position of a particle to search for optimal solutions. If velocity is too low, the algorithm may be slow to converge, and high velocity results in instability. If a new particle position moves beyond the boundary, a new position is set as X_{min} and X_{max} . The value of velocity is also clamped to the range $[V_{min}$ and $V_{max}]$ to reduce the likelihood of the particle leaving the search space. V_{min} and V_{max} are set to 10% of the particle position range.

5.5.2. Acceleration and Inertia Coefficients

Based on the definition of the velocity as described in Equation (7), c_1 and c_2 are the acceleration coefficient for the cognitive and social components that regularize the step size in the direction of the personal and global best position of the particles. Thus, these constants control the length and time that particles need to reach the most optimum position. By trial and error, these values were set to $c_1 = 1.5$ and $c_2 = 2$. Values of $c_1 = c_2 = 2$ have shown fast global convergence to the optimum solution for most of the problems [19].

The inertia coefficient w tunes the local and global search ability of the PSO. A value of 1 was selected for this parameter, which yields a better result, maintaining a balance between exploration and exploitation and speeding up the convergence of the load forecasting model.

6. Empirical Validation

In this section, we describe the experiments performed for validating our approach of applying the GA and PSO metaheuristics to an electricity consumption dataset. The input and LSTM model hyperparameters described in Section 4 and Tables 1 and 2 were optimized to achieve the best electric load forecasting performance. Optimal parameters obtained after the search are described, and results are compared with the benchmarks. Finally, model validation using a sliding window approach and conducting a t -test for the difference in means was performed to ascertain if the differences in the average CV(RMSE) scores of the best solution, found using the metaheuristic approach, were significantly different from those of the machine learning and multi-sequence benchmark models.

6.1. GA and PSO Search Results

The time series LSTM models were developed in Python using the Keras API running on top of the TensorFlow backend. A detailed design configuration for the LSTM network, the GA,

and PSO metaheuristics representation and parameters, as defined in Section 5, was used for the experiment. The search space for the lagged inputs and hyperparameter selection as stated by the two metaheuristics ran for 30 generations. To speed up computations, we ran both metaheuristics on a reduced dataset. Fitness graphs of the normalized RMSE scores for the GA and PSO search are shown in Figures 10 and 11. The continuous curve is the minimum fitness, and the dashed curve is the average fitness across the generations.

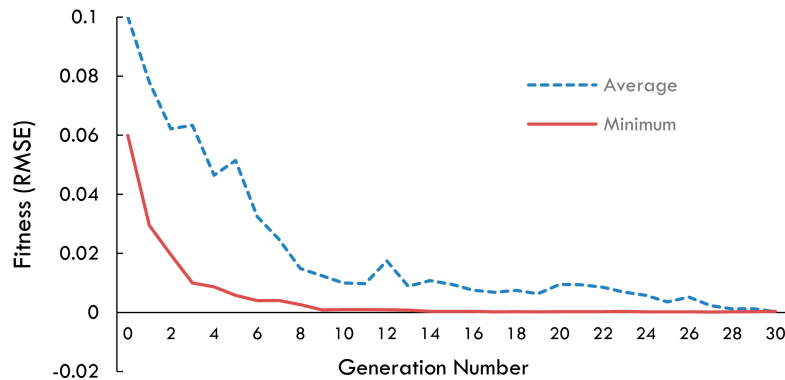


Figure 10. Convergence Plot for Genetic Algorithm.

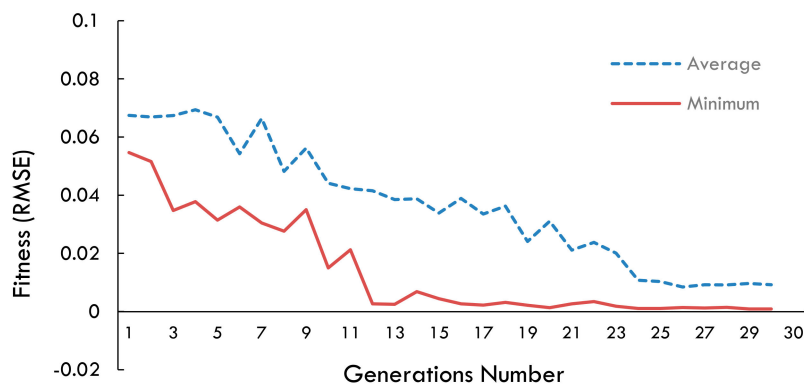


Figure 11. Convergence plot for the PSO algorithm.

6.2. Optimal Parameters suggested by GA and PSO

The optimal parameters suggested by the best performing GA and PSO models are described in Table 3. It was found that the models suggest multi-sequence inputs with two and three different time scales, window lengths of 7 and 10, and starting points that were immediate and approximately one to five weeks earlier. The LSTM cell numbers were 63 and 54, the batch sizes were 34 and 35, the activation functions suggested were ReLU and Leaky ReLU, and the optimizers were Adamax and ADAM, respectively.

Table 3. Optimal parameters found by the metaheuristics solution.

No	Parameter	Best Value
1	No of Sequence	2, 3
2	Sequence Length	7–11
3	Starting Points	Immediate up to 05 weeks
4	No of LSTM units	53–64
5	Batch Size	34–40
6	Activation Function	ReLU, Leaky ReLU
7	Optimizer	ADAM and Adamax

These parameters found by the GA and PSO were then used to train the LSTM model on the complete dataset, and performance was compared with the benchmark models, including random forest, SVR (Support Vector Regression), an ANN, and an Extra Trees Regressor. The results of the comparison are shown in Table 4. The obtained performance values demonstrate that metaheuristic-based calibration of LSTM for load prediction derives a more accurate forecasting model that outperforms the manual/expert calibration of the deep learning model (i.e., the multi-sequence LSTM), ensemble learning models (i.e., random forest and Extra Trees Regressor), the regression model (i.e., SVR), and the neural network model (i.e., ANN). A more rigorous validation of our model using K-fold cross-validation and a *t*-test is presented in Section 6.4 and Table 5, where we compare the performance of our metaheuristic-based LSTM to the best benchmarks of Table 4.

Table 4. Evaluation results of the benchmarks and the GA and PSO models.

Metrics	LSTM_GA	LSTM_PSO	Multi-Seq LSTM	Random Forest	SVR	ANN	Extra Trees Regressor
RMSE	311.44	342.57	353.38	437.43	479.87	725.89	492.13
MAE	231.50	248.83	263.14	335.24	385.12	559.63	346.44
CV(RMSE)	0.621	0.622	0.643	0.805	0.835	1.311	0.889

Table 5. Two-sample *t*-test for differences in the mean CV(RMSE).

Model	Multi-Seq LSTM	RF	SVR	Metaheuristic- LSTM
Mean	0.7630	0.9855	1.0155	0.6799
Std Dev	0.1378	0.1334	0.1336	0.0820
<i>p</i> -value	0.047	0.00001	0.00001	-
<i>t</i> -value	−1.79542	−7.41852	−7.8886	

The multi-sequence inputs suggested by both the GA and the PSO indicate that our metropolitan electric consumption dataset has a multi-scale property, which implies that the consumption sequence can exhibit distinct patterns in different time scales. Since the GA-optimized LSTM slightly outperformed the PSO, we used the GA-searched parameters to develop the LSTM learning curve and further conduct validation tests.

6.3. Learning Curve

A model fit is considered suitable when the performance of the model on both the training and validation sets is good and does not overfit. The learning curve in Figure 12 shows the decay of the loss function with epochs. This curve can help predict whether the model has overfitted, under-fitted, or is fit for the training and testing datasets. The plot shows that loss function decays quite rapidly to a low value given the large training set. Training and validation losses decrease and stabilize around the same point. The model thus successfully captures the electricity consumption patterns.

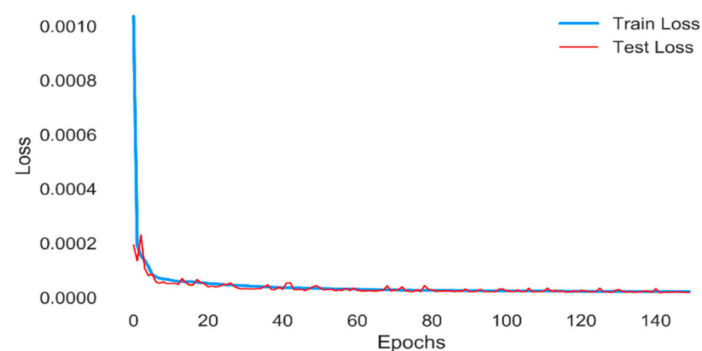


Figure 12. Decay of loss function for the training and testing sets for the multi-sequence LSTM.

6.4. Model Validation using a Sliding Window Approach: A *t*-Test for the Difference in Means

A two-sample *t*-test at a 0.05 level of significance was conducted to ascertain if the differences in the average CV(RMSE) scores of the best solution found using the metaheuristic approach were significantly better those of the best two machine learning benchmarks (i.e., random forest and SVR, as shown in Table 4) and of the expertly calibrated multi-sequence. A walk forward sliding window *k*-fold validation approach was used to test these models. For each of the windows, the testing and training dataset sizes remain fixed while traversing the complete dataset, as illustrated in Figure 13.

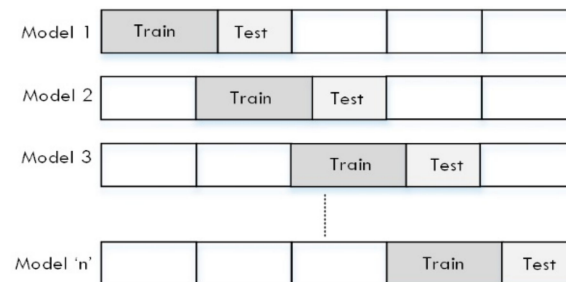


Figure 13. Train test split for model validation.

Fifteen different train test splits were created, and performance metrics were calculated for the training and validation of the proposed models. Differences in means were compared using a one-tailed *t*-test assuming equal variances, and the population standard deviations are not known. The following hypotheses were tested:

$$\text{Null, } H_0 : \mu_1 - \mu_2 = 0$$

$$\text{Alternate, } H_1 : \mu_1 < \mu_2$$

A *p*-value <0.05 and a high *t*-value for the various models, when compared to the optimal LSTM model, indicates that the mean CV(RMSE) score of the LSTM model optimized using a metaheuristic approach is lower than that of the multi-sequence and machine learning benchmark model at a 0.05 level of significance, with a sample size of 15. The results are shown in Table 5.

7. Conclusions and Future Works

Load forecasting is a critical area of interest in the energy domain, particularly for energy distributors seeking to minimize costs. This work describes a new approach to using an LSTM deep model to forecast short-term electricity consumption for a metropolitan area. A vital issue in the design of an appropriate LSTM forecasting model is selecting appropriate time lag sequences along with model hyperparameters for a particular time series data. It was found that the GA and PSO metaheuristics-based calibration was a successful solution that circumvented the combinatorial complexity of finding optimal parameters over a discrete search space in the context of electric energy consumption.

Our findings provide support for both GA and PSO calibration using a multi-sequence input model with variable-length sequences. Given that LSTM is constrained by constant-length input sequences, variable-length sequences are possible using a zero-padding mechanism. Indeed, metaheuristics-based calibration has made the input lag selection more flexible, which is needed in a load forecasting context and subsequently has enabled more precise learning of long-term dependencies in load consumption patterns. Therefore, by using multiple and different timescale sequences, LSTM easily carries crucial information over a long distance. Thus, the complexity, non-stationary and non-linearity of load patterns can be learned to provide a more accurate forecast. The results of comparison with strong benchmark models demonstrate that our metaheuristic-based calibration of LSTM for load prediction derives more accurate forecasting models that outperform a manual/expert-calibrated deep learning

model (i.e., multi-sequence LSTM), ensemble learning models (i.e., random forest and Extra Trees Regressor), a regression model (i.e., SVR), and a neural network model (i.e., ANN).

In our future work, we will capitalize on the multi-scale property of our input sequences of the proposed LSTM model to improve the cost of our model. This will be done by introducing attention mechanisms to the optimized LSTM model. An attention mechanism can lead to better results by assigning different degrees of attention to sub-window lagged features within multiple time series. Such an attention mechanism will emphasize learning from particular time periods rather than a wide history. In another direction, we will also extend our work by tailoring the metaheuristic calibration for different load forecasting horizons.

Author Contributions: This paper is a collaborative work of all authors. Conceptualization, S.B. and A.F.; Methodology, S.B. and A.F.; Software, A.F.; Validation, S.B., A.F. and A.O.; investigation, S.B.; resources, S.B. and A.O.; Data Curation, A.F.; Writing-Original Draft Preparation, S.B., A.F., A.O. and M.A.S.; Writing-Review & Editing, S.B., A.F., A.O. and M.A.S.; Supervision, S.B.; Project Administration, S.B.; Funding Acquisition, S.B. and A.O. All authors have read and agreed to the published version of the manuscript.

Acknowledgments: The authors would like to acknowledge United Arab Emirates University for supporting the present work by a Start-Up grant under grant number G00002211 and providing essential facilities

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

References

1. Wang, C.H.; Grozev, G.; Seo, S. Decomposition and statistical analysis for regional electricity demand forecasting. *Energy* **2012**, *41*, 313–325. [\[CrossRef\]](#)
2. Chou, J.S.; Ngo, N.T. Time series analytics using sliding window metaheuristic optimization-based machine learning system for identifying building energy consumption patterns. *Appl. Energy* **2016**, *177*, 751–770. [\[CrossRef\]](#)
3. Hyndman, R.; Koehler, A.B.; Ord, J.K.; Snyder, R.D. *Forecasting with Exponential Smoothing: The State Space Approach*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2008.
4. Janacek, G. Time series analysis forecasting and control. *J. Time Ser. Anal.* **2010**, *31*, 229–303. [\[CrossRef\]](#)
5. Bontempi, G.; Taieb, S.B.; Le Borgne, Y.A. Machine learning strategies for time series forecasting. In *European Business Intelligence Summer School*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 62–77.
6. Ahmed, N.K.; Atiya, A.F.; Gayar, N.E.; El-Shishiny, H. An empirical comparison of machine learning models for time series forecasting. *Econom. Rev.* **2010**, *29*, 594–621. [\[CrossRef\]](#)
7. Khashei, M.; Bijari, M. An artificial neural network (p, d, q) model for timeseries forecasting. *Expert Syst. Appl.* **2010**, *37*, 479–489. [\[CrossRef\]](#)
8. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [\[CrossRef\]](#)
9. Bouktif, S.; Fiaz, A.; Ouni, A.; Serhani, M.A. Optimal Deep Learning, LSTM Model for Electric Load Forecasting using Feature Selection and Genetic Algorithm: Comparison with Machine Learning Approaches. *Energies* **2018**, *11*, 1636. [\[CrossRef\]](#)
10. Bouktif, S.; Fiaz, A.; Ouni, A.; Serhani, M. Single and multi-sequence deep learning models for short and medium term electric load forecasting. *Energies* **2019**, *12*, 149. [\[CrossRef\]](#)
11. Yildiz, B.; Bilbao, J.I.; Sproul, A.B. A review and analysis of regression and machine learning models on commercial building electricity load forecasting. *Renew. Sustain. Energy Rev.* **2017**, *73*, 1104–1122. [\[CrossRef\]](#)
12. Hansen, J.V.; McDonald, J.B.; Nelson, R.D. Time Series Prediction with Genetic-Algorithm Designed Neural Networks: An Empirical Comparison with Modern Statistical Models. *Comput. Intell.* **1999**, *15*, 171–184. [\[CrossRef\]](#)
13. Yang, X.S. *Engineering Optimization: An Introduction with Metaheuristic Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2010.
14. Bonissone, P.P.; Subbu, R.; Eklund, N.; Kiehl, T.R. Evolutionary algorithms+ domain knowledge= real-world evolutionary computation. *IEEE Trans. Evol. Comput.* **2006**, *10*, 256–280. [\[CrossRef\]](#)
15. Zheng, J.; Xu, C.; Zhang, Z.; Li, X. Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network. In Information Sciences and Systems (CISS). In Proceedings of the 51st Annual Conference, Baltimore, MD, USA, 22 March 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–6.

16. Narayan, A.; Hipel, K.W. Long short term memory networks for short-term electric load forecasting. In Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC) IEEE International Conference, Banff, AB, Canada, 5–8 October 2017; pp. 2573–2578.
17. Marino, D.L.; Amarasinghe, K.; Manic, M. Building energy load forecasting using deep neural networks. In Proceedings of the IECON 2016–42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy, 23–26 October 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 7046–7051.
18. Wang, J.; Jin, S.; Qin, S.; Jiang, H. Swarm intelligence-based hybrid models for short-term power load prediction. *Math. Probl. Eng.* **2014**. [\[CrossRef\]](#)
19. Ozerdem, O.C.; Olaniyi, E.O.; Oyedotun, O.K. Short term load forecasting using particle swarm optimization neural network. *Procedia Comput. Sci.* **2017**, *120*, 382–393. [\[CrossRef\]](#)
20. Ren, G.; Wen, S.; Yan, Z.; Hu, R.; Zeng, Z.; Cao, Y. Power load forecasting based on support vector machine and particle swarm optimization. In Proceedings of the 2016 12th World Congress Intelligent Control and Automation (WCICA), Guilin, China, 12–15 June 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 2003–2008.
21. Islam, B.U.; Baharudin, Z.; Raza, M.Q.; Nallagownden, P. Optimization of neural network architecture using genetic algorithm for load forecasting. In Proceedings of the 2014 5th IEEE International Conference Intelligent and Advanced Systems (ICIAS), Kuala Lumpur, Malaysia, 3–5 June 2014; pp. 1–6.
22. Defilippo, S.B.; Neto, G.G.; Hippert, H.S. Short-term load forecasting by artificial neural networks specified by genetic algorithms—a simulation study over a Brazilian dataset. In Proceedings of the XIII Simposio Argentino de Investigación Operativa (SIO)—JAIIO 44, Rosario, Santa Fe, Argentina, 31 August–1 September 2015.
23. Srivastava, S.; Lessmann, S. A comparative study of, L.STM neural networks in forecasting day-ahead global horizontal irradiance with satellite data. *Sol. Energy* **2018**, *162*, 232–247. [\[CrossRef\]](#)
24. 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. [\[CrossRef\]](#)
25. Bandara, K.; Bergmeir, C.; Smyl, S. Forecasting Across Time Series Databases using Long Short-Term Memory Networks on Groups of Similar Series. *arXiv* **2017**, arXiv:1710.03222.
26. Xiao, Y.; Yin, Y.; Hybrid, L. STM Neural Network for Short-Term Traffic Flow Prediction. *Information* **2019**, *10*, 105. [\[CrossRef\]](#)
27. Roman, R.C.; Precup, R.E.; David, R.C. Second order intelligent proportional-integral fuzzy control of twin rotor aerodynamic systems. *Procedia Comput. Sci.* **2018**, *139*, 372–380. [\[CrossRef\]](#)
28. Mahanipour, A.; Nezamabadi-Pour, H. GSP: An automatic programming technique with gravitational search algorithm. *Appl. Intell.* **2019**, *49*, 1502–1516. [\[CrossRef\]](#)
29. Liu, Y.; Guan, L.; Hou, C.; Han, H.; Liu, Z.; Sun, Y.; Zheng, M. Wind Power Short-Term Prediction Based on, L.STM and Discrete Wavelet Transform. *Appl. Sci.* **2019**, *9*, 1108. [\[CrossRef\]](#)
30. Salehinejad, H.; Baarbe, J.; Sankar, S.; Barfett, J.; Colak, E.; Valaee, S. Recent Advances in Recurrent Neural Networks. *arXiv* **2017**, arXiv:1801.01078.
31. Liu, P.; Qiu, X.; Chen, X.; Wu, S.; Huang, X. Multi-timescale long short-term memory neural network for modelling sentences and documents. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 2326–2335.
32. tf.keras.layers. ZeroPadding1D TensorFlow Core r1.14 TensorFlow. Available online: https://www.tensorflow.org/api_docs/python/tf/keras/layers/ZeroPadding1D (accessed on 18 April 2019).
33. Bergstra, J.; Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
34. Brownlee, J. *Clever Algorithms: Nature-Inspired Programming Recipes*; Lulu Press: Morrisville, NY, USA, 2011.
35. Holland, J.H.; Goldberg, D. *Genetic Algorithms in Search, Optimization and Machine Learning*; Addison-Wesley: Boston, MA, USA, 1989.
36. Fraser, A.S. Simulation of genetic systems by automatic digital computers Introduction. *Aust. J. Biol. Sci.* **1957**, *10*, 484–491. [\[CrossRef\]](#)
37. Holland, J.H. *Adaptation in Natural and Artificial Systems*; University of Michigan Press: Ann Arbor, MI, USA, 1975.
38. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the M.HS’95 Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43.
39. Poli, R.; Kennedy, J.; Blackwell, T. Particle swarm optimization. *Swarm Intell.* **2007**, *1*, 33–57. [\[CrossRef\]](#)

40. Meyer-Baese, A.; Schmid, V.J. *Pattern Recognition and Signal Analysis in Medical Imaging*; Elsevier: Amsterdam, The Netherlands, 2014.
41. Lütkepohl, H.; Krätzig, M. (Eds.) *Applied Time Series Econometrics*; Cambridge University Press: Cambridge, UK, 2004.
42. RTE France. Bilans Électriques Nationaux. Available online: <http://www.rte-france.com/fr/article/bilanselectriques-nationaux> (accessed on 7 April 2018).
43. Geman, S.; Bienenstock, E.; Doursat, R. Neural networks and the bias/variance dilemma. *Neural Comput.* **1992**, *4*, 1–58. [[CrossRef](#)]
44. Goldberg, D.E.; Deb, K. A comparative analysis of selection schemes used in genetic algorithms. *Found. Genet. Algorithms* **1991**, *1*, 69–93.



© 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/>).