A Modified Weight Optimization for Artificial Higher Order Neural Networks in Physical Time Series

Noor Aida Husaini[1], Rozaida Ghazali[2]
Nureize Arbaiy[3], Norhamreeza Abdul Hamid[4]
Faculty of Computer Science & Information Technology
Universiti Tun Hussein Onn Malaysia, Johor, Malaysia

Lokman Hakim Ismail[5]
Faculty of Civil Engineering and Built Environment
Universiti Tun Hussein Onn Malaysia
Johor, Malaysia

***Abstract*—Many methods and approaches have been proposed for analyzing and forecasting time series data. There are different Neural Network (NN) variations for specific tasks (e.g., Deep Learning, Recurrent Neural Networks, etc.). Time series forecasting are a crucial component of many important applications, from stock markets to energy load forecasts. Recently, Swarm Intelligence (SI) techniques including Cuckoo Search (CS) have been established as one of the most practical approaches in optimizing parameters for time series forecasting. Several modifications to the CS have been made, including Modified Cuckoo Search (MCS) that adjusts the parameters of the current CS, to improve algorithmic convergence rates. Therefore, motivated by the advantages of these MCSs, we use the enhanced MCS known as the Modified Cuckoo Search-Markov Chain Monté Carlo (MCS-MCMC) learning algorithm for weight optimization in Higher Order Neural Networks (HONN) models. The Lévy flight function in the MCS is replaced with Markov Chain Monté Carlo (MCMC) since it can reduce the complexity in generating the objective function. In order to prove that the MCS-MCMC is suitable for forecasting, its performance was compared with the standard Multilayer Perceptron (MLP), standard Pi-Sigma Neural Network (PSNN), Pi-Sigma Neural Network-Modified Cuckoo Search (PSNN-MCS), Pi-Sigma Neural Network-Markov Chain Monté Carlo (PSNN-MCMC), standard Functional Link Neural Network (FLNN), Functional Link Neural Network-Modified Cuckoo Search (FLNN-MCS) and Functional Link Neural Network-Markov Chain Monté Carlo (FLNN-MCMC) on various physical time series and benchmark dataset in terms of accuracy. The simulation results prove that the HONN-based model combined with the MCS-MCMC learning algorithm outperforms the accuracy in the range of 0.007% to 0.079% for three (3) physical time series datasets.**

Keywords*—Modified Cuckoo Search-Markov Chain Monté Carlo; MCS-MCMC; neural networks; higher order; time series forecasting

I. Introduction

Time series forecasting involves developing a model or method that captures or describes the observed time series in order to understand the underlying causes. This research field looks for the "why" behind the time series dataset. This often involves making assumptions about data forms and breaking down time series into constitutional components [1, 2]. The challenge in time series forecasting is to provide a selection of techniques to better understand a dataset. In order to understand the past and predict the future event, it is important to analyze and optimize time series data using appropriate algorithms to understand underlying causes. There are many types of time series. For example; physical, financial and so forth [1, 3-5]. Time series forecasting have been addressed using classic methods such as the Autoregressive Integrated Moving Average (ARIMA) [6, 7], the Autoregressive Moving Average (ARMA) [7] and more. This linear model is the perfect choice for modeling time series events. However, they did not produce satisfactory results because they assumed a linear relationship between the past values of the series and ignored the non-linear relationships between these models.

Contrary, non-linear model such as Neural Networks (NN) has shown better performance as compared to linear models. Not to mention, it has been applied in dealing with issues of time series forecasting [8-12]. The NN is a type of parallel computer structure, which several of processing units are linked together thus that the computer's memory is distributed, and information is passed in a parallel manner. Many NN architectures and algorithms have been developed thus far, namely multilayer feedforward networks, deep learning methods and so on [12-14]. Of these networks, the interest is gradually shifting towards using feedforward networks. Multilayer Perceptron (MLP), a class of feedforward networks, has been found to perform best in broader applications related to forecasting issues [1, 8-11]. The MLP is well-known for having the ability to map both linear and non-linear relationship if the number of nodes and layers are given sufficiently. However, MLP needs excessive learning time which may lead to overfitting [15, 16]. This is more likely to happen to the networks with many processing units and results in poor generalizability. The ability to generalize, that is to produce outputs from unknown inputs, is critical when the NN is used in time series forecasting. For this reason, networks with few parameters are preferred, fair enough to provide an adequate fit in order to avoid over-training [2, 15].

Therefore, to correct this failing, some Higher Order Neural Networks (HONN) is suggested. In this study, two (2) types of HONN were highlighted; Pi-Sigma Neural Network (PSNN) [17] and Functional Link Neural Network (FLNN) [18]. The PSNN utilizes product units at the output units that indirectly incorporate the capabilities of HONN while using a fewer number of weights and processing units. It has a regular structure, exhibits much faster learning, and is open to the incremental addition of units to attain a desired level of complexity. Meanwhile, the FLNN removes the need for hidden layers and hidden nodes by utilizing a higher order term

This work was funded by the Research Management Centre, Universiti Tun Hussein Onn Malaysia (Research Fund E15501).

to expand its input spaces into higher dimensional space within the single layer units. This simple architecture reduced the number of trainable parameters needed whilst reduces the learning complexity during the network training [19]. Taken as a whole, HONN are simple in their architecture and have fewer number of trainable parameters to deliver the input-output mappings as compared to the standard NN.

The standard method to train the NN is the well-known Backpropagation (BP) algorithm [20]. The existing BP algorithm, however, has several limitations including easily stuck into local minima, especially when dealing with highly non-linear problems [15]. The BP algorithm is also very dependent on the choices of initial values of the weights as well as other parameters. For instance, the BP algorithm is generally very slow as it requires small learning rates for stable learning. The momentum variation is usually faster than straightforward gradient descent since it allows higher learning rates while maintaining stability. However, it is still too slow for many practical applications.

Therefore, we used the Modified Cuckoo Search-Markov Chain Monté Carlo (MCS-MCMC) learning algorithm [21], that employs the learning rules to find the optimal weights in HONN models, thus overcome the BP drawbacks for this forecasting issue, and apply this method to several physical time series datasets. The results were compared with standard MLP and several HONN-based models. This MCS-MCMC used to enhanced the Modified Cuckoo Search (MCS) [22] by adopting Markov Chain Monté Carlo (MCMC) random walk. Those can be achieved by Markov chain mixing and integrated autocorrelation of a function of interest [23]. Therefore, it is useful in speeding up the convergence rate and obtaining higher accuracy rate.

Following this section, this paper is organized as follows: Section II presents the Related Works, followed by Section III which discuss the Architecture of HONN. Section IV poses the Experimental Results and Section V examines the Computational Results. Finally, Section VI concludes the work done.

II. Related Works

Weight optimizations are made in a wide range of diverse disciplines. Some methods that can be used to update weights in NN are BP, Genetic Algorithm (GA) [24, 25], Support Vector Machine (SVM) [26] and more. The concept of weight optimization by NN has become an active research field. It goes without saying that Swarm Intelligence (SI) played a role too. Among those swarm-based algorithms that have achieved significant popularity in the last few years are Evolutionary Algorithm (EA) [27, 28], Differential Evolution (DE) [29, 30], Artificial Bee Colony [16, 31] and Cuckoo Search [32].

The work presented in [33] combines Particle Swarm Optimization (PSO) and Extreme Learning Machine (ELM) to forecast the inflation rate in Indonesia. It uses PSO to optimize weight in order to obtain the optimal input values in ELM. In [34], the work binds the Ant Colony Optimization (ACO), PSO and 3-Opt algorithms. The PSO algorithm is used to optimize the parameter values used in the ACO algorithm for city selection operations, and defines the significance of inter-city pheromone and distances. 3-Opt heuristic approach to boost the local solutions is applied to the proposed method. The performance of the combined method becomes very significant in terms of solution quality and robustness. In the meantime, the research in [35] dealt with Whale Optimization Algorithm to optimize the weights and biases. Based on the findings, this algorithm has demonstrated the ability to solve a wide range of optimization issues and surpass the BP algorithm.

In conjunction with that, [36] presented GA with DE to change the weight parameters encoded within the structure by optimizing the network topology using GA and set the network weights using DE. Similar to [36], [37] combines GA and NN to increase the NN performance in diagnosing coronary artery disease. This somewhat shows surprising results which make the levels of accuracy, sensitivity and specificity achieved by that combination. In another study, [38] optimized the weight to speed up the convergence rate by reparametrizing the weight vectors in NN. Weight optimization is also studied by [39] using PSO. In his work, he combined the multiresolution analysis techniques with NN to forecast the next-day event. The findings suggested both results and good forecasting efficiency. Other research conducted by [40] used grid search technique to calculate the best value of SVM parameters. The use of those technique is crucial to forecast the time series event. The result shows that the SVM outperformed NN in terms of Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE).

In particular, the key reason why weight parameters are optimized is to prevent local minima and convergence speed. This is because, weights are the relative strength of node-to-node connections in NN. Besides, those optimization treats important topics such as having a particular way of manipulating and expanding the problem's search space, which provides a detailed overview of how to manage such continuous domains. Instead, one of the well-known solutions is to find values of the variables that optimize the objectives. However, the variables are always limited, or somehow constrained. Therefore, in order to identify those values, experiments should focus on optimizing the objective functions or error functions due to the use of a common randomization arbitration and local search. Those parameter needs to be optimized subsequently to build up such appropriate and effective models. Once the effective models being developed, then the parameter is in its optimality conditions. It is however, the need for thorough research in order to evaluate the correct parameter measurement is still in doubt.

III. Architecture of HONN

In this study, the MCS-MCMC learning algorithm [21] is used to search for optimal weight parameters than can minimize the objective function in PSNN and FLNN network models. We replaced BP algorithm in the standard PSNN and FLNN with MCS-MCMC learning algorithm. The replacement is made to overcome the gradient-based learning algorithm drawbacks in BP algorithm that are slow, and easily get stuck into local minima [15]. Table I indicates the needs of MCS-MCMC that overcome the existing BP and MCS learning algorithm.

TABLE. I. COMPARISON OF BP, MCS AND MCS-MCMC LEARNING ALGORITHM

BP	MCS	MCS-MCMC
• Stuck into local minima. • Very dependent on the initial weights. • Need more parameter to be set up.	• Caters slow convergence encountered by BP. • Less parameter to be set up.	• Reduce complexity. • Speed up convergence rate. • Initialize weight value for better way/solutions and abondoned poor values.

According to Table I, the MCS-MCMC is used for weight initialization and weight update (replacing the BP algorithm in the standard PSNN and FLNN). The weights and biases were calculated and updated for the complete training that represents the architecture. Those can be achieved by starting it with random values followed by several repeated attempts on discovering better solutions and abandoning the poor values. The architecture of Pi-Sigma Neural Network-Markov Chain Monté Carlo (PSNN-MCMC) and Functional Link Neural Network-Markov Chain Monté Carlo (FLNN-MCMC) are presented in Fig. 1 and Fig. 2.

$x_1, x_2, \ldots x_n$ denotes input vectors, w_{ij} denotes adjustable weights for input vectors to linear summing unit, σ is the non-linear activation function, $h_1, h_2, \ldots h_l$ indicates the summing units, y is the output node and w_{jk} is the fixed weights from linear summing units to the output layer. Step-by-step process in PSNN-MCMC:

Step 1: Initialize weights w_{ij} from input vector to the linear summing unit h_l with a random number using MCS-MCMC learning algorithm. Those random weights are evaluated from layer-to-layer to improve the searching strategies to get the optimal weights set.

Step 2: Transform the optimization parameters (weights and biases) into the objective function.

Step 3: Feed the objective function into the MCS-MCMC learning algorithm to search for optimal weight parameters.

Step 4: Calculate error.

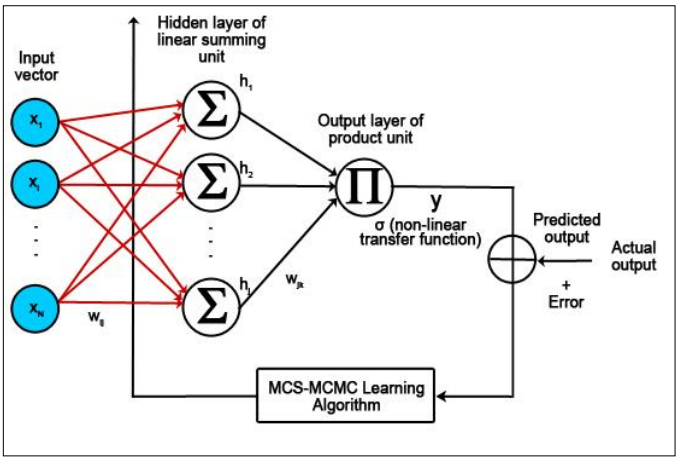


Fig. 1. The Architecture of PSNN-MCMC.

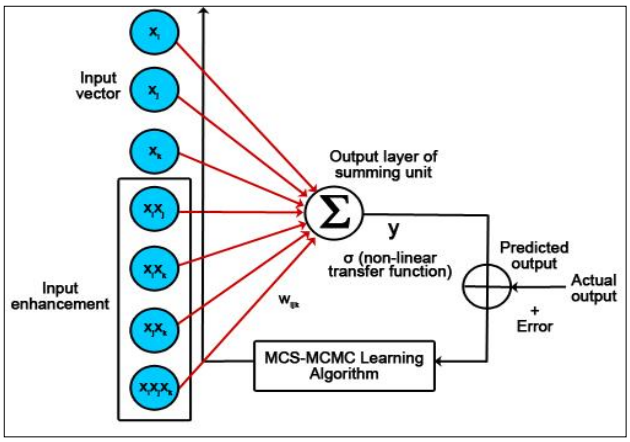


Fig. 2. The Architecture of FLNN-MCMC.

x_i, x_j, x_k is the input vector, w_{ijk} is the adjustable weight, y is the output, and σ is the non-linear activation function.

Step-by-step process in FLNN-MCMC:

Step 1: Initialize weights w_{ijk} with a random number using MCS-MCMC learning algorithm.

Step 2: In the initial process, transform the standard FLNN architecture (weight and biases) into the objective function.

Step 3: Feed the objective function, along with the training data, into the MCS-MCMC learning algorithm to search for optimal weight parameters to minimize the objective function.

Step 4: Tune the weight changes using the MCS-MCMC learning algorithm based on the error calculation (the difference between actual and predicted outputs).

Step 5: Obtain the optimal weights set from the training phase and used upon unseen data for forecasting.

IV. EXPERIMENTAL RESULTS

A. Data Preparation

Appropriate datasets should be provided to determine the problems encountered and evaluate the performance of the proposed PSNN-MCMC and FLNN-MCMC, and other models; standard PSNN, Pi-Sigma Neural Network-Modified Cuckoo Search (PSNN-MCS), standard FLNN, Functional Link Neural Network-Modified Cuckoo Search (FLNN-MCS), and standard MLP. The performance are evaluated based on the lowest Mean Squared Error (MSE) [41, 42] and Root Mean Squared Error (RMSE) [43]. Based on the previous records, the maximum, minimum and average measurements of three (3) datasets are tabulated in Table II.

TABLE. II. THE DATASETS EVALUATIONS

Dataset	Minimum	Maximum	Average	Data Size
Relative Humidity	69.5000	98.1000	85.9035	50, 840
Temperature	23.7000	29.5000	26.7543	1, 813
Santa Fe Laser	0	255	59.8661	3, 972

Relative Humidity: The datasets were collected from Malaysian Meteorological Department (MMD). Each dataset consists of 50, 840 instances which are covered from year of 1992 until 2009 [44].

Temperature: The datasets were collected from MMD. Each dataset consists of 1, 813 that covers over year of 1992 until 2009 [44].

Santa Fe Laser: A univariate time series derived from laser-generated data recorded from a Far-Infrared-Laser in a chaotic state. This benchmark datasets are composed of a clean low-dimensional non-linear and stationary time series with the total number of 3, 972 instances.

The reason for choosing these datasets are due to the stability they owned compared to other datasets. The stability is depending on the types of data and factors affecting them [45, 46]. For instance, the time series signals were observed on a highly non-stationary and/or non-linear range [47, 48]. Non-stationary is a common property to vary time-series models, which means, a variable has no clear tendency to return to a constant value or a linear trend. To note, the stability is the key to predictability. Therefore, a stable dataset is needed to predict the current trend. These physical time series data, later, were fed to all NN to capture the underlying rules of the movement.

B. Data Pre-processing

Mostly, data gathering somehow are loosely controlled. Thus, resulting in outliers, impossible data combinations, and may contains missing values. Therefore, the data need to be pre-processed to avoid errors and misleading results Fig. 3. The data pre-processing involves cleaning, shifting and normalizing the raw data into a format that improves the performance of the subsequent modules [18, 49].

C. Data Partition

Data partitioning is highly required by NN to obtain best NN models. Hence, in this study, we divide the datasets into three (3) partitions: 60% for training, while 20% for both testing and validation.

Training Set: Served the model for training purposes which allows the model to produce an output closer to the target value. Therefore, it must have more significant portion than the data being used for testing and validation.

Validation Set: Used to evaluate a given model, in which the sample of data used to provide an unbiased evaluation of a model fit on the training dataset fine-tunes the model. This set is also essential to avoid overfitting.

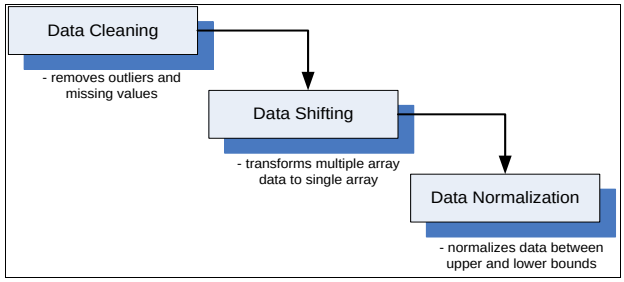


Fig. 3. Data Pre-Processing Process.

Testing Set: Describes how the models will perform on new, unseen data in order to evaluate the model. This sample provides an unbiased evaluation of a final model fit on the training dataset. It is only used once a model is thoroughly trained.

The split ratio of the datasets mainly relies on two (2) criteria. First, the total number of samples in the dataset. Second, the actual model going to be trained. Some models need substantial data to train upon. Therefore, in this study, more massive training sets should be optimized. Models with very few hyperparameters (e.g., momentum, learning rate, etc.) will be easy to validate and tune. As is, the validation set can probably reduce. However, if the model has many hyperparameters, an extensive validation must be set as well. All in all, like many other things in NN, the training-testing-validation split ratio is also quite specific based on some instances, and it gets easier to make a judgment as more training used.

D. Parameters Settings

The parameters of an NN are learned during the training stage. Learning (or training) is a process by which the tunable weights of a network are adapted through a continuous process of simulation whereas the network is embedded. The most basic method of training a network is a trial-and-error procedure [15]. During the learning phase, the network learns until its weight continues to tweak. The same set of data is then processed many times as the connection weight continues to improve. Parameters must be specified during training for any given NN architecture. For all network models, input nodes are set between 5 and 7 nodes, higher nodes / nodes between 2 and 5 (except for standard MLP) and one (1) for output nodes. The parameter settings for all network models are tabulated in Table III.

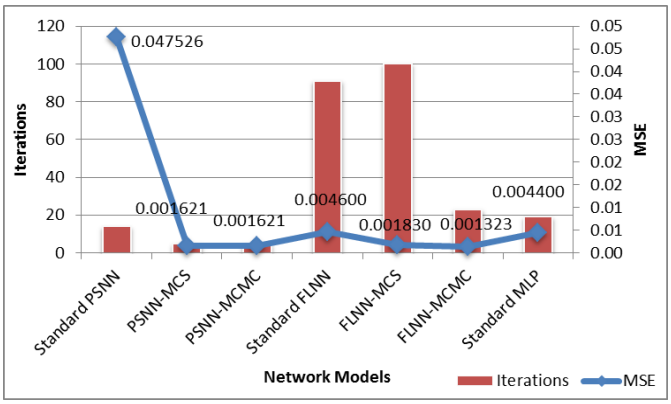
TABLE. III. PARAMETER SETTINGS FOR ALL NETWORK MODELS

Parameters	Values	References
Initial weights	$[0.25,0.75]$	[15]
Learning Rate	0.2	[15]
Momentum	0.3	[15]
Minimum Error	0.001	[15]
Epoch	1000	[15]
Initial Value, A	1	[23]
Step size, α	0.01	[23]
Probability, P_α	0.25	[22]
Initial Value, θ	1	[23]
n	5	[23]
a	4	[23]
Minimum Error	0.001	[15]
Number of Generation	1000	[22]
Input Nodes	5 to 7	[15]
Network's Order	2 to 5 (for rest of NN models) 3 to 8 (for MLP)	[15]
Output Node	1	[15]
Transfer Function	Sigmoid	[15]

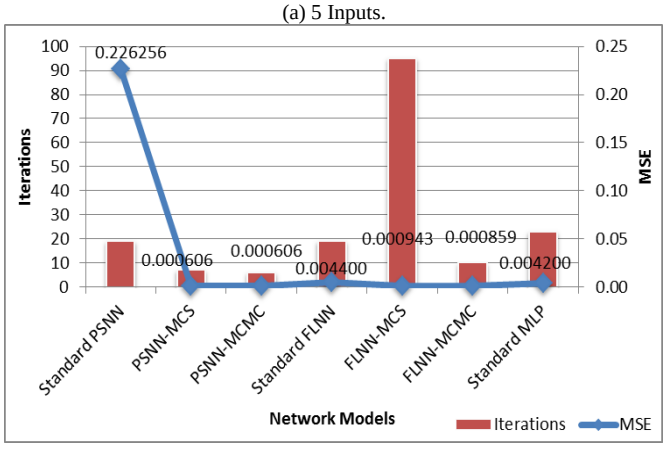
V. Computational Results

A. Relative Humidity Dataset

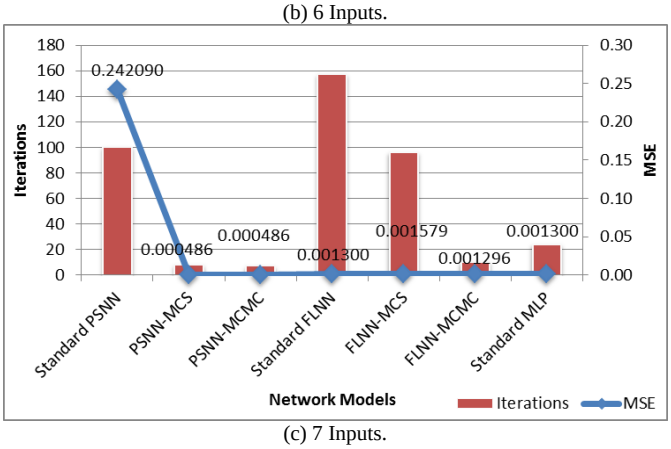
Referring to Fig. 4, the MSE results for Relative Humidity with 5 to 7 input nodes are visualized. As the 5 inputs were supplied, FLNN-MCMC, PSNN-MCMC and PSNN-MCS lead the ranks. When inputs 6 and 7 were loaded, PSNN-MCMC, PSNN-MCS and FLNN-MCMC outperformed. Seemingly, based on the results, the performances of the network in which the learning method had been replaced by MCS-MCMC learning algorithm are much preferable compared to the networks with standard MCS algorithm.



(a) 5 Inputs.



(b) 6 Inputs.



(c) 7 Inputs.

Fig. 4. Performance Comparison on Relative Humidity.

B. Temperature Dataset

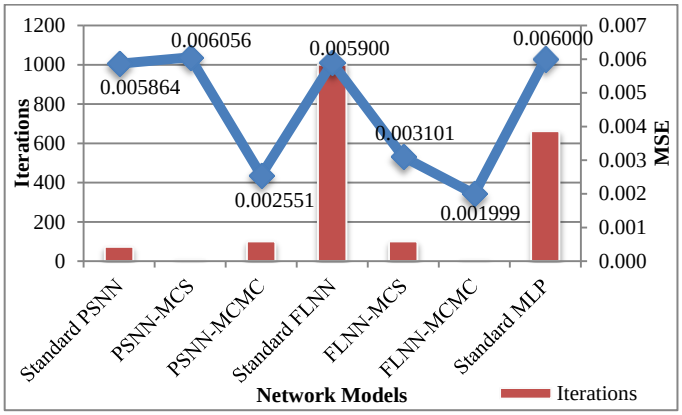
Fig. 5 graphically shows the performance comparison for all the networks on Temperature dataset. According to the results plotted in Fig. 5, the first, second and third ranks are FLNN-MCMC, PSNN-MCMC and FLNN-MCS for 5 inputs, FLNN-MCMC, FLNN-MCS and PSNN-MCMC for 6 and 7 inputs. From these results, it is said that the incorporation of MCS-MCMC learning algorithm into both PSNN and FLNN network models could help to minimize the error rate, thus assists the network to converge quickly. As it has been pointed out, FLNN-MCMC shows the least MSEs compared to all network models generated. Therefore, by having the least MSE, it combines both the estimator's variance and its bias to the extent that the estimated value is derived from the truth. In addition, the positive tendency in the Temperature dataset itself indicates that the data have a strong influence / fluctuation that is stable enough to handle the network model integrated with the MCS-MCMC learning algorithm.

C. Santa Fe Laser Dataset

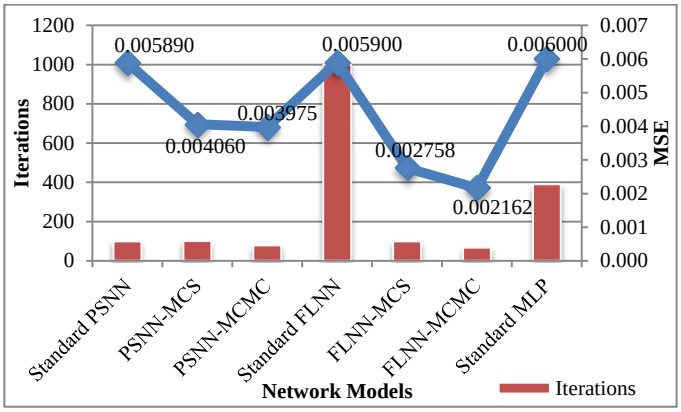
In view of inputs 5, 6 and 7, the FLNN-MCMC also outperformed the other network models for 60:20:20 data partition. Fig. 6 shows the results with respect to iterations and MSE values. From these statistics, it can be noted that the FLNN-MCMC network model performed better than the other network models with stable results even when dealing with the Santa Fe Laser dataset's temporal behavior.

The current study includes trials of MCS-MCMC learning algorithm on various network models. From the results, it is proved that, in this study, it is affirmative that the networks with MCS-MCMC learning algorithm were well generalized and showed least error compared to other network models, which could represent non-linear function. The MCS-MCMC's existence as the learning algorithm that replaces the existing BP algorithm enabled fast and rapid training. A significant advantage of the MCS-MCMC is that the learning algorithm can automatically adjust better parameters to find excellent parameter values with little user interference, which being accomplished through Markov chain mixing and a functional of interest integrated autocorrelation. Overall, the use of MCS-MCMC learning algorithm was discovered to be able to perform on various ranges of datasets.

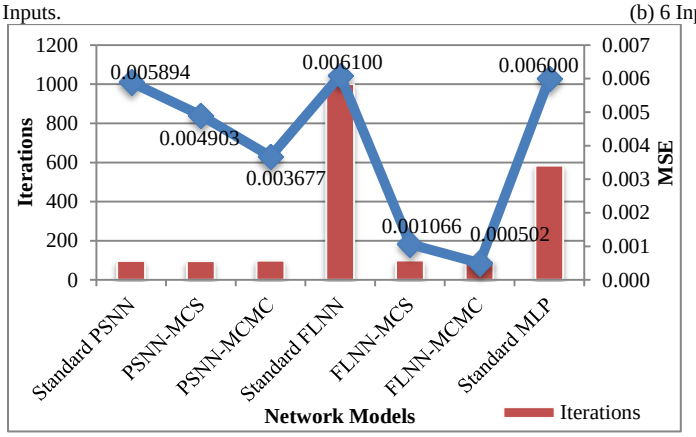
The MCS-MCMC is developed for initializing and updating the weights in HONN-based models. The use of Swarm Intelligence (SI) techniques in MCS-MCMC allows it to expand their input space to a higher dimensional space where linearity separable is possible has led to a significant effect on improving the network performance. The network is computationally efficient and is capable of modelling non-linear input-output mappings when learning the time series data, thus justified the potential use of this model by practitioners. Besides, the results clearly showed that the MCS-MCMC substantially at par with the computational efficiency of the training process, and has been developed in order to produce more realistic and acceptable results.



(a) 5 Inputs.

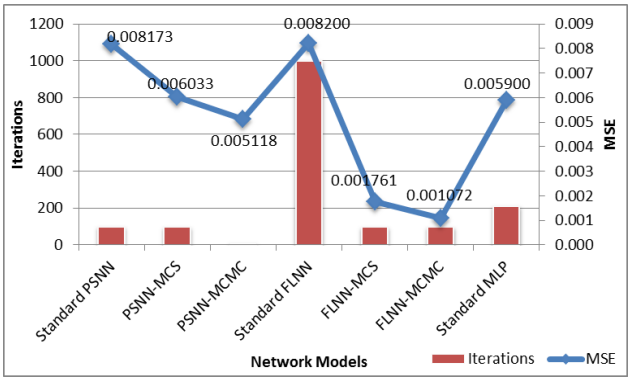


(b) 6 Inputs.

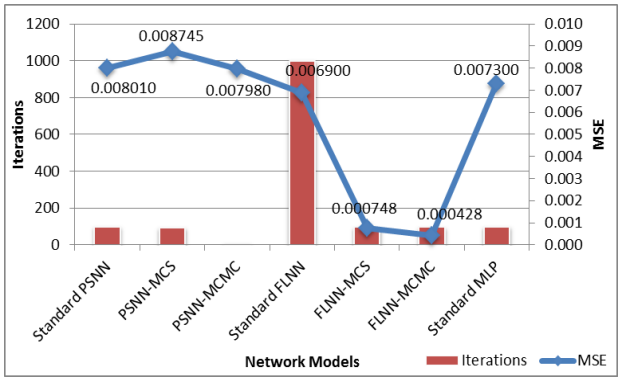


(c) 7 Inputs.

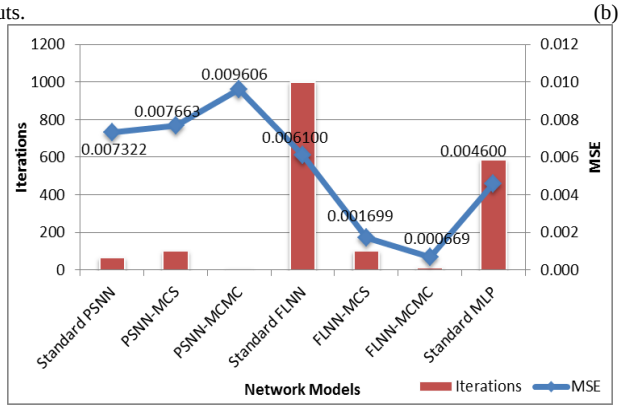
Fig. 5. Performance Comparison on Temperature.



(a) 5 Inputs.



(b) 6 Inputs.



(c) 7 Inputs.

Fig. 6. Performance Comparison on Santa Fe Laser.

D. Discussions

In this section, several issues raised by different NN comparisons are addressed. Because the results presented previously include extensive simulations, this section describes the observations obtained from the entire experimental results.

1) Model performances based on ranking: The simulation results in Section V, Subsection A were summarized in Tables IV to VI. This tables cover inputs ranges from 5 to 7 and seven (7) network models. Table IV shows the overall rank for Relative Humidity on all networks.

From Table IV, the PSNN-MCMC outperformed other network models by getting the highest average ranking. This demonstrates that the accuracy rate is enhanced by integrating the MCS-MCMC learning algorithm with HONN. Table V indicates the overall rank for Temperature on all networks.

According to Table V, FLNN-MCMC outperformed the other network models by having the highest average rank. This is followed by FLNN-MCS and PSNN-MCMC in the second and third rank, respectively. Basically, those swarm-based learning algorithm helps to overcome the drawbacks of the existing BP algorithm. Table VI summarizes data on all networks from the Santa Fe Laser dataset.

The results in Table VI show that the FLNN-MCMC provides a lower MSE than the other network models. This is accompanied by FLNN-MCS that falls into the second place and standard MLP in the third place. Based on these outcomes, it is concluded that implementing the swarm-based learning algorithm in HONN helps network models converge with lower iterations and lower error rate. Therefore, improves the network performance indirectly.

TABLE. IV. OVERALL RANK FOR RELATIVE HUMIDITY ON ALL NETWORKS

Inputs	Standard PSNN	PSNN-MCS	PSNN-MCMC	Standard FLNN	FLNN-MCS	FLNN-MCMC	Standard MLP
5	7	3	2	6	4	1	5
6	7	2	1	6	4	3	5
7	7	2	1	5	6	3	4
Mean Rank	7.00	2.33	1.33	5.67	4.67	2.33	4.67
Overall Rank	7	2	1	6	4	2	4

TABLE. V. OVERALL RANK FOR TEMPERATURE ON ALL NETWORKS

Inputs	Standard PSNN	PSNN-MCS	PSNN-MCMC	Standard FLNN	FLNN-MCS	FLNN-MCMC	Standard MLP
5	4	7	2	5	3	1	6
6	5	4	3	6	2	1	7
7	5	4	3	7	2	1	6
Mean Rank	4.67	5.00	2.67	6.00	2.33	1.00	6.33
Overall Rank	4	5	3	6	2	1	7

TABLE. VI. OVERALL RANK FOR SANTA FE LASER ON ALL NETWORKS

Inputs	Standard PSNN	PSNN-MCS	PSNN-MCMC	Standard FLNN	FLNN-MCS	FLNN-MCMC	Standard MLP
5	6	5	3	7	2	1	4
6	6	7	5	3	2	1	4
7	5	6	7	4	2	1	3
Mean Rank	5.67	6.00	5.00	4.67	2.00	1.00	3.67
Overall Rank	6	7	5	4	2	1	3

2) The accuracy: In this section, we presented the result based on the percentage of RMSE and Accuracy. The RMSE used to measures how much error there is between the actual and the target output [42]. In other words, it tells how concentrated the data is around the line of best fit. In general, if the value of RMSE getting lower, the better performance will be produced.

Tables VII to IX show the experimental results on all datasets. The table consisted of six (6) elements. The first element indicates the network model; and the second element designates the best network structure. This is accomplished by the method of trial-and-error procedure [15]. The third element specifies the number of trainable weights. Those values are collected during experiments. The fourth element is the RMSE value acquired through Equation (1):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(P_i - \tilde{P}_i\right)^2}{n}} \quad (1)$$

where n is the total number of data patterns, P_i and $\tilde{P}_i$ represent the actual and predicted output value, respectively. Equation (2) provides the sixth element (Accuracy in percentage). The simulation results later being compared in the form of accuracy rate.

$$Accuracy = \left(1 - \frac{MSE}{2}\right) \times 100 \quad (2)$$

where MSE is mean squared error [42].

TABLE. VII. EXPERIMENTAL RESULTS ON RELATIVE HUMIDITY

Network Model	Best Network Structure	No. of Trainable Weights	RMSE	Accuracy (%)
Standard PSNN	5-2-1	10	0.21801	97.624
PSNN-MCS	7-5-1	35	0.02205	99.976
PSNN-MCMC	7-2-1	14	0.02205	99.976
Standard FLNN	7-4-1	127	0.03606	99.935
FLNN-MCS	6-4-1	57	0.03071	99.953
FLNN-MCMC	6-4-1	57	0.02931	99.957
Standard MLP	7-7-1	56	0.03606	99.935

TABLE. VIII. EXPERIMENTAL RESULTS ON TEMPERATURE

Network Model	Best Network Structure	No. of Trainable Weights	RMSE	Accuracy (%)
Standard PSNN	5-3-1	15	0.07658	99.707
PSNN-MCS	6-2-1	12	0.06372	99.797
PSNN-MCMC	5-2-1	10	0.05051	99.872
Standard FLNN	5-4-1	31	0.07681	99.705
FLNN-MCS	6-4-1	57	0.05252	99.862
FLNN-MCMC	7-3-1	120	0.02241	99.975
Standard MLP	6-7-1	49	0.07746	99.700

TABLE. IX. EXPERIMENTAL RESULTS ON SANTA FE LASER

Network Model	Best Network Structure	No. of Trainable Weights	RMSE	Accuracy (%)
Standard PSNN	7-4-1	28	0.08557	99.634
PSNN-MCS	5-2-1	10	0.07767	99.698
PSNN-MCMC	5-3-1	15	0.07154	99.744
Standard FLNN	7-4-1	127	0.07810	99.695
FLNN-MCS	6-4-1	57	0.02735	99.963
FLNN-MCMC	6-4-1	57	0.02069	99.979
Standard MLP	7-8-1	64	0.06782	99.770

According to the results on Relative Humidity dataset (refer to Table VII), the HONN-based models being incorporated with MCS-MCMC learning algorithm give significant percentage around 99.953% to 99.976%. while for Temperature dataset (refer to Table VIII), the values vary from 99.797% to 99.975%. For Santa Fe Laser dataset (refer to Table IX), the FLNN-MCMC achieved highest percentage of Accuracy with the value of 99.979%.

The experimental results vary depending on the datasets. The algorithm can readily mapped the function if the data is sufficiently stable, thus delivering much better and stable outcomes. Otherwise, it could result in an extensive training algorithm. As the time series datasets exhibit a very strong trend, it shows obvious up and down movement. Therefore, during the training of such datasets, the networks were used to learn the precise values of each data point. This sometimes could lead the networks failed to respond well to the underlying chaotic structure within the data behaviour. Hence, to correctly predict the value from one point to another point is a challenging task.

3) Threat to validity and improvements: In this study, the fairness of experimentations involving SI technique are levelled to minimize threats to validity. One of major concerned was regarding the validity of parameter setting for each SI technique. In order to ensure fair comparisons, all parameter settings for all the network models, involving the input settings, learning rate and stopping criteria are set with the same value (revisit Section III, D). Another concerned was regarding the network structure for all the network models; the standard PSNN, PSNN-MCS, PSNN-MCMC, standard FLNN, FLNN-MCS, FLNN-MCMC and standard MLP. The network structure for those network models cannot be equivalent for all datasets in the experiment as they may yield unfair results. Therefore, to ensure fair prediction performance results, the network structure issue is addressed.

The critical part is on generalization. It is on how the network generates lowest MSE. For this reason, the best model is regarded to the NN structure that offers the greatest proportion of improvements. The simulation results are benchmarked against seven (7) NN models. The improvements for MCS-MCMC learning algorithm on both PSNN and FLNN for all datasets are measured in Equations (3) and (4). Let a be standard PSNN, b be PSNN-MCS, c be PSNN-MCMC, d be standard FLNN, e be FLNN-MCS, f be FLNN-MCMC and g be standard MLP.

$$Improvement_t(\%) = \frac{\left| \left(c - \frac{a+b+c+d+e+f+g}{7} \right) \right|}{c} \times 100\% \quad (3)$$

$$Improvement_f(\%) = \frac{\left| \left(f - \frac{a+b+c+d+e+f+g}{7} \right) \right|}{f} \times 100\% \quad (4)$$

$Improvement_t$ denotes improvement for PSNN-MCMC while $Improvement_f$ denotes improvement for FLNN-MCMC [42]. The overall improvements for PSNN-MCMC and FLNN-MCMC are tabulated in Tables X to XI. The findings on Table X show that the PSNN-MCMC provides significant improvement in all datasets where the PSNN-MCMC can improve the accuracy. This is also applicable to FLNN-MCMC in Table XI.

As can be seen from Tables X and XI, the MCS-MCMC learning algorithm can train and improve the accuracy of the HONN network model. Thus, it makes the best improvement on Relative Humidity dataset with the value of 0.707% on PSNN-MCMC and 0.670% on FLNN-MCMC when compared to other datasets. Both network models operate approximately 0.007 % to 0.079%.

TABLE. X. THE OVERALL IMPROVEMENTS OF PSNN-MCMC

Datasets	Network Structure	Improvement of PSNN-MCMC (%)
Relative Humidity	7-2-1	0.707
Temperature	5-2-1	0.116
Santa Fe Laser	5-3-1	0.079

TABLE. XI. THE OVERALL IMPROVEMENTS OF FLNN-MCMC

Datasets	Network Structure	Improvement of FLNN-MCMC (%)
Relative Humidity	6-4-1	0.670
Temperature	7-3-1	0.320
Santa Fe Laser	6-3-1	0.391

As the time series have chaotic behavior, this approach offers significant advantages over the standard network models such as improved simulations and lower error rate, due to their ability to better approximate complex, non-smooth and often discontinuous training datasets. To conclude, it is confirmed that HONN, when incorporated with MCS-MCMC learning algorithm, helps to overcome the drawback of the existing BP algorithm that prone to overfit and stuck into local minima. Thus, improve the network performance and increase the accuracy by getting the highest average ranking.

VI. Conclusion

The higher demands for SI techniques justify the need for a more effective, better solutions approach. The findings of this study will redound to the benefit of the SI field, considering that SI plays a vital role in optimization issues today. Therefore, the MCS-MCMC learning algorithm nailing down the optimal weight values in HONN which helped in dealing with slow convergence and poor generalization. Those are derived from the findings which later will be used to predict the time series event better. This study may also advantageous for certain sectors such as meteorological department that applies the non-linearity relationship in meteorological process. On the other hand, by obtaining outstanding performance on various ranges of time series datasets, it may reduce the risk in decision making. Thus, this approach significantly matches the idea. Therefore, the effectiveness of any decision depends upon the nature of a sequence of events preceding the decision. Furthermore, this study would be beneficial to the researchers, as it can provide baseline information on the different approach of SI and NN.

Acknowledgment

This work was funded by the Research Management Centre, Universiti Tun Hussein Onn Malaysia (Research Fund E15501).

References

[1] Gershenfeld, N.A. and A.S. Weigend, The future of time series: Learning and understanding, in Pattern Formation In The Physical And Biological Sciences. 2018, CRC Press. p. 349-429.

[2] Husaini, N.A., et al. Jordan pi-sigma neural network for temperature prediction. in International Conference on Ubiquitous Computing and Multimedia Applications. 2011. Springer.

[3] Costa, M., A.L. Goldberger, and C.-K. Peng, Multiscale entropy analysis of complex physiologic time series. Physical review letters, 2002. 89(6): p. 068102.

[4] Batt, R.D., S.R. Carpenter, and A.R. Ives, Extreme events in lake ecosystem time series. Limnology and Oceanography Letters, 2017. 2(3): p. 63-69.

[5] Bao, W., J. Yue, and Y. Rao, A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS one, 2017. 12(7): p. e0180944.

[6] Zhang, G.P., Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 2003. 50: p. 159-175.

[7] Said, S.E. and D.A. Dickey, Testing for unit roots in autoregressive-moving average models of unknown order. Biometrika, 1984. 71(3): p. 599-607.

[8] Bishop, C.M., Neural networks for pattern recognition. 1995: Oxford university press.

[9] Kolarik, T. and G. Rudorfer. Time series forecasting using neural networks. in ACM Sigapl Apl Quote Quad. 1994. ACM.

[10] Brath, A., A. Montanari, and E. Toth, Neural networks and non-parametric methods for improving real-time flood forecasting through conceptual hydrological models. Hydrology and Earth System Sciences Discussions, 2002. 6(4): p. 627-639.

[11] Shrestha, R.R., S. Theobald, and F. Nestmann, Simulation of flood flow in a river system using artificial neural networks. Hydrology and Earth System Sciences Discussions, 2005. 9(4): p. 313-321.

[12] Ali, Z., et al., Forecasting drought using multilayer perceptron artificial neural network model. Advances in Meteorology, 2017. 2017.

[13] Ryu, S., J. Noh, and H. Kim, Deep neural network based demand side short term load forecasting. Energies, 2017. 10(1): p. 3.

[14] Hewamalage, H., C. Bergmeir, and K. Bandara, Recurrent neural networks for time series forecasting: Current status and future directions. arXiv preprint arXiv:1909.00590, 2019.

[15] Ghazali, R., et al., The application of ridge polynomial neural network to multi-step ahead financial time series prediction. Neural Computing & Applications, 2008. 17: p. 311-323.

[16] Shah, H., et al., A quick gbest guided artificial bee colony algorithm for stock market prices prediction. Symmetry, 2018. 10(7): p. 292.

[17] Shin, Y. and J. Ghosh. The pi-sigma network: An efficient higher-order neural network for pattern classification and function approximation. in IJCNN-91-Seattle International Joint Conference on Neural Networks. 1991. IEEE.

[18] Giles, C.L. and T. Maxwell, Learning, invariance, and generalization in high-order neural networks. Applied optics, 1987. 26(23): p. 4972-4978.

[19] Garro, B.A., H. Sossa, and R.A. V´azquez. Design of artificial neural networks using differential evolution algorithm. in Proceedings of the 17th international conference on Neural information processing: models and applications. 2010. Springer-Verlag.

[20] Leung, H. and S. Haykin, The complex backpropagation algorithm. IEEE Transactions on Signal Processing, 1991. 39(9): p. 2101-2104.

[21] Husaini, N.A., R. Ghazali, and I.T.R. Yanto. Enhancing modified cuckoo search algorithm by using MCMC random walk. in 2016 2nd International Conference on Science in Information Technology (ICSITech). 2016. IEEE.

[22] Walton, S., et al., Modified cuckoo search: A new gradient free optimisation algorithm. Chaos, Solitons & Fractals, 2011. 44(9): p. 710-718.

[23] Hastings, W.K., Monte Carlo sampling methods using Markov chains and their applications. Biometrika, 1970. 57(1): p. 97-109.

[24] Holland, J., Adaptation in natural and artificial systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. 1992, MIT Press: Ann Arbor, USA.

[25] Goldberg, D., Genetic algorithms in search, optimization and machine learning. 1989, Boston, USA: Addison Wesley.

[26] Vapnik, V.N., The nature of statistical learning. Theory, 1995.

[27] De Jong, K., Analysis of the behavior of a class of genetic adaptive systems. 1975, University of Michigan: Ann Arbor, MI.

[28] Fogel, L., A. Owens, and W. MJ, Artificial intelligence through simulated evolution. 1966, Chichester, UK: John Wiley.

[29] Storn, R. Differential evolution design of an IIR-filter. in IEEE International Conference on Evolutionary Computation. 1996. Nagoya.

[30] dos Santos Coelho, L. and D.L. de Andrade Bernert, An improved harmony search algorithm for synchronization of discrete-time chaotic systems. Chaos, Solitons & Fractals, 2009. 41(5): p. 2526-2532.

[31] Karaboga, D., B. Akay, and C. Ozturk. Artificial bee colony (abc) optimization algorithm for training feed-forward neural networks. in Proceedings of the 4th international conference on Modeling Decisions for Artificial Intelligence, ser. MDAI '07. 2007. Springer-Verlag.

[32] Yang, X.S. and S. Deb. Cuckoo search via Lévy flights. in Proceedings of the World Congress on Nature & Biologically Inspired Computing (NaBIC '09. 2009. India: IEEE Publications.

[33] Alauddin, M.W., W.F. Mahmudy, and A.L. Abadi, Extreme Learning Machine Weight Optimization using Particle Swarm Optimization to Identify Sugar Cane Disease. Journal of Information Technology and Computer Science, 2019. 4(2): p. 127-136.

[34] Gülcü, Ş., et al., A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem. Soft Computing, 2018. 22(5): p. 1669-1685.

[35] Aljarah, I., H. Faris, and S. Mirjalili, Optimizing connection weights in neural networks using the whale optimization algorithm. Soft Computing, 2018. 22(1): p. 1-15.

[36] Mason, K., J. Duggan, and E. Howley. Neural network topology and weight optimization through neuro differential evolution. in Proceedings of the Genetic and Evolutionary Computation Conference Companion. 2017.

[37] Arabasadi, Z., et al., Computer aided decision making for heart disease detection using hybrid neural network-Genetic algorithm. Computer Methods and Programs in Biomedicine, 2017. 141: p. 19-26.

[38] Salimans, T. and D.P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. in Advances in neural information processing systems. 2016.

[39] Lahmiri, S., A variational mode decompoisition approach for analysis and forecasting of economic and financial time series. Expert Systems with Applications, 2016. 55: p. 268-273.

[40] Samsudin, R., A. Shabri, and P. Saad, A comparison of time series forecasting using support vector machine and artificial neural network model. Journal of applied sciences, 2010. 10(11): p. 950-958.

[41] Chae, Y.T., et al., Artificial neural network model for forecasting sub-hourly electricity usage in commercial buildings. Energy and Buildings, 2016. 111: p. 184-194.

[42] Hassim, Y.M.M. and R. Ghazali, Optimizing functional link neural network learning using modified bee colony on multi-class classifications, in Advances in Computer Science and its Applications. 2014, Springer. p. 153-159.

[43] Leva, S., et al., Analysis and validation of 24 hours ahead neural network forecasting of photovoltaic output power. Mathematics and computers in simulation, 2017. 131: p. 88-100.

[44] Department, M.M. Weather Forecast. 2010 [cited 2011 February, 18]; Available from: http://www.met.gov.my.

[45] Ribeiro, H.V., et al., Characterizing time series via complexity-entropy curves. Physical Review E, 2017. 95(6): p. 062106.

[46] Rounaghi, M.M. and F.N. Zadeh, Investigation of market efficiency and financial stability between S&P 500 and London stock exchange: Monthly and yearly forecasting of time series stock returns using ARMA model. Physica A: Statistical Mechanics and its Applications, 2016. 456: p. 10-21.

[47] Akram, U., et al., An Improved Pi-Sigma Neural Network with Error Feedback for Physical Time Series Prediction. International Journal of Advanced Trends in Computer Science and Engineering, 2019. 8: p. 276-284.

[48] Al-Jumeily, D., R. Ghazali, and A. Hussain, Predicting Physical Time Series Using Dynamic Ridge Polynomial Neural Networks. PLOS ONE, 2014. 9(8): p. e105766.

[49] García, S., J. Luengo, and F. Herrera, Data preprocessing in data mining. 2015: Springer.