axioms

MDPI

Article

Metaheuristic-Based Hyperparameter Tuning for Recurrent Deep Learning: Application to the Prediction of Solar Energy Generation

Catalin Stoean [1], Miodrag Zivkovic [2], Aleksandra Bozovic [3], Nebojsa Bacanin [2,*], Roma Strulak-Wójcikiewicz [4], Milos Antonijevic [2] and Ruxandra Stoean [1]

[1] Department of Computer Science, University of Craiova, A.I.Cuza, 13, 200585 Craiova, Romania
[2] Faculty of Informatics and Computing, Singidunum University, Danijelova 32, 11010 Belgrade, Serbia
[3] Academy of Applied Technical Studies, Katarine Ambrozic 3, 11000 Belgrade, Serbia
[4] Faculty of Economics and Transport Engineering, Maritime University of Szczecin, Wały Chrobrego 1/2, 70-500 Szczecin, Poland
* Correspondence: nbacanin@singidunum.ac.rs; Tel.: +381-60-7490326

Abstract: As solar energy generation has become more and more important for the economies of numerous countries in the last couple of decades, it is highly important to build accurate models for forecasting the amount of green energy that will be produced. Numerous recurrent deep learning approaches, mainly based on long short-term memory (LSTM), are proposed for dealing with such problems, but the most accurate models may differ from one test case to another with respect to architecture and hyperparameters. In the current study, the use of an LSTM and a bidirectional LSTM (BiLSTM) is proposed for dealing with a data collection that, besides the time series values denoting the solar energy generation, also comprises corresponding information about the weather. The proposed research additionally endows the models with hyperparameter tuning by means of an enhanced version of a recently proposed metaheuristic, the reptile search algorithm (RSA). The output of the proposed tuned recurrent neural network models is compared to the ones of several other state-of-the-art metaheuristic optimization approaches that are applied for the same task, using the same experimental setup, and the obtained results indicate the proposed approach as the better alternative. Moreover, the best recurrent model achieved the best results with R^2 of 0.604, and a normalized MSE value of 0.014, which yields an improvement of around 13% over traditional machine learning models.

Keywords: metaheuristic optimizers; deep learning; long short-term memory networks; solar energy generation; time series

MSC: 68T07

Citation: Stoean, C.; Zivkovic, M.; Bozovic, A.; Bacanin, N.; Strulak-Wójcikiewicz, R.; Antonijevic, M.; Stoean, R. Metaheuristic-Based Hyperparameter Tuning for Recurrent Deep Learning: Application to the Prediction of Solar Energy Generation. *Axioms* **2023**, *12*, 266. https://doi.org/10.3390/axioms12030266

Academic Editors: Freddy A. Lucay and Wenceslao Palma

Received: 3 February 2023
Revised: 25 February 2023
Accepted: 2 March 2023
Published: 4 March 2023

1. Introduction

Energy consumption worldwide has risen by 40% in the last three decades and this is expected to increase [1]. Governments are taking measures to reduce it, and the recent crisis in Ukraine has led to the exploration of alternative sources of energy. Green energy has seen significant growth worldwide in recent years, as the world shifts towards a more sustainable future. Governments, organizations, and individuals alike are recognizing the importance of reducing reliance on traditional non-renewable energy sources and transitioning to clean, renewable energy sources such as solar, wind, and hydropower. This growth in green energy has not only helped mitigate the effects of climate change but has also created new job opportunities and stimulated local economies [2]. As awareness and technology continue to improve, the green energy sector is in for continued growth in the coming years [3].

The rising concerns about environmental and climate changes and the escalating growth in renewable sources have made the accurate forecasting of its generation a key

point in the shift toward sustainable energy worldwide. Considering the topic of the current paper, i.e., solar energy production, various techniques have been suggested in the last decade for the prediction of its trend and it is widely acknowledged that varying forecasting time frames necessitate different methodologies to reduce the effects of solar variability. Since environmental factors significantly impact renewable energy systems, using methods and models to anticipate these changes is crucial. This is why having the means to comprehend the correlation between various exogenous parameters and utilizing this information effectively is vital [4].

The prediction of generated power can be achieved through statistical methods, machine learning (ML), or deep learning (DL) techniques, with the foremost sector in these studies being the solar energy [5]. The DL application shows an abundance of recurrent architectures, especially Long Short-Term Memory (LSTM) networks. However, there is still much room for improvement in the application of DL recurrent models for multi-variate forecasting of energy production in power systems. Better architectures would result not only from the neural network mechanisms but also from a proper establishment of their hyperparameters [6]. In [6], it is demonstrated that the fine-tuning for the hyperparameters of a basic LSTM via metaheuristics leads to clearly better results than when performing a grid search for finding adequate values for them. A data set concerning energy load forecasting is used. Therefore, the current paper attempts to fill a part of this gap by studying the enhancement brought by a novel nature-inspired metaheuristic, i.e., the reptile search algorithm (RSA), that is also adapted through hybridization with mechanisms of other recent metaheuristics towards the optimization of the hyperparameters of two LSTM models for short-term forecasting of solar energy production in Spain. Accordingly, the current work not only deals with a different data set than [6], but also employs more complex DL models and proposes a new hybridized metaheuristic specifically developed for the current task.

The article is structured as follows. Section 2 reviews the current state of the art on solar energy forecasting and outlines the basic concepts related to the LSTM and the metaheuristic search strategy. Section 3 introduces the metaheuristic approach that will be employed for LSTM hyperparameter tuning, its improved hybridized version and the use for the hyperparameter tuning task. Experiments on the real-world data set with the versions of the two LSTM architectures tuned by the proposed metaheuristic and comparative alternatives are done in Section 4. Conclusions on the potential of the given approach are drawn in Section 5.

2. Background

The recent entries in the literature regarding the forecasting of solar energy generation by means of ML are reviewed in the following subsection. An overview of LSTM architectures is subsequently given, along with a brief introduction to metaheuristic optimization.

2.1. Literature Survey

The number of articles on the application of ML algorithms in the field has increased significantly in recent years, indicating their importance, high use and significant ability to analyze issues related to energy systems [4]. There are numerous approaches for photovoltaic (PV) output power prediction. Several studies have previously summarized PV output power forecasting from different angles.

Accurately predicting solar power generation requires site-specific analysis and consideration of factors such as model type, forecast time horizon, local climate, and many other data and model characteristics, which makes it difficult to generalize results. To overcome this, models must be tested across various conditions and locations to average out their impact. To truly understand a model's performance, it should be tested in multiple locations while preserving the relevant scenario [7].

Anuradha et al. [8] highlight that weather and physical elements affect the electrical power output of a solar PV panel. Solar irradiance, cloud cover, humidity, and ambient

temperature are the main meteorological factors that influence solar power generation. Predicted weather parameters can be used as model inputs, while solar power forecasts can be used as the model output. Voyant et al. [9] recognize that in relation to photovoltaic systems and solar power plants, it is necessary to predict the intensity and direction of solar radiation in a given area in order to estimate production capacity.

Ahmed et al. [5] also highlight that incorporating PV into power grids is challenging due to the irregular, climate and location-based nature of solar energy, leading to problems such as surges, instability, poor scheduling, and financial losses. Predictive models can assist, but factors such as timing, forecast range, data preparation, weather classification, network optimization, uncertainty quantification, and performance assessments need to be considered. The same study shows that solar irradiance has the strongest correlation with photovoltaic output, making weather classification and cloud motion analysis crucial. Kuo et al. [10] reach a similar conclusion, i.e., that accurate prediction of PV power generation is difficult due to the complex interactions between many environmental conditions and uncontrollable factors. Therefore, the study of weather data is crucial for predicting the impact of solar power generation. In [11], it is likewise stated what weather conditions (observable, forecast, or both) and weather variables (air temperature, relative humidity, pressure, cloud cover/sky cover, wind speed and direction) should be used, and in what specific cases, to obtain better forecast results.

Many studies compare different ML algorithms such as linear regression, bagging, decision trees, random forests, support vector machines and generalized additive models [9,12,13]. Voyant et al. [9] describe the use of ML algorithms in modeling and forecasting weather and solar energy, divided into supervised learning (linear regression, generalized linear models, nonlinear regression, support vector machines/support vector regression, decision tree learning, nearest neighbor, Markov chain), unsupervised learning (K-means and k-methods clustering, hierarchical clustering, Gaussian mixture models, cluster evaluation) and ensemble learning (boosting, bagging, random subspace, predictors ensemble). Carrera et al. [11] evaluate and categorize supervised learning algorithms for multiple independent variables in the prediction of solar energy generation in the following three sections: single regression models (linear regression, Huber, Ridge, lasso, elastic net, decision tree, k-NN, SVR), bagging ensemble methods (bagging, random forest, extra trees) and boosting ensemble methods (AdaBoost, gradient boosting, CatBoost, XGBoost). A similar overview of the use of ML algorithms for solar energy forecasting can be found in the review by Wu et al. [14].

Zhang et al. [15] classify ML methods from different perspectives and provide a systematic and critical overview of their use for recent PV power output applications in terms of temporal and spatial prediction scales. The authors state that the artificial neural network and support vector machines are used much more often than other methods. They discuss in detail and explore the potential advantages of optimizing the machine model to improve prediction performance.

In the comparative study of Anuradha et al. [8], three regression models from ML techniques such as support vector machines, random forests and the linear regression model are considered for experimentation and the random forest regression proves to be the most accurate.

According to Markovics and Mayer [16], one of the possible directions of the development of ML methods for photovoltaic power forecasting is the use of more advanced ML models, including the popular deep learning methods. The other possibility is to delve into the hybridization of physical and ML models, as the results revealed that adding even as simple theoretically calculated data to predictors as the angles of the sun's position significantly increased the accuracy of power predictions. Finally, extending the study to probabilistic PV power prediction and model testing in such a context is also a potential topic.

Vennila et al. [17] elaborates on a study where a hybrid model that used both ML and statistics performed better than a model that used only ML. It is further suggested that the proposed method can increase efficiency and accuracy using several DL mechanisms.

The proposed model for stable power generation forecasting in [18] is a hybrid of a Convolutional Neural Network (CNN) and LSTM. The CNN classifies weather conditions and the LSTM identifies power generation patterns based on those conditions. The hybrid model is effective in accurately forecasting power generation by considering weather fluctuations given by CNN. Results from a qualitative evaluation show that the forecasted power signals respond to fluctuations and closely follow the actual power signal trend.

The research conducted by Alkhayat and Mehmood [19] also underlines that ML and DL models have a good ability to discover nonlinear relationships and superior performance. DL specifically has a promising future because of its generalization capability, of continuing to improve the quality of results by increasing the quantity of the data used for training and unsupervised feature learning. This is evidenced by the number of articles on solar energy research based on DL from recent years [4,10,20,21].

Wang et al. [22] also note that DL applications have grown rapidly due to their ability to deal with large data sets and high-performance computing power. However, the study points out two major challenges: (a) theoretical issues: the deep learning models are generally nonconvex functions, and it is thus theoretically difficult for DL to train the deep network and optimize its parameters, and (b) modeling problems - the key issue is how to design a hierarchical model with powerful feature learning and establishing the most appropriate DL-based prediction model for a specific forecasting data set.

Kuo et al. [10] compare the results of one-day-ahead PV power forecasting using three models, ANN, LSTM, and gated recurrent unit (GRU) with three groups of weather data (central weather bureau, local weather station, and hybrid data given by the combination of the other two). The hybrid data improved measurements compared to the other two groups, and the LSTM model was the most accurate in various weather conditions. The combination of the LSTM model and hybrid data proved to be the most accurate with a one-month forecasting accuracy.

A comparison of different DL models for short-term PV power forecasting is conducted in [23]. The models are LSTM, BiLSTM, GRU, BiGRU, CNN1D, and hybrid models like CNN1D-LSTM and CNN1D-GRU. The study also confirms the effectiveness of DL models, like LSTM, over traditional neural networks.

Similar results are reported by Jebli et al. [24] who explored the use of DL techniques to predict solar energy, in particular recursive neural network (RNN), LSTM, and GRUs.

Li et al. [25] propose a hybrid DL approach combining CNN and LSTM for PV power forecasting. The CNN extracts nonlinear features and invariant structures from previous power data for prediction, while the LSTM models temporal changes in the latest data to predict the next step. The proposed method showed smaller prediction errors when compared to several benchmark methods.

In the comprehensive review of the literature [19] on the subject, the most used architectures are the hybrid models followed by RNN models such as LSTM and GRU, and, in the third place, CNN.

As the literature review shows, various models that employ optimization algorithms for estimating solar energy generation have received an increasing amount of attention, with the more commonly used optimal algorithms being: particle swarm optimization (PSO), which is also the most frequent [26], genetic algorithms (GA) [27] and whale optimization algorithm (WOA). The major factors that affect the forecasting results were identified as solar irradiance, wind speed, and temperature. Accordingly, they are naturally the more prevalent used inputs. Although probabilistic forecasts with uncertainty information are highly useful for system operations, deterministic forecasts remain the primary methods employed; however, it is expected that the importance of the former will increase.

Panda et al. [28] emphasize that the choice of parameters for ML systems in renewable energy forecasting can have a major impact on the predictions. The new possibility for

further research is based on the use of improved metaheuristics for ML variable selection for renewable-energy predictions.

According to the current discussions in the literature, the present paper proposes two LSTM architectures powered by an improved version of a nature-inspired metaheuristic to perform the tuning of the hyperparameters of the considered DL models.

2.2. Long Short-Term Memory

Artificial neural networks (ANNs) are computer systems modeled after the structure and function of the human brain. They use interconnected neurons to learn and process information and can be used for tasks such as classification and prediction. Different types of ANNs include shallow networks, deep networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs) [29,30].

Forecasting time series data is difficult because traditional neural network models only consider current input data, without taking into account historical information. RNNs are able to consider previous inputs, but LSTM networks can retain historical data for even longer periods. In this research, an LSTM type of ANN will be used to predict the direction and magnitude of changes in solar energy generation. Unlike traditional ANNs, LSTM networks use memory cells to retain long sequences of data, as described in [31]. An LSTM cell includes three gates: forget, input, and output. These gates determine which data is retained in memory and which is discarded.

Data that is input into an LSTM network goes through the forget gate, which determines if it should be forgotten by the neural cell. The function of the forget gate f_t can be represented by Equation (1).

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

The current forget gate, denoted by f_t, is between 0 and 1 and is controlled by the sigmoid function σ. The weight matrices W_f and U_f, the bias value b_f, the input values x_t and the previous data h_{t-1} are all used to calculate the forget gate value.

The input gate is the next stage where data is processed. The behavior of the input gate is represented by Equations (2) and (3). The result of the sigmoid function, denoted by i_t, determines which data should be stored in the memory cell. The weight matrices W_i, U_i, and the bias value b_i are the parameters that need to be adjusted for this gate.

$$i_t = \sigma(W_i X_t + U_i h_{t-1} + b_i) \tag{2}$$

The input gate's output is determined by the potential update vector $\tilde{C}_t$, which is calculated by Equation (3). The vector is the output of the *tanh* function, which ranges between -1 and 1.

$$\tilde{C}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{3}$$

Lastly, to determine the final state of the output gate, Equation (4) is utilized to obtain the potential collection of values for the vector and which information should be updated.

$$C_t = F_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{4}$$

The values to be removed are represented by C_{t-1}. The new data that need to be stored in the memory are given by $f_t \odot \tilde{C}_t$, while the fresh information that is retained in the cell is $i_t \odot \tilde{C}_t$. The last output gate that determines the real values of the hidden layers can be obtained by Equation (5). The gate o_t denotes the sigmoid function, while the output is the product of the sigmoid output and *tanh* from the previous couple of gates' outputs, determined by Equation (6).

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{5}$$

$$h_t = o_t \odot tanh(C_t) \tag{6}$$

The LSTM model has become popular in scientific research because of its ability to perform well in time series prediction tasks. This can be seen in the recent successful applications in areas such as stock price prediction [32–34], medical applications [35,36], estimating COVID-19 cases [37,38], and petroleum production [39] to name the few.

2.3. Bidirectional Long Short-Term Memory

A BiLSTM processes the inputs from both the past and the future, requiring two directions of the same sequence simultaneously. The BiLSTM model is comprised of a forward and backward LSTM, designed to handle the exploding gradient problem. The forward line handles past information opposite to the backward line, also known as the reverse LSTM, which handles the future information for the input data. The information going through both lines is fused and forwarded. As a result, the BiLSTM has a better performance compared to the basic RNN and LSTM networks with respect to data processing. The hidden and output layers are determined at time t as stated by Equations (7)–(9):

$$\overrightarrow{h}_t = \sigma(\overrightarrow{W}_i x_t + \overrightarrow{V}_i \overrightarrow{h}_{t-1} + \overrightarrow{b}) \tag{7}$$

$$\overleftarrow{h}_t = \sigma(\overleftarrow{W}_i x_t + \overleftarrow{V}_i h_{t+1} + \overleftarrow{b}) \tag{8}$$

$$y_t = \sigma(U[\overrightarrow{h}_t; \overleftarrow{h}_t] + c) \tag{9}$$

The structure of BiLSTM can be seen in Figure 1.

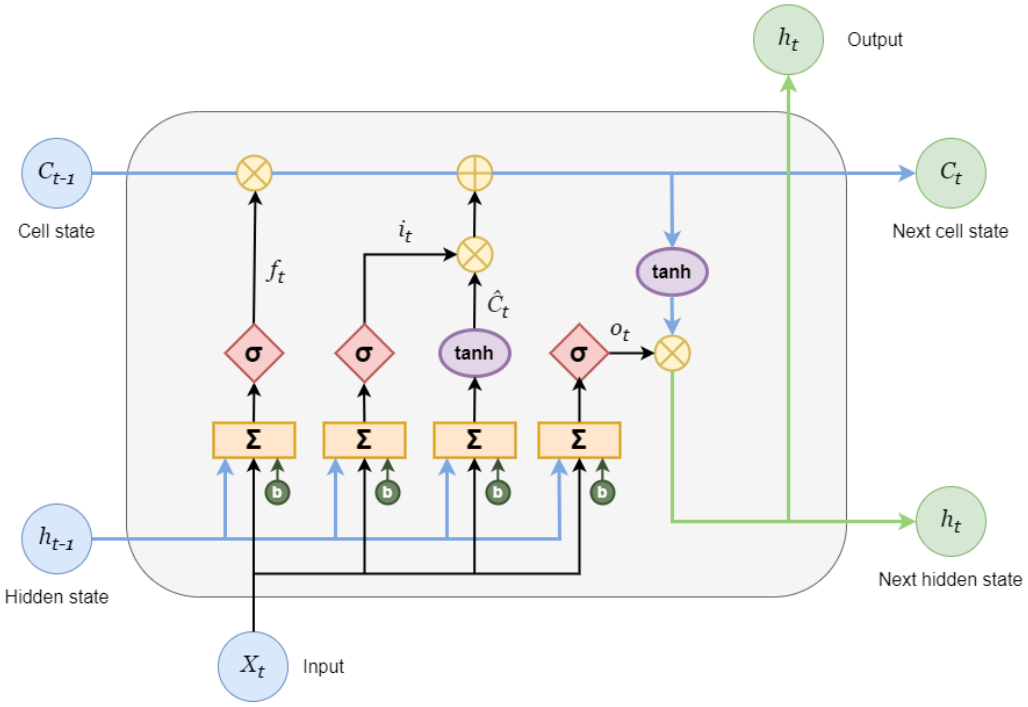

Figure 1. BiLSTM structure.

2.4. Metaheuristic Optimization

NP-hard problems, which are prevalent in computer science, require the use of stochastic algorithms, such as metaheuristics, as deterministic approaches are infeasible. Metaheuristic algorithms can be grouped into different families based on the natural phenomena they use to guide the search process, such as, for example, evolution or ant behavior [40–42].

The main families of metaheuristic algorithms are nature-inspired methods (further divided into genetic algorithms and swarm intelligence), methods based on some physical phenomena (such as storms, gravitational waves, electromagnetic field etc.), algorithms that mimic human behavior (for example teaching and learning, or brainstorming process), and approaches that use mathematical laws to guide the search (such as oscillations of the fundamental trigonometrical functions).

Swarm intelligence algorithms are based on the behavior of large groups of simple individuals, such as animals in a swarm, who display coordinated and complex behavior during activities like hunting, feeding, mating, and migration [43,44]. Swarm intelligence methods are known to be effective at solving a wide range of NP-hard problems in real life. Some notable examples of algorithms in this group include ant colony optimization (ACO) [45], particle swarm optimization (PSO) [46], artificial bee colony (ABC) [47], bat algorithm (BA) [48,49], and firefly algorithm (FA) [50]. On the other hand, algorithms that use mathematical functions to guide the search include the sine-cosine algorithm (SCA) [51] and the arithmetic optimization algorithm [52]. These are known for their efficiency and effectiveness.

The existence of a wide variety of population-based algorithms can be attributed to the no-free-lunch theorem (NFL) [53], which states that there is no universal approach that can find the best solution for all optimization challenges. As a result, one method may perform well for one problem but poorly for another, and hence the reason for the diversity of metaheuristic methods and the need to discover and adjust the appropriate algorithm for each specific optimization task.

Population-based methods have been used to solve a wide range of real-world problems in recent years, including monitoring and forecasting COVID-19 cases [54,55], cloud and cloud-edge computing [56–59], energy-aware and carbon-efficient cloud computing [60], IoT and sensor networks tuning [61–64], feature selection [65], image classification and medical applications [66,67], global optimization problems [68,69], credit card fraud detection [70,71], air pollution forecasting [72,73], network intrusion detection [74,75], and optimization of a variety ML models [76–81].

3. Methods

The original form of the metaheuristic that will be used in the current study is outlined next, followed by its proposed improved variant, based on the hybridization with complementing working mechanisms from other recent nature-inspired approaches.

3.1. Original Reptile Search Algorithm

The cooperative and highly coordinated hunting technique exhibited by the crocodiles, which consists of surrounding the prey, followed by the attack, has been a motivation for the recent reptile search algorithm (RSA) [82]. The initialization phase starts by generating a matrix X of arbitrary solutions $x_{i,j}$ with respect to Equation (10), where i denotes the index of the individual, j represents its current location, N marks the total count of individuals, while n represents the dimensionality of the particular problem [82]:

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,j} & x_{1,n-1} & x_{1,n} \\ x_{2,1} & \cdots & x_{2,j} & \cdots & x_{2,n} \\ \cdots & \cdots & x_{i,j} & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N-1,1} & \cdots & x_{N-1,j} & \cdots & x_{N-1,n} \\ x_{N,1} & \cdots & x_{N,j} & x_{N,n-1} & x_{N,n} \end{bmatrix} \quad (10)$$

Equation (11) is utilized to produce arbitrary individuals. Here, *rand* is an arbitrary value within the range $[0,1]$, and LB and UB denote the lower and upper limits of the search domain for the particular problem [82].

$$x_{ij} = \text{ rand } \times (UB - LB) + LB, j = 1, 2, \ldots, n \tag{11}$$

The search phase has been split into two main procedures (encircling the target, and hunting) accompanied by the four distinctive behaviors to emphasize both exploration and exploitation. Exploration employs two walking methods exhibited by crocodiles: elevated walk and stomach walk. The main goal of the crocodiles here is to widen the search realm and provide support for the second hunting phase. The elevated walk technique is employed when $t \leq \frac{T}{4}$, while the stomach walk is triggered when $t > \frac{T}{4}$ and $t \leq 2\frac{T}{4}$. Equation (12) is responsible for updating of the crocodile's position [82]:

$$x_{(i,j)}(t+1) = \begin{cases} \text{Best}_j(t) \times -\eta_{(i,j)}(t) \times \beta - R_{(i,j)}(t) \times \text{ rand}, t \leq \frac{T}{4} \\ \text{Best }_j(t) \times x_{(r_1,j)} \times ES(t) \times \text{ rand}, t > \frac{T}{4} \text{ and } t \leq 2\frac{T}{4} \end{cases} \tag{12}$$

$$\eta_{(i,j)} = \text{Best}_j(t) \times P_{(i,j)} \tag{13}$$

where Best_j denotes the latest best individual at location j, t represents the ongoing iteration, while T specifies the maximum count of iterations. The hunting operator $\eta_{(i,j)}$ has been defined by Equation (13), where β represents the sensitive variable fixed at 0.1, governing the exploration accuracy [82].

The search space is shrunk by applying the reduction function, defined by Equation (14), where r_1 is an arbitrary value within range $[1, N]$, $x_{r1,j}$ denotes the i_{th}'s solution arbitrary position, while ϵ denotes a small value.

$$R_{(i,j)} = \frac{\text{Best}_j(t) - x_{(r_1,j)}}{\text{Best}_j(t) + \epsilon} \tag{14}$$

Equation (15) calculates the probability ratio, named "Evolutionary Sense", that arbitrarily alternates from -2 to 2 as rounds pass by [82]:

$$ES(t) = 2 \times r_2 \times \left(1 - \frac{1}{T}\right) \tag{15}$$

where r_2 denotes an arbitrary value inside $[-1, 1]$.

Equation (16) is used to determine the percentage difference between the positions of the observed and best-attained individual [82]:

$$P_{(i,j)} = \alpha + \frac{x_{(i,j)} - M(x_i)}{\text{Best}_j(t) \times \left(UB_{(j)} - LB_{(j)}\right) + \epsilon} \tag{16}$$

where α marks the sensitive parameter, with a predefined value 0.1, controlling the fluctuations among potential individuals suitable for the cooperated hunt. The respective upper and lower limits of the j_{th} location have been specified by $UB_{(j)}$ and $LB_{(j)}$ [82].

The average position $M(x_i)$ of the ith individual has been given by Equation (17) [82].

$$M(x_i) = \frac{1}{n} \sum_{j=1}^{n} x_{(i,j)} \tag{17}$$

The RSA exploitation procedure is split into hunting coordination (in case $t \leq 3\frac{T}{4}$ and $t > \frac{T}{2}$) and cooperation (if $t \leq T$ and $t > 3\frac{T}{4}$) methods, aiming to intensify the local

investigation of the search realm and closing to the best potential individual. The hunting behavior exhibited by crocodiles can be summarized by Equation (18) [82].

$$x_{(i,j)}(t+1) = \begin{cases} \text{Best}_j(t) \times P_{(i,j)}(t) \times \text{ rand}, t \leq 3\frac{T}{4} \text{ and } t > \frac{T}{2} \\ \text{Best}_j(t) - \eta_{(i,j)}(t) \times \epsilon - R_{(i,j)}(t) \times \text{ rand}, t \leq T \text{ and } t > 3\frac{T}{4} \end{cases} \quad (18)$$

The elementary RSA displays the time complexity of the $O(N \times (T \times D + 1))$, where N represents the count of candidates, T denotes the count of rounds, while D stands for the dimensionality of the solutions space.

3.2. Improved RSA Algorithm

The RSA is one of the most recent metaheuristics, and in its basic implementation, it proved as a very powerful optimizer [82], although it has some observed drawbacks. The comprehensive simulations with benchmark functions (bound-constrained and constrained taken from the Congress on Evolutionary Computation benchmark suites) have exposed that the algorithm does not have enough power in exploitation, especially in later rounds of the execution, despite having satisfactory exploration capability.

More specifically, by executing extensive simulations with the original RSA, it was observed that the algorithm's search process in most runs manages to find a proper domain of the search area; however, it does not have enough exploitation power to execute a fine-tuned search around this region. As a consequence, more iterations are needed for the basic RSA to converge. This issue can be also viewed from the perspective of diversification-intensification trade-off (balance), which is known as one of the most common problems with metaheuristics [79,83], and in the case of RSA it is noticed that this trade-off is biased towards exploration.

On the contrary, the firefly algorithm (FA) is famous of its potent exploitation, as described in [50]. Therefore, one reasonable way to improve the RSA is by hybridizing it with the FA and hence the main idea proposed by this research is to address the RSA's identified flaws by creating a low-level hybrid approach between RSA and FA, as both algorithms complement each other. However, when creating hybrid metaheuristics, researchers need to be careful in order to establish a proper trade-off between exploitation and exploration. At the beginning of a run, when the most important goal is to identify an optimal zone of the search region, exploration should be more intensive, conversely, as the iterations progress, fine-tuned search around the current best solutions is more needed, hence the exploitation should be strengthened. In this particular case, if the FA search expression is triggered at the start of a run, in some executions, the algorithm may not be able to find the optimum region, therefore proposed low-level hybrid metaheuristics takes this into account by introducing hard-coded control parameters.

The exact mechanism is defined as follows. When the algorithm begins executing, the individuals are updated with respect to the RSA search Equation (12), but in later iterations, when the search should be more focused on converging within the promising areas of the search domain, the exploitation is improved by employing the FA search procedure, defined by Equation (19), where α denotes the randomization variable, while κ is a random number taken from the Gaussian distribution [50]. Finally, $r_{i,j}$ represents the distance among individuals i and j.

$$X_i^{t+1} = X_i^t + \beta_0 \cdot e^{-\gamma r_{i,j}^2}(X_j^t - X_i^t) + \alpha^t(\kappa - 0.5) \quad (19)$$

The alternation between the two search options in the later iterations during the algorithm run is controlled by two additional parameters. The first parameter, named varying search vs, has the role to activate the mixed search procedure, and it is triggered when $t > vs$, by starting to alternate between the elementary RSA and FA search mechanisms. This parameter has the fixed value of $maxIter/5$ (in the executed simulations

it is equal to 1, as there is a maximum of five rounds), which was established through empirical experiments.

The second introduced control parameter, named search mode sm, is used to determine for every individual in the population if the search should be performed by utilizing the RSA or FA mechanism. Every individual generates an arbitrary value rnd within $[0, 1]$, and, if $rnd < sm$, it will execute the RSA search, else it will proceed with the FA search. The value of this parameter is decreased dynamically as the rounds go by, giving additional focus to the FA search when the algorithm has converged towards the promising regions of the search space. The starting value of this parameter is fixed to 0.8, and it is reduced as defined by Equation (20), which was established through empirical experiments.

$$sm_t = sm_{t-1} - (sm_{t-1}/10) \tag{20}$$

An additional mechanism that has been employed was inspired by the *trial* parameter in the ABC algorithm [47]. This *trial* parameter is initially set to 0, and if the solution is not improved in the current round, the value of *trial* is increased, and when it reaches $limit = 2$ threshold (empirically determined for this particular problem $2 * (solutions/maxIter)$, which is in this case 2), it is removed from the population, and replaced with the quasi-reflective opposite solution x_{qrl}, that has been produced by employing the quasi-reflection-based learning (QRL) procedure [84]. Previous research indicated that the QRL can efficiently produce the solution in the opposite section of the search domain [85].

The hybridized algorithm was simply given the name hybrid RSA–HRSA, and the pseudo-code is presented in Algorithm 1.

Algorithm 1 Pseudo-code of introduced hybrid reptile search algorithm (HRSA)

Initialization phase
Initialize RSA parameters α, β, etc.
Initialize $vs = maxIter/5$
Initialize $trial = 0$ for every individual
Initialize the locations of the individuals in a random fashion. $X : i = 1, ..., N$.
while $(t < T)$ **do**
    Determine the objective function for the candidate solutions (X)
    Determine the best individual so far
    Update the ES using Equation (15)
    The beginning of the RSA
    for (j=1 to N) **do**
        for (j=1 to n) **do**
            if $t > vs$ **then**
                Produce random value rnd
            end if
            if $(t > vs)$ & $!(rnd < sm)$ **then**
                Execute FA search according to Equation (19)
            else
                Execute RSA search
                Update the η, R, P and values using Equations (13), (14) and (16), respectively
                if $(t \leq \frac{T}{4})$ **then**
                $x_{(i,j)}(t+1) = Best_j(t) \times -\eta_{(i,j)}(t) \times \beta - R_{(i,j)}(t) \times rand,$ ▷ **{High walking}**
                else if $(t \leq 2\frac{T}{4}$ *and* $t > \frac{T}{4})$ **then**
                $x_{(i,j)}(t+1) = Best_j(t) \times x_{(r_1,j)} \times ES(t) \times rand,$ ▷ **{Belly walking}**
                else if $(t \leq 3\frac{T}{4}$ *and* $t > 2\frac{T}{4})$ **then**
                $x_{(i,j)}(t+1) = Best_j(t) \times P_{(i,j)}(t) \times rand,$ ▷ **{Hunting coordination}**
                else
                $x_{(i,j)}(t+1) = Best_j(t) - \eta_{(i,j)}(t) \times \epsilon - R_{(i,j)}(t) \times rand,$ ▷ **{Hunting cooperation}**
                end if
            end if
        end for
        if solution not improved **then**
            $trial = trial + 1$
            if $trial = limit$ **then**
                Replace current solution with quasi-reflective opposite solution x_{qrl}
                $trial = 0$
            end if
        end if
    end for
    $t = t + 1$
end while
Return the best solution $(Best(X))$

3.3. Multivariate Time-Series Prediction Framework Based on the HRSA and Complexity of Introduced Approach

The suggested multivariate time-series prediction framework is based on the solutions within the population, where every individual represents a potential LSTM or BiLSTM network structure. Each of these solutions is denoted by the collection of chosen hyperparameters.

The research presented in this manuscript takes into consideration a set of 6 hyperparameters that are subjected to the tuning process, which include the count of units in the first layer, the learning rate, the count of epochs used for training, the dropout rate, the count of layers within the structure, and the count of units in the second layer (in case a two-layered structure has been produced).

Moreover, the conducted research aims to examine the capabilities of computationally light structures, focusing on the simple model architectures, consisting of a maximum of two layers (therefore the limit for this parameter was set to 2 for every individual in the population). Solutions utilize a flat encoding, as every individual is comprised of six encoded parameters (one for every hyperparameter subjected to tuning). This particular problem is a mixed NP-hard challenge, as certain hyperparameters have integer values, while others are continuous variables.

The boundaries for every hyperparameter are detailed in Section 4. The fitness function that is required to be minimized for every individual in the population is the overall mean square error (MSE), as it is discussed in more detail in the next Section as well. In the end, the best-produced model has been interpreted by utilizing the SHAP method. The entire framework is summarized in Figure 2.

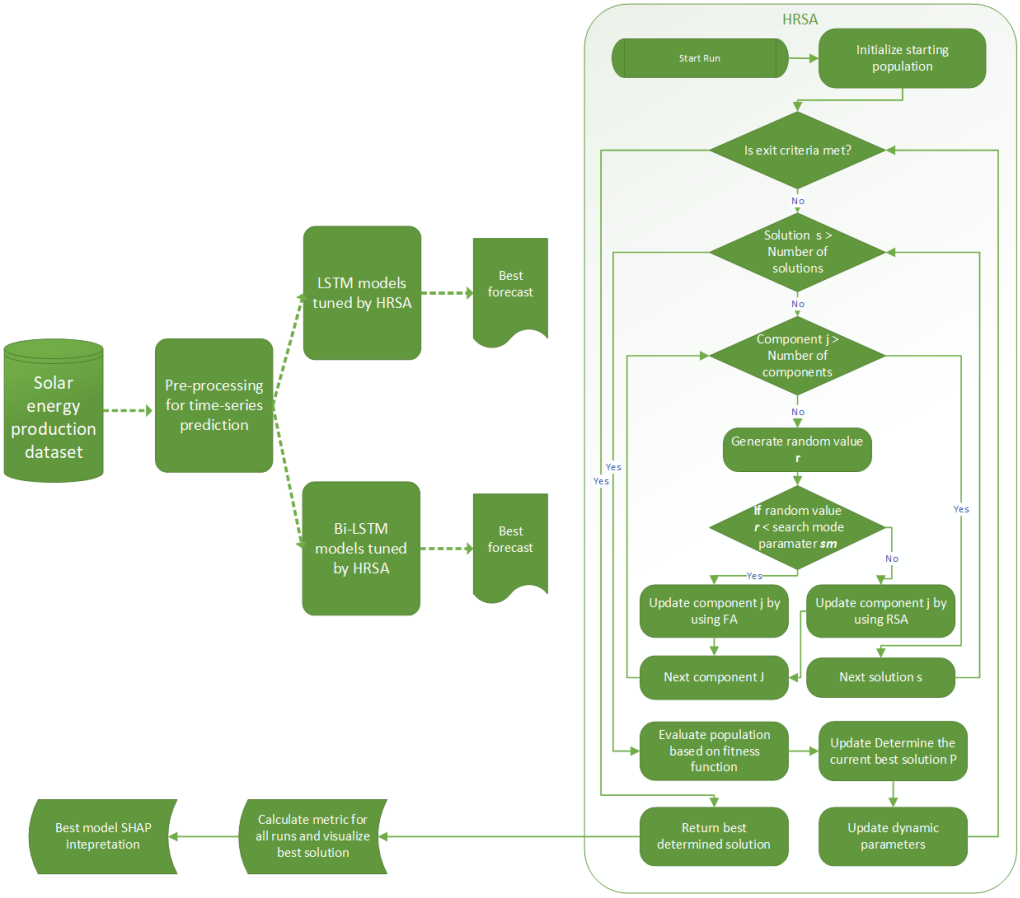

Figure 2. Flowchart of the proposed framework.

Finally, the complexity of the proposed method is worth mentioning. When comparing the complexity of metaheuristics, execution time is not an objective criterion because it depends on the computation platform to a large extent. Therefore, according to the well-

established practice from the modern literature [50], the complexity of metaheuristics should be calculated based on the number of fitness function evaluations (FFEs) in the run since it is the most computationally intensive (expensive) operation. This particularly applies to this research because when tuning LSTM/BiLSTM, one FFE requires the deep learning model to be generated based on the metaheuristics solution and then trained, which is a very resource-intensive operation.

Although the introduced HRSA algorithm employs a mechanism that replaces all solutions whose *trial* attribute reaches the threshold value with the generated QRL individual, this procedure does not add overhead in terms of FFEs because the newly created solution is not evaluated at this point of execution. Therefore, the computation complexity in terms of FFEs of the proposed HRSA does not increase from the baseline RSA, and HRSA complexity is given as $O(N) = N + N \cdot T$.

4. Experimental Results

This section will first address the real-world problem under consideration and then discuss the experimental setup and findings using the proposed enhanced metaheuristic in comparison to competing alternatives for the parametrization of the two LSTM networks.

4.1. Data set Description

The study utilizes two data sets that were created from actual data sources. One data set is related to hourly energy demand and production and it is sourced from the ENTSO-E portal (https://transparency.entsoe.eu/dashboard/show, accessed on 19 January 2023). The second data set used in the research pertains to weather data and it is obtained from the OpenWeather API (https://openweathermap.org/api, accessed on 19 January 2023). The weather data in the study include hourly meteorological measurements for Valencia, Spain. The energy data contain hourly details about the quantity of power generated from renewable sources in Spain. The entire collection of data sources is openly available to the general public (https://www.kaggle.com/datasets/nicholasjhana/energy-consumption-generation-prices-and-weather, accessed on 19 January 2023). The data set combines both sources of data at an hourly pace for all variables. To handle the high computational needs of the experiments that involve comparative analysis, a smaller, representative subset of the available data was utilized. The final data set used in the research includes data from 1 October 2018 to 28 December 2018. It can be visualized in Figure 3.

The data collection includes the most important hourly variables for the multivariate prediction of the PV output. It is further divided into training, validation, and testing subsets with 70%, 10%, and 20% of the data, highlighted in blue, green, and red colors in Figure 3, respectively.

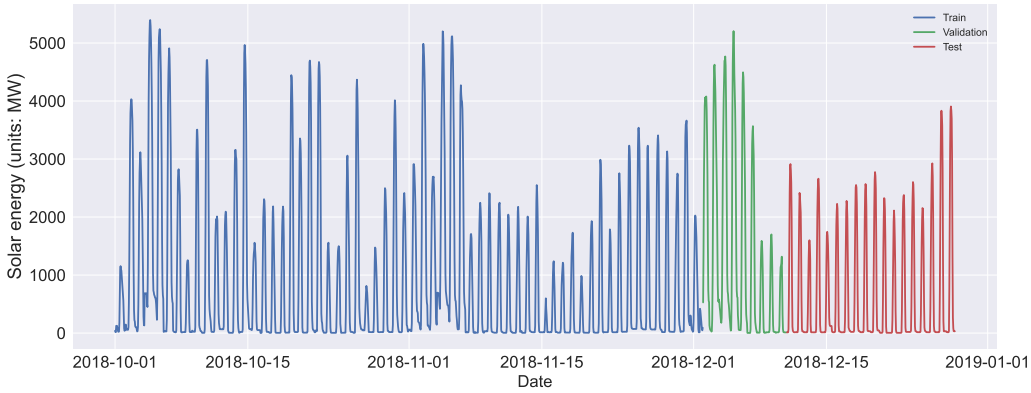

Figure 3. Visualization of the utilized solar energy data set.

4.2. Metrics

Aiming to evaluate the overall performance of the regarded approaches, four traditional ML measurements have been employed, including mean absolute error (MAE),

mean square error (MSE), root mean square error ($RMSE$), and coefficient of determination (R^2), that can be calculated with respect to Equations (21)–(24), respectively.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x}_i| \tag{21}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\frac{(x_i - \hat{x}_i)^2}{x_i} \tag{22}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2} \tag{23}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \tag{24}$$

x_i being the actual value, while $\hat{x}_i$ corresponds to the predicted value for the i-th observation, $\bar{x}$ represents the arithmetic mean value of actual data, and finally, n denotes the number of data points within the data set.

The prediction challenge represents an instance of a minimization task. The tuning process takes the overall MSE ($oMSE$) as the fitness function defined according to Equation (25).

$$Obj = min(oMSE) \tag{25}$$

4.3. Experimental Setup

The presented research utilizes two experiments for the regarded solar power data set. Both LSTM and BiLSTM models were independently developed, tuned, and validated for the three-step ahead forecasting task. The lag value, i.e., the one controlling how much historical data is taken, was set to 6.

As the executed experiments are extremely computationally intensive, simulations were conducted with a relatively low number of iterations and individuals in the population. Each tested metaheuristic algorithm used 5 solutions in the population ($N = 5$), and 5 rounds ($T = 5$), over 15 separate runs ($R = 15$). Moreover, early stopping with a $patience = epochs/3$ was used to address the overfitting issue. The neural networks that were used are relatively simple and efficient, consisting of maximum of two layers.

The simulation framework was developed in the Python programming language with the help of a collection of supporting ML libraries such as TensorFlow 2.0, Keras, Pandas, Sklearn, Numpy, Seaborn, Matplotlib and Shap. The utilized hardware consisted of Intel i9 11900K CPU, accompanied by 128GB RAM memory, and finally the Nvidia 1080 11G GPU.

When making predictions for the next three steps, the output layer of the network produces three values, each corresponding to a prediction for one step in the future. The accuracy of each individual step prediction is measured independently by utilizing the above-mentioned metrics, and the overall accuracy of the network is obtained by evaluating its performance on all three-step predictions. Comprehensive measurements were tagged as $oR2$, $oMAE$, $oMSE$, and $oRMSE$, and they are not simply taken as the arithmetic mean values obtained from the individual step calculations, due to the fact that when performing predictions for several steps in the future, the initial data points were not present to be compared. In the case of single-step forecasts, the overall measurements are equal to the individual metrics.

The suggested HRSA and competitor algorithms were employed to tune the hyperparameters of the observed models. Aiming to retain the computational requirements feasible, interval constraints were established for each of the tuned parameters. The set of parameters that are tuned, accompanied by their respective numeric type, are provided as follows:

- count of units in the first layer—limits $[100, 200]$, integer variable,

- learning rate—boundaries $[0.001, 0.01]$, real variable,
- count of epochs used for training—in range $[300, 600]$, integer variable,
- dropout—range $[0.05, 0.2]$, real variable,
- number of layers—1 or 2 layers, integer,
- count of units in the second layer—boundaries $[100, 200]$, integer variable

This research used seven other powerful metaheuristics to compare the performance level of the proposed HRSA method. The contending algorithms were the elementary RSA, ABC [47], FA [50], SSA [86], Harris Hawks Optimization (HHO) [87], SCA [51] and Chimp optimization algorithm (ChOA) [88]. Every competing algorithm was implemented independently by the authors, by employing the suggested control parameters' configuration as proposed by their respective creators.

4.4. Simulation Results

This subsection brings forward the simulation results of the executed LSTM and BiLSTM experiments. All tables containing the outcomes have the best result of each category marked with bold text.

4.4.1. LSTM Experiments

Table 1 depicts the LSTM simulation results with respect to the fitness function (MSE), presenting the standard metrics that include the best, worst, mean and median values, accompanied by the standard deviation and variation determined over 15 separate runs. It is possible to note, based on Table 1, that the suggested LSTM-HRSA approach attained the best scores in terms of best, worst, mean, and median metrics. On the other hand, the LSTM-RSA method established the most stable results, indicated by the best standard deviation and variance measurements.

Table 1. Overall results of the objective function for LSTM experiments. The best result in each column is written in bold.

Method	Best	Worst	Mean	Median	Std	Var
LSTM-HRSA	**0.0137**	**0.0142**	**0.014**	**0.0139**	2.23×10^{-4}	4.99×10^{-8}
LSTM-RSA	0.0142	0.0144	0.0143	0.0143	**6.58** $\times 10^{-5}$	**4.33** $\times 10^{-9}$
LSTM-ABC	0.014	0.0147	0.0144	0.0144	2.46×10^{-4}	6.06×10^{-8}
LSTM-FA	0.0137	0.015	0.0145	0.0146	4.97×10^{-4}	2.47×10^{-7}
LSTM-SSA	0.0142	0.015	0.0145	0.0144	3.51×10^{-4}	1.23×10^{-7}
LSTM-HHO	0.0138	0.015	0.0144	0.0147	4.69×10^{-4}	2.20×10^{-7}
LSTM-SCA	0.0138	0.0148	0.0144	0.0145	3.76×10^{-4}	1.41×10^{-7}
LSTM-ChOA	0.0143	0.0147	0.0145	0.0144	1.63×10^{-4}	2.67×10^{-8}

Tables 2 and 3 present the normalized and denormalized metrics with respect to the one-step ahead, two-step ahead, and three-step ahead predictions, together with the overall results, achieved by the best run of each one of the 8 argued LSTM models. LSTM-HHO achieved the best scores for one-step and two-step ahead forecasts, while LSTM-SCA attained the best outcomes for three-step ahead. Nevertheless, when analyzing the best overall results, the suggested LSTM-HRSA model achieved the best scores for each regarded metric - MSE (the fitness function), R^2, MAE, and RMSE. Finally, Table 4 presents the best set of hyperparameters determined by each metaheuristic.

Table 2. Normalized metrics for one-step, two-step, three-step ahead, and overall predictions for LSTM experiments. The best result in each column is written in bold.

	Error Indicator	LSTM-HRSA	LSTM-RSA	LSTM-ABC	LSTM-FA	LSTM-SSA	LSTM-HHO	LSTM-SCA	LSTM-ChOA
One-step ahead	R^2	0.426279	0.379680	0.405602	0.411128	0.389334	**0.462406**	0.403854	0.398311
	MAE	0.086843	0.092011	0.090283	0.089682	0.092903	**0.084873**	0.090301	0.091209
	MSE	0.019809	0.021418	0.020523	0.020332	0.021084	**0.018562**	0.020583	0.020775
	RMSE	0.140744	0.146348	0.143258	0.142590	0.145205	**0.136241**	0.143468	0.144134
Two-step ahead	R^2	0.671760	0.663221	0.671169	0.672800	0.663446	**0.687558**	0.665462	0.655853
	MAE	0.069301	0.071177	0.073110	0.070798	0.073262	**0.069035**	0.072266	0.074587
	MSE	0.011333	0.011628	0.011354	0.011297	0.011620	**0.010788**	0.011551	0.011882
	RMSE	0.106457	0.107833	0.106553	0.106288	0.107797	**0.103864**	0.107474	0.109006
Three-step ahead	R^2	0.714377	0.720823	0.705576	0.723159	0.716281	0.650704	**0.730167**	0.703660
	MAE	0.067204	0.066277	0.071115	**0.064111**	0.069485	0.073157	0.065808	0.067980
	MSE	0.009862	0.009639	0.010166	0.009558	0.009796	0.012060	**0.009317**	0.010232
	RMSE	0.099306	0.098179	0.100825	0.097767	0.098975	0.109819	**0.096522**	0.101152
Overall Results	R^2	**0.604139**	0.587908	0.594116	0.602362	0.589687	0.600223	0.599828	0.585941
	MAE	**0.074450**	0.076488	0.078169	0.074863	0.078550	0.075688	0.076125	0.077925
	MSE	**0.013668**	0.014228	0.014014	0.013729	0.014167	0.013803	0.013817	0.014296
	RMSE	**0.116910**	0.119282	0.118381	0.117172	0.119025	0.117487	0.117545	0.119567

Table 3. Denormalized metrics for one-step, two-step, three-step ahead, and overall predictions for LSTM experiments. The best result in each column is written in bold.

	Error Indicator	LSTM-HRSA	LSTM-RSA	LSTM-ABC	LSTM-FA	LSTM-SSA	LSTM-HHO	LSTM-SCA	LSTM-ChOA
One-step ahead	R^2	0.426279	0.379680	0.405602	0.411128	0.389334	**0.462406**	0.403854	0.398311
	MAE	468.432157	496.305283	486.988569	483.743753	501.119031	**457.804513**	487.083637	491.982836
	MSE	576,343.262267	623,155.500741	597,114.680222	591,563.458038	613,456.735227	**540,051.415624**	598,870.356156	604,439.245545
	RMSE	759.172749	789.401989	772.731959	769.131626	783.234789	**734.881906**	773.867144	777.456909
Two-step ahead	R^2	0.671760	0.663221	0.671169	0.672800	0.663446	**0.687558**	0.665462	0.655853
	MAE	373.809655	383.928146	394.353153	381.882793	395.174567	**372.374631**	389.800946	402.323210
	MSE	329,740.096164	338,318.466458	330,333.731958	328,695.304008	338,092.018974	**313,869.497973**	336,067.378930	345,719.755906
	RMSE	574.230003	581.651499	574.746668	573.319548	581.456807	**560.240572**	579.713187	587.979384
Three-step ahead	R^2	0.714377	0.720823	0.705576	0.723159	0.716281	0.650704	**0.730167**	0.703660
	MAE	362.500937	357.500181	383.596842	345.813909	374.802066	394.607170	**354.967121**	366.683273
	MSE	286,928.697207	280,452.719981	295,770.081252	278,106.126201	285,015.651643	350,892.734230	**271,065.861908**	297,694.506240
	RMSE	535.657257	529.577870	543.847480	527.357683	533.868572	592.361996	**520.639858**	545.613880
Overall Results	R^2	**0.604139**	0.587908	0.594116	0.602362	0.589687	0.600223	0.599828	0.585941
	MAE	**401.580916**	412.577870	421.646188	403.813485	423.698555	408.262105	410.617235	420.329773
	MSE	**397,670.685213**	413,975.562393	407,739.497811	399,454.962749	412,188.135281	401,604.549276	402,001.198998	415,951.169230
	RMSE	**630.611358**	643.409327	638.544828	632.024495	642.018797	633.722770	634.035645	644.942764

Table 4. Best attained collection of LSTM parameter values by each observed algorithm.

Method	Units in First Layer	Learning Rate	Epochs	Dropout	Layers	Units in Second Layer
LSTM-HRSA	150	0.006738	551	0.176168	2	149
LSTM-RSA	100	0.010000	588	0.060360	1	/
LSTM-ABC	134	0.007830	458	0.200000	1	/
LSTM-FA	100	0.008356	389	0.050000	2	200
LSTM-SSA	147	0.006232	600	0.148325	1	/
LSTM-HHO	171	0.008932	450	0.150124	2	143
LSTM-SCA	163	0.010000	600	0.200000	2	100
LSTM-ChOA	168	0.006622	585	0.141010	1	/

The graphical representation of the conducted LSTM simulations are given in Figures 4 and 5, where the results of the MSE (fitness function) and R^2 metrics have been outlined through the convergence graphs of the best runs, box plot diagrams and violin plots over 15 runs, as well as swarm plots that show the diversity of the population during the last round of the best run of each algorithm. The swarm plots reveal that the ABC expresses the highest diversity of solutions in the last iteration, contrary to the proposed HRSA method, where all 5 individuals from the population ended up in the proximity of the best region of the search space. This particular kind of behavior is anticipated, as the ABC employs the *limit* control variable that assures high diversity of the population. On the other hand, HRSA has powerful exploitation in the direction of the recent best individual.

Figure 6 presents the Kernel Distribution Estimation (KDE) diagrams for the fitness function and R^2, visualizing the probability density function. According to these diagrams, which show the distribution of the outcomes of the runs, the results are not coming from the normal distribution. In the end, Figure 7 presents the best predictions of the LSTM tuned by the proposed HRSA method, compared to the predictions of the LSTM tuned by the elementary RSA.

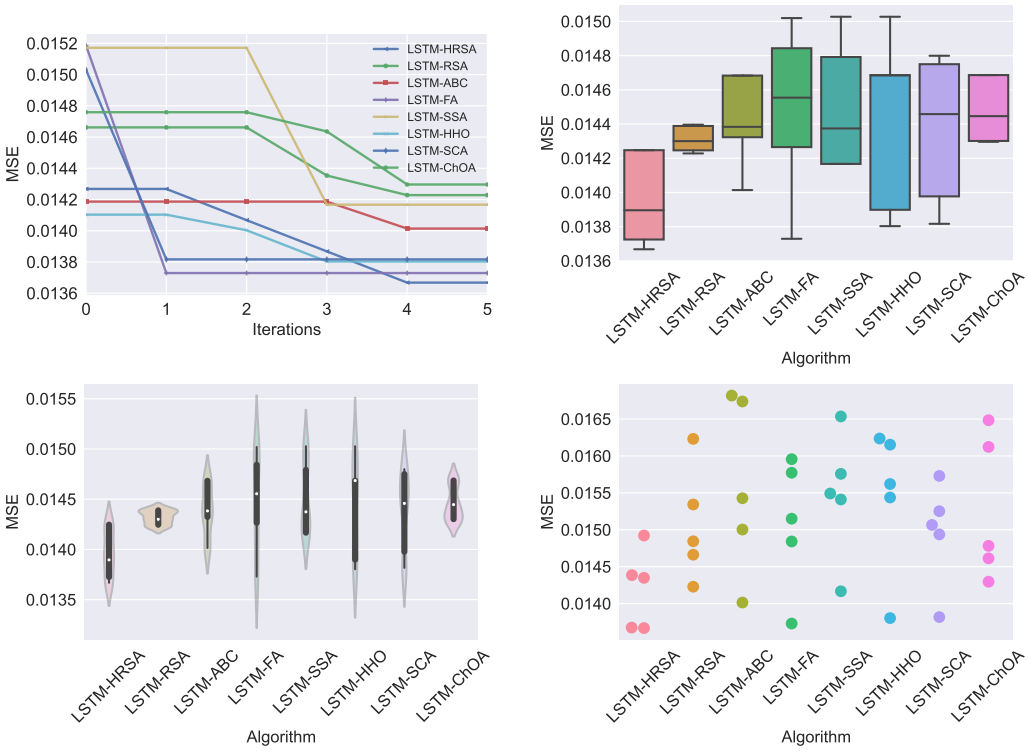

Figure 4. Visual LSTM simulations for each of the 8 optimization methods with respect to the convergence, box plot, violin diagrams, and swarm diversity diagrams of the fitness function (MSE).

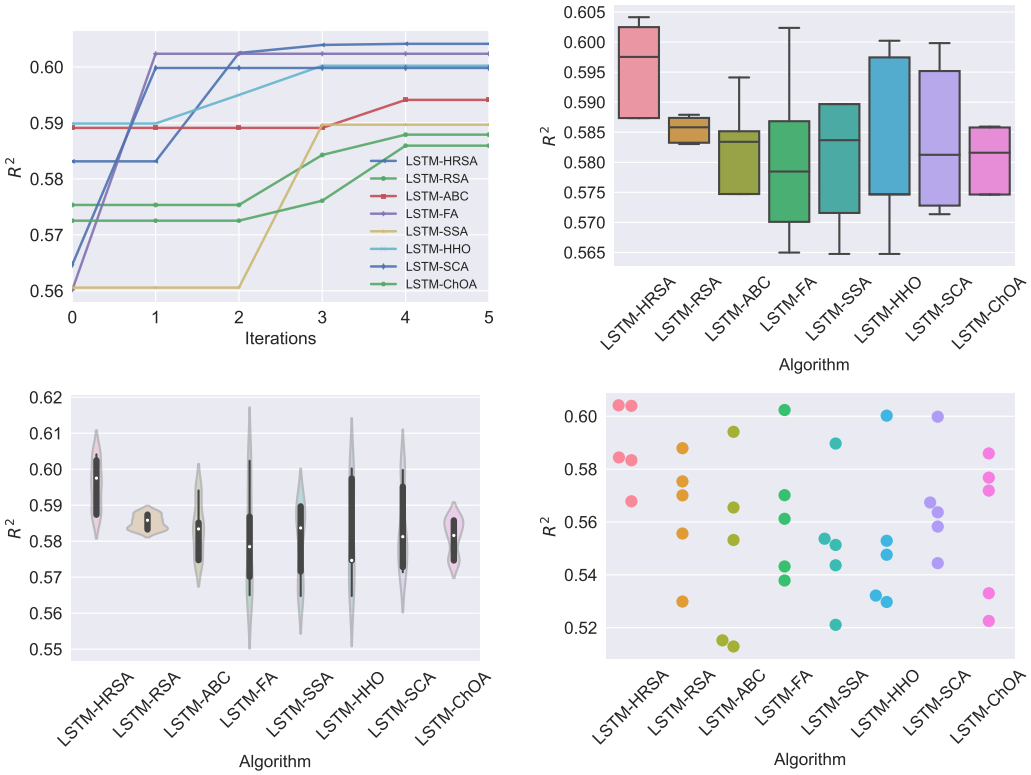

Figure 5. Visual LSTM simulations for each of 8 optimization methods with respect to the convergence, box plot, violin diagrams, and swarm diversity diagrams of the R^2 criterion.

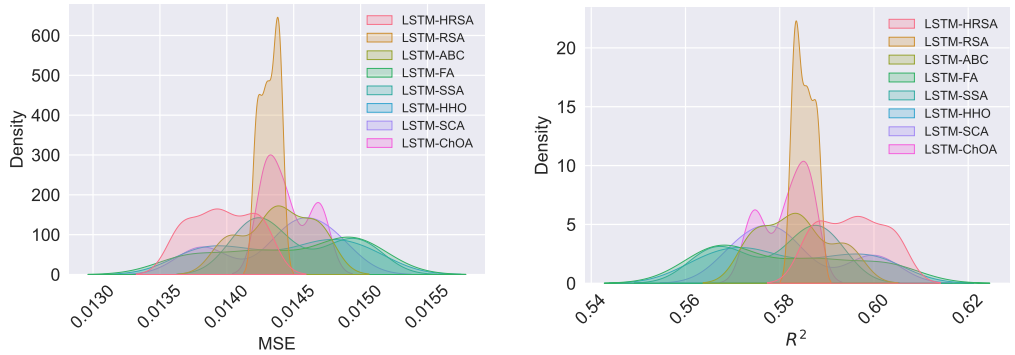

Figure 6. Graphical representation of the Kernel Distribution Estimation plots of LSTM in terms of fitness function (MSE) and R2.

Finally, to conclude the part with LSTM experiments, it is also worthwhile to provide the data about execution time on the platform which was used for experiments (details given in Section 4.3) in the worst-case scenario when taking into consideration the choice of hyperparameters (details provided in Section 4.3). One LSTM objective function evaluation for LSTM with 2 layers with 200 neurons per layer, dropout of 0.05, and learning rate of 0.0001, trained over 600 epochs with early stopping criteria, took about 195 s. However, most of the generated models by metaheuristics were much less complex and the average time for one objective function evaluation was around 90 s.

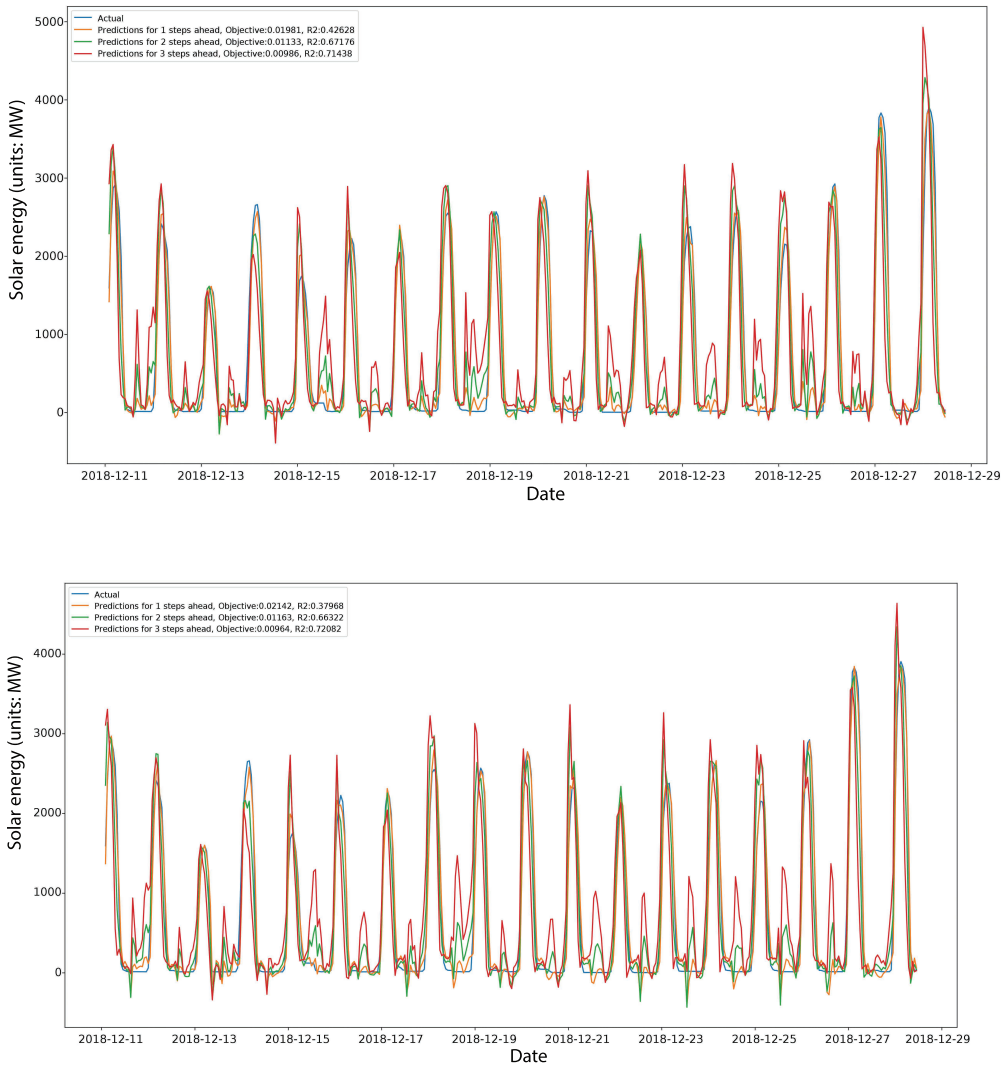

Figure 7. Graphical representation of the best predictions of the proposed LSTM-HRSA (**top**), compared to the best predictions of the elementary LSTM-RSA method (**bottom**).

4.4.2. BiLSTM Experiments

Table 5 provides the BiLSTM simulation outcomes with respect to the fitness function (MSE), yielding the standard metrics such as the best, worst, mean and median values, together with the standard deviation and variation established again over the course of 15 separate runs. One can note, based on Table 1, that the proposed BiLSTM-HRSA method again attained the best scores in terms of best, worst, mean, and median metrics. On the other hand, the LSTM-ABC method established the most stable results, indicated by the best standard deviation and variance measurements.

Tables 6 and 7 present the normalized and denormalized metrics related to the one-step ahead, two-step ahead, and three-step ahead predictions, accompanied by the overall results, achieved by the best run of each one of the 8 regarded LSTM models. The BiLSTM-HRSA achieved the best scores for one-step and two-step ahead forecasts, while BiLSTM-ChOA attained the best outcomes for three-step ahead. Moreover, when analyzing the best overall results, the suggested BiLSTM-HRSA model achieved the best scores for MSE (the fitness function), R^2, and RMSE, while BiLSTM-HHO established the best MAE overall result. In the end, Table 8 presents the best set of hyperparameters determined by each metaheuristic. It is interesting to note that almost all metaheuristics (except one - HHO) generated BiLSTM networks with just one layer.

Table 5. Overall results of the objective function for BiLSTM experiments. The best result in each column is written in bold.

Method	Best	Worst	Mean	Median	Std	Var
BiLSTM-HRSA	**0.01371**	**0.01419**	**0.01402**	**0.01412**	1.77×10^{-4}	3.14×10^{-8}
BiLSTM-RSA	0.01381	0.01464	0.01434	0.01434	3.22×10^{-4}	1.03×10^{-7}
BiLSTM-ABC	0.01437	0.01465	0.0145	0.0145	**1.14** $\times 10^{-4}$	**1.29** $\times 10^{-8}$
BiLSTM-FA	0.01396	0.01475	0.01438	0.01447	2.95×10^{-4}	8.73×10^{-8}
BiLSTM-SSA	0.01423	0.01507	0.01454	0.01444	2.93×10^{-4}	8.59×10^{-8}
BiLSTM-HHO	0.01399	0.01499	0.01449	0.01452	3.97×10^{-4}	1.58×10^{-7}
BiLSTM-SCA	0.01407	0.01464	0.01425	0.01417	2.18×10^{-4}	4.77×10^{-8}
BiLSTM-ChOA	0.01411	0.01511	0.01463	0.01445	4.18×10^{-4}	1.75×10^{-7}

The graphical representation of the performed BiLSTM simulations are given in Figures 8 and 9, where the same measurements have been outlined (for both the fitness function— MSE and R^2): the convergence graphs of the best runs, box plot diagrams and violin plots over 15 runs, and swarm plots for population diversity view. The swarm plots here reveal that in this case, the ChOA expresses the highest diversity of solutions in the last iteration, contrary to the ABC and FA, where all 5 individuals from the population were concentrated near each other.

Lastly, Figure 10 brings forward the KDE diagrams for the fitness function and R^2, visualizing once more the probability density function for the BiLSTM experiments. According to these diagrams that show the distribution of the outcomes of the runs, the results are not coming from the normal distribution, a similar conclusion that has also been drawn for the LSTM experiments. In the end, Figure 11 presents the best predictions of the BiLSTM tuned by the proposed HRSA method, compared to the predictions of the BiLSTM tuned by the elementary RSA.

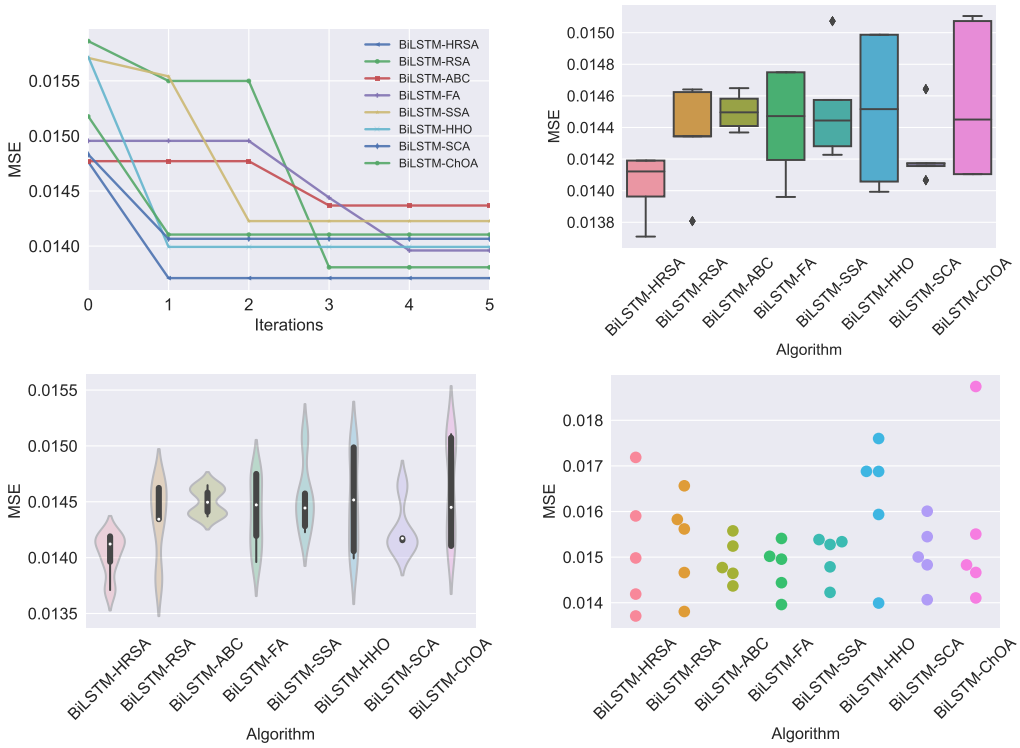

Figure 8. Visual BiLSTM simulations for each of 8 considered optimization methods with respect to the convergence, box plot, violin diagrams, and swarm diversity diagrams of the fitness function (MSE).

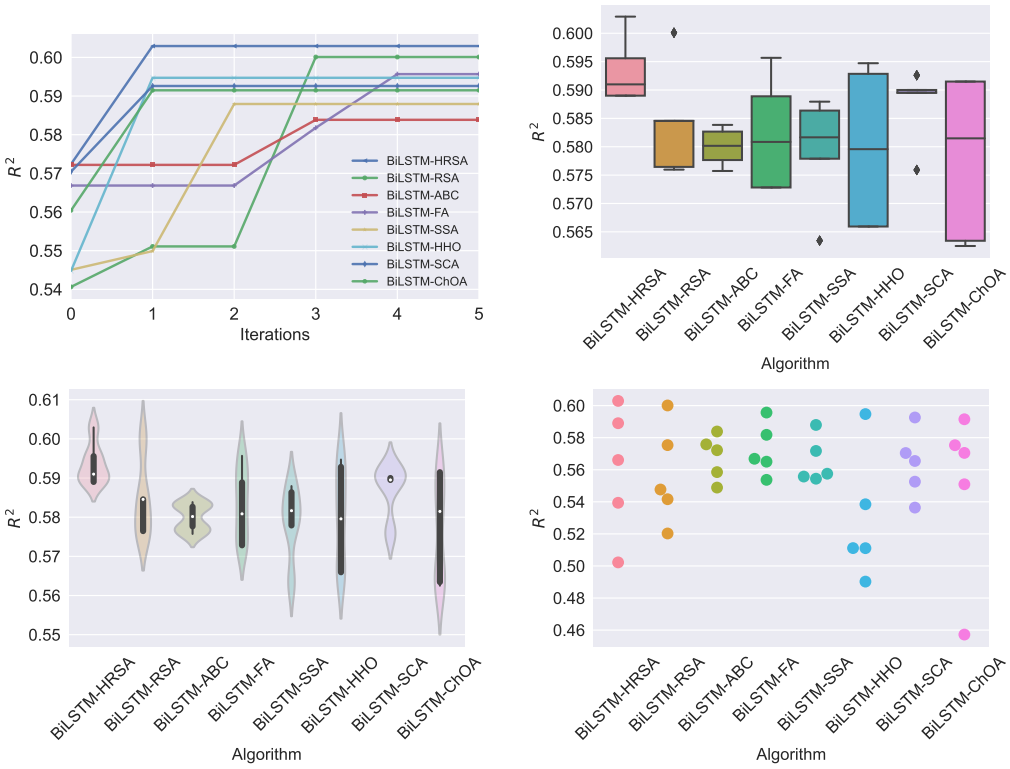

Figure 9. Visual BiLSTM simulations for each of 8 considered optimization methods with respect to the convergence, box plot, violin diagrams, and swarm diversity diagrams of the R^2 criterion.

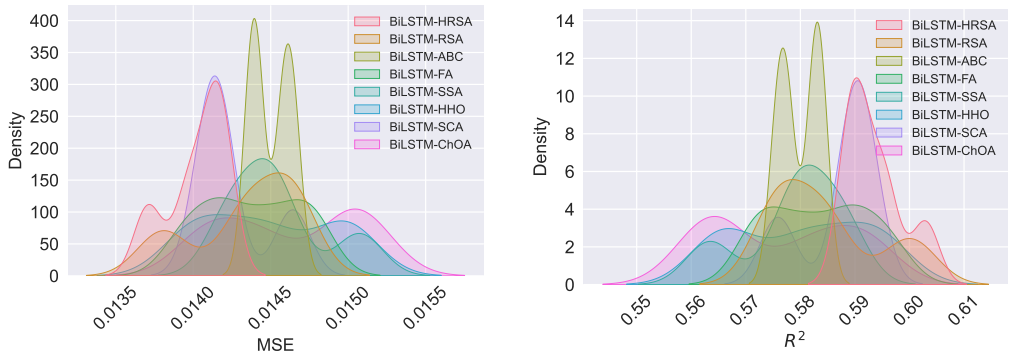

Figure 10. Graphical representation of the Kernel Distribution Estimation plots of BiLSTM in terms of fitness function (MSE) and R2.

Similarly as in the case of LSTM experiments, to finalize the section with BiLSTM experiments, it is worthwhile pointing out the execution time in the worst-case scenario. One BiLSTM objective function evaluation for BiLSTM structure with 2 layers and 200 neurons per layer, a dropout rate of 0.05 and learning rate of 0.0001, trained over 600 epochs with early stopping criteria, took about 245 seconds, which is even higher in the case of LSTM. However, most of the generated models by metaheuristics were much less complex and the average time for one BiLSTM objective function evaluation was around 105 s.

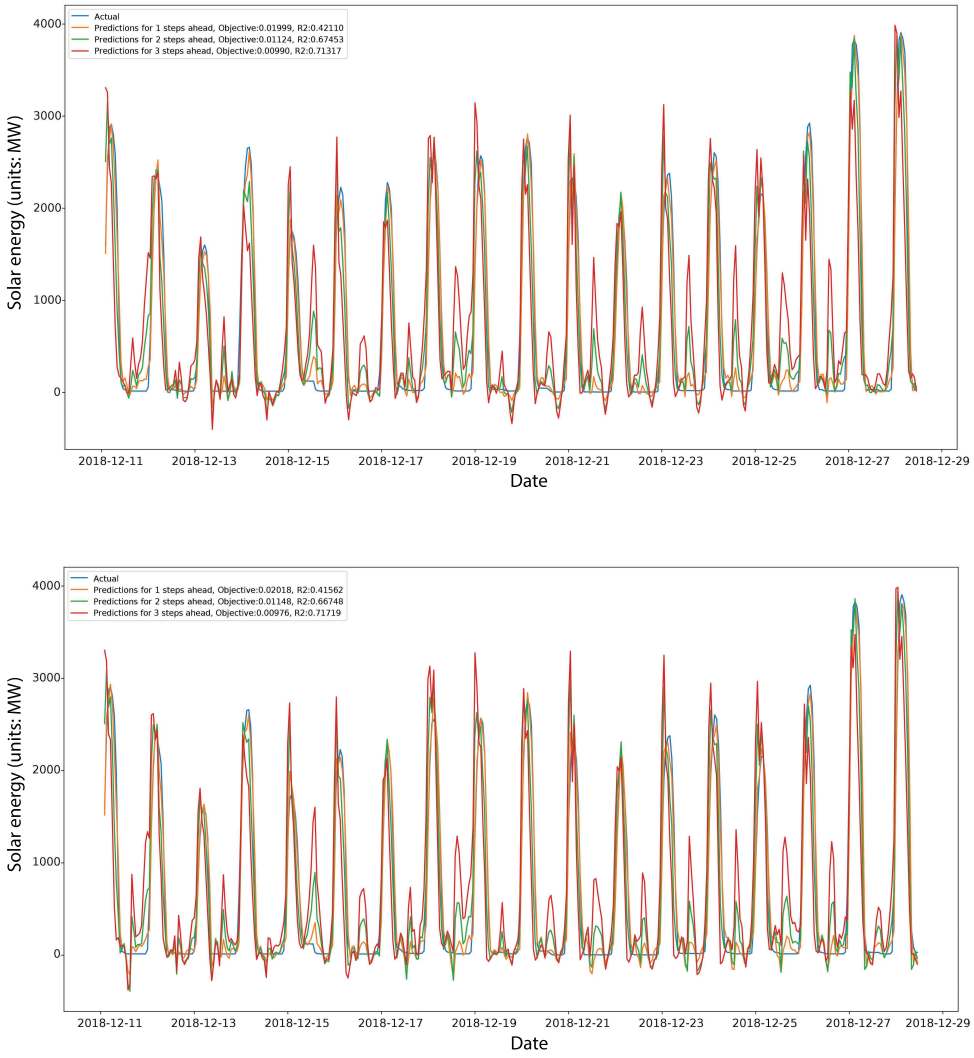

Figure 11. Graphical representation of the best predictions of the proposed BiLSTM-HRSA (**top**), compared to the best predictions of the elementary BiLSTM-RSA method (**bottom**).

Table 6. Normalized metrics for one-step, two-step, three-step ahead, and overall predictions for BiLSTM experiments. The best result in each column is written in bold.

	Error Indicator	BiLSTM-HRSA	BiLSTM-RSA	BiLSTM-ABC	BiLSTM-FA	BiLSTM-SSA	BiLSTM-HHO	BiLSTM-SCA	BiLSTM-ChOA
One-step ahead	R^2	**0.421099**	0.415616	0.394359	0.388672	0.371305	0.407744	0.382719	0.371755
	MAE	0.088153	0.087888	0.089462	0.089296	0.090744	**0.086110**	0.091255	0.092336
	MSE	**0.019988**	0.020177	0.020911	0.021107	0.021707	0.020449	0.021313	0.021691
	RMSE	**0.141378**	0.142046	0.144606	0.145283	0.147333	0.142999	0.145989	0.147280
Two-step ahead	R^2	**0.674531**	0.667477	0.659167	0.672847	0.654548	0.670620	0.669736	0.661434
	MAE	0.068554	0.069248	0.070329	0.069242	0.071110	**0.067229**	0.068945	0.070249
	MSE	**0.011237**	0.011481	0.011768	0.011296	0.011927	0.011373	0.011403	0.011690
	RMSE	**0.106007**	0.107149	0.108480	0.106281	0.109213	0.106642	0.106785	0.108119
Three-step ahead	R^2	0.713170	0.717192	0.698024	0.725492	0.737979	0.705799	0.725345	**0.741276**
	MAE	0.067599	0.066682	0.069706	0.067219	0.069153	0.068083	**0.066007**	0.064085
	MSE	0.009903	0.009765	0.010426	0.009478	0.009047	0.010158	0.009483	**0.008933**
	RMSE	0.099516	0.098815	0.102109	0.097355	0.095115	0.100786	0.097381	**0.094514**
Overall Results	R^2	**0.602933**	0.600095	0.583850	0.595670	0.587944	0.594721	0.592600	0.591488
	MAE	0.074769	0.074606	0.076499	0.075252	0.077002	**0.073807**	0.075402	0.075557
	MSE	**0.013710**	0.013808	0.014368	0.013960	0.014227	0.013993	0.014066	0.014105
	RMSE	**0.117088**	0.117505	0.119868	0.118154	0.119277	0.118292	0.118601	0.118763

Table 7. Denormalized metrics for one-step, two-step, three-step ahead, and overall predictions for BiLSTM experiments. The best result in each column is written in bold.

	Error Indicator	BiLSTM-HRSA	BiLSTM-RSA	BiLSTM-ABC	BiLSTM-FA	BiLSTM-SSA	BiLSTM-HHO	BiLSTM-SCA	BiLSTM-ChOA
One-step ahead	R^2	**0.421099**	0.415616	0.394359	0.388672	0.371305	0.407744	0.382719	0.371755
	MAE	475.494774	474.069836	482.560360	481.662702	489.475820	**464.476906**	492.232039	498.062444
	MSE	**581,546.657808**	587,054.952015	608,408.713021	614,121.623212	631,568.107513	594,962.847662	620,102.183796	631,116.615843
	RMSE	**762.592065**	766.195114	780.005585	783.659124	794.712594	771.338348	787.465672	794.428484
Two-step ahead	R^2	**0.674531**	0.667477	0.659167	0.672847	0.654548	0.670620	0.669736	0.661434
	MAE	369.779544	373.525155	379.353701	373.488815	383.568145	**362.634358**	371.889565	378.923944
	MSE	**326,956.104522**	334,042.303713	342,390.554819	328,648.215533	347,031.006821	330,885.790177	331,773.092965	340,113.527815
	RMSE	**571.800756**	577.963929	585.141483	573.278480	589.093377	575.226729	575.997477	583.192531
Three-step ahead	R^2	0.713170	0.717192	0.698024	0.725492	0.737979	0.705799	0.725345	**0.741276**
	MAE	364.630339	359.681027	375.996556	362.581967	373.008805	367.237735	**356.040109**	345.675295
	MSE	288,141.221075	284,100.488216	303,356.285497	275,762.724491	263,218.500005	295,545.992973	275,910.628013	**259,906.610666**
	RMSE	536.787873	533.010777	550.777891	525.131150	513.048243	543.641419	525.271956	**509.810367**
Overall Results	R^2	**0.602933**	0.600095	0.583850	0.595670	0.587944	0.594721	0.592600	0.591488
	MAE	403.301552	402.425339	412.636872	405.911161	415.350924	**398.116333**	406.720571	407.553894
	MSE	**398,881.327802**	401,732.581314	418,051.851112	406,177.521079	413,939.204780	407,131.543604	409,261.968258	410,378.918108
	RMSE	**631.570525**	633.823778	646.569293	637.320580	643.381073	638.068604	639.735858	640.608241

Table 8. Best attained collection of BiLSTM parameter values by each observed algorithm.

Method	Units in First Layer	Learning Rate	Epochs	Dropout	Layers	Units in Second Layer
BiLSTM-HRSA	104	0.008206	600	0.099072	1	/
BiLSTM-RSA	200	0.010000	600	0.200000	1	/
BiLSTM-ABC	200	0.006756	599	0.200000	1	/
BiLSTM-FA	156	0.010000	598	0.200000	1	/
BiLSTM-SSA	200	0.006827	515	0.078874	1	/
BiLSTM-HHO	139	0.008175	484	0.174655	2	120
BiLSTM-SCA	200	0.005856	600	0.200000	1	/
BiLSTM-ChOA	146	0.010000	549	0.155236	1	/

4.4.3. Comparison with Other ML/DL Models

Looking at the experimental results of the chosen models, it can be noted that the best performance was achieved by the LSTM-HRSA approach, concluding that it was the best-generated model throughout the simulations. To further emphasize the level of performance attained by the LSTM-HRSA model, it was compared to the recent traditional ML and DL methods, tested on the same data set with the same forecasting task. Additionally, the best-performing model was compared to the baseline LSTM and baseline BiLSTM architectures, as well.

For baseline LSTM and BiLSTM models, two networks without the dropout layer were created, the first with one, and the second with two layers, each layer containing 300 units. These networks were trained for 600 epochs, and the same early stopping criterion was used as described in the simulations with metaheuristics, but models were not optimized, and Keras default learning rate was used. These models were employed in the following comparisons to see their behavior if they have not been tuned by metaheuristics. The number of units in these baseline models was determined empirically, through experiments of trial and error.

Moreover, to compare the behavior of the traditional feed-forward deep neural networks (FFDNNs), two models were implemented in Keras, with one and two layers, each layer comprising 300 neurons, with the same early stopping condition, and with default values from Keras. Finally, support vector regression (SVR) and XGBoost were implemented with the default values from the *scikit learn* library, without optimization, and included in the comparison. The comparative analysis of the performances is shown in Table 9, where the superiority of the suggested LSTM-HRSA can be acknowledged. Looking at the results presented in Table 9, it is also possible to note that baseline LSTM and BiLSTM networks are suitable for this particular prediction problem, and perform better than other observed models, however, it is still possible to optimize them further to attain an even better level of performance, as suggested by the results achieved by the LSTM-HRSA method.

Table 9. The results of the best-produced model compared to the traditional ML methods and baseline LSTM and BiLSTM. The best result in each column is written in bold.

Method/Metric	R^2	MAE	MSE	RMSE
LSTM-HRSA	**0.604139**	**0.074450**	**0.013668**	**0.116910**
Baseline LSTM 1 layer 300n	0.532955	0.089421	0.015621	0.124983
Baseline LSTM 2 layers 300n	0.556421	0.083249	0.015267	0.123561
Baseline BiLSTM 1 layer 300n	0.534006	0.088724	0.015245	0.123470
Baseline BiLSTM 2 layers 300n	0.558395	0.081506	0.014931	0.122192
FFDNN 1 layer 300n	0.446716	0.097807	0.017917	0.133854
FFDNN 2 layers 300n	0.480219	0.092917	0.017021	0.130465
SVR	0.457883	0.096177	0.017618	0.132734
XGBoost	0.429964	0.100255	0.018365	0.135517

4.5. Statistical Analysis and Results Interpretation

A statistical consideration of the results and the interpretation of the factors the models deemed as most decisive was conducted next.

4.5.1. Statistical Tests

Once the simulations have been concluded, it is necessary to validate the acquired results to establish if these are statistically important. To achieve this, the best results of each of the 15 runs, for every algorithm and both problem instances (LSTM and BiLSTM), are taken and treated as data series. Next, it must be decided if it is possible to safely use a parametric test, or it is necessary to proceed with a non-parametric one. To establish if the parametric tests can be safely utilized, it is essential to check the independence, normality and homoscedasticity requirements [89]. The independence requirement is fulfilled as each separate run of every metaheuristic algorithm begins by producing a set of pseudo-random individuals. However, the normality criterion is not fulfilled, as the KDE diagrams in Figures 6 and 10 obviously show that the results are not originating from the normal distribution. It must be noticed here that it was expected that the results will not originate from the normal distribution, as metaheuristic stochastic algorithms typically require a large number of executions to express their nature. The results may have come from the normal distribution if the number of independent runs was greater than 15, however, since the experiments are extremely demanding, unfortunately, this was not possible.

To verify this conclusion, the normality criterion has also been evaluated by applying Shapiro–Wilk's single problem analysis [90], where the Shapiro–Wilk's p-values have been determined for each regarded method separately. As every calculated p-value is lower than the threshold value 0.05, the H0 hypothesis can be rejected at level $\alpha = 0.05$. Consequently, it is safe to conclude that the simulation outcomes are not originating from the normal distribution. The Shapiro–Wilk outcomes are summarized within Table 10.

Table 10. Shapiro–Wilk p-values for LSTM and BiLSTM instances used to verify the normality requirement for safe application of parametric tests.

Methods	HRSA	RSA	ABC	FA	SSA	HHO	SCA	ChOA
LSTM	0.032	0.027	0.021	0.034	0.027	0.018	0.017	0.026
BiLSTM	0.027	0.031	0.024	0.029	0.018	0.019	0.028	0.024

Since the Shapiro–Wilk test outcomes have shown that it is not safe to proceed with parametric tests as the normality condition failed, it is necessary to apply the non-parametric alternative. Therefore, Wilcoxon signed-rank test [91] was taken with the same set of data series comprised of the best results for each run. The introduced HRSA was taken as the control algorithm. The attained results are summarized in Table 11.

For LSTM experiments, determined p-values are in every case lower than the threshold value 0.05 (p-values were namely 0.039 vs. RSA, 0.036 vs. ABC, 0.032 vs. FA, 0.024 vs. SSA, 0.033 vs. HHO, 0.034 vs. SCA, and 0.029 vs. ChOA). Therefore, it can be concluded that the suggested HRSA method is statistically significantly superior to the other contenders for both thresholds limits $\alpha = 0.1$ and $\alpha = 0.05$.

In the case of BiLSTM simulations, established p-values are lower than 0.05 for every algorithm except SCA (p-values were namely 0.031 vs. RSA, 0.023 vs. ABC, 0.029 vs. FA, 0.022 vs. SSA, 0.026 vs. HHO, 0.057 vs. SCA, and 0.019 vs. ChOA). Consequently, it can be noted that the proposed HRSA in this case is significantly better than the other contending algorithms except SCA for the threshold limit $\alpha = 0.05$, and significantly better than all contenders for limit $\alpha = 0.1$.

Table 11. *p*-values determined by the Wilcoxon signed-rank test on both problem instances.

Methods	HRSA	RSA	ABC	FA	SSA	HHO	SCA	ChOA
LSTM	N/A	0.039	0.036	0.032	0.024	0.033	0.034	0.029
BiLSTM	N/A	0.031	0.023	0.029	0.022	0.026	0.057	0.019

4.5.2. Best Model Interpretation

It is important to be able to understand and explain the behavior of an ML model in order to fully comprehend the process and results. The Shapley Additive Explanations (SHAP) method was used to explain the LSTM model that performed the best on the solar energy prediction data set. SHAP allows for a clear and meaningful interpretation of the LSTM predictions while avoiding the trade-off between accuracy and interpretability. It uses Shapley values, based on game theory, to determine which features have the greatest impact on the predictions [92].

In simple terms, Shapley values are a way to distribute payouts among players (in this case, features) based on their contribution to the overall payout (representing the prediction). The SHAP method uses this concept to assign an importance metric to each feature, measuring how much that feature contributed to a particular prediction by comparing the model's prediction to what the prediction would be if that feature had a baseline value.

The most successful LSTM model from experiments, LSTM-HRSA, was selected and Shapley values were determined for it. The model structure is shown in Figure 12. To understand the effect of each predictor on the final prediction, SHAP summary plots were created for all features, both with and without the target variable, that has also been utilized as a predictor. These plots are presented in Figure 13. The plots in the first row of the figure, in the shape of bar charts, show the relative significance of each feature, as determined by obtaining the average absolute value of the Shapley values. The plots provided in the subsequent row illustrate the effect each observation has on the target variable.

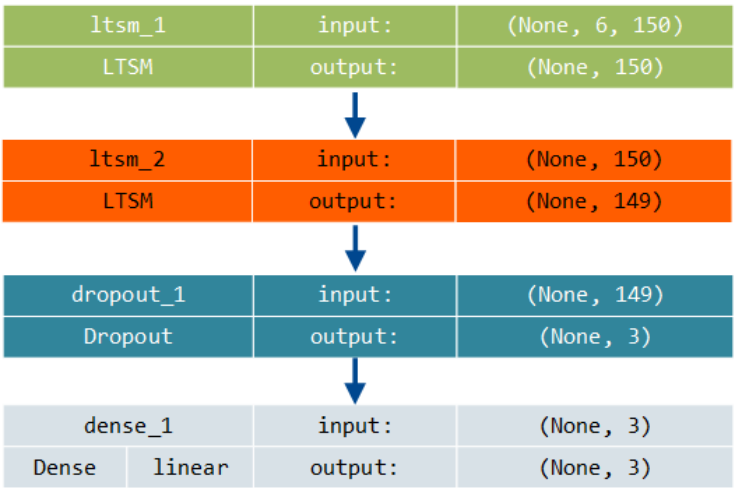

Figure 12. The structure of the best produced model - LSTM-HRSA.

Analyzing Figure 13, it is possible to draw several conclusions. At the beginning, the most significant feature is the solar variable for the past period, the second most significant predictor is the temperature, followed by the humidity and clouds indicators, while the least significant predictor is rain. From the summary plots per observations, it is possible to see that the increase in temperature will generally imply a rise in solar-generated power, while the increase in humidity and clouds will cause a decrease in solar power. This is in line with the real-world observations, as the relative humidity is in inverse relation to the air temperature, reaching the maximum values overnight (when temperatures are lower), and decreasing over the day with the increase of the temperature [93]. On the other hand,

solar radiation power is positively related to the air temperature and in inverse relation to the relative humidity [94,95]. Naturally, the clear skies without clouds will allow more solar radiation, increasing the temperature and decreasing the humidity in the process [96,97].

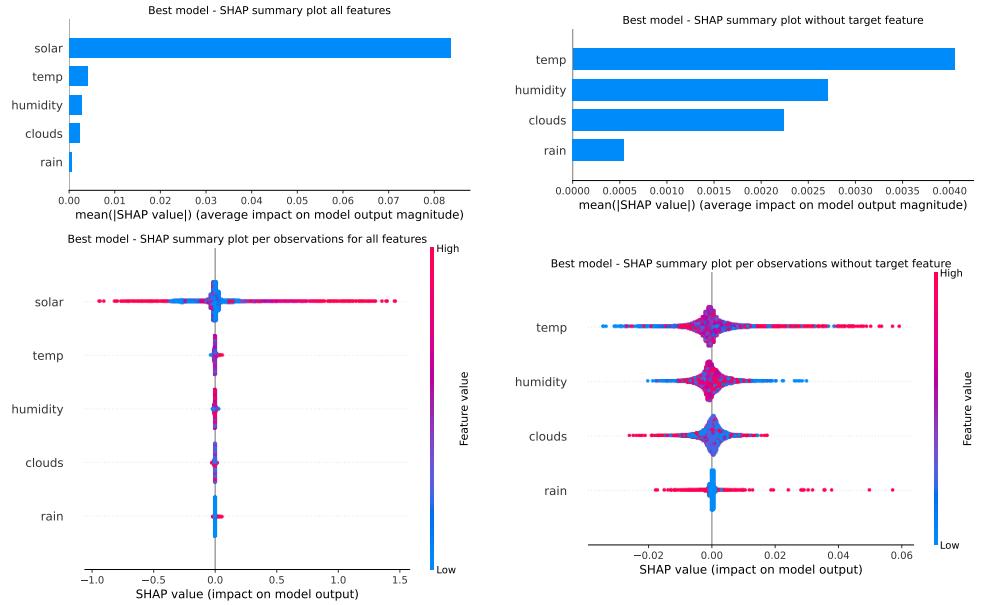

Figure 13. SHAP summary plots for executed tests with respect to the LSTM-HRSA optimized model.

5. Conclusions

The current paper presented the use of a modified novel metaheuristic to boost the performance of two LSTM architectures for the multivariate prediction of solar energy production. The reptile search algorithm resulted from the hybridization of the original method with characteristics of other recent nature-inspired optimization metaheuristics. Both a standard LSTM and a BiLSTM architecture were appointed to model the time series of PV output accompanied by exogenous weather data, with six hyperparameters tuned by the enhanced RSA. The results were compared with the outcomes of the manually parameterized networks, as well as those when the two were tuned by seven other recent metaheuristics.

The LSTM model tuned by the HRSA metaheuristics attained the best results with R^2 of 0.604139, normalized MAE, MSE and RMSE values of 0.074450, 0.013668 and 0.116910, respectively, that were significantly better than other models tuned by metaheuristics. The comparisons with baseline LSTM and BiLSTM models, and other traditional ML methods have also shown significant improvement in the results, as the best MSE value achieved by the best-performing baseline method was 0.016 (baseline LSTM), which is around 13% improvement, while the R^2 was improved by 0.071 (from 0.533 to 0.604).

The findings of this study demonstrated again the importance of automatic parametrization of deep learning architectures, as well as the necessity to further look for new ways to model the adaptation power that exists in the inspiring nature and to include these findings in novel robust metaheuristics.

Future work will analyze other recurrent architectures (such as GRU) or transformer models in their application for energy problems, while powered by metaheuristic optimization algorithms. Additionally, it is targeted to develop a multi-objective approach that will follow not only the accuracy of the results but also the complexity of the involved DL model and provide the output in the form of a Pareto front. Moreover, time-series forecasting tasks from other domains, such as cryptocurrency trends and stock markets predictions, will also be addressed.

Author Contributions: Conceptualization, M.Z., N.B. and C.S.; methodology, N.B., C.S. and M.Z.; software, N.B. and M.Z.; validation, M.Z. and N.B.; formal analysis, M.Z.; investigation, C.S., N.B. M.Z. and R.S.; resources, N.B., M.Z., A.B. and C.S.; data curation, M.Z., A.B. and N.B.; writing—original draft preparation, M.Z., R.S.-W., C.S. and R.S.; writing—review and editing, R.S., C.S., M.Z., M.A. and N.B.; visualization, N.B., M.Z. and M.A.; supervision, N.B. and C.S.; project administration, M.Z. and N.B.; funding acquisition, N.B. and C.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by a grant of the Romanian Ministry of Research and Innovation, CCCDI – UEFISCDI, project number 178PCE/2021, PN-III-P4-ID-PCE-2020-0788, within PNCDI III.

Data Availability Statement: Complete data sources are publicly available at https://www.kaggle.com/datasets/nicholasjhana/energy-consumption-generation-prices-and-weather. In the current work, the used data is from 1 January 2015 to 28 December 2018, with a total of 35,065 data points.

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

Abbreviations

The following abbreviations are used in this manuscript:

ABC	Artificial Bee Colony
ANN	Artificial Neural Network
BA	Bat Algorithm
BiLSTM	Bidirectional Long Short-Term Memory
ChOA	Chimp Optimization Algorithm
CNN	Convolutional Neural Networks
DL	Deep Learning
FA	Firefly Algorithm
GA	Genetic Algorithm
GRU	Gated Recurrent Unit
HHO	Harris Hawks Optimization
HRSA	Hybrid Reptile Search Algorithm
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
ML	Machine Learning
MSE	Mean Squared Error
PSO	Particle Swarm Optimization
PV	Photovoltaic
RMSE	Root Mean Squared Error
RSA	Reptile Search Algorithm
SCA	Sine Cosine Algorithm

References

1. Ahmad, T.; Zhang, D. A critical review of comparative global historical energy consumption and future demand: The story told so far. *Energy Rep.* **2020**, *6*, 1973–1991. [CrossRef]
2. Sen, S.; Ganguly, S. Opportunities, barriers and issues with renewable energy development—A discussion. *Renew. Sustain. Energy Rev.* **2017**, *69*, 1170–1181. [CrossRef]
3. Cantarero, M.M.V. Of renewable energy, energy democracy, and sustainable development: A roadmap to accelerate the energy transition in developing countries. *Energy Res. Soc. Sci.* **2020**, *70*, 101716. [CrossRef]
4. Forootan, M.M.; Larki, I.; Zahedi, R.; Ahmadi, A. Machine Learning and Deep Learning in Energy Systems: A Review. *Sustainability* **2022**, *14*, 4832. [CrossRef]
5. Ahmed, R.; Sreeram, V.; Mishra, Y.; Arif, M. A review and evaluation of the state-of-the-art in PV solar power forecasting: Techniques and optimization. *Renew. Sustain. Energy Rev.* **2020**, *124*, 109792. [CrossRef]
6. Bacanin, N.; Stoean, C.; Zivkovic, M.; Rakic, M.; Strulak-Wójcikiewicz, R.; Stoean, R. On the Benefits of Using Metaheuristics in the Hyperparameter Tuning of Deep Learning Models for Energy Load Forecasting. *Energies* **2023**, *16*, 1434. [CrossRef]
7. Blaga, R.; Sabadus, A.; Stefu, N.; Dughir, C.; Paulescu, M.; Badescu, V. A current perspective on the accuracy of incoming solar energy forecasting. *Prog. Energy Combust. Sci.* **2019**, *70*, 119–144. [CrossRef]
8. Anuradha, K.; Erlapally, D.; Karuna, G.; Srilakshmi, V.; Adilakshmi, K. Analysis Of Solar Power Generation Forecasting Using Machine Learning Techniques. *E3S Web Conf.* **2021**, *309*, 01163. [CrossRef]

9. Voyant, C.; Notton, G.; Kalogirou, S.; Nivet, M.L.; Paoli, C.; Motte, F.; Fouilloy, A. Machine learning methods for solar radiation forecasting: A review. *Renew. Energy* **2017**, *105*, 569–582. [CrossRef]
10. Kuo, W.C.; Chen, C.H.; Hua, S.H.; Wang, C.C. Assessment of Different Deep Learning Methods of Power Generation Forecasting for Solar PV System. *Appl. Sci.* **2022**, *12*, 7529. [CrossRef]
11. Carrera, B.; Kim, K. Comparison Analysis of Machine Learning Techniques for Photovoltaic Prediction Using Weather Sensor Data. *Sensors* **2020**, *20*, 3129. [CrossRef]
12. Kim, S.G.; Jung, J.Y.; Sim, M.K. A Two-Step Approach to Solar Power Generation Prediction Based on Weather Data Using Machine Learning. *Sustainability* **2019**, *11*, 1501. [CrossRef]
13. Zamo, M.; Mestre, O.; Arbogast, P.; Pannekoucke, O. A benchmark of statistical regression methods for short-term forecasting of photovoltaic electricity production, part I: Deterministic forecast of hourly production. *Sol. Energy* **2014**, *105*, 792–803. . [CrossRef]
14. Wu, Y.K.; Huang, C.L.; Phan, Q.T.; Li, Y.Y. Completed Review of Various Solar Power Forecasting Techniques Considering Different Viewpoints. *Energies* **2022**, *15*, 3320. [CrossRef]
15. Zhang, W.; Li, Q.; He, Q. Application of machine learning methods in photovoltaic output power prediction: A review. *J. Renew. Sustain. Energy* **2022**, *14*, 022701. [CrossRef]
16. Markovics, D.; Mayer, M.J. Comparison of machine learning methods for photovoltaic power forecasting based on numerical weather prediction. *Renew. Sustain. Energy Rev.* **2022**, *161*, 112364. [CrossRef]
17. Vennila, C.; Titus, A.; Sudha, T.; Sreenivasulu, U.; Reddy, N.; Jamal, K.; Lakshmaiah, D.; Jagadeesh, P.; Belay, A. Forecasting solar energy production using machine learning. *Int. J. Photoenergy* **2022**, *2022*, 7797488. [CrossRef]
18. Lim, S.C.; Huh, J.H.; Hong, S.H.; Park, C.Y.; Kim, J.C. Solar Power Forecasting Using CNN-LSTM Hybrid Model. *Energies* **2022**, *15*, 8233. [CrossRef]
19. Alkhayat, G.; Mehmood, R. A review and taxonomy of wind and solar energy forecasting methods based on deep learning. *Energy AI* **2021**, *4*, 100060. [CrossRef]
20. Khan, W.; Walker, S.; Zeiler, W. Improved solar photovoltaic energy generation forecast using deep learning-based ensemble stacking approach. *Energy* **2022**, *240*, 122812. [CrossRef]
21. Alkhayat, G.; Hasan, S.H.; Mehmood, R. SENERGY: A Novel Deep Learning-Based Auto-Selective Approach and Tool for Solar Energy Forecasting. *Energies* **2022**, *15*, 6659. [CrossRef]
22. Wang, H.; Lei, Z.; Zhang, X.; Zhou, B.; Peng, J. A review of deep learning for renewable energy forecasting. *Energy Convers. Manag.* **2019**, *198*, 111799. . [CrossRef]
23. Mellit, A.; Pavan, A.M.; Lughi, V. Deep learning neural networks for short-term photovoltaic power forecasting. *Renew. Energy* **2021**, *172*, 276–288. [CrossRef]
24. Jebli, I.; Belouadha, F.Z.; Kabbaj, M.I.; Tilioua, A. Deep learning based models for solar energy prediction. *Adv. Sci. Technol. Eng. Syst. J.* **2021**, *6*, 349–355. [CrossRef]
25. Li, G.; Xie, S.; Wang, B.; Xin, J.; Li, Y.; Du, S. Photovoltaic power forecasting with a hybrid deep learning approach. *IEEE Access* **2020**, *8*, 175871–175880. [CrossRef]
26. Aljanad, A.; Tan, N.M.L.; Agelidis, V.G.; Shareef, H. Neural Network Approach for Global Solar Irradiance Prediction at Extremely Short-Time-Intervals Using Particle Swarm Optimization Algorithm. *Energies* **2021**, *14*, 1213. [CrossRef]
27. Zhou, Y.; Zhou, N.; Gong, L.; Jiang, M. Prediction of photovoltaic power output based on similar day analysis, genetic algorithm and extreme learning machine. *Energy* **2020**, *204*, 117894. [CrossRef]
28. Panda, S.; Dhaka, R.K.; Panda, B.; Pradhan, A.; Jena, C.; Nanda, L. A review on application of Machine Learning in Solar Energy & Photovoltaic Generation Prediction. In Proceedings of the 2022 International Conference on Electronics and Renewable Systems (ICEARS), Tuticorin, India, 16–18 March 2022; pp. 1180–1184. [CrossRef]
29. Sherstinsky, A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Phys. D Nonlinear Phenom.* **2020**, *404*, 132306. [CrossRef]
30. Gers, F.A.; Schmidhuber, J.; Cummins, F. Learning to forget: Continual prediction with LSTM. *Neural Comput.* **2000**, *12*, 2451–2471. [CrossRef]
31. Aditya Pai, B.; Devareddy, L.; Hegde, S.; Ramya, B. A time series cryptocurrency price prediction using lstm. In *Emerging Research in Computing, Information, Communication and Applications*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 653–662.
32. Chen, K.; Zhou, Y.; Dai, F. A LSTM-based method for stock returns prediction: A case study of China stock market. In Proceedings of the 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, USA, 29 October–1 November 2015; pp. 2823–2824.
33. Stoean, C.; Paja, W.; Stoean, R.; Sandita, A. Deep architectures for long-term stock price prediction with a heuristic-based strategy for trading simulations. *PLoS ONE* **2019**, *14*, e0223593. [CrossRef]
34. Bukhari, A.H.; Raja, M.A.Z.; Sulaiman, M.; Islam, S.; Shoaib, M.; Kumam, P. Fractional neuro-sequential ARFIMA-LSTM for financial market forecasting. *IEEE Access* **2020**, *8*, 71326–71338. [CrossRef]
35. Stoean, C.; Stoean, R.; Atencia, M.; Abdar, M.; Velázquez-Pérez, L.; Khosravi, A.; Nahavandi, S.; Acharya, U.R.; Joya, G. Automated Detection of Presymptomatic Conditions in Spinocerebellar Ataxia Type 2 Using Monte Carlo Dropout and Deep Neural Network Techniques with Electrooculogram Signals. *Sensors* **2020**, *20*, 3032. [CrossRef] [PubMed]

36. Stoean, R.; Stoean, C.; Atencia, M.; Rodríguez-Labrada, R.; Joya, G. Ranking Information Extracted from Uncertainty Quantification of the Prediction of a Deep Learning Model on Medical Time Series Data. *Mathematics* **2020**, *8*, 1078. [CrossRef]
37. Shahid, F.; Zameer, A.; Muneeb, M. Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM. *Chaos Solitons Fractals* **2020**, *140*, 110212. [CrossRef] [PubMed]
38. Chimmula, V.K.R.; Zhang, L. Time series forecasting of COVID-19 transmission in Canada using LSTM networks. *Chaos Solitons Fractals* **2020**, *135*, 109864. [CrossRef]
39. Sagheer, A.; Kotb, M. Time series forecasting of petroleum production using deep LSTM recurrent networks. *Neurocomputing* **2019**, *323*, 203–213. [CrossRef]
40. Stegherr, H.; Heider, M.; Hähner, J. Classifying Metaheuristics: Towards a unified multi-level classification system. *Nat. Comput.* **2022**, *21*, 155–171. [CrossRef]
41. Emmerich, M.; Shir, O.M.; Wang, H. Evolution strategies. In *Handbook of Heuristics*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 89–119.
42. Fausto, F.; Reyna-Orta, A.; Cuevas, E.; Andrade, Á.G.; Perez-Cisneros, M. From ants to whales: Metaheuristics for all tastes. *Artif. Intell. Rev.* **2020**, *53*, 753–810. [CrossRef]
43. Beni, G. Swarm intelligence. In *Complex Social and Behavioral Systems: Game Theory and Agent-Based Models*; Sotomayor, M., Pérez-Castrillo, D., Castiglione, F., Eds.; Springer: New York, NY, USA, 2020; pp. 791–818.
44. Abraham, A.; Guo, H.; Liu, H. Swarm intelligence: Foundations, perspectives and applications. In *Swarm Intelligent Systems*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 3–25.
45. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [CrossRef]
46. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
47. Karaboga, D.; Basturk, B. On the performance of artificial bee colony (ABC) algorithm. *Appl. Soft Comput.* **2008**, *8*, 687–697. [CrossRef]
48. Yang, X.S. A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 65–74.
49. Yang, X.S.; Gandomi, A.H. Bat algorithm: A novel approach for global engineering optimization. *Eng. Comput.* **2012**, *29*, 464–483. [CrossRef]
50. Yang, X.S. Firefly algorithms for multimodal optimization. In Proceedings of the International Symposium on Stochastic Algorithms, Zurich, Switzerland, 13–14 September 2007; Springer: Berlin/Heidelberg, Germany, 2009; pp. 169–178.
51. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
52. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
53. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
54. Zivkovic, M.; Bacanin, N.; Venkatachalam, K.; Nayyar, A.; Djordjevic, A.; Strumberger, I.; Al-Turjman, F. COVID-19 cases prediction by using hybrid machine learning and beetle antennae search approach. *Sustain. Cities Soc.* **2021**, *66*, 102669. [CrossRef]
55. Zivkovic, M.; Venkatachalam, K.; Bacanin, N.; Djordjevic, A.; Antonijevic, M.; Strumberger, I.; Rashid, T.A. Hybrid Genetic Algorithm and Machine Learning Method for COVID-19 Cases Prediction. In *Proceedings of the International Conference on Sustainable Expert Systems: ICSES 2020*; Springer Nature: Berlin/Heidelberg, Germany, 2021; Volume 176, p. 169.
56. Bacanin, N.; Bezdan, T.; Tuba, E.; Strumberger, I.; Tuba, M.; Zivkovic, M. Task scheduling in cloud computing environment by grey wolf optimizer. In Proceedings of the 2019 27th Telecommunications Forum (TELFOR), Belgrade, Serbia, 26–27 November 2019; pp. 1–4.
57. Bezdan, T.; Zivkovic, M.; Tuba, E.; Strumberger, I.; Bacanin, N.; Tuba, M. Multi-objective Task Scheduling in Cloud Computing Environment by Hybridized Bat Algorithm. In Proceedings of the International Conference on Intelligent and Fuzzy Systems; Springer: Berlin/Heidelberg, Germany, 2020; pp. 718–725.
58. Bezdan, T.; Zivkovic, M.; Antonijevic, M.; Zivkovic, T.; Bacanin, N. Enhanced Flower Pollination Algorithm for Task Scheduling in Cloud Computing Environment. In *Machine Learning for Predictive Analysis*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 163–171.
59. Zivkovic, M.; Bezdan, T.; Strumberger, I.; Bacanin, N.; Venkatachalam, K. Improved Harris Hawks Optimization Algorithm for Workflow Scheduling Challenge in Cloud–Edge Environment. In *Computer Networks, Big Data and IoT*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 87–102.
60. Abbasi-khazaei, T.; Rezvani, M.H. Energy-aware and carbon-efficient VM placement optimization in cloud datacenters using evolutionary computing methods. *Soft Comput.* **2022**, *26*, 9287–9322. [CrossRef]
61. Zivkovic, M.; Bacanin, N.; Tuba, E.; Strumberger, I.; Bezdan, T.; Tuba, M. Wireless Sensor Networks Life Time Optimization Based on the Improved Firefly Algorithm. In Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 15–19 June 2020; pp. 1176–1181.
62. Zivkovic, M.; Bacanin, N.; Zivkovic, T.; Strumberger, I.; Tuba, E.; Tuba, M. Enhanced Grey Wolf Algorithm for Energy Efficient Wireless Sensor Networks. In Proceedings of the 2020 Zooming Innovation in Consumer Technologies Conference (ZINC), Novi Sad, Serbia, 26–27 May 2020; pp. 87–92.

63. Bacanin, N.; Tuba, E.; Zivkovic, M.; Strumberger, I.; Tuba, M. Whale Optimization Algorithm with Exploratory Move for Wireless Sensor Networks Localization. In Proceedings of the International Conference on Hybrid Intelligent Systems, Sehore, India, 10–12 December 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 328–338.
64. Zivkovic, M.; Zivkovic, T.; Venkatachalam, K.; Bacanin, N. Enhanced Dragonfly Algorithm Adapted for Wireless Sensor Network Lifetime Optimization. In *Data Intelligence and Cognitive Informatics*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 803–817.
65. Bezdan, T.; Cvetnic, D.; Gajic, L.; Zivkovic, M.; Strumberger, I.; Bacanin, N. Feature Selection by Firefly Algorithm with Improved Initialization Strategy. In Proceedings of the 7th Conference on the Engineering of Computer Based Systems, Novi Sad, Serbia, 26–27 May 2021; pp. 1–8.
66. Bezdan, T.; Zivkovic, M.; Tuba, E.; Strumberger, I.; Bacanin, N.; Tuba, M. Glioma Brain Tumor Grade Classification from MRI Using Convolutional Neural Networks Designed by Modified FA. In Proceedings of the International Conference on Intelligent and Fuzzy Systems, İzmir, Turkey, 21–23 July 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 955–963.
67. Zivkovic, M.; Bacanin, N.; Antonijevic, M.; Nikolic, B.; Kvascev, G.; Marjanovic, M.; Savanovic, N. Hybrid CNN and XGBoost Model Tuned by Modified Arithmetic Optimization Algorithm for COVID-19 Early Diagnostics from X-ray Images. *Electronics* **2022**, *11*, 3798. [CrossRef]
68. Strumberger, I.; Tuba, E.; Zivkovic, M.; Bacanin, N.; Beko, M.; Tuba, M. Dynamic search tree growth algorithm for global optimization. In Proceedings of the Doctoral Conference on Computing, Electrical and Industrial Systems, Costa de Caparica, Portugal, 8–10 May 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 143–153.
69. Preuss, M.; Stoean, C.; Stoean, R. Niching Foundations: Basin Identification on Fixed-Property Generated Landscapes. In Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO '11), Dublin, Ireland, 12–16 July 2011; Association for Computing Machinery: New York, NY, USA, 2011; pp. 837–844. [CrossRef]
70. Jovanovic, D.; Antonijevic, M.; Stankovic, M.; Zivkovic, M.; Tanaskovic, M.; Bacanin, N. Tuning Machine Learning Models Using a Group Search Firefly Algorithm for Credit Card Fraud Detection. *Mathematics* **2022**, *10*, 2272. [CrossRef]
71. Petrovic, A.; Bacanin, N.; Zivkovic, M.; Marjanovic, M.; Antonijevic, M.; Strumberger, I. The AdaBoost Approach Tuned by Firefly Metaheuristics for Fraud Detection. In Proceedings of the 2022 IEEE World Conference on Applied Intelligence and Computing (AIC), Sonbhadra, India, 17–19 June 2022; pp. 834–839.
72. Bacanin, N.; Sarac, M.; Budimirovic, N.; Zivkovic, M.; AlZubi, A.A.; Bashir, A.K. Smart wireless health care system using graph LSTM pollution prediction and dragonfly node localization. *Sustain. Comput. Inform. Syst.* **2022**, *35*, 100711. [CrossRef]
73. Jovanovic, L.; Jovanovic, G.; Perisic, M.; Alimpic, F.; Stanisic, S.; Bacanin, N.; Zivkovic, M.; Stojic, A. The Explainable Potential of Coupling Metaheuristics-Optimized-XGBoost and SHAP in Revealing VOCs' Environmental Fate. *Atmosphere* **2023**, *14*, 109. [CrossRef]
74. Bacanin, N.; Zivkovic, M.; Stoean, C.; Antonijevic, M.; Janicijevic, S.; Sarac, M.; Strumberger, I. Application of Natural Language Processing and Machine Learning Boosted with Swarm Intelligence for Spam Email Filtering. *Mathematics* **2022**, *10*, 4173. [CrossRef]
75. Stankovic, M.; Antonijevic, M.; Bacanin, N.; Zivkovic, M.; Tanaskovic, M.; Jovanovic, D. Feature Selection by Hybrid Artificial Bee Colony Algorithm for Intrusion Detection. In Proceedings of the 2022 International Conference on Edge Computing and Applications (ICECAA), Tamilnadu, India, 13–15 October 2022; pp. 500–505.
76. Milosevic, S.; Bezdan, T.; Zivkovic, M.; Bacanin, N.; Strumberger, I.; Tuba, M. Feed-Forward Neural Network Training by Hybrid Bat Algorithm. In Proceedings of the Modelling and Development of Intelligent Systems: 7th International Conference, MDIS 2020, Sibiu, Romania, 22–24 October 2020; Revised Selected Papers 7; Springer International Publishing: Berlin/Heidelberg, Germany, 2021; pp. 52–66.
77. Gajic, L.; Cvetnic, D.; Zivkovic, M.; Bezdan, T.; Bacanin, N.; Milosevic, S. Multi-layer Perceptron Training Using Hybridized Bat Algorithm. In *Computational Vision and Bio-Inspired Computing*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 689–705.
78. Bacanin, N.; Zivkovic, M.; Al-Turjman, F.; Venkatachalam, K.; Trojovskỳ, P.; Strumberger, I.; Bezdan, T. Hybridized sine cosine algorithm with convolutional neural networks dropout regularization application. *Sci. Rep.* **2022**, *12*, 1–20. [CrossRef] [PubMed]
79. Bacanin, N.; Stoean, C.; Zivkovic, M.; Jovanovic, D.; Antonijevic, M.; Mladenovic, D. Multi-Swarm Algorithm for Extreme Learning Machine Optimization. *Sensors* **2022**, *22*, 4204. [CrossRef] [PubMed]
80. Jovanovic, L.; Jovanovic, D.; Bacanin, N.; Jovancai Stakic, A.; Antonijevic, M.; Magd, H.; Thirumalaisamy, R.; Zivkovic, M. Multi-Step Crude Oil Price Prediction Based on LSTM Approach Tuned by Salp Swarm Algorithm with Disputation Operator. *Sustainability* **2022**, *14*, 14616. [CrossRef]
81. Bukumira, M.; Antonijevic, M.; Jovanovic, D.; Zivkovic, M.; Mladenovic, D.; Kunjadic, G. Carrot grading system using computer vision feature parameters and a cascaded graph convolutional neural network. *J. Electron. Imaging* **2022**, *31*, 061815. [CrossRef]
82. Abualigah, L.; Abd Elaziz, M.; Sumari, P.; Geem, Z.W.; Gandomi, A.H. Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer. *Expert Syst. Appl.* **2022**, *191*, 116158. [CrossRef]
83. Bacanin, N.; Bezdan, T.; Tuba, E.; Strumberger, I.; Tuba, M. Optimizing convolutional neural network hyperparameters by enhanced swarm intelligence metaheuristics. *Algorithms* **2020**, *13*, 67. [CrossRef]
84. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M. Quasi-oppositional differential evolution. In Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007; pp. 2229–2236.

85. Bacanin, N.; Bezdan, T.; Venkatachalam, K.; Zivkovic, M.; Strumberger, I.; Abouhawwash, M.; Ahmed, A.B. Artificial neural networks hidden unit and weight connection optimization by quasi-refection-based learning artificial bee colony algorithm. *IEEE Access* **2021**, *9*, 169135–169155. [CrossRef]
86. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
87. Heidari, A.A.; Faris, H.; Aljarah, I.; Mirjalili, S.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. Available online: https://aliasgharheidari.com/HHO.html (accessed on 12 January 2023). [CrossRef]
88. Khishe, M.; Mosavi, M.R. Chimp optimization algorithm. *Expert Syst. Appl.* **2020**, *149*, 113338. [CrossRef]
89. LaTorre, A.; Molina, D.; Osaba, E.; Poyatos, J.; Del Ser, J.; Herrera, F. A prescription of methodological guidelines for comparing bio-inspired optimization algorithms. *Swarm Evol. Comput.* **2021**, *67*, 100973. [CrossRef]
90. Shapiro, S.S.; Francia, R. An approximate analysis of variance test for normality. *J. Am. Stat. Assoc.* **1972**, *67*, 215–216. [CrossRef]
91. Wilcoxon, F. Individual comparisons by ranking methods. In *Breakthroughs in Statistics*; Springer: Berlin/Heidelberg, Germany, 1992; pp. 196–202.
92. Lundberg, S.M.; Lee, S.I. A unified approach to interpreting model predictions. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 4768–4777.
93. Shrestha, A.K.; Thapa, A.; Gautam, H. Solar radiation, air temperature, relative humidity, and dew point study: Damak, Jhapa, Nepal. *Int. J. Photoenergy* **2019**, *2019*, 8369231. [CrossRef]
94. Behr, H.D. Trends and Interdependence of Solar Radiation and Air Temperature—A Case Study from Germany. *Meteorology* **2022**, *1*, 341–354. [CrossRef]
95. Tao, H.; Ewees, A.A.; Al-Sulttani, A.O.; Beyaztas, U.; Hameed, M.M.; Salih, S.Q.; Armanuos, A.M.; Al-Ansari, N.; Voyant, C.; Shahid, S.; et al. Global solar radiation prediction over North Dakota using air temperature: Development of novel hybrid intelligence model. *Energy Rep.* **2021**, *7*, 136–157. [CrossRef]
96. Gürel, A.E.; Ağbulut, Ü.; Biçen, Y. Assessment of machine learning, time series, response surface methodology and empirical models in prediction of global solar radiation. *J. Clean. Prod.* **2020**, *277*, 122353. [CrossRef]
97. Pyrgou, A.; Santamouris, M.; Livada, I. Spatiotemporal analysis of diurnal temperature range: Effect of urbanization, cloud cover, solar radiation, and precipitation. *Climate* **2019**, *7*, 89. [CrossRef]