Recurrent Neural Network Training using ABC Algorithm for Traffic Volume Prediction

Adrian Bosire
Department of Computer Science
Kiriri Womens University of Science and Technology, Kasarani, Kenya
E-mail: bosire.adrian@gmail.com

Student paper

Keywords: deep neural network, recurrent neural network, artificial bee colony

Received: March 8, 2019

This study evaluates the use of the Artificial Bee Colony (ABC) algorithm to optimize the Recurrent Neural Network (RNN) that is used to analyze traffic volume. Related studies have shown that Deep Neural Networks are superseding the Shallow Neural Networks especially in terms of performance. Here we show that using the ABC algorithm in training the Recurrent Neural Network yields better results, compared to several other algorithms that are based on statistical or heuristic techniques that were preferred in earlier studies. The ABC algorithm is an example of swarm intelligence algorithms which are inspired by nature. Therefore, this study evaluates the performance of the RNN trained using the ABC algorithm for the purpose of forecasting. The performance metric used in this study is the Mean Squared Error (MSE) and ultimately, the outcome of the study may be generalized and extended to suit other domains.

Povzetek: Ocena uspešnosti algoritma umetne kolonije čebelje pri optimizaciji ponavljajoče se nevronske mreže.

1 Introduction

The Artificial Bee Colony (ABC) algorithm is based on the intelligent foraging behavior of the honey-bee swarm, which makes it suitable for optimization problems [14]. In his proposal of the ABC algorithm, Karaboga aimed to solve multi-dimensional and multi-modal optimization problems [12]. A function is considered to be multi-modal if it has several local optima. Furthermore, it is multi-dimensional if the local optima are distributed randomly in the search space, essentially complicating the process of finding the optimal solution. The ABC algorithm has been applied to solve many kinds of real-world problems such as leaf-constrained minimum spanning tree problem, flow shop scheduling problem, inverse analysis problem and radial distribution system network reconfiguration problem among others [21], [29].

Basturk and Karaboga [1] evaluated the ABC algorithm based on five multi-dimensional benchmark functions: sphere function, Rosenbrock Valley, Griewank function, Rastrigin function and Step function. The results obtained show that the ABC algorithm is quite robust for multi-modal problems, since it has multi-agents that work independently and in parallel. This is also echoed by the results they obtained after comparing the performance of the ABC with that of the Particle Swarm Optimization algorithm, Particle Swarm Inspired Evolutionary Algorithm and Genetic Algorithm [14].

Karaboga et. al. [17] used the ABC algorithm to train Feed-Forward Artificial Neural Networks with an aim to overcome drawbacks such as getting stuck in local minima and computational complexity. They discovered that the algorithm had good exploration and exploitation capabilities especially in searching for the optimal weight-set which is crucial in training Neural Networks. In this case, exploration refers to the ability to examine the viability of numerous unknown sections in order to discover the global optimum in the search space and exploitation refers to ability to utilize knowledge of the preceding good solutions to find improved solutions.

The data used in this study in the evaluation of the optimized neural network represents the vehicle count at specific junctions of select motorways in the whole of Britain. However, the optimized neural network can be trained for any other road network whose data is available.

The rest of this paper is organized as follows: Section 2 begins with an overview on swarm intelligence followed by Section 3 which explains the fundamental concept of the ABC algorithm. Later, Section 4 looks at the implementation of the ABC algorithm in optimizing the Recurrent Neural Network. In Section 5, we find the experiments and results. Eventually, a summary of the findings of this paper is presented in Section 6.

2 Swarm intelligence

Swarm intelligence refers to the collective intelligence exhibited by the collaborative behavior of social insect colonies or animal societies in pursuit of a defined purpose. This means that the entities that collaborate form a swarm, which is alternatively defined as a set of agents which act on their environment with an aim of solving a distributed problem [23]. These entities work together with a common goal thus increasing their chances of finding the best or optimal solution to the task at hand. In so doing, they inadvertently enhance the exploration and exploitation of their environment. Furthermore, this process serves to break down the problem into smaller and simpler tasks which are easily solved by sub-groups whose solutions are aggregated to formulate the overall solution. So, the time used to find a solution is decreased exponentially with an increase in the agents involved and also because some of these smaller tasks can be solved concurrently. The dedicated effort of such agents to a single, simplified and well-defined task also minimizes occurrence of errors as may be experienced when a single agent is tasked with the same problem. Therefore, the collective effort is useful in cases where a problem can be compartmentalized into smaller manageable tasks.

Examples of swarm intelligence algorithms include Artificial Bee Colony, Ant Colony Optimization, Particle Swarm Optimization, Immune Algorithm, Bacterial Foraging Optimization, Cat Swarm Optimization, Cuckoo Search Algorithm, Firefly Algorithm, Gravitational Search Algorithm among others [15], [23]. These algorithms are evidence of various assortments of swarms in the world and their varied level of intelligence but self-organization and labor division are key features they collectively possess.

3 Artificial bee colony algorithm

The ABC algorithm is a swarm-based algorithm presented by Karaboga [12]. This algorithm is inspired by the intelligent-search behavior of honeybees, known for their systematic collection of nectar that they process into honey. Nectar (food) is collected from flowers located in the neighboring fields (food sources) away from their hives. The bees communicate with each other by means of a waggle dance so as to share information about the quality of food sources. This information shared among the colony members includes the location and proximity of the food source to the hive, the quality of food source and quantity of food. This majorly governs the foraging range with correct accuracy thus enabling the swarm to direct its efforts to the best food source. Their mutual dependence is pegged on their distinct but partially evolving roles that adapt to the needs of the colony. The needs of the colony, decentralized decision-making and the age of the bees as well as their physical structure serve as a control for their social life. Therefore, self-organization, autonomy, distributed functioning and division of labor constitute the swarms' ability to solve distributed problems as a unit and adapt to any environment. [23], [24], [27].

The intelligence exhibited by the collective behaviour of swarms via local interactions may be characterized into four distinctive features. The firrst one is positive feedback which refers to the creation of convenient structures such as recruitment and reinforcement. Then we have negative feedback that involves counterbalancing of the positive feedback in order to stabilize the collective pattern and avoid saturation The third is fluctuations which involve the variations incurred in form of errors, random task switching among swarm individuals which stimulates creativity and discovery of new structures. Lastly, we have multiple interactions tha refer to the relationship and cooperation between the various agents in the swarm that result in the overall development [17], [18].

The honeybee forage selection model is based on three components: food sources (alternative solutions), employed foragers (active solution seekers) and unemployed foragers (passive solution seekers) made up of onlookers and scouts. In addition, two leading modes of the behavior are expressed: recruitment to a food source and abandonment of a food source. Thus, the position of a food source represents a potential solution to the optimization problem and the quantity of a food source corresponds to the calculated fitness value of the associated solution [12], [13], [14], [26].

In essence, food sources signify the profitability of the proposed solution in terms of complexity involved in attaining it. This complexity is evaluated based on proximity, ease of extraction, energy concentration which is calculated as a probability value. Employed foragers are associated with a particular food source or simply a solution they are working on, whereas, the unemployed foragers are looking for potential food sources to exploit or simply looking out for alternative solutions. Thus, the scouts find alternative food sources while the onlookers establish viable solutions from the information given to them by the employed foragers through the waggle dance.

At the beginning, the number of employed bees and the number of available food sources. Additionally, an employed bee turns into a scout when the position of a food source declines after a predetermined limit of foraging attempts, at that time exploitation ceases. Thus, the employed and onlooker bees usually perform the exploitation whereas the scouts perform the exploration of the search space. This process of foraging can be viewed as a complex problem broken down into many parts and the ultimate task is to find a viable solution since there are many ways in reaching the goal [9], [18], [23]. Let us examine figure 1 as illustrated by Karaboga [12], for a better understanding of this foraging behaviour.

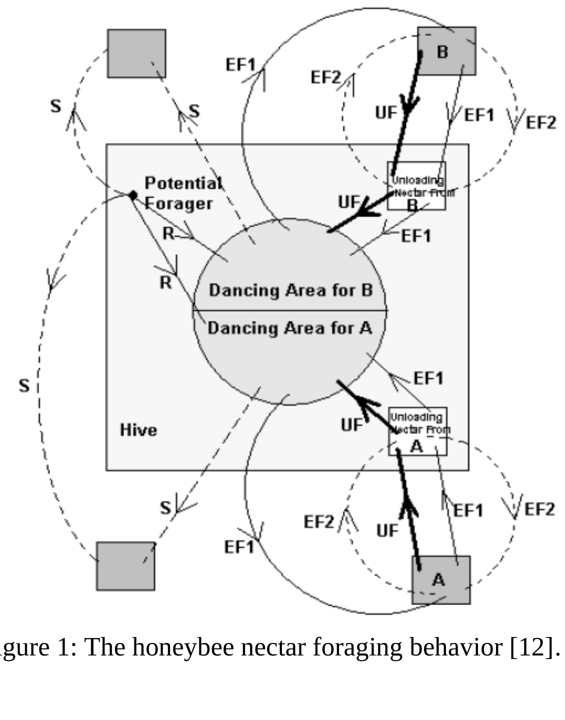

Figure 1: The honeybee nectar foraging behavior [12].

In figure 1 above, there are two discovered food sources: A and B. Any potential forager will always start as an unemployed forager and will not have any knowledge about the food sources around the nest. This limits the prospective options for such a bee to the following:

i. To become a scout and instinctively start searching around the nest for food (S).
ii. To become a recruit after watching the waggle dances for the available food sources (R).

This bee then evaluates the available food sources, memorizes a food source location and immediately starts exploiting it thus becoming an employed forager. The foraging bee takes with it a load of nectar from the source and unloads it to a food store back in the hive after which the bee takes on one of the three roles below:

i. It recruits other bees (onlookers) and returns to the same food source (EF1).
ii. It continues to forage at the same food source without recruiting other bees (EF2).
iii. It becomes an uncommitted follower after abandoning the food source (UF).

Therefore, this formulates the procedure of the ABC algorithm which is separated into five distinct phases; Initialization phase, Employed bee phase, Probabilistic selection phase, Onlooker bee phase and the Scout bee phase [12], [23]:

i. Initialization Phase
The Food Source locations are randomly initialized within the search space as calculated using equation (1) below.

$$\mathrm{x_{ij}} = \mathrm{x_j^{min}} + \mathrm{rand}(0,1)(\mathrm{x_j^{max}} - \mathrm{x_j^{min}}) \qquad (1)$$

where *i = 1, 2, ..., SN* and *SN* indicates the number of Food Sources (equal to half of the bee colony);
j = 1, 2, ..., D and *D* is the dimension of the problem;
x_{ij} represents the parameter for *i*th employed bee on *j*th dimension, meaning that they are dependent on each other;
x_j^{max} and x_j^{min} are upper and lower bounds of x_{ij}.

ii. Employed Bee Phase
Every Employee Bee is assigned to the resultant Food Source generated by equation (2) below for further exploitation.

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \qquad (2)$$

where *k* is a neighbor of *i*, $i \neq k$;
φ_{ij} is a random number in the range $[-1, \ 1]$ to control the production of neighbor solutions around x_{ij};
v_{ij} is the new solution for x_{ij}.

The value of the new Food Source is measured using a fitness value calculated by equation (3) below.

$$fit_i = \begin{cases} \frac{1}{1 + absf_i}, \ f_i \geq 0 \\ 1 + abs(f_i), \ f_i < 0 \end{cases} \qquad (3)$$

where $absf_i$ is the absolute objective function associated with each Food Source;
fit_i is the fitness value.

The two food sources x_{ij} (Original Food Source) and v_{ij} (New Food Source) are compared and the best is chosen based on a greedy selection of their fitness values.

iii. Probabilistic Selection Phase
Then, a probability value for each Food Source is calculated using equation (4) which is useful for Onlooker Bees when they evaluate the viability of a Food Source amongst the available options.

$$p_i = \frac{fit_i}{\sum_{j=1}^{N} fit_j} \qquad (4)$$

where fit_i is the fitness value of *i*-th solution;
p_i is the selection probability of ***i***-th solution.

iv. Onlooker Bee Phase
The Employed Bees advertise the viability of their Food Sources to the Onlooker Bees which select a Food Source to exploit based on the fitness and probability values associated with it i.e., the more fitness, the higher the probability. The Food Sources that are picked are further exploited using equation (2). This improves the solution and their fitness values are also calculated using equation (3). Once again, to yield an improved solution, a greedy selection process is performed on the original and new Food Sources, similar to Employed Bee Phase.

v. Scout Bee Phase
The Employed Bee for a Food source that doesn't generate better results over time becomes a Scout Bee and the Food Source is abandoned. This leads to the random generation of a new Food Source in the search space using equation (1). Subsequently, the Employed bee phase, Probabilistic selection phase, Onlooker bee phase and Scout bee phases will execute until termination criterion is satisfied. The best food source solution is obtained as output. Note that the steps of the algorithm presented in section 4 are quite elaborate than the fore mentioned summary. [12], [13], [15], [18].

4 RNN training using ABC algorithm

Artificial Neural Networks are based on the simulated network of biological neurons in which neurons are the essential computational units [22]. Hence, the underlying concept is to train a mathematical model so that it can reproduce some physical phenomena or make some predictions. The model is presented with training samples that are the actual outputs of the studied system corresponding to the actual inputs of the problem. Later, the error obtained between the actual and the predicted value serves as the metric for measuring the performance of the algorithm in terms of prediction [5].

Artificial Neural Networks can broadly be categorized into Shallow Neural Network and Deep Neural Network techniques. Shallow Neural Networks generally have only one hidden layer as opposed to Deep Neural Networks which have several levels of hidden layers. Therefore, Deep Neural Networks utilize functions whose complexity is of a higher magnitude contrary to

Shallow Neural Networks, given that all resources remain constant [3].

Shallow Neural Network (SNN) techniques contain less than two layers of nonlinear feature transformations. Examples of the SNN techniques are Conditional Random Fields (CRFs), Gaussian Mixture Models (GMMs), Support Vector Machines (SVMs), Maximum Entropy (MaxEnt) models, Logistic Regression, Kernel Regression, Multi-Layer Perceptron's (MLPs) with a single hidden layer including Extreme Learning Machines (ELMs). SNN techniques effectively solve well-constrained problems due to their limited modeling and representational power which poses a challenge when dealing with complicated real-world applications. A well-constrained problem is one for which a function is to be minimized or maximized with respect to well defined constraints [3], [6].

Deep Neural Networks (DNN) are Artificial Neural Networks composed of several interconnected hidden layers. These hidden layers have multiple hidden perceptrons between the network input layer and its network output for computational use. Dynamic environments require Deep Neural Network techniques which are useful in extracting complex structure and building internal representation. Examples of DNNs are Recurrent Neural Network (RNN), Convolutional Neural Networks (Conv.Net), Deep Boltzmann Machines (DBM), Deep Belief Networks (DBN) [30].

So, the basic concept behind Artificial Neural Networks owes to their imitation of biological neurons as shown in figure 2 which is an elementary neuron with several inputs and one output. Here, each input *x* is fed to the next layer, in our case an output layer *y*, with an appropriate weight *w*. The sum of the weighted inputs and the bias forms the input to the transfer function *f*. The bias is a threshold that represents the minimum level that a neuron needs for activating and is represented by *b*. Neurons can use any differentiable transfer function *f* to generate their output. Therefore, in multi-layer networks, the input values to the inputs of the first layer, allow the signals to propagate through the network, and read the output values where output of the i^{th} node can be described by the function in Eq. 4.1 below [25], [28].

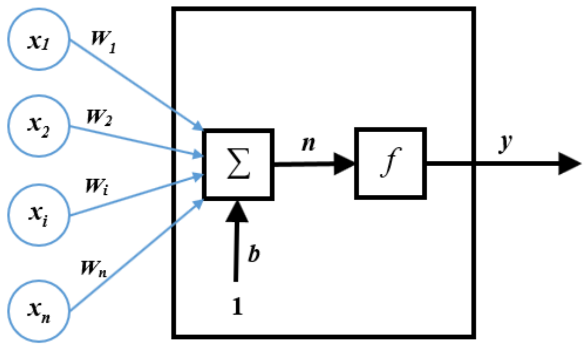

Figure 2: Representation of an Elementary Neuron.

$$y_i = f_i\left(\sum_{j=1}^{n} w_{ij}x_j + b_i\right) \quad (5)$$

where y_i is the output of the node;
x_j is the j^{th} input to the node;
w_{ij} is the connection weight between the node and input x_j;
b_i is the threshold (or bias) of the node;
f_i is the node transfer function.

Multilayer networks often use the sigmoid transfer function which generates outputs between 0 and 1 as the neuron's net input goes from negative to positive infinity. This is used for models where we have to predict the probability as an output. Hence, its suitability because the probability of real-world entities exist in the range of 0 and 1. Sigmoid output neurons are often used for pattern recognition, clustering and prediction problems.

The information from a layer to the next one is transmitted by means of the activation function, represented in equation (6). The activation function relies on the weighted sum and bias to make a calculation on whether a neuron will be activated or not, thus introducing non-linearity to the network. This non-linear transformation performed on the inputs and sent through the network enables it to learn and perform complex tasks.

$$y = f(n) = \frac{1}{1+e^{-n}} \quad (6)$$

The main goal is to minimize the cost function by optimizing the network weights. The fundamental idea of this optimization approach is to individually interpret and change the weight values. Also, note that dynamic environments present a relatively higher network complexity which suggests the need for Deep Neural Networks. Therefore, the data presented to the network has to be split into three sets; training set, validation set and the testing set. This facilitates the training, verification and evaluation of the networks' performance. Furthermore, the complexity of the challenge is represented by the Mean Squared Error (MSE) in equation (7). The MSE is obtained while comparing the target input against the predicted output could determine the number of hidden layers. The optimization of the network is achieved by minimizing the MSE which is essentially a network error function. Henceforth, the training algorithm is used to find the optimal weights that are used for initializing the Neural Network. In this case, the ABC algorithm is used to find the precise weights that enable the network connections to make accurate decisions. The algorithm uses a cost function as a measure for our progress in determining the right weights [19], [25].

$$E\big(w(t)\big) = \frac{1}{n}\sum_{k=1}^{n}(d_k - O_k)^2 \quad (7)$$

where, $E\big(w(t)\big)$ is the error at the t^{th} iteration;
$\big(w(t)\big)$, the weights in the connections at the t^{th} iteration;
d_k and O_k represent the desired and the actual values of k^{th} output node;
k is the number of output nodes;
n is the number of inputs.

A Recurrent Neural Network (RNN) is an extension of the conventional feed-forward neural network described above. The major difference is that RNNs have cyclic connections which make them reliable for modeling time-series data in dynamic environments. This means that at any given the output is related to the present input and the input at previous timestamps. Therefore, we build on the concept above of the elementary neuron in relation to the RNN. Here, we have the input sequence denoted by $x = (x_1, x_2, ..., x_t)$, the hidden layer denoted by $h = (h_1, h_2, ..., h_t)$ and the output vector sequence denoted by $y = (y_1, y_2, ..., y_t)$. Usually the RNN calculates the hidden vector sequence h using equation (8) and the output vector sequence y using equation (9) with t = 1 to T [20];

$$h_t = f_t\ (w_{xh}x_t + w_{hh}h_{t-1} + b_h) \tag{8}$$

$$y_t = f_t\ (w_{hy}h_t + b_y) \tag{9}$$

where function f_t is the activation function;
w is a weight matrix;
b is the bias term.

However, the Long Short-Term Memory (LSTM) architecture is preferable because it resolves the underlying vanishing and exploding gradient problems of the traditional RNN. The LSTM – RNN uses three gates that form a cell which consequently solves the problems mentioned above thus making the network robust. Thus, the LSTM cell replaces the recurrent hidden cell in Eq. 4.4 above. The equations to compute the values for the three gates are described below [11], [20].

$$i_t = f_t\ (w_{xi}x_t + w_{hi}h_{t-1} + w_{ci}c_{t-1} + b_i) \tag{10}$$

$$g_t = f_t\ (w_{xg}x_t + w_{hg}h_{t-1} + w_{cg}c_{t-1} + b_g) \tag{11}$$

$$c_t = f_t c_{t-1} + i_c \tan h\,(w_{xc}x_t + w_{hc}h_{t-1} + b_c) \tag{12}$$

$$O_t = f_t\ (w_{xo}x_t + w_{ho}h_{t-1} + w_{co}c_{t-1} + b_o) \tag{13}$$

$$h_t = O_t \tan h\,(c_t) \tag{14}$$

Where, f_t is the logistic sigmoid function;
i, g, o and c are respectively the input gate, forget gate, output gate and cell state;
w_{ci}, w_{cg} and w_{co} are denoted weight matrices for peephole connections.

In LSTM – RNN, the input gate ***i***, the forget gate ***g***, and the output gate ***o*** control the information flow. The input gate decides the ratio of input which has an effect when calculating the cell state, *c*. The forget gate calculates the ratio of the previous memory h_{t-1} using equation (11) and decides whether to pass it onwards or not. The result obtained is used for determining the cell state in equation (12). The output gate which is based on equation (13) determines whether pass out the output of the memory cell or not. This process as represented by the ratios from the three gates is denoted by equation (14) and also depicted diagrammatically in the figure 3 [20].

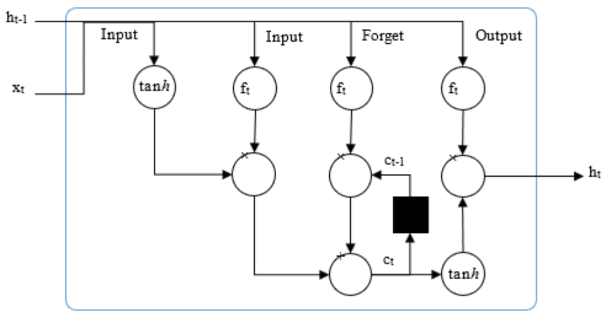

Figure 3: Long-Short Term memory Cell.

Therefore, the algorithm below outlines the optimization process for the deep neural network using the ABC algorithm [10], [12], [19], [25].

```
1.  Set Cycle=0.
2.  Load training samples from dataset.
3.  Initialize a population of scout bee with random
    solution $x_i$, $i = 1,2, ..., SN$ using equation (1).
4.  Evaluate fitness ($fit_i$) of the population using equation
    (3)
a.  Initialize weight and bias for the Recurrent Neural
    Network
5.  Set Cycle=1: while Maximum cycle not reached,
    repeat step 6 – step 12
6.  FOR each employed bee {
        Produce new solution $v_i$ by using equation (2)
        Calculate the value $fit_i$ on the new population
        Apply greedy selection process between $x_{ij}$ and
    $v_{ij}$}
7.  Calculate the probability values $p_i$ for the solutions ($x_i$)
    using equation (4)
8.  FOR each onlooker bee {
        Select a solution $x_i$ depending on $p_i$
        Produce new solution $v_i$
        Calculate the value $fit_i$
        Apply greedy selection process}
9.  If there is an abandoned solution for the scout then
    replace it with a new solution which will be randomly
    produced by equation (1)
10. Memorize the best solution so far
11. Update new weight and bias for the Recurrent Neural
    Network
12. Increment Cycle + 1 until Cycle=MCN
```

where x_i represents a solution;
fit_i is the fitness value of x_i;
v_i indicates a neighbor solution of x_i;
p_i is the probability value of x_i;
MCN is the maximum cycle number in the algorithm.

Remember that at the beginning, one half of the colony consists of onlooker bees and the second half constitutes the employed bees which are equal to the number of food sources (viable solutions) and any employed bee whose food source has been exhausted becomes a scout bee. Therefore, the algorithm starts by generating a randomly distributed initial population (SN food source positions), where SN denotes the size of population. Each solution xi (i = 1, 2, ..., SN) is a D-dimensional vector. D being the number of optimization parameters. After initialization, the population of the solutions is subjected to repeated cycles, C = 1, 2, ..., MCN, of the search process until a termination criterion is achieved. Each cycle of the search consists of three steps: engaging the employed bees with their food sources and evaluating their viability; sharing the food sources viability information with the onlookers which select a food source and again assess its viability; determining the scout bees and sending them out randomly to explore new food sources. An employed bee produces a modification on the solution in its memory depending on the probability and fitness tests. Thereby, generating optimal weights that serve to minimize the cost function and with each cycle the RNN is adequately trained with varying parameters using the ABC algorithm until optimal conditions are met [10], [18], [19].

5 Experiments and results

During the training phase, the Recurrent Neural Network is presented with a set of the training data from the dataset and the input weights are adjusted by using the ABC algorithm as a learning algorithm. The dataset can be acquired from the Road Traffic Statistics website for Great Britain [7]. The purpose of the weight adjustment is to enable the RNN to learn so that it would adapt to the given training data [10]. The dataset also has to be split to a suitable ratio to enable the training of the network, validation and testing of the results obtained. Thereafter, the performance of the network is evaluated based on the Mean Squared Error (MSE) obtained between the desired output and the actual output thus testing the validity of the network in terms of its prediction efficiency. Figure 4 below depicts the performance graph obtained on execution of the algorithm in MATLAB [2].

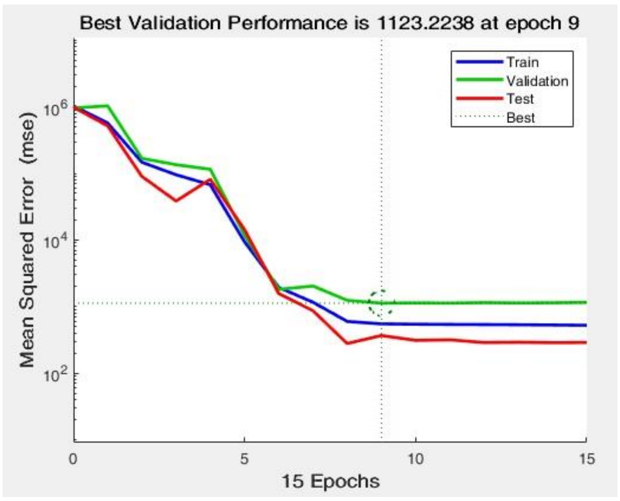

Figure 4: Performance of the ABC Optimized RNN.

The figure 4 represents the best validation performance of the network. On several runs of the algorithm the MSE obtained was 1.1232e3. This is the value obtained on epoch 9 after which the error gradually starts to increase due to overfitting but in this case, it gradually maintains a constant level. In other experiments, the MSE of the RNN before it was optimized was 3.853e3 [4]. The difference between the two MSEs basically shows that the ABC algorithm is actually efficient in terms of optimization. Generally, lower MSEs translate to high accuracy. Graphically, this is seen in the regression plots for the dataset in figure 6.

Figure 5 shows the respective regression values of the three different sets of the dataset. Splitting of the dataset helps with the early stopping of the network in order to achieve its generalization capability. The three sets of data all obtain value greater than 0.9 and the aggregate regression value is 0.93625 which borders 1. This shows a high relationship between the desired outputs and the obtained outputs, which shows a high accuracy in the

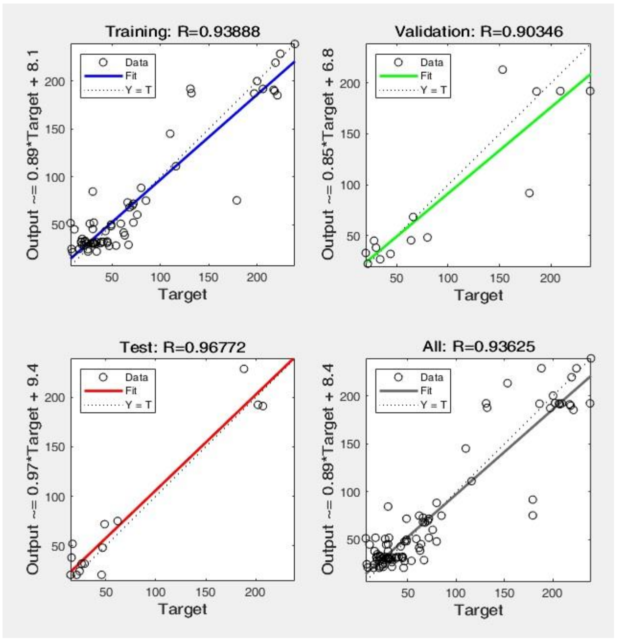

Figure 5: ABC Optimized RNN regression plot.

networks ability to forecast efficiently. Furthermore, there is a high cross-correlation between the input data and the error time-series as depicted in the graph in figure 6 below.

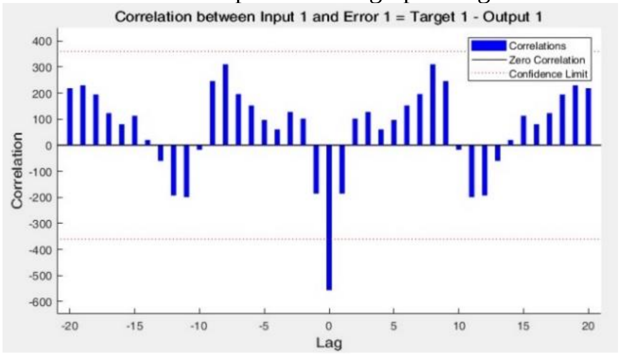

Figure 6: Correlation between the input and the output error.

The figure 6 above means that the network is able to model the predictive characteristics of the time-series lag which is the difference between the expected and the actual values. This correlation is depicted in the figure and the values fall in between the acceptable confidence limits as shown by the dotted red line. This is further exemplified in the time series plot of figure 7 below which shows the relationship between the predicted values and the actual values

The figure 7 above shows the desired output values plotted against the actual values obtained by the RNN optimized by the ABC algorithm. This time-series graph shows the level of accuracy that can be obtained during prediction with a well-trained RNN. The high efficacy of the ABC algorithm is also depicted in the graph regardless of one incorrectly predicted value. However, the other values fall between the confidence limits and as such with further training and fine-tuning of the parameters the RNN can actually produce reliable results. This means that the generalized model can actually produce accurate predictions. The values of the RNN after optimization

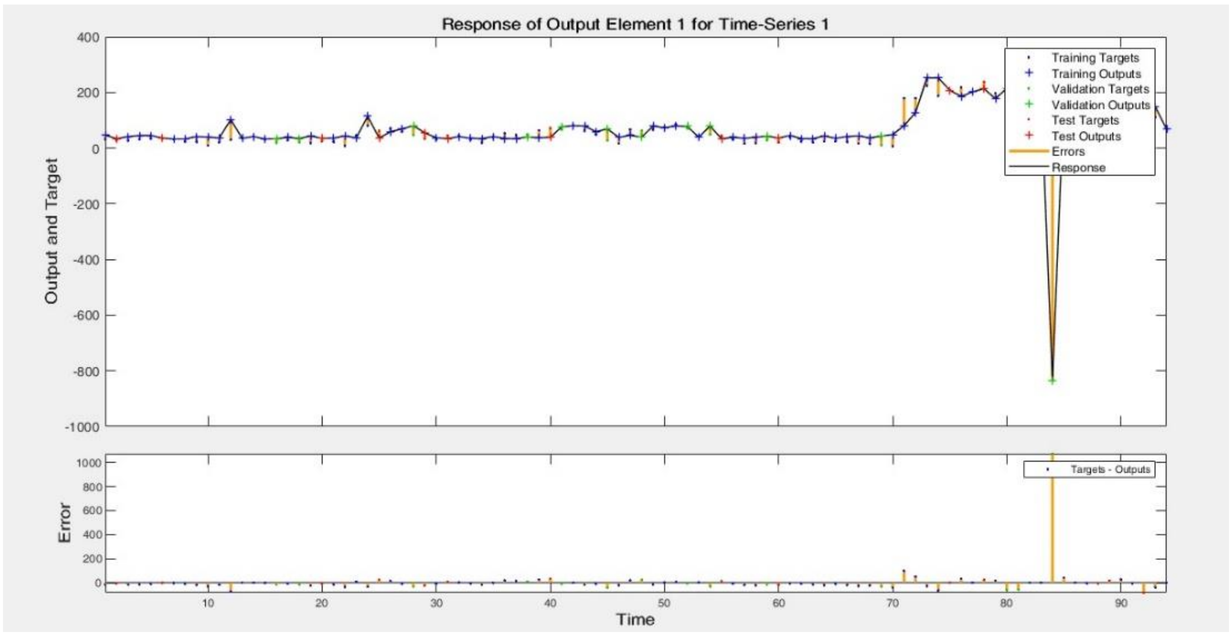

Figure 7: Time-series response of the RNN output values.

Training Algorithm	Hidden Layer Size	Performance (MSE)		
		Training	Validation	Testing
Artificial Bee Colony (ABC) Algorithm	20	721.2039	998.2733	1689.7351
	40	412.1763	569.0172	673.4512
	60	359.8409	920.2444	1.0031e+01
	80	492.2976	1.1457e+01	2.4915e+01
	100	540.0012	2.4345e+01	2.7031e+02

Table 1: MSEs obtained by the ABC optimized Recurrent Neural Network.

using the ABC algorithm are illustrated in the table 1 below.

The results in table 1 above reflect the MSEs obtained during the training, validation and testing phases of the experiment. The optimal MSE for the training phase was 359.8409, the validation phase had an optimal MSE of 569.0172 and the optimal MSE at the Test phase was 673.4512. These values show that the error rate reduced gradually with an increase in the number of hidden layers. Other experiments have been performed using other algorithms for optimization of the deep neural networks. These algorithms include the Levenberg-Marquardt Backpropagation algorithm which had an optimal MSE of 360.2578 at the training phase, the Scaled Conjugate Gradient Backpropagation algorithm which had a least MSE of 480.9656 at the validation phase and the Resilient Backpropagation algorithm which had 467.9015 as the MSE at the testing phase [4]. The ABC trained RNN has peak performance when the hidden layer size is between 40 and 80 given an input vector size of 500. In comparison to the fore-mentioned training algorithms, the ABC algorithm surpasses the other training algorithms in similar conditions.

6 Conclusion

It is evident from the results that the ABC algorithm out-performs the backpropagation algorithms. However, the parameter settings for the algorithm need to be refined for the model to be generalized. Moreover, different architectures of other deep neural networks can be implemented especially in distributed computing environments so as to sustain a greater number of the hidden layers or even produce a sustainable hybrid thereof. Furthermore, deep neural networks need to be optimized so as to enhance the practicability of a model that yields reliable forecasting in dynamic environments.

7 Acknowledgement

This work was supported by Kiriri Womens University of Science and Technology.

References

[1] Basturk, B., & Karaboga, D. (2006). An Artificial Bee Colony Algorithm (ABC) for Numeric Function Optimization. *IEEE Swarm Intelligence Symposium.* Indianapolis, Indiana, USA.

[2] Beale, M. H., Hagan, M. T., & Demuth, H. B. (2018, February). MATLAB R2018a. *Neural Network Toolbox Version 11.1*. MathWorks Inc. Retrieved from: https://www.mathworks.com/products/deep-learning.html

[3] Bianchini, M., & Scarselli, F. (2014, August). On the complexity of Neural Network Classifiers: A Comparison Between Shallow and Deep Architectures. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(8), 1553-1565.

https://doi.org/10.1109/TNNLS.2013.2293637

[4] Bosire, A., Okeyo, G., & Cheruiyot, W. (2018, October). Performance of Deep Neural Networks in the Analysis of Vehicle Traffic Volume. *International Journal of Research and Scientific Innovation, 5*(10), 57-66.

[5] De Luca, G., & Gallo, M. (2017). Artificial Neural Networks for forecasting user flows in transportation networks: literature review, limits, potentialities and open challenges. *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems* (pp. 919-923). Naples, Italy: IEEE. https://doi.org/10.1109/MTITS.2017.8005644

[6] Deng, L., & Yu, D. (2013). Deep Learning: Methods and Applications. *Foundations and Trends in Signal Processing, 7*, 197-387. https://doi.org/10.1561/2000000039

[7] Department for Transport. (2018, June). *Traffic counts*. Retrieved June 2018, from Traffic counts: https://www.dft.gov.uk/traffic-counts/about.php

[8] Garro, B. A., & Vázquez, R. A. (2015). Designing Artificial Neural Networks Using Particle Swarm Optimization Algorithms. *Computational Intelligence and Neuroscience*. https://doi.org/10.1155/2015/369298

[9] Haris, P., Gopinathan, E., & Ali, C. (2012, July). Artificial Bee Colony and Tabu Search Enhanced TTCM Assisted MMSE Multi-User Detectors for Rank Deficient SDMA-OFDM System. *Wireless Personal Communications, 65*(2), 425-442. https://doi.org/10.1007/s11277-011-0264-0

[10] Hassim, Y. M., & Ghazali, R. (2012). Training a Functional Link Neural Network Using an Artificial Bee Colony for Solving a Classification Problems. *Journal of Computing, 4*(9), 110-115.

[11] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

[12] Karaboga, D. (2005). *An Idea Based on Honey Bee Swarm for Numerical Optimization.* Technical Report, Erciyes University, Computer Engineering Department, Turkey.

[13] Karaboga, D., & Akay, B. (2009, August). A comparative study of Artificial Bee Colony algorithm. *Applied Mathematics and Computation, 214*(1), 108-132. https://doi.org/10.1016/j.amc.2009.03.090

[14] Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm. https://doi.org/10.1007/s10898-007-9149-x

[15] Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing*, 687–697. https://doi.org/10.1016/j.asoc.2007.05.007

[16] Karaboga, D., & Ozturk, C. (2009). Neural Networks training by Artificial Bee Colony algorithm on pattern classification. *Neural Network World, 19*(3), 279–292.

[17] Karaboga, D., Basturk, B., & Ozturk, C. (2007). Artificial Bee Colony (ABC) Optimization Algorithm for Training Feed-Forward Neural Networks. In *Modeling Decisions for Artificial Intelligence* (pp. 318-329). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-73729-2_30

[18] Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2012). A comprehensive survey: Artificial Bee Colony (ABC) algorithm and applications. *Artif Intell Rev, 42*, 21–57. https://doi.org/10.1007/s10462-012-9328-0

[19] Kayabasi, A. (2018). An Application of ANN Trained by ABC Algorithm for Classification of Wheat Grains. *International Journal of Intelligent Systems and Applications in Engineering, 6*(1), 85-91. https://doi.org/10.18201/ijisae.2018637936

[20] Kim, J., Kim, J., Thu, H. L., & Kim, H. (2016). Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection. 1-5. https://doi.org/10.1109/PlatCon.2016.7456805

[21] Koc, E., Ersoy, N., Andac, A., Camlidere, Z. S., Cereci, I., & Kilic, H. (2012). An empirical study about search-based refactoring using alternative multiple and population-based search techniques. In E. Gelenbe, R. Lent, & G. Sakellari (Ed.), *Computer and information sciences II* (pp. 59-66). Springer, London. https://doi.org/10.1007/978-1-4471-2155-8_7

[22] Kriegeskorte, N. (2015). Deep Neural Networks: A New Framework for Modeling Biological Vision and Brain Information Processing. *Annual Review of Vision Science, 1*, 417–446. https://doi.org/10.1146/annurev-vision-082114-035447

[23] Kumar, A., Kumar, D., & Jarial, S. K. (2017). A Review on Artificial Bee Colony Algorithms and Their Applications to Data Clustering. *Cybernetics and Information Technologies, 17*(3). https://doi.org/10.1515/cait-2017-0027

[24] Omkar, S., & Senthilnath, J. (2009). Artificial Bee Colony for Classification of Acoustic Emission Signal Source. *International Journal of Aerospace Innovations, 1*(3), 129-143. https://doi.org/10.1260/175722509789685865

[25] Ozturk, C., & Karaboga, D. (2011). Hybrid Artificial Bee Colony Algorithm for Neural Network Training. *2011 IEEE Congress of Evolutionary Computation (CEC)*, 84-88. https://doi.org/10.1109/CEC.2011.5949602

[26] Seyyed, R. K., Maleki, I., Hojjatkhah, S., & Bagherinia, A. (2013, August). Evaluation of the Efficiency of Artificial Bee Colony and Firefly Algorithm in Solving the Continuous Optimization Problem. *International Journal on Computational Sciences & Applications, 3*(4).

[27] Shukran, M. A., Chung, Y. Y., Yeh, W.-C., Wahid, N., & Zaidi, A. M. (2011, August). Artificial bee colony based data mining algorithms for classification tasks. *Modern Applied Science, 5*(4), 217–231. https://doi.org/10.5539/mas.v5n4p217

[28] Vazquez, R. A., & Garro, B. A. (2015). Training Spiking Neural Models Using Artificial Bee Colony. *Computational Intelligence and Neuroscience*. https://doi.org/10.1155/2015/947098

[29] Xiangyu, K., Liu, S., & Wang, Z. (2013). An Improved Artificial Bee Colony Algorithm and Its Application. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, *6*(6), 259-274. https://doi.org/10.14257/ijsip.2013.6.6.24

[30] Yann, L., & Ranzato, M. (2013). Deep learning tutorial. Tutorials in International Conference on Machine Learning (ICML'13).

List of Abbreviations

ABC	Artificial Bee Colony
Conv.Net	Convolutional Neural Network
CRF	Conditional Random Field
DBM	Deep Boltzmann Machine
DBN	Deep Belief Network
DNN	Deep Neural Network
EF	Employed Forager
ELM	Extreme Learning Machine
GMM	Gaussian Mixture Model
LSTM	Long-Short Term Memory
MaxEnt	Maximum Entropy
MCN	Maximum Cycle Number
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NN	Neural Network
R	Recruited Bee
RNN	Recurrent Neural Network
S	Scout Bee
SNN	Shallow Neural Network
SVM	Support Vector Machine
UF	Unemployed Forager