A Novel Evolutionary-based Deep Convolutional Neural Network Model for Intelligent Load Forecasting

Seyed Mohammad Jafar Jalali, *Student Member, IEEE*, Sajad Ahmadian, Abbas Khosravi, *Senior Member, IEEE*, Miadreza Shafie-khah, *Senior Member, IEEE*, Saeid Nahavandi, *Fellow, IEEE*, and João P.S. Catalão, *Senior Member, IEEE*

***Abstract*—The problem of electricity load forecasting has emerged as an essential topic for power systems and electricity markets seeking to minimize costs. However, this topic has a high level of complexity. Over the past few years, convolutional neural networks (CNNs) have been used to solve several complex deep learning challenges, making substantial progress in some fields and contributing to state of the art performances. Nevertheless, CNN architecture design remains a challenging problem. Moreover, designing an optimal architecture for CNNs leads to improve their performance in the prediction process. This paper proposes an effective approach for the electricity load forecasting problem using a deep neuroevolution algorithm to automatically design the CNN structures using a novel modified evolutionary algorithm called enhanced grey wolf optimizer (EGWO). The architecture of CNNs and its hyperparameters are optimized by the novel discrete EGWO algorithm for enhancing its load forecasting accuracy. The proposed method is evaluated on real time data obtained from data sets of Australian Energy Market Operator (AEMO) in the year 2018. The simulation results demonstrated that the proposed method outperforms other compared forecasting algorithms based on different evaluation metrics.**

***Index Terms*—Deep convolutional neural networks, Electricity load forecasting, Evolutionary computation, Neuroevolution, Optimization.**

I. Introduction

Electricity load forecasting is valuable for the energy companies to handle the demand response system effectively throughout the day ahead of the electricity energy sector. Through obtaining the raw data on the demand for electricity to be charged by customers, energy suppliers can approximate how much electricity is expected in the power system grid [1]. The aim of the supplier is to decrease energy generation and procurement expenses. Thus, this situation will allow utilities to better schedule the planning of power systems and the resources to be supplied by a previous awareness about energy demand [2], [3]. The accurate and effective load forecasting models can assist electricity organizations take significant decisions in order to provide a reliable safety bias for the grid management system. The costs of the energy operator are saved by 10 million pounds according to Bunn and Farmer [4] due to a drop of 1% in the load forecasting errors. In other words, providing reliable needed power by keeping the operational cost as low as possible is the primary goal of the electricity load forecasting models.

S. M. J. Jalali, A. Khosravi and S. Nahavandi are with the Institute for Intelligent Systems Research and Innovation (IISRI), Deakin University, Waurn Ponds, VIC, 3216, Australia, e-mail: sjalali@deakin.edu.au.

S. Ahmadian is with the Faculty of Information Technology, Kermanshah University of Technology, Kermanshah, Iran

M. Shafie-khah is with the School of Technology and Innovations, University of Vaasa, 65200 Vaasa, Finland.

J.P.S. Catalão is with the Faculty of Engineering of University of Porto and INESC TEC, Porto, Portugal.

Time series forecasting is a well-regarded field of research, aiming to use best models for predicting the future trends depending on previously observed trends. As forecasting of electricity load is considered as part of the time series forecasting paradigm, there are three categories in accordance with the forecasting horizon: short-term (minutes to hours ahead), medium-term (one day to weeks ahead) and long-term (months to years ahead) [5]. Throughout the multiple load forecasting strategies, medium-term load forecasting is highly essential for electrical power planning and scheduling, electric energy demand management, and improvising electricity sharing agreements [6]. This technique further leads to the generation of capacity planning for future network expansions due to the rising electricity load demand [6]. On the other hand, the research in regards of medium-term load forecasting is not actively explored by the researchers compared to shot-term and long-term load forecasting. Thus, the focus of this paper is on medium-term (one-day ahead) load forecasting.

Achieving highest accuracy forecasting models for electricity load demand is a major challenge in the electric power domain. From the year 1960 on-wards, several statistical and machine learning techniques have been proposed and utilized for electricity load forecasting [7].

Recently, with rapid development of artificial intelligence, deep learning (DL) is the latest achievement of the machine learning era attracting remarkable attention due to its outstanding performance in various domains including machine vision, recommendation systems and natural language processing [8]. Due to high ability for analyzing complex and deep nonlinear relationships through distributed feature representation, DL techniques have shown excellent performance in comparison with traditional machine learning techniques for various domains. Generally, for time series forecasting problems, the most DL algorithms which have been used by researchers in the literature are recurrent neural network (RNN), stacked autoencoder (SAE), long short-term memory (LSTM), deep belief networks (DBN), and their modification [9].

Among different DL techniques, convolutional neural networks (CNN) are the most commonly applied technique in order to solve complex tasks specifically for image processing domain. The strongest advantage of CNNs is in automating the procedure of feature extraction for a given problem. They are able to optimize the extracted features directly from the raw input during the training phase. Moreover, they are inviolable for small input data transformations such as scaling, translation, and distortion.

Although CNNs can save significant human effort in feature extraction, adjusting many hyperparameters in their architecture is a difficult and time-consuming task. An effective and useful way to obtain the most optimal values of the hyperparameters is to use deep neuroevolution (DNE) methods (also called automated neural architecture search). DNE is a form of computational intelligence techniques that utilizes evolutionary algorithms (EAs) to optimize hyperparameters and architecture of deep neural networks (DNNs) [10]. This approach brought a number of benefits over the still well-known gradient based algorithm called as back propagation [11] [12]. Some of these advantages include: being able to not trapped into the local optima, being not more sensitive to the initial connection weights and creating automatically optimal topologies for DNNs [13], [14].

Nevertheless, the application of CNNs in the field of time series forecasting remains few to the best of our knowledge. Binkowski et al. [15] utilized a CNN-autoregressive model for electricity and financial time series data. They showed that their algorithm outperforms LSTM and CNN networks. Koprinska et al. [16] applied CNNs to forecast electricity load and photo-voltaic solar power for the next day. They showed that CNNs outperformed other state-of-the-art methods. Tsantekidis et al. [17] used a CNN in order to forecast stock prices. They demonstrated that their proposed methodology is competitive with MLP and SVM algorothms. Although the mentioned studies investigated the strength of CNNs, many hyperparameters of CNNs largely affect in forecasting performance. On the other hand, the withdraw of the previous works is in selecting hyperparameters manually for designing the architecture of CNNs. Thus, using an intelligent algorithm to efficiently obtain the optimal values of hyperparameters for improving the forecasting accuracy is a critical issue. Accordingly, in this work, we adopt a DNE method based on GWO (grey wolf optimizer) algorithm to efficiently obtain the hyperparameters of deep CNNs for a time series problem.

GWO algorithm [18] is a newcomer among population-based optimization algorithms, characterized by a number of attractive advantages including flexibility, simplicity, derivation-free and local minima avoidance, as well as high exploration and exploitation capability. Moreover, this algorithm has fewer control parameters for adjusting, is easy to implement, and has a fast convergence characteristic [19]. These excellent features make GWO algorithm highly appropriate for solving highly nonlinear and multi-modal function optimization problems. Although this algorithm advantages by stochastic efficient controllers, it is also vulnerable to local stagnation and fast convergence in solving a problem with several variables. Therefore, in order to improve the search capabilities of the gray wolf optimization algorithm to find the effective optimal solutions, several efforts have been made in recent years by applying operators such as dimension learning-based hunting [19], opposition-based learning strategy [20], levy-flight [21] and elite-based crossover [22] to reduce the discrepancy between both the phases of exploration and exploitation of the GWO algorithm. These findings motivate us to further enhance the GWO performance by modifying this algorithm using two powerful evolutionary operators including nonlinear convergence factor and Gaussian mutation strategies. Our modified two-stage GWO provides some potential advantages such as gaining a greater capacity for local optima prevention and conducting a more consistent balance between patterns in exploration and exploitation phases. It should be noted that for the most portion, the superior performance of GWO and its modifications are primarily employed to optimize the continuous functions for engineering problems. On the other hand, in the real world, there are some discretization optimization problems. Since CNN hyperparameter tuning is a discretization problem that can only be taken the discrete (integer) values, the standard version of GWO does not work for such discretization problems. As a result, our improved Gray Wolf Optimizer expands the functionality of the GWO algorithm to a discretization optimization problem such as CNN hyperparameter tuning.

In addition, the GWO algorithm and its modified versions have been applied to classical computational intelligence models such as support vector regression (SVR) [23], [24], radial basis function neural network (RBFNN) [25], ELMAN neural network [26], regularized extreme learning machine (RELM) [27] and generalized regression neural network (GRNN) [28] to solve forecasting problems such as short-term load forecasting, sea clutter forecasting, and wind speed forecasting. From the point of view of deep learning network optimization, to the best of our knowledge, a modified version of GWO [29] has been applied to three different datasets of various prediction problems, including individual household electric power consumption, air pollution, and human activity recognition by considering the optimization of CNN-LSTM deep neural network. Compared to this work, in our work, we tune the CNN hyperparameters while they considered the tuning of CNN-LSTM architecture which is a different deep learning model with CNN. Second, the hyperparameters we tune are different with that work in batch size, number of epochs, and kernel size which they did not tune them. Third, the individual household electric power consumption dataset considered in their work is a multivariate dataset and the three datasets used in our work are from the univariate type of forecasting model. Forth, our focus is on electricity load forecasting and they applied their models over different classification and regression tasks. Therefore, our work is the first attempt for optimization of important hyperparameters of CNN deep learning model by a novel and powerful version of the GWO algorithm that focuses on the problem of electricity load forecasting. In Table I, a comparison between the GWO algorithm for real-world applications is shown. As can be seen from this table, the work proposed by us is the first one which presented on optimizing the CNN model hyperparameters with a strongly improved version of the GWO for the problem of electricity load forecasting, which optimizes effectively the highest number of hyperparameters of the CNN model.

Among the extensive literature of computational intelligence techniques on load forecasting [7], some authors have focused on modeling the load demand, for example, using ARIMA, neural networks, fuzzy logic, evolutionary optimization techniques, and hybrid algorithms [31], [32]. Most of these methods have evaluated their proposed methods with just one dataset, a very short horizon or one evaluation performance metric such as well-known RMSE [7]. In this work, three datasets, one-day ahead horizon, and three evaluation metrics are utilized to validate the effectiveness of the compared algorithms.

The novelty of the proposed algorithm in this work lies in the characterization of the response of the electricity load by an evolutionary neural architecture search. Moreover, according to the best of our knowledge, this paper is the first one to benefit from a bio-inspired algorithm (i.e., GWO) for optimizing the

TABLE I: A taxonomy of the reviewed works for GWO and its variants.

Work	Improved GWO	Application	Deep learning model	Hyperparameters				
				Batch size	Number of epochs	No.filters	Kernel size	Pooling size
[19]	Yes	Functional engineering problem	-	-	-	-	-	-
[20]	Yes	Functional engineering problem	-	-	-	-	-	-
[21]	Yes	Functional engineering problem	-	-	-	-	-	-
[22]	Yes	Feature selection	-	-	-	-	-	-
[23]	Yes	Electricity load forecasting	-	-	-	-	-	-
[24]	Yes	Electricity load forecasting	-	-	-	-	-	-
[25]	Yes	Sea clutter forecasting	-	-	-	-	-	-
[26]	Yes	Wind speed forecasting	-	-	-	-	-	-
[30]	-	Wind speed forecasting	-	-	-	-	-	-
[28]	-	Electricity load forecasting	-	-	-	-	-	-
[29]	Yes	Household electric power consumption	CNN-LSTM	-	-	Yes	-	Yes
Our work	Yes	Electricity load forecasting	CNN	Yes	Yes	Yes	Yes	Yes

hyperparameters of deep CNNs in order to improve the load forecasting accuracy.

Briefly, the contributions of this paper are as follows:

- This paper is the first to use a novel discrete GWO algorithm for optimizing CNN hyperparameters in time series regression forecasting problems (here an electricity load forecasting problem).
- A two-stage modification is applied in order to enhance the search capabilities of GWO. We name our modified method as enhanced GWO (EGWO).
- Hyperparameter tuning of CNNs is of paramount significance since computational performance is strongly linked to architecture of CNNs. On the other hand, the architecture of CNNs is designed manually which is a time-consuming and difficult task to do. The major contribution of this work is in introducing an optimization neuro-evolution paradigm for exploring the full spectrum of CNN hyperparameters, automatically. To this end, EGWO is utilized to sequentially look for optimum combination of hyperparameters under the restriction of the error rate. Such a DNE methodology for CNNs can design more efficient architectures and aid in the learning processes, automatically.
- The proposed EGWO algorithm is utilized to select a set of CNN hyperparameters in order to achieve more effective and accurate forecasting results.
- The proposed method is compared with eleven state-of-the-art and hybrid evolutionary forecasting algorithms on three real-world electricity load demand time series. The experimental results showed that the proposed method significantly outperforms other forecasting algorithms and obtain the lowest error metrics.

The remainder of the paper is organized as follows: Section II describes the enhanced GWO algorithm, deep CNNs and the steps of the proposed model in detail. The analysis of numerical results and discussions are presented in section III. Finally, section IV highlights the conclusion of the work.

II. Methodology

This section presents a novel electricity load forecasting method called EGWO-CNN which is based on deep convolutional neural networks optimized by grey wolf optimizer. To this end, deep CNN is employed for forecasting future electricity load using available time series data in the past. In addition, the proposed EGWO algorithm is applied to obtain the optimal values of hyperparameters used in deep CNN leading to improve its performance in forecasting time series. The overall steps of EGWO-CNN is presented in Fig. 1. The detailed descriptions of GWO

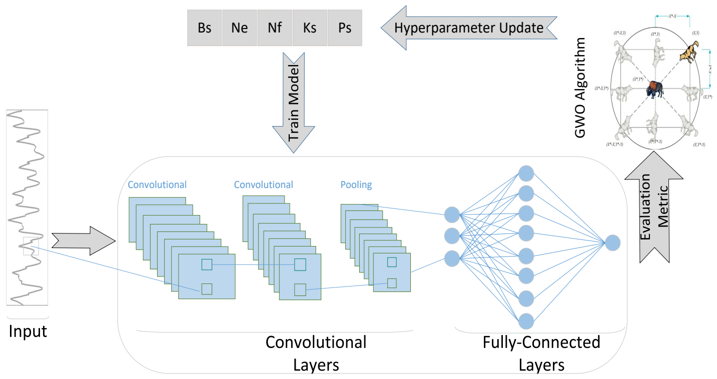

Fig. 1: EGWO-CNN overall procedure

algorithm, deep CNN, and the proposed model are represented in the following subsections.

A. *Modified Grey Wolf Optimizer*

GWO is a recently developed swarm-based meta-heuristic algorithm proposed which imitates the behavior of grey wolves in nature for hunting and social leadership [18]. In this algorithm, the population of individuals (wolves) is split into four groups including alpha (α), beta (β), delta (δ), and omega (ω) wolves. The first three best wolves are regarded to be α, β, and δ that guide other wolves (ω) towards promising search space areas. The wolves update their positions around α, β, or δ during the optimization process based on the following formulas:

$$\vec{D} = |\vec{C}.\vec{X_p}(t) - \vec{X}(t)| \tag{1}$$

$$\vec{X}(t+1) = |\vec{X_p}(t) - A.\vec{D}| \tag{2}$$

where t represents the current iteration, $A = 2a.\vec{r_1}a$ and $C = 2.\vec{r_2}$ indicate the coefficients. $\vec{X}$ and $\vec{X_p}$ represent the position vectors for grey wolf and prey (optimum solution), respectively. During the iterations, the value of a is linearly decreased from 2 to 0, while, r_1 and r_2 are random vectors in the interval of $[0, 1]$. Grey wolves use Eqs. (1) and (2) to update their positions around the prey, respectively. The grey wolves navigate around the best solution acquired so far in the m-dimensional search space.

For the process of hunting, the wolves update their positions based on the (α), (β) and (δ). The hunting mathematical model is provided as:

$$\begin{aligned}\vec{D_\alpha} &= |C_1.\vec{X_\alpha} - \vec{X}|\\ \vec{D_\beta} &= |C_2.\vec{X_\beta} - \vec{X}|\\ \vec{D_\delta} &= |C_3.\vec{X_\delta} - \vec{X}|\end{aligned} \quad (3)$$

where X_α, X_β and X_δ indicate the positions of α, β and δ, respectively. C_1, C_2 and C_3 are the different sets of random vectors and $\vec{X}$ represents the current solution position. Indeed, the formulas used in Eq. (3) calculate the estimated distance between the present solution and X_α, X_β, and X_δ, respectively. It should be noted that the main task of Eq. (3) is to set the step size of the ω wolf with regard to X_α, X_β and X_δ, respectively.

After defining the distances, the final position vectors for the solution will be calculated by the following equations:

$$\begin{aligned}\vec{X_1} &= \vec{X_\alpha} - A_1.\vec{D_\alpha}\\ \vec{X_2} &= \vec{X_\beta} - A_2.\vec{D_\beta}\\ \vec{X_3} &= \vec{X_\delta} - A_3.\vec{D_\delta}\end{aligned} \quad (4)$$

$$\vec{X}(t+1) = \frac{\vec{X_1} + \vec{X_2} + \vec{X_3}}{3} \quad (5)$$

where A_1, A_2 and A_3 show random vectors and the number of iterations is represented by t. It should be noted that to provide the exploration and exploitation phases in GWO algorithm, two random and adaptive vectors of $\vec{A}$ and $\vec{C}$ play a significant role in providing exploration and exploitation for GWO algorithm. The phase of exploration takes place when the vector $\vec{A}$ is less than -1 and greater than 1. The exploitation phase occurs, when $\vec{C}$ is greater than 1.

In order to enhance the search capabilities of GWO, we design a two-modification phase as follows:

-First modification: In this phase, we employ nonlinear convergence factor as shown in Eq. (6) for balancing the convergence of parameter a. Half of the iterations are for exploration phase in the original GWO algorithm, while the other half are for exploitation. Using this operator, a greater number of iterations for exploration are used, which is worthwhile to prevent into local optima.

$$a = 2\left(1 - \left(\frac{t-1}{t_{max}-1}\right)^{1.5}\right) \quad (6)$$

-Second modification

In this stage, We apply Gaussian mutation strategy to address the diversity loss and take smaller steps to make it easier for exploring every corner of the search space. The Gaussian density function is as following:

$$f_{\Upsilon,\Psi^2}(x) = \frac{1}{\Psi\sqrt{2\pi}} e^{-\frac{(x-\Upsilon)^2}{2\Psi^2}} \quad (7)$$

where Υ and Ψ denote to the mean and the standard deviation, respectively. This formula is also simplified to provide a random single D-dimension variable with Υ = 0 and Ψ =1:

$$X^{'} = X_i \oplus Gaussian(\Upsilon) X_i \quad (8)$$

where $X^{'}$ is a mutated search agent and $Gaussian(\Upsilon)$ represents a Gaussian D-dimension vector obtained by Eq. (7) between [0,1] interval. By applying these two modifications, the diversity of population as well as the exploration and exploitation of the GWO is enhanced. We name this two stage-modification as Enhanced-GWO (EGWO).

The procedure of training CNNs is regarded as a difficult and challenging topic with an unknown search space due to choosing the optimal sets of hyperparameters and architectures. On the other hand, the balance between exploration and exploitation phases in GWO algorithm is effective which can be very helpful for solving challenging problems such as training of CNNs. Therefore, high exploration of GWO algorithm needs to be effective as a CNN learner, excellently.

For further specifying the contribution performed on the original version of GWO algorithm, we will represents more details here. We can see two linear functions in the exploration phase (Functions A and C used in Eqs. 1 and 2) of the basic version of GWO algorithm. On the other hand, the hyperparameter optimization of deep CNNs is considered as a non-linear problem in which the original GWO utilizes of linear functions in the exploration phase which leads to decreasing the performance of GWO in finding the optimal CNN hyperparameters. Besides, this weakness increases the probability of tapping into the local optima as well as neglecting the global solutions and decreasing the performance of the forecasting model. To alleviate this issue, we use nonlinear convergence factor (Eq. 6) by replacing the linear functions utilized in basic GWO that leads to increasing search capability and decrease the probability of tapping into local optima.

The original version of the GWO algorithm employs random vectors (namely the vectors C_1, C_2 and C_3 in Eq. 3 and vectors A_1, A_2 and A_3 in Eq. 4) in the phase of exploitation without allowing any probabilistic distribution. This concern leads to a loss of search diversity and the neglect of global optimum. The key explanation is that evolutionary algorithms have to conduct their search in the exploitation process by small movements in order to reach the global optimum. However, the utilization of random vectors throughout the exploitation stage can result in disregarding global optimum due to their chaotic nature. In the proposed EGWO algorithm, we apply the Gaussian mutation strategy (Eq. 7) instead of using random vectors in the exploitation phase to address the diversity loss and take smaller steps to make it easier for exploring every corner of the search space.

B. Deep Convolutional Neural Network

In the proposed method, we use 1D convolutional neural network which has recently been introduced and immediately achieved the state-of-the-art performance levels in several applications. It should be noted that time series forecasting problems mainly contain one-dimensional vectors of data that have been ordered in specific time steps. Therefore, 1D CNN can be applied on such data to predict missing values. Generally, 1D CNNs consist of an input layer, an output layer and a number of CNN-layers. The input layer is a passive layer that receives the raw 1D input data and the output layer is a layer with a number of neurons equal to the number of outputs. In addition, both 1D convolutions and sub-sampling (pooling) occur in CNN-layers of 1D CNNs.

Forward- and back-propagation are two main methods to train CNNs. Forward propagation can be applied in each CNN-layer using the following equation:

$$x_k^l = b_k^l \sum_{i=1}^{l-1} Conv1D(w_{ik}^{l-1}, s_i^{l-1}) \quad (9)$$

where x_k^l represents the k^{th} input at layer l, b_k^l is the bias of k^{th} neuron at layer l, s_i^{l-1} indicates the output of the i^{th} neuron at layer $l-1$, w_{ik}^{l-1} is the weight from the i^{th} neuron at layer $l-1$ to the k^{th} neuron at layer l. $Conv1D(.,.)$ shows 1D convolution without zero-padding in 1D CNNs. The intermediate output in each layer can be obtained by passing the corresponding input through the activation function as follows:

$$s_k^l = y_k^l = f(x_k^l) \downarrow ss \quad (10)$$

where s_k^l indicates the output of the k^{th} neuron at layer l, and $f(.)$ represents the activation function used in 1D CNNs.

The back-propagation algorithm is used for propagating the error from the output layer. Suppose that the number of layers is L and N_L be the number of outputs in the output layer. Moreover, t^p and $[y_1^L, ..., y_{N_L}^L]'$ are the target and output vectors of an input vector p, respectively. Therefore, the mean square error (MSE) for the input p in the output layer can be calculated as follows:

$$E_p = MSE(t^p, [y_1^L, ..., y_{N_L}^L]') = \sum_{i=1}^{N_L} (y_i^L - t_i^p)^2 \quad (11)$$

The purpose of training 1D CNNs is to minimize the error calculated by Eq. (11). To this end, gradient descent algorithm can be applied to minimize the error by calculating the deviation of E_p by each network parameter.

In this paper, two convolutional layers and one pooling layer are accompanied by a dense fully connected layer. Pooling layer's role is to distill the convolutional layer output to the most important elements and dense layer interprets the features drawn from convolutional component of the models. Rectified Linear Unit (ReLU) activation function within the convolutional layer is used in the proposed method. In order to lessen the feature maps to a single 1D vector, a flatten layer is utilized between the pooling layer and dense layer. Finally, training procedure is performed by stochastic gradient descent (SGD).

C. EGWO-CNN based electricity load forecasting approach

In this section, we present the proposed electricity load forecasting method (called EGWO-CNN) which is based on one-dimensional convolutional neural network optimized by grey wolf optimizer. The purpose of EGWO-CNN method is to estimate the electricity load demand for a time step or multiple time steps in the future by using historical electricity load demand data. To this end, first, the hyperparameters of 1D CNN are optimized by grey wolf algorithm based on training data. Then, the trained 1D CNN is used to predict unknown values in test data. Two main issues should be addressed before performing EGWO algorithm including representation of solutions in population and calculation of fitness function.

1) Representation of solutions: One-dimensional CNN has several hyperparameters and its performance closely depends on determining the optimal values for them. The aim of using EGWO algorithm in the proposed method is to find optimal values for these hyperparameters leading to an improved predication accuracy. To this end, in the proposed method, we consider five critical hyperparameters including number of filters, kernel size, number of epochs, batch size, and pooling size. Therefore, each solution in the population space of EGWO algorithm contains five values corresponding to the considered parameters. It should be noted that the original EGWO algorithm can be applied for the problems with continuous space for the individuals. On the other hand, the hyperparameters values of CNN should be considered as discrete values. To this end, we employ an encoding transformation function to convert the real number vector, the position of an individual in continuous space into an integer vector by the following equation:

$$y_{ij} = \lfloor b_j * \frac{x_{ij} - lb}{ub - lb} + 0.5 \rfloor, j = 1, ..., n \quad (12)$$

where x_{ij} is the real number as the position X_i in the jth dimension, y_{ij} is the transformed integer value for the jth dimension of individual i, b_j is the total number of the item of type j, lb and ub are the lower and upper bounds of the search space, respectively.

2) Calculation of fitness function: The optimization procedure starts with an initialization step where a number of individuals are initialized with random values as the positions. The number of positions is equal to the number of CNN hyperparameters optimized by EGWO algorithm. It should be noted that each individual refers to a solution containing the values of hyperparameters for CNN. After the initialization step, the new generations of the first population can be obtained by repeating the search procedure of EGWO algorithm to find the optimal solution corresponding to the optimal values of CNN hyperparameters. To evaluate the effectiveness of each solution, we need to define a fitness function for the optimization process. In the proposed method, the prediction error obtained by CNN based on the training samples is used as the fitness function. It is worth noting that our goal in this work is to predict the next day (next 24 hours) based on weekly input data. Suppose that the historical electricity load demand data for M time steps is expressed by a vector as follows:

$$\vec{y} = (y_{(0)}, y_{(1)}, ..., y_{(M-1)}) \quad (13)$$

where $y_{(t)}$ is the actual electricity load demand for the time step t. The aim of the proposed method is to predict the electricity load demand for the next N time steps using CNN model. The predicted electricity load demand for N time steps can be represented as follows:

$$\vec{\hat{y}} = (\hat{y}_{(M)}, \hat{y}_{(M+1)}, ..., \hat{y}_{(M+N-1)}) \quad (14)$$

where $\hat{y}_{(t)}$ is the predicted electricity load demand for the time step t. In the proposed method, each individual is used to configure a CNN based on the obtained values of hyperparameters. Then, the CNN is applied to predict the electricity load demand data in training set. It should be noted that the electricity load demand data is represented as an one-dimensional input vector. Therefore, 1D CNN can be used to predict the time series data. The MSE metric is used to evaluate the performance of performed CNN for each solution in the search space which can be calculated using Eq. (11). In EGWO algorithm, the calculated MSE metric for each CNN is considered as the fitness function. After performing EGWO algorithm, the best individual can be obtained which is used to find the optimal values of CNN hyperparameters. Then, the CNN with the optimal hyperparameters is applied to predict the electricity load data in the test set. The overall procedure of the proposed method is represented in Algorithm 1.

Algorithm 1 Pseudo-code of the proposed electricity load forecasting model (EGWO-CNN)

```
1: Input: $pop\_size$ (population size) and $n$ (maximum number of iterations).
2: Output: Predicted electricity load demand.
3: Begin algorithm:
4: Split dataset into two sets including training set $Tr$ and test set $Te$;
5: Initialize the grey wolf population $X_i$ ($i$=1,2,..., $pop\_size$);
6: Initialize parameter $a$, A and C;
7: for (each solution $X_i$ in the grey wolf population) do
8:     Set a CNN model based on the values of solution $X_i$ as the hyperparameters;
9:     Calculate the fitness of solution $X_i$ using Eq. (11) as the MSE error of CNN model
       obtained based on the training set $Tr$;
10: end for
11: Let $X_\alpha$ be the best solution;
12: Let $X_\beta$ be the second best solution;
13: Let $X_\delta$ be the third best solution;
14: while (number of iterations < n) do
15:     for each solution $X_i$ in the grey wolf population do
16:         Apply the Gaussian mutation operator by using Eq. (8);
17:         Update the position of $X_i$ using Eq. (5);
18:         Set a CNN model based on the values of solution $X_i$ as the hyperparameters;
19:         Calculate the fitness of solution $X_i$ using Eq. (11) as the MSE error of CNN
            model obtained based on the training set $Tr$;
20:     end for
21:     Update $a$ using Eq. (6), A and C;
22:     Update $X_\alpha$, $X_\beta$ and $X_\delta$
23:     Increase the number of iterations by 1;
24: end while
25: Set a CNN model based on the values of solution $X_\alpha$ as the hyperparameters;
26: Predict the electricity load demand data in the test set $Te$ using the CNN model;
27: End algorithm
```

III. Numerical Results and Discussions

A. Data

In order to evaluate the effectiveness of the proposed method, the electricity load demand datasets from Australian Energy Market Operator (AEMO) are utilized [33]. The experiments are conducted on the datasets from the states of New South Wales (NSW), Queensland (QLD), and Victoria (VIC) for the year 2018. For each state, similar to [34], four months including January, April, July and October are considered to represent the different seasons: summer, autumn, winter, and spring, respectively. Generally, each time series data consists of multiple data points that are adjacent based on a specific time step. The used datasets represent the time series data as a sequence of data points with one hour time-steps. As a day consists of 24 hours, each day in the used datasets contains 24 data points. On the other hand, we use the data of each month in the used datasets as an independent dataset in the experiments. Therefore, for a 30-days month, we have $30 * 24$ data points, while this value is $31 * 24$ for a 31-days month. To perform the experiments, we need to divide each dataset into two separate sets including training set and test set. To this end, 75% of the earlier data points in each dataset are considered as training set while the remaining are used as test set. Eqs. (13) and (14) represent the data points of training set and test set, respectively. According to these equations, the number of data points in the training set and test set is N and M, respectively. For a dataset containing the data points of a 30-days month, the values of N and M are $0.75 * 30 * 24$ and $0.25 * 30 * 24$, respectively. In addition to splitting the used datasets into training set and test set, we need to form the input vectors based on these two sets for all implemented learning algorithms. To this end, each input vector is formed by considering a sequence of the data points for one week. Therefore, the number of data points in each input vector is equal to 168 (i.e. 7*24 data points). On the other hand, the purpose of forecasting model is to predict electricity load for the next day in which we have an one-day ahead forecasting model. Since our goal is to predict the next day (24 hours) ahead, based on reference [5], this type of forecasting is classified as medium-term (one day to weeks ahead). In order to perform the experiments, the original data points of the datasets are normalized into an interval of $[0, 1]$.

B. Experiment Procedure and Parameters

The optimal hyperparameters for the deep learning models used in the experiments are reported in Table II. It should be mentioned that for the models such as CNN and LSTM, we first examine their performances based on their reference articles. Then, for obtaining their best performances, we determine their optimal values through trial and error sets of simulation. For the remaining non-deep learning models, we first use the values determined by their corresponding reference articles, then we conduct a trial and error set of simulations to report their best performances. We set the population size of wolves and maximum number of iteration in 40 and 20, respectively. Through trail and error, these initialized parameter settings for EGWO ensured to avoid local minima and converge to an acceptable solution within the shortest time. The dimension of problem is set to five as there are five key CNN hyperparameters. These hyperparameters and their ranges are reported in Table III. These ranges are quite wide and have been set based on the existing CNN literature [35], [36]. Moreover, the reason behind selecting the optimization of these five hyperparameters is in their higher performance in architecture design of CNNs as indicated by recommended literature [36], [37]. It should be noted that Other hyperparameters that are not involved in the optimization process are trained with fixed values, which included dropout rate equal to 0.25, learning rate equal to 0.006, activation function as ReLU and optimizer as SGD.

TABLE II: Hyperparameters of deep learning benchmark algorithms used in the experiments

Model	Parameter	Value
PDRNN	No.hidden neuron	[5, 10, 20, 30, 50, 100]
	Optimizer	Adam
	Learning rate	0.001, 0.002, 0.005, 0.01
CNN	Act.function	Relu
	Batch size	512
	Learning rate	0.001
	Epochs	100
	No.filters	32
	Pooling size	3
	Dropout rate	0.3
LSTM	Act.function	Relu
	Batch size	256
	Learning rate	0.001
	Epochs	200
	No.filters	64
	Pooling size	4
	Dropout rate	0.2
GWO-LSTM EGWO-LSTM	Epoch	[1-500]
	No.hidden neuron	[1-60]
	Batch size	[1-200]
	Learning rate	[0.0001–0.1]
	Optimizer	Adam
PSO-CNN GA-CNN GWO-CNN IGWO-CNN MGWO-CNN and EGWO-CNN (Proposed)	Batch size	[10-100]
	Epochs	[1-200]
	No.filters	[1-300]
	Kernel size	[1-20]
	Pooling size	[1-15]
	Act.function	Relu
	Optimizer	SGD

C. Results and Analysis

In this work, for evaluating the error of prediction models, we employ three metrics including root mean square error (RMSE),

TABLE III: List of hyperparameters and their values

Hyperparameter	Definition	Value
Batch size	B_s	[10-100]
No.epochs	N_e	[1-200]
No.filters	N_f	[1-300]
Kernel size	K_s	[1-20]
Pooling size	P_s	[1-15]

TABLE IV: The best obtained configuration of hyperparameters for CNNs found by EGWO

Dataset	Month	RMSE	Hyperparameters				
			N_f	K_s	N_e	B_s	P_s
VIC	Jan	0.159	50	1	50	70	6
	Apr	0.090	106	1	50	50	1
	Jul	0.069	170	8	45	50	6
	Oct	0.089	196	7	55	60	3
NSW	Jan	0.126	50	20	50	30	1
	Apr	0.062	50	1	65	50	4
	Jul	0.075	47	19	50	40	1
	Oct	0.063	87	6	50	80	2
QLD	Jan	0.088	50	1	70	30	1
	Apr	0.113	75	8	86	50	1
	Jul	0.087	138	2	50	60	1
	Oct	0.094	50	15	50	50	6

mean absolute percentage error (MAPE) and mean absolute error (MAE).

We also compare the performance of the proposed algorithm with several state of the arts and hybrid algorithms. These algorithms are ARIMA (auto regressive integrated moving average) [38], CNN (convolutional neural network) [16], LR (logistic regression) [39], LSTM (long short-term memory) [40], MLP (multilayer perceptron) [41], RFR (random forest regression) [42], SVR (support vector regression) [43] and xgboost [44]. Besides, in order to show the search capabilities of our proposed model, the combination of deep CNN model with original GWO, two state-of-the-art GWO algorithms including improved GWO (IGWO) benefiting from dimension learning-based hunting search strategy [19] and the modified GWO (MGWO) which benefits from opposition-based learning strategy [20] alongside two powerful evolutionary algorithms including genetic algorithm (GA) and particle swarm optimization (PSO) are examined. Besides, the hybridization of GWO and EGWO with LSTM deep neural network model are investigated as well. We also validate the efficiency of our proposed method with three recently powerful algorithms in the literature including pooling-based deep recurrent neural network (PDRNN) [45], ensemble wavelet transform extreme learning machine (EWELM) [46] and principal component correlation analysis combined with LSTM (PCCA–LSTM) [47].

For having a fair comparison, the initialized configuration of state of the art algorithms is based on their corresponding papers in which the authors reported their best obtained results. Furthermore, a separate sensitivity analysis is used for these algorithms to ascertain their optimum setting parameters. For instance, for configuring the CNN and LSTM models, in addition to the settings considered in their original works, we also trained them with the greedy search method and the same architecture used in the proposed method with the non-optimized hyperparameters. Thus, by this procedure, we obtained the best possible models. Regarding hybrid models, the initial population size and the maximum iteration number are assumed the same for all models. For initializing the parameters of GA and PSO, we perform a

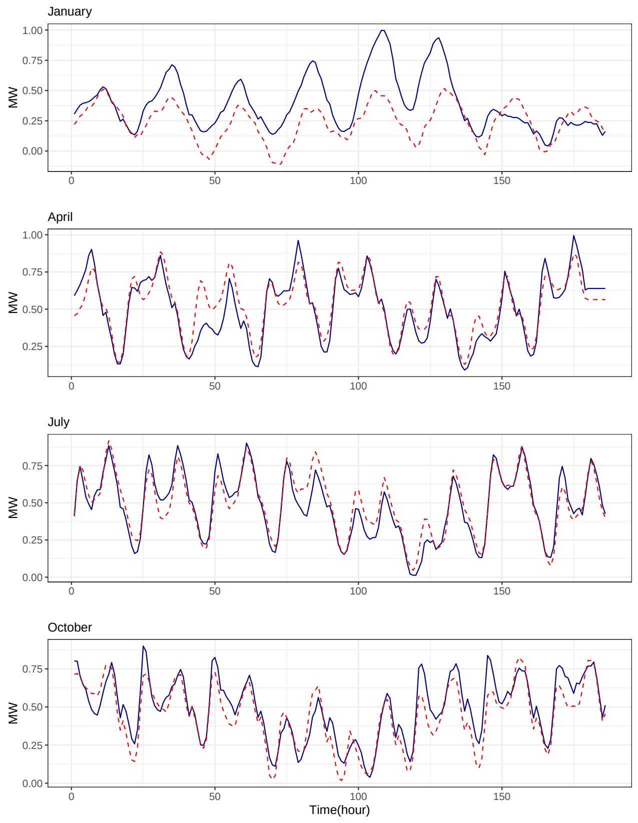

Fig. 2: Actual vs predicted points for VIC dataset. The real line represents the actual values and the dotted line represents the predicted values.

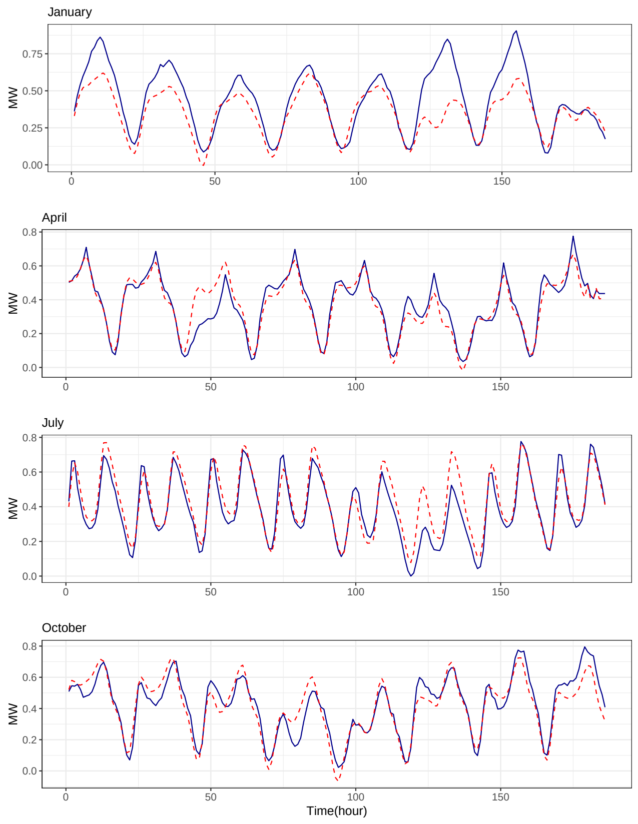

Fig. 3: Actual vs predicted points for NSW dataset. The real line represents the actual values and the dotted line represents the predicted values.

TABLE V: The average results of forecasting performance of proposed EGWO-CNN vs other forecasting benchmarks

Dataset	Month	Metric	Prediction Model																		
			LR	ARIMA	CNN	RFR	XGBOOST	LSTM	SVR	MLP	PDRNN	PCCA–LSTM	EWELM	GWO-LSTM	EGWO-LSTM	PSO-CNN	GA-CNN	GWO-CNN	IGWO-CNN	MGWO-CNN	**Proposed**
VIC	Jan	RMSE	0.196	0.284	0.169	0.199	0.213	0.173	0.188	0.182	0.168	0.172	0.17	0.171	0.167	0.168	0.167	0.166	0.165	0.163	**0.161**
		MAE	0.139	0.237	0.138	0.147	0.163	0.129	0.159	0.144	0.122	0.131	0.138	0.142	0.123	0.123	0.122	0.119	0.119	0.118	**0.115**
		MAPE	41.854	73.288	40.666	44.171	40.951	34.638	48.259	52.235	33.279	35.898	41.029	44.908	32.142	33.554	31.908	30.004	29.877	28.619	**26.593**
	Apr	RMSE	0.195	0.121	0.108	0.201	0.118	0.104	0.112	0.131	0.103	0.104	0.109	0.102	0.098	0.101	0.099	0.096	0.096	0.095	**0.092**
		MAE	0.155	0.097	0.086	0.161	0.088	0.081	0.089	0.105	0.073	0.08	0.084	0.074	0.063	0.071	0.068	0.067	0.068	0.067	**0.064**
		MAPE	36.029	26.913	23.21	39.832	22.147	20.965	24.799	31.398	21.023	20.464	22.61	20.117	18.743	20.143	19.055	17.867	17.885	16.252	**14.571**
	Jul	RMSE	0.166	0.114	0.096	0.152	0.081	0.086	0.107	0.111	0.079	0.086	0.094	0.08	0.077	0.078	0.076	0.075	0.074	0.073	**0.071**
		MAE	0.125	0.093	0.078	0.121	0.063	0.064	0.081	0.091	0.061	0.064	0.076	0.063	0.058	0.059	0.058	0.056	0.056	0.055	**0.053**
		MAPE	50.882	29.568	29.518	65.052	28.733	32.806	28.669	58.212	27.335	32.052	29.032	27.186	25.591	26.287	25.109	24.676	23.918	23.451	**21.296**
	Oct	RMSE	0.162	0.136	0.111	0.158	0.112	0.089	0.104	0.117	0.101	0.088	0.107	0.101	0.097	0.098	0.095	0.094	0.095	0.094	**0.091**
		MAE	0.116	0.113	0.086	0.128	0.086	0.079	0.083	0.095	0.08	0.078	0.083	0.081	0.076	0.078	0.076	0.075	0.076	0.074	**0.072**
		MAPE	33.633	29.593	21.168	40.447	20.956	21.119	22.534	32.141	20.784	20.332	20.316	22.516	18.474	19.222	17.307	16.945	17.282	16.535	**15.721**
NSW	Jan	RMSE	0.144	0.242	0.184	0.148	0.168	0.166	0.196	0.165	0.141	0.164	0.181	0.14	0.137	0.138	0.135	0.133	0.132	0.131	**0.129**
		MAE	0.105	0.186	0.151	0.121	0.129	0.117	0.161	0.131	0.106	0.114	0.147	0.106	0.104	0.104	0.102	0.101	0.101	0.1	**0.099**
		MAPE	29.666	44.051	42.458	36.424	32.918	28.651	45.361	38.732	30.953	27.668	41.683	30.276	27.916	28.558	26.712	26.097	25.889	25.454	**24.431**
	Apr	RMSE	0.131	0.109	0.089	0.118	0.077	0.072	0.132	0.103	0.07	0.07	0.084	0.07	0.069	0.069	0.067	0.066	0.066	0.066	**0.064**
		MAE	0.103	0.082	0.071	0.096	0.056	0.045	0.115	0.081	0.046	0.042	0.066	0.045	0.044	0.044	0.043	0.042	0.041	0.041	**0.039**
		MAPE	35.287	31.934	28.738	40.011	20.298	17.966	47.908	51.731	17.512	15.855	28.131	17.132	16.139	16.664	16.148	15.807	15.559	15.323	**14.033**
	Jul	RMSE	0.139	0.111	0.118	0.122	0.092	0.109	0.131	0.131	0.089	0.108	0.113	0.086	0.085	0.086	0.084	0.083	0.083	0.081	**0.078**
		MAE	0.107	0.082	0.088	0.098	0.068	0.084	0.095	0.107	0.07	0.083	0.084	0.065	0.064	0.066	0.063	0.059	0.058	0.057	**0.056**
		MAPE	39.439	37.957	44.026	47.654	30.778	36.309	44.601	60.347	31.784	35.766	42.729	28.881	28.643	29.066	28.563	27.442	27.152	26.458	**25.832**
	Oct	RMSE	0.124	0.103	0.081	0.119	0.069	0.075	0.074	0.099	0.075	0.073	0.078	0.074	0.072	0.073	0.072	0.069	0.07	0.069	**0.066**
		MAE	0.096	0.084	0.062	0.095	0.055	0.056	0.061	0.073	0.055	0.053	0.059	0.059	0.053	0.054	0.052	0.051	0.052	0.051	**0.049**
		MAPE	33.934	33.762	22.451	42.343	19.219	17.963	23.344	43.725	19.203	17.906	21.414	17.701	16.674	17.388	16.895	16.402	16.668	16.284	**15.972**
QLD	Jan	RMSE	0.113	0.171	0.131	0.101	0.138	0.114	0.141	0.141	0.103	0.112	0.129	0.1	0.098	0.099	0.097	0.096	0.096	0.095	**0.094**
		MAE	0.085	0.137	0.102	0.077	0.107	0.091	0.117	0.119	0.08	0.088	0.099	0.077	0.075	0.076	0.076	0.075	0.074	0.074	**0.073**
		MAPE	23.473	32.848	25.508	19.503	24.972	22.342	31.997	37.828	20.965	20.878	24.193	19.429	18.979	19.336	18.771	17.844	17.794	17.283	**16.538**
	Apr	RMSE	0.159	0.136	0.247	0.167	0.129	0.146	0.231	0.182	0.128	0.143	0.243	0.125	0.124	0.125	0.123	0.119	0.119	0.118	**0.116**
		MAE	0.126	0.101	0.228	0.138	0.097	0.117	0.209	0.154	0.095	0.115	0.222	0.09	0.09	0.091	0.088	0.086	0.085	0.085	**0.084**
		MAPE	585.921	535.178	1626.104	1038.221	255.478	1220.361	2031.133	2347.967	279.445	1145.676	1431.886	264.303	259.604	267.454	249.098	228.343	225.312	220.791	**210.335**
	Jul	RMSE	0.137	0.112	0.104	0.134	0.103	0.099	0.121	0.125	0.1	0.097	0.102	0.099	0.098	0.097	0.095	0.092	0.091	0.091	**0.089**
		MAE	0.105	0.089	0.087	0.111	0.087	0.083	0.103	0.108	0.083	0.081	0.084	0.082	0.082	0.079	0.076	0.073	0.072	0.072	**0.071**
		MAPE	33.979	33.655	36.929	53.152	32.666	29.277	39.426	57.607	30.116	27.047	34.875	28.906	28.661	28.744	27.806	27.005	26.918	26.706	**25.403**
	Oct	RMSE	0.161	0.121	0.118	0.152	0.105	0.112	0.121	0.131	0.104	0.11	0.115	0.102	0.101	0.101	0.098	0.097	0.097	0.096	**0.095**
		MAE	0.113	0.093	0.079	0.117	0.076	0.075	0.088	0.105	0.077	0.072	0.074	0.077	0.075	0.075	0.074	0.073	0.072	0.073	**0.071**
		MAPE	32.612	28.315	21.539	33.927	22.361	20.288	27.918	36.841	21.166	18.887	19.773	19.989	19.202	19.961	19.046	18.568	18.339	18.161	**17.307**

TABLE VI: Time complexity (per second) of the proposed model and other compared benchmark models for VIC case study.

Model	Optimization time				Training time				Testing time			
	Jan	Apr	Jul	Oct	Jan	Apr	Jul	Oct	Jan	Apr	Jul	Oct
LR	-	-	-	-	59.1	61.2	58.4	59.2	30.8	28.8	30.2	29.1
ARIMA	-	-	-	-	53.2	55.9	53.5	56.3	23.3	24.1	22.9	23.9
CNN	-	-	-	-	62.5	59.1	61.7	61.4	32.5	31.9	32.7	32.9
RFR	-	-	-	-	63.2	56.6	58.7	58.5	32.8	33.1	33.6	32.5
XGBOOST	-	-	-	-	51.1	51.7	54.6	53.8	22.8	23.7	22.2	22.9
LSTM	-	-	-	-	64.4	63.3	59.6	60.3	33.5	33.8	34.2	34.7
SVR	-	-	-	-	58.1	56.6	55.1	57.9	29.8	27.8	29.1	28.6
MLP	-	-	-	-	56.8	57.4	56.3	58.6	26.8	26.4	27.1	27.4
PDRNN	270.3	285.4	281.1	274.7	61.4	60.9	56.4	60.2	31.5	31.2	30.9	30.8
PCCA–LSTM	-	-	-	-	62.2	67.4	65.9	63.3	33.4	32.5	32.9	33.4
EWELM	-	-	-	-	58.7	60.2	61.4	59.1	28.8	27.6	28.3	28.1
GWO-LSTM	271.6	277.8	280.3	279.5	66.5	65.1	68.4	67.4	35.4	35.2	34.1	33.8
EGWO-LSTM	266.3	262.5	271.6	269.5	70.1	70.9	69.5	71.8	37.8	38.1	37.1	37.4
PSO-CNN	261.8	264.4	268.1	265.5	58.9	60.5	57.1	58.8	29.9	28.7	29.1	30.1
GA-CNN	262.9	266.1	261.2	262.4	61.1	59.4	60.8	59.5	30.3	29.6	29.8	30.5
GWO-CNN	258.8	260.3	255.1	256.9	58.1	59.2	58.3	57.6	29.4	28.4	28.3	29.3
IGWO-CNN	260.9	265.5	267.9	258.5	61.3	62.2	61.9	60.2	31.1	30.9	31.2	31.6
MGWO-CNN	262.7	268.8	259.4	259.1	61.2	63.3	62.1	61.1	29.8	30.5	30.4	30.8
Proposed	255.7	254.8	256.1	251.7	50.8	49.7	52.6	53.4	22.5	22.9	22.1	21.5

sensitivity analysis to obtain their optimal values. Based on the results of sensitivity analysis, the values for inertia weight factor and social parameters are set to 0.7 and 1.4, respectively. Regarding GA, the values of crossover and mutation probability are set to 0.6 and 0.07, respectively. Besides, the CNN hyperparameters optimized in the proposed EGWO-CNN model are also used in the hybrid algorithms (GWO-CNN, PSO-CNN and GA-CNN). All of these benchmarks are implemented in Python version 3.7 programming environment and all the experiments are repeated 10 times for each model to have a solid performance evaluation. Table IV shows the optimal values for the hyperparameters of the CNNs optimized by EGWO. These values are chosen with respect to the lowest value of RMSE in each run of the proposed EGWO-CNN algorithm. These findings show that the EGWO-CNN chooses the values that are not heavy in simulation results and obtains the lowest RMSE. Thus, we use optimal values for the compared forecasting models that have resulted in the best output in each month. In other words, we set the values of the parameters of the compared approaches according to the optimal values obtained for each month. In this way, we can claim that the experiments are fair and the superiority of the proposed method is easily proved.

In Table V, the average values of prediction results for one day ahead load forecasting of three states are tabulated. The values in bold format represent the lowest value of their corresponding error metric. From this table, it can be observed that our proposed EGWO-CNN outperforms other eighteen benchmark models in terms of lower RMSE, MAE and MAPE in all cases. The obtained results also highlight the potential of the EGWO-CNN for one-day ahead forecasting. Among the compared models, the hybrid

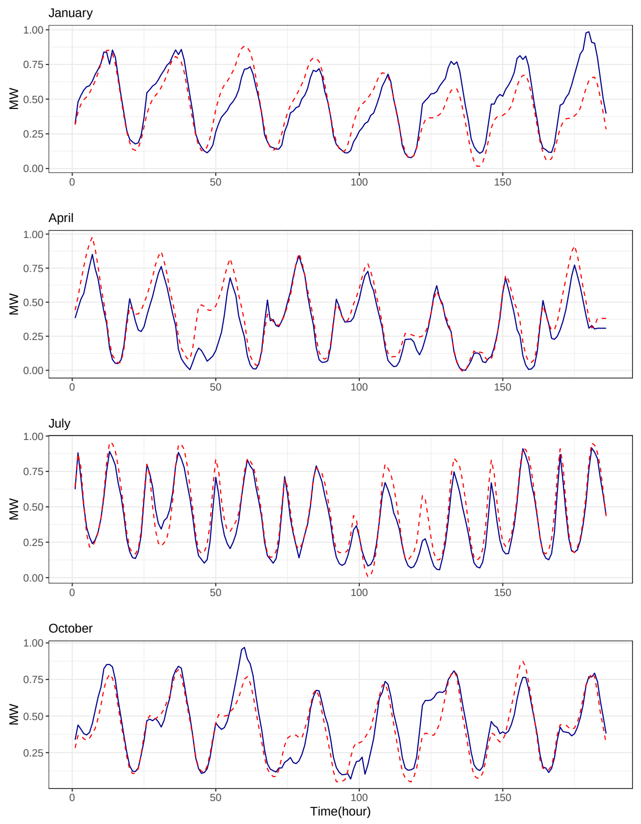

Fig. 4: Actual vs predicted points for QLD dataset. The real line represents the actual values and the dotted line represents the predicted values.

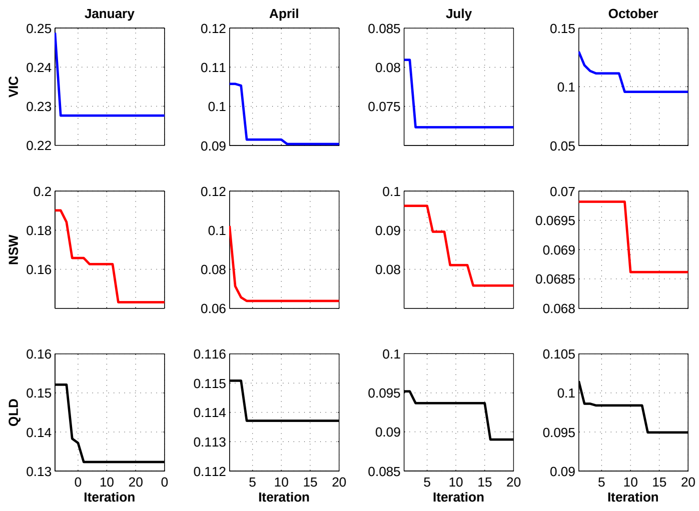

Fig. 5: Convergence profiles of proposed EGWO-CNN model for four months and three states based on RMSE metric.

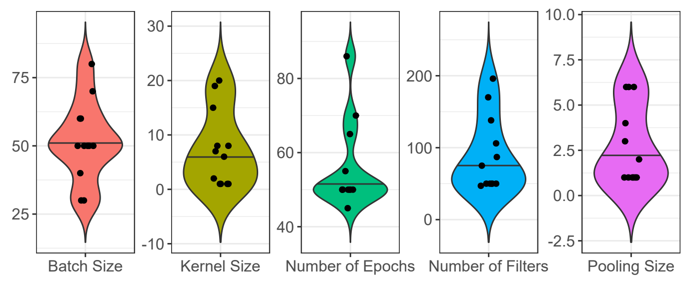

Fig. 6: Violin plots of best CNN configurations obtained by EGWO

models achieve competitive results in comparison with the state of the art algorithms. The closest competitor to the proposed method is the state of the art version of GWO called MGWO hybridized with CNN obtained close performance to the EGWO-CNN model. In Figs. 2, 3, and 4, the actual and predicted values generated by the proposed algorithm are represented. The blue and red lines represent the actual and predicted values obtained by EGWO-CNN, respectively. The obtained results closely agree with the actual electricity load which meet the min or max picks.

Fig. 5 represents the convergence curves of EGWO-CNN for four months of each state based on RMSE values for all training samples over the course of 20 iterations. This figure shows that EGWO-CNN can be converged in a faster way to obtain the optimal solution for each case. In Fig. 6, we illustrate the distributions of the optimized hyperparameters of CNNs using violin plots. This illustration is important since it shows a worthy estimate of the hyperparameters values for training of CNNs. Besides, they confirm and justify golden setting of CNN hyperparameters during training procedure for obtaining the best performance of the network.

In Table VI, the time performance of the proposed algorithm and other benchmarked algorithms are reported based on three time criteria including optimization, training and testing times for VIC dataset. It is worth noting that optimization time refers to the time of algorithms in which the optimization operators are used in them. As can be deduced from this table, the proposed algorithm in terms of optimization time, has the least time for different scenarios among the optimization algorithms. Also, in terms of training and testing time, our proposed EGWO-CNN algorithm has the shortest time for different months.

In summary, the extensive experiments support the efficient forecasting performance of EGWO-CNN approach for one-day ahead period. The capability of the proposed approach to optimize the hyperparameters of CNNs from load data results in accurate forecasting of future loads. The key advantages of the proposed EGWO-CNN algorithm are: 1) automating CNN hyperparameters, 2) avoiding being trapped in local minima, 3) being computationally cheap and 4) having fast convergence profiles during optimization process. The effective performance of the proposed method is verified by the lower forecasting errors such as RMSE, MAE and MAPE. The competitive performance of EGWO-CNN approach reveals that it can be used as an effective forecasting tool for time series prediction problems.

IV. Conclusion

In this study, we presented a novel evolutionary deep neural architecture search for the problem of load forecasting. To this end, EGWO is applied to fine tune the CNN hyperparameters automatically. The proposed approach was assessed with datasets from three regions of AEMO. In order to evaluate the robustness of the proposed method, several benchmark algorithms were also compared with the proposed method. Three error metrics including RMSE, MAE and MAPE were utilized for evaluating the performance of these forecasting algorithms. The obtained results indicated that the proposed neural architecture search method results in models outperforming benchmarks in most cases. For the future works, more advanced evolutionary methods will be applied to improve the convergence profile of CNNs. Also, the proposed neural architecture search will be applied to other datasets including renewable ones.

References

[1] M. Mishra, J. Nayak, B. Naik, and A. Abraham, "Deep learning in electrical utility industry: A comprehensive review of a decade of research," *Engineering Applications of Artificial Intelligence*, vol. 96, p. 104000, 2020.

[2] A. Heydari, M. M. Nezhad, E. Pirshayan, D. A. Garcia, F. Keynia, and L. De Santoli, "Short-term electricity price and load forecasting in isolated power grids based on composite neural network and gravitational search optimization algorithm," *Applied Energy*, vol. 277, p. 115503, 2020.

[3] A. Rafati, M. Joorabian, and E. Mashhour, "An efficient hour-ahead electrical load forecasting method based on innovative features," *Energy*, p. 117511, 2020.

[4] A. R. Hapka, "Dw bunn, ed farmer, comparative models for electrical load forecasting, wiley, belfast (1985), pp. 232,£ 24.95," 1986.

[5] X. Qiu, Y. Ren, P. N. Suganthan, and G. A. Amaratunga, "Empirical mode decomposition based ensemble deep learning for load demand time series forecasting," *Applied Soft Computing*, vol. 54, pp. 246–255, 2017.

[6] G. Paredes, L. Vargas, and S. Maldonado, "Reconfiguration and reinforcement allocation as applied to hourly medium-term load forecasting of distribution feeders," *IET Generation, Transmission & Distribution*, vol. 14, no. 9, pp. 1791–1798, 2020.

[7] M. Q. Raza and A. Khosravi, "A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings," *Renewable and Sustainable Energy Reviews*, vol. 50, pp. 1352–1372, 2015.

[8] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, p. 92, 2019.

[9] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[10] J. Long, S. Zhang, and C. Li, "Evolving deep echo state networks for intelligent fault diagnosis," *IEEE Transactions on Industrial Informatics*, 2019.

[11] S. Ahmadian and A. R. Khanteymoori, "Training back propagation neural networks using asexual reproduction optimization," in *7th Conference on Information and Knowledge Technology (IKT)*. IEEE, 2015, pp. 1–6.

[12] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy *et al.*, "Evolving deep neural networks," in *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Elsevier, 2019, pp. 293–312.

[13] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 24–35, 2019.

[14] A. Baldominos, Y. Saez, and P. Isasi, "On the automated, evolutionary design of neural networks: past, present, and future," *Neural Computing and Applications*, pp. 1–27, 2019.

[15] M. Bińkowski, G. Marti, and P. Donnat, "Autoregressive convolutional neural networks for asynchronous time series," *Proceedings of the 35th International Conference on Machine Learning*, pp. 580–589, 2017.

[16] I. Koprinska, D. Wu, and Z. Wang, "Convolutional neural networks for energy time series forecasting," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[17] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *2017 IEEE 19th Conference on Business Informatics (CBI)*, vol. 1. IEEE, 2017, pp. 7–12.

[18] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46–61, 2014.

[19] M. H. Nadimi-Shahraki, S. Taghian, and S. Mirjalili, "An improved grey wolf optimizer for solving engineering problems," *Expert Systems with Applications*, vol. 166, p. 113917, 2020.

[20] J. C. Bansal and S. Singh, "A better exploration strategy in grey wolf optimizer," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–20, 2020.

[21] M. Khubroo and S. J. Mousavirad, "A levy flight-based decomposition multi-objective optimization based on grey wolf optimizer," in *2019 9th International Conference on Computer and Knowledge Engineering (ICCKE)*. IEEE, 2019, pp. 155–161.

[22] H. Chantar, M. Mafarja, H. Alsawalqah, A. A. Heidari, I. Aljarah, and H. Faris, "Feature selection using binary grey wolf optimizer with elite-based crossover for arabic text classification," *Neural Computing and Applications*, vol. 32, no. 16, pp. 12 201–12 220, 2020.

[23] F. Jiang, Z. Peng, and J. He, "Short-term load forecasting based on support vector regression with improved grey wolf optimizer," in *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*. IEEE, 2018, pp. 807–812.

[24] S. Dai, D. Niu, and Y. Li, "Daily peak load forecasting based on complete ensemble empirical mode decomposition with adaptive noise and support vector machine optimized by modified grey wolf optimization algorithm," *Energies*, vol. 11, no. 1, p. 163, 2018.

[25] S. Shang, K.-N. He, Z.-B. Wang, T. Yang, M. Liu, and X. Li, "Sea clutter suppression method of hfswr based on rbf neural network model optimized by improved gwo algorithm," *Computational Intelligence and Neuroscience*, vol. 2020, 2020.

[26] M. Madhiarasan, S. Deepa *et al.*, "Elman neural network with modified grey wolf optimizer for enhanced wind speed forecasting," *Circuits and Systems*, vol. 7, no. 10, p. 2975, 2016.

[27] H. Liu, H. Wu, and Y. Li, "Smart wind speed forecasting using ewt decomposition, gwo evolutionary optimization, relm learning and iewt reconstruction," *Energy Conversion and Management*, vol. 161, pp. 266–283, 2018.

[28] L. Ge, Y. Xian, Z. Wang, B. Gao, F. Chi, and K. Sun, "A gwo-grnn based model for short-term load forecasting of regional distribution network," *CSEE Journal of Power and Energy Systems*, 2020.

[29] H. Xie, L. Zhang, and C. P. Lim, "Evolving cnn-lstm models for time series prediction using enhanced grey wolf optimizer," *IEEE Access*, vol. 8, pp. 161 519–161 541, 2020.

[30] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–34.

[31] A. Kavousi-Fard, H. Samet, and F. Marzbani, "A new hybrid modified firefly algorithm and support vector regression model for accurate short term load forecasting," *Expert systems with applications*, vol. 41, no. 13, pp. 6047–6056, 2014.

[32] H.-Z. Li, S. Guo, C.-J. Li, and J.-Q. Sun, "A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm," *Knowledge-Based Systems*, vol. 37, pp. 378–387, 2013.

[33] [Online]. Available: https://aemo.com.au/en/energy-systems/electricity/national-electricity-market-nem/data-nem

[34] N. A. Shrivastava, A. Khosravi, and B. K. Panigrahi, "Prediction interval estimation of electricity prices using pso-tuned support vector machines," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 2, pp. 322–331, 2015.

[35] W. Zhu, W. Yeh, J. Chen, D. Chen, A. Li, and Y. Lin, "Evolutionary convolutional neural networks using abc," in *Proceedings of the 2019 11th International Conference on Machine Learning and Computing*. ACM, 2019, pp. 156–162.

[36] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Transactions on Evolutionary Computation*, 2019.

[37] A. Baldominos, Y. Saez, and P. Isasi, "Evolutionary convolutional neural networks: An application to handwriting recognition," *Neurocomputing*, vol. 283, pp. 38–52, 2018.

[38] G. E. Box and G. M. Jenkins, "Time series analysis: Forecasting and control san francisco," *Calif: Holden-Day*, 1976.

[39] J. Friedman, T. Hastie, R. Tibshirani *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[40] L. Peng, S. Liu, R. Liu, and L. Wang, "Effective long short-term memory with differential evolution algorithm for electricity price prediction," *Energy*, vol. 162, pp. 1301–1314, 2018.

[41] L. Hernández, C. Baladrón, J. M. Aguiar, B. Carro, A. Sánchez-Esguevillas, and J. Lloret, "Artificial neural networks for short-term load forecasting in microgrids environment," *Energy*, vol. 75, pp. 252–264, 2014.

[42] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[43] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[44] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

[45] H. Shi, M. Xu, and R. Li, "Deep learning for household load forecasting—a novel pooling deep rnn," *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 5271–5280, 2017.

[46] S. Li, L. Goel, and P. Wang, "An ensemble approach for short-term load forecasting by extreme learning machine," *Applied Energy*, vol. 170, pp. 22–29, 2016.

[47] N. Wei, C. Li, J. Duan, J. Liu, and F. Zeng, "Daily natural gas load forecasting based on a hybrid deep learning model," *Energies*, vol. 12, no. 2, p. 218, 2019.