International Journal of Mathematics and Computational Science
Vol. 4, No. 4, 2018, pp. 128-138
http://www.aiscience.org/journal/ijmcs
ISSN: 2381-7011 (Print); ISSN: 2381-702X (Online)

A Hybrid Technique Between BOSOM and LSTM for Data Analysis

Kernan Mzelikahle[1, *], John Trimble[2], Dumisani John Hlatywayo[3]

[1]Computer Science Department, National University of Science and Technology, Bulawayo, Zimbabwe
[2]Industrial Engineering Department, Tshwane University of Technology, Tshwane, South Africa
[3]Applied Physics Department, National University of Science and Technology, Bulawayo, Zimbabwe

Abstract

In the field of machine learning, many applications require techniques that are able to respond in real time. However, such abilities are usually achieved by trading off the accuracy rates in favour of good time complex performance. The Long Short Term Memory (LSTM) is a typical example where the technique provides good time complex performance, yet it has fairly low accuracy rates. The Bat Optimised Self Organising Map (BOSOM), on the other hand, is a relatively slow processing technique for unsupervised classification problems; however, it has fairly high accuracy rates. In this paper, a hybrid technique between the standard LSTM network and BOSOM, for use in time and space unconstrained applications, is proposed. The objective of the paper is to demonstrate that higher accuracy rates and higher recall rates are better achieved by striking a balance between turn-over time and exhaustive learning. To achieve generalisation of performance of the BOSOM-LSTM model, datasets of varying sizes from multiple domains are used. The results in this paper show that the BOSOM-LSTM hybrid model has considerably better accuracy and recall performance compared to other considered techniques. This performance establishes good ground that the BOSOM-LSTM hybrid technique is a competitive technique that can be used in application areas that require high accuracy and high reliability.

Keywords

Long Short Term Memory, Bat Optimised Self Organising Map, Artificial Neural Networks, Unsupervised Learning

Received: September 26, 2018 / Accepted: October 14, 2018 / Published online: December 21, 2018

@ 2018 The Authors. Published by American Institute of Science. This Open Access article is under the CC BY license.
http://creativecommons.org/licenses/by/4.0/

1. Introduction

Machine learning is a subject discipline that focuses on techniques for artificially acquiring knowledge, and using such knowledge in solving problems. There are many methods that are used in learning, and they may be categorised into two distinct branches; supervised learning and unsupervised learning [3, 20]. Supervised learning uses a training set that has a corresponding desired set of outputs. If there is a discrepancy between the output of the learning technique and the desired output, then relevant weight structures are adjusted in order to correct the discrepancy. On the other hand, in unsupervised learning, there is no known desired state, thus the learning technique must adjust its structures (weights) according to the underlying patterns in the training set. Normally, in Artificial Neural Networks (ANNs), the fundamental building unit is a neuron [4]. A neuron may be drafted into a neural network through the use of structural connections. These connections between neurons determine the nature of the neural network. For example, if neurons are connected in successive layers such that neurons in preceding layers strictly feed their outputs to succeeding layers, then such a neural network is called a Feed Forward Artificial Neural Network (FFANN) [8, 18]. Neural networks with feedback loops among neurons are called Recurrent Neural Networks (RNN) [3, 20]. Many

* Corresponding author
E-mail address: kernan.mzelikahle@nust.ac.zw (K. Mzelikahle)

variants of both types of networks have been proposed over the years. In this paper, a hybridisation process of an unsupervised learning network and a supervised learning network is proposed.

The objective of hybridising such networks, despite their varying methods of achieving learning, is to seek greater accuracy rates. The Long Short Term Memory (LSTM) is a recurrent neural network that uses a variant of supervised learning proposed by Gers et al. [5]. In their work, Gers et al. [5] were building on foundational precepts that were developed by Hochreiter and Schmidhuber [9]. The LSTM has been shown to be a solution to the vanishing gradient problem during learning for the Backpropagation Through Time (BPTT) algorithm [7]. The vanishing gradient problem occurs when the propagation time window grows too long. In other words, the problem becomes apparent and clearly evident when a task contains long range contextual dependencies [17]. The range is determined by the number of time steps that are spanned by a sequence learning algorithm in order for it to label a task. Gers et al. [5] proposed the LSTM with a forget gate, however, they proceeded to extend the LSTM architecture in order to achieve more objectives [6, 7, 17]. In this paper, the basic and un-extended LSTM is adopted for use in the hybridisation process. However, the peephole LSTM is used in the subsequent comparative analysis in order to establish performance evaluation of the BOSOM-LSTM hybrid technique.

The Bat Optimised Self Organising Map (BOSOM) was proposed by Mzelikahle et al. [12] as a method to optimise a Self Organising Map (SOM). In their study, the initial weight vector was updated using the Bat Algorithm [12]. This implies that the SOM section of BOSOM begins its unsupervised learning process with non-random weights that have been pushed to a near global optimum point. Similarly, in this paper, the output of BOSOM is fed into an adopted LSTM for training. Effectively, the BOSOM network is used to first cluster the training set into inherent clusters before feeding the result into the LSTM. In practice, this requires the use of a temporary storage (buffer mechanism) during execution.

2. Long Short Term Memory

The LSTM uses a variant of Backpropagation Through Time (BPTT) for adjusting its learning structures [5]. The technique used in the LSTM is for truncating the gradient in order to bridge minimal time lags. This technique allows the LSTM to span long discrete time steps [7]. Through the use of input gates, output gates and buffer gates, the LSTM is able to learn when to open and when to close for constant error flow. In essence, the LSTM solves the vanishing gradient problem by introducing truncation of error into the gradient descent method [7]. Through the use of this method, the LSTM network, therefore, is made up of memory blocks that employ recurrent feedback loops. The memory blocks may be organised into layers, with one or more hidden layers. The structure of the LSTM is such that it has three gates with special functions;

(1) The Input gate: This gate controls the activation of the weighted sum input signals. This may be achieved by learning the underlying trends in the training set.

(2) The Output gate: This gate controls the flow of activations on output points of the LSTM memory block. The activation of this gate is learned from the desired output set of the training set.

(3) The Forget gate: This gate controls the duration for which a memory cell is to recall the stored state. When activated, the forget gate forces the memory cell to reset, thereby forget its state.

Through the use of these gates, the LSTM is able to process and manipulate continuous input streams, even if they are not segmented into sub-sequences [17]. Figure 1 shows the basic structure of a memory block that has a single memory cell.

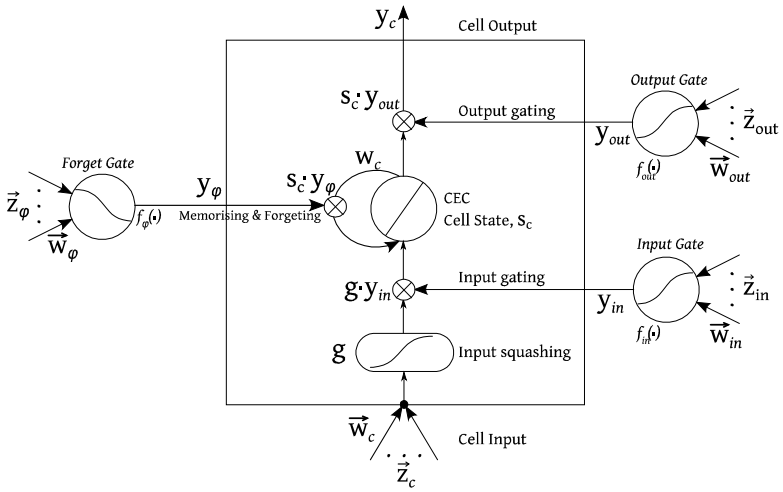

Figure 1. LSTM memory block with one cell [5].

The LSTM network receives, through input nodes, the input vector $\vec{x} = (x_1, \ldots, x_T)$ and it can perform a non-linear mapping of input to a sequence output y, where $\vec{y} = (y_1, \ldots, y_T)$. This is achieved by calculating unit activations in the LSTM network as;

$$y_t = \phi(W_{ym} m_t + b_y) \quad (1)$$

where W_{ym} is a vector matrix of weights from some node m to an output node y, with a bias vector b_y, and ϕ is the output activation function [6]. This calculation can be conducted recursively for all active nodes. By applying this to a memory block in a network, it is possible to maintain a network state across multiple batches for training. This implies that both discrete and continuous samples can be processed across time lags.

3. Bat Optimised SOM

BOSOM is a hybrid technique that combines the Bat Algorithm and a Self Organising Map [12]. In BOSOM, the connections to neurons in a SOM are initialised to random weights, and the bat algorithm is used to adjust these weights in order to determine a near global optimum set of weights. When this set of weights has been determined, the SOM is set to begin the unsupervised learning process at this near global optimum point. The Self Organising Map uses neurons in one layer, and they are allowed to compete among themselves. This process has been shown to be effective for discovering underlying structures in the data. The BOSOM model uses an objective function that determines an acceptable solution in a solution space [12]. The objective function makes use of a learning rate for the purpose of controlling possible oscillations during learning; therefore, there are few occurrences of twisted map incidents. It is observed that for BOSOM, the lower the learning rate the lower the tendency to oscillate. However, if the learning rate is too low, BOSOM may take unreasonably long periods of time to complete the process. BOSOM operates optimally if the output layer has been configured into a lattice structure. In order to adjust the weights during learning, BOSOM uses a weight changing rule calculated as;

$$\Delta w_{ij} = \gamma x_i (y_j - y_j^2 w_{ij}) \quad (2)$$

where Δw_{ij} is the change in weight of a connection that links the input node i to a neuron j on the BOSOM surface, y_j is the output from neuron j, and w_{ij} is the current weight of a connection between i and j. Figure 2 shows the BOSOM structure that uses a lattice grid for competing neurons on the surface.

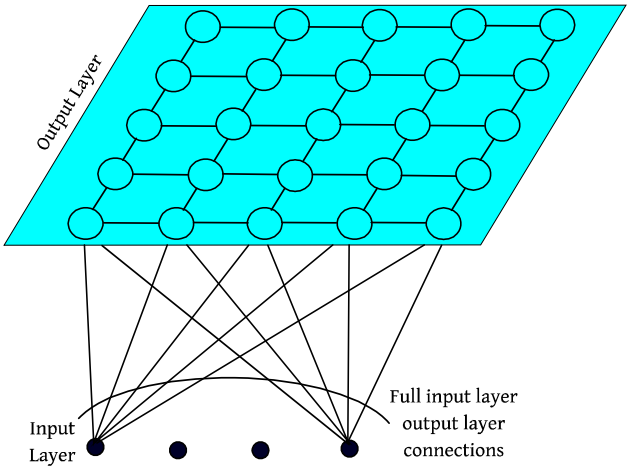

Figure 2. BOSOM Lattice Surface [12].

When adjusting connection weights in BOSOM, during learning, there is need to determine the neurons that shall have their weights adjusted. In this paper, soft competition is used. That is, to determine the neurons for which to adjust connection weights, and to calculate topological weight adjustments, a neighbourhood calculation function is applied as follows;

$$\lambda(j, k) = \exp\left(-\frac{||r_j - r_k||^2}{2\sigma_\lambda^2}\right) \quad (3)$$

where $\lambda(\cdot)$ is the neighbourhood function, k is the winning neuron and j is the neuron for which a determination is required, and r_j is the radius of neuron j from k, while σ_λ is the allowable standard deviation on the neighbourhood function $\lambda(\cdot)$.

3.1. Experimentation Using BOSOM

Before the hybridisation process between the LSTM and BOSOM can be addressed, it is imperative to assess the performance of BOSOM against other competing techniques. To achieve this objective, BOSOM was compared against the standard SOM [10] and the K-Means clustering algorithm [20]. Three well known datasets were used for this experiment and these are the IRIS dataset, the SERVO dataset and the LENSES dataset. All the three datasets were obtained from the University of California, Irvine (UCI) Machine Learning Repository. Table 1 summarises the properties of the datasets used.

Table 1. Dataset summary on BOSOM experimentation.

Property	Iris	Servo	Lenses
Input node	4	4	4
Data Size	150	167	24
Training Size	120 (80%)	134 (80%)	9 (37.5%)
Testing Size	30 (20%)	33 (20%)	15 (62.5%)
Classes/Range	3	0.13 – 7.10	3

3.2. Results on BOSOM Experimentation

Both BOSOM and SOM were configured in this experimentation process using a 5×5 lattice structure. On the other hand, the K-Means clustering algorithm was configured using a default 2 dimensional value set. To comparatively assess performance of these techniques, three metrics were used. These are;

(1) The Clustering Accuracy (CA): This is a property metric that indicates how well the classes are separated on a cluster map.

(2) The Quantisation Error (QE): This is a property metric that measures how accurately the network or technique responds to input vectors. Its range is [0,1] and a network seeks to minimise the error.

(3) The Convergence Error (CE): This is a property metric that measures the rate of convergence to a solution by calculating the error associated with oscillation. This error is in range [0,1] and the network seeks to minimise this error.

Table 2. Performance summary of BOSOM on metrics.

	BOSOM	SOM	K-Means
Clustering Accuracy (%)	94.7418	92.8724	93.1052
Quantisation Error	0.0563	0.1129	0.1375
Convergence Error	0.1056	0.1875	0.1692
Runs	30	30	30
Epochs	2000	2000	2000

Table 2 shows the average performance of techniques measured over 30 repetitions (runs), with each run having 2000 epochs. As shown in Table 2, all the competing techniques perform significantly well. These results appear to be consistent with results reported by other researchers [16, 19, 21]. On a comparative assessment, BOSOM shows better performance compared to both the standard SOM and K-Means algorithm. For clustering accuracy, the K-Means algorithm shows better performance compared to SOM, however, BOSOM appears to have the best performance. SOM performs better than K-Means on the quantisation error, implying that SOM has very good response to input vectors. This result corroborates results reported by Spanakis and Weiss [19]. Based on these results, it may thus be put forth that the BOSOM technique is a highly competitive technique and may therefore be adopted for hybridisation with the LSTM.

4. BOSOM-LSTM Hybridisation

BOSOM is connected to the LSTM network through the use of a buffer mechanism. The LSTM memory blocks are ordered in a $p \times q$ matrix of Constant Error Carousels (CECs). Given that the LSTM has to be fed BOSOM's output, a parallel connection is used. In this parallel connection, a buffering mechanism is used to synchronise the read-write operations. In the buffering mechanism, BOSOM first directs output to a temporary storage and sends an activation signal to the LSTM network for it to start accessing the buffer. In the buffer, BOSOM writes a set of connection weight vectors corresponding to both the connection weights and the topological weights. This process preserves the topological structure of the BOSOM surface. When data is read through the buffering mechanism into the LSTM, the weights to the connections are initialised from the topology connection weights, depending on the clusters and strengths between nodes on the BOSOM surface. During learning, the weights are adjusted for each input vector, on connections between BOSOM and the LSTM. The LSTM uses an input gate and an output gate to regulate when data is to be written into, and when data is to be read from the LSTM memory block, respectively. For this reason, there is need for a method to synchronise operations, such that no data may be overwritten in the LSTM before being read. On a $p \times q$ LSTM, p properties of data can be fed into the network and q time steps can be accounted for before activating the forget action. The reading and writing actions are thus synchronised as shown in Table 3. In Table 3, the LSTM is only activated for performing both read and write operations when the input signal corresponds to the output signal from the buffering mechanism. This means that the activation values from firing neurons are able to flow through the network, with connections undergoing weight adjustments during learning, without disruption by untimely overwriting.

Table 3. Activation of LSTM for Read and Write operations.

Input	Output	Read	Write
0	0	1	1
0	1	0	0
1	0	0	0
1	1	1	1

Figure 3 shows the structure of connections between BOSOM and the LSTM by making use of a buffer. In this figure, one CEC column is shown, however, more memory blocks can be added in parallel to form an array of memory cells.

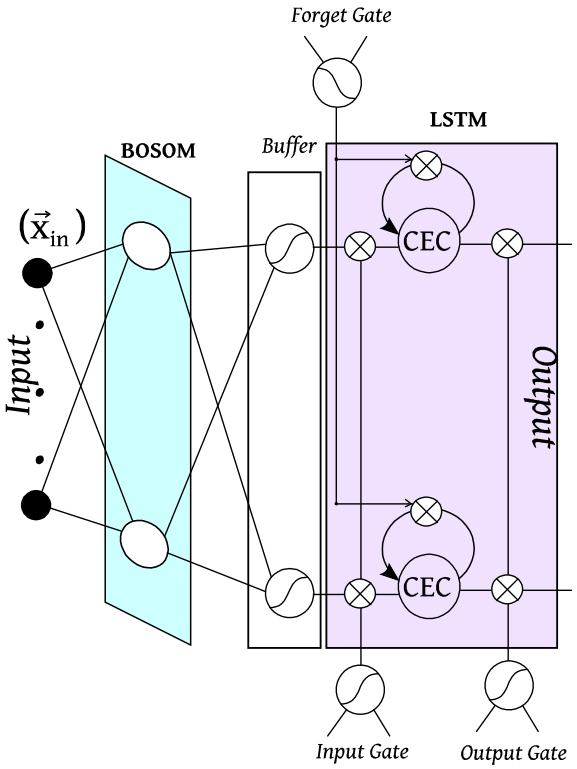

Figure 3. BOSOM-LSTM connection.

The learning process within the LSTM is adopted from the standard LSTM proposed by Gers et al. [5] without modifications. On the other hand, adjusting weights connecting BOSOM to the buffer mechanism needs to be accounted for. These weights are updated as follows;

$$\Delta w_{km}(t) = \alpha\delta_k(t)y_m(t-1), \quad where$$
$$\delta_k(t) = -\frac{\partial E(t)}{\partial z_k(t)} \tag{4}$$

In this case, $\Delta w_{km}(t)$ is the change in weight of a connection between unit m on the BOSOM surface and an input unit k in the buffer mechanism connecting to the LSTM, at a time step t. The learning rate of the LSTM, α, is maintained constant during a run, and $\delta_k(t)$ is an error function defined for BOSOM. The previous output of unit m is $y_m(t-1)$.

5. Experiments Configuration

In order to assess the utility of the BOSOM-LSTM hybrid technique, experiments were conducted in a highly controlled environment. Three well known datasets were utilised to assess the performance of the BOSOM-LSTM technique in a comparative study against other known techniques. The BOSOM surface was maintained as a 5×5 lattice and thus the LSTM was configured to a 25×40 memory block array. For all the three datasets used, the configuration of the BOSOM-LSTM structure was maintained a constant. In the BOSOM-LSTM connection, there are three (3) segments of data that are represented within the buffer mechanism, and these are:-

(1) Neighbourhood connection weights: These are weight vectors for the connections between neurons (nodes) on the BOSOM surface. Contours and clusters can be extracted through analysis of these connection weights.

(2) Bridging weights: These are connection weight vectors of links that connect the BOSOM surface to the buffer gates in the buffering mechanism. It is possible to change the BOSOM surface structure into some other organisation of nodes; however, in this experiment the BOSOM structure is maintained as a square lattice.

(3) Activation values: Each of the surface nodes (neurons) has an activation value that is obtained from the BOSOM learning process. These activation values have their highest excitation at the centre of a learned cluster, and they gradually decrease as the radius from the winning neuron increases. As influence of the next cluster is felt, the activation values increase in the direction of the new focal point. Such an activation value is a vector in nature because it has both magnitude and direction in its quantity.

Table 4. Segment properties of data from BOSOM training.

Property	Segment Size
Nodes	25
Neighbourhood connections	40
Bridging vectors	38
Activations	25
Maximum Classes	15

Table 4 presents a summary of the segments of data obtained from the BOSOM configuration employed in this paper. In the experiment, these segments of data are ordered such that they are fed in parallel into the LSTM structure (see Figure 3). The LSTM structure is configured such that it perfectly aligns with the nodes (neurons) in the buffering mechanism. For example, in this experiment, there are 25 BOSOM surface nodes, implying 25 input parameters into the buffer, and therefore 25 input nodes for the LSTM structure. However, the length of the LSTM memory blocks rather depends on the problem at hand. In general, the more columns of memory blocks, the higher the LSTM's ability to process bigger dataset sizes. Both these configuration parameters may be adjusted at the beginning of each experiment or use. In this experimentation process, three (3) datasets were used and Table 5 presents a summary of their properties. In general, the dataset splitting rule used is 80% training set, and 20% testing set. All datasets were acquired for use from the UCI, Irvine repository. The three (3) well

known datasets used in this experiment are; the Stanford Sentiment Treebank dataset [13], the Yahoo! Music User Rating dataset, and the Skytrax User Reviews dataset [14]. In these datasets, there were some few missing data-points. In this paper, the missing data points were filled using the window averaging data filling method.

Table 5. Summary of dataset properties on BOSOM-LSTM experimentation.

Property	Stanford	Yahoo!	Skytrax
Inputs	19	7	21
Size (Instances)	11855	10000000	41396
Training Size	9645	8000000	33117
Testing Size	2210	2000000	8279
Classes	5	13	9

In this experimentation procedure, the training vectors for the input gate and output gate of the LSTM were obtained from the random Gaussian normal distribution, with $\mu = 0$ and $\sigma^2 = 1$. Due to the need to synchronise the activation of the input gate and that of the output gate, the Gaussian normal distribution was observed to exhibit the best behaviour. In order to assess the performance of the BOSOM-LSTM model, a comparative study was done, where four (4) other techniques were used as control models. These four (4) competing techniques are; the FFANN, the peephole LSTM (p.LSTM) [7], the Gated Recurrent Unit (GRU) network [2], and the Independent Recurrent Neural Network (IndRNN) [11]. These competing techniques were carefully chosen because of their high performance and wide usage in industry.

In this experimentation process, the FFANN was configured to have 3 hidden layers with specification: 25 input nodes, 3 hidden layers at 25-30-25 nodes, and 15 output nodes. The peephole LSTM was configured with specifications: 25 input nodes, 40 CEC nodes, and 15 output nodes. The Gated Recurrent Unit network was configured with specifications: 25 input nodes, 40 GRU units, and 15 output nodes. The Independent Recurrent Neural Network was configured with specifications: 25 input nodes, 40 IndRNN units, and 15 output nodes.

The experiments were carried out on a high performance computer system, and the allocated resources on login are: Intel Xeon E5-2670 CPU, @ 2.5 GHz, 16 cores, 10 nodes, 128 GB memory (RAM), 1 TB storage space, 850 GB Temp in /tmp. Operating system: Red Hat Enterprise Linux Sever (HPC), v7.5. The basic tools used were clang v4.0.0 –posix (cc), R v3.5.0 and python v2.7_3.

6. Results

All the five (5) techniques were independently run across the datasets, with each technique run 30 times and results recorded. This approach, of independently running the techniques, allows for performance metrics to be measured in a controlled environment. The performance metrics were measured during and after execution of an experimental instance. In this set of experiments, five metrics were used to assess performance of the competing techniques. Further, the time complexity and space complexity are measured on all the competing techniques, for all experimental instances. The five (5) performance metrics measured are as discussed below.

The Mean Square Error (MSE). This metric is a measure of the averaged value of errors that a technique suffers compared to the desired output of a testing set. This measure calculates the average of squares of the determined errors for every epoch in a run. Since the differences of these errors are squared, then the MSE is a non-negative quantity. By this definition, the values closer to zero reflect better performance of the measured technique (or quantity), and the values farther away from zero reflect poor performance. In this paper, we calculate the MSE as;

$$\mathrm{MSE}(\mathrm{y}) = \mathrm{E}[(\delta(\mathrm{Y}) - \mathrm{y})^2] \tag{5}$$

where $\delta(\mathrm{Y})$ is an estimator of the mean of the random variable y.

The Average Convergence Error (ACE). The ACE measures the rate of convergence of a technique towards its determined solution. The ACE, in a way, measures a technique's tendency to oscillate away from settling on a solution. For a given technique; the lower the average convergence error, the better the performance. The ACE is an iterative measure, therefore it is measured during execution of an experimental instance, where each epoch is a contributor. In this paper, the average convergence error is a quantity in the range [0, 1] and is calculated as;

$$\mathrm{ACE} = \lim_{\mathrm{k}\to\mathrm{N}} \frac{|\mathrm{x}_{\mathrm{k}+1}+\varepsilon|}{|\mathrm{x}_{\mathrm{k}}-\varepsilon|} \tag{6}$$

where ε is the epoch error margin, x is the epoch on step k, and N is the total number of epochs.

The Precision Recall Graph. This is a graph that shows the relationship between positive predictive values of a technique against its sensitivity for all epochs in a run. The precision recall graph runs in range [0, 1] for both axes, and thus the area under the curve for a perfect measure is 1. By calculating the area under the graph, the performance of a technique is thus determined.

The Pearson Correlation Coefficient (PCC). The PCC is a bivariate correlation measure of a linear correlation between two variables, say X and Y. The PCC varies between $+1$ and -1, wherein values closer to $+1$ indicate a strong linear relationship, values closer to zero indicate that there is no

linear correlation between the variables, and values closer to -1 indicate existence of a negative linear correlation. In this paper, the PCC ($\rho_{X,Y}$) between two variables is calculated as;

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} \quad (7)$$

where $\text{cov}(X, Y)$ is the covariance of variables X and Y, σ_X is the standard deviation of variable X, and σ_Y is the standard deviation of variable Y.

The Cross Entropy loss (CE). The CE loss between two variable distributions, p and q drawn from the same underlying set, measures the fraction average number of data points required to identify an event drawn from the set. The CE loss runs in the range $[0, \log(N)]$, with values closer to zero reflecting less loss rate than values closer to $\log(N)$. Therefore, a technique with loss levels closer to zero performs better than a technique with loss levels closer to $\log(N)$. In this paper, the CE loss is calculated as;

$$H(p, q) = -\sum_x p(x)\log(q(x)) \quad (8)$$

where $H(p, q)$ is the cross entropy between variable distributions p and q, and x is the data point position on either distribution. Table 6 presents the performance of the competing techniques on the Mean Square Error (MSE) metric. As shown in Table 6, all the competing techniques perform considerably well; however, there are some noticeable performance differences. On average, the BOSOM-LSTM technique outperforms all the other four (4) competing techniques, at 1.7895 error rate. However, when a direct comparison is made against the peephole LSTM, it can be observed that the performance difference is quite narrow. This performance indicates that the BOSOM-LSTM hybrid technique is a very competitive technique on the MSE metric, but the peephole LSTM is not significantly inferior.

Table 6. Performance of Techniques on the MSE metric.

Dataset	Bosom-Lstm	FFANN	p.LSTM	GRU	IndRNN
Stanford	1.4371	13.5784	2.8742	3.7546	4.6217
Yahoo!	2.9681	9.6322	3.2566	5.3321	3.1436
Skytrax	0.9634	7.6131	1.5631	3.1625	2.8731
Average	1.7895	10.2746	2.5646	4.0831	3.5461
Epoch	10000	10000	10000	10000	10000
Runs	30	30	30	30	30

All the other techniques have considerably inferior performances, with the least performer being the FFANN, on the MSE metric. Table 7 presents comparative performances of the competing techniques on the Average Convergence Error (ACE) metric. As shown in Table 7, the BOSOM-LSTM technique appears to have the best performance because it has the lowest average ACE value, at 0.1453. This means that the BOSOM-LSTM has quick convergence, by epochs count, to a solution and has minimal oscillatory tendencies, in this experiment.

Table 7. Performance of techniques on the ACE metric.

Dataset	Bosom-Lstm	FFANN	p.LSTM	GRU	IndRNN
Stanford	0.1436	0.3897	0.2639	0.2978	0.3219
Yahoo!	0.0856	0.1172	0.0974	0.1477	0.2420
Skytrax	0.2068	0.1923	0.1438	0.0736	0.1351
Average	0.1453	0.2331	0.1684	0.1730	0.2330
Epoch	10000	10000	10000	10000	10000
Runs	30	30	30	30	30

The peephole LSTM has equally very good performance on the ACE metric, averaging 0.1684. The oscillatory tendencies for the peephole LSTM are considerably significant; however, compared to the other competing techniques, the peephole LSTM has sufficient competitive performance. Both the IndRNN and the FFANN have poor performances on the ACE metric. The IndRNN and FFANN have their performances averaging 0.2330 and 0.2331 respectively. This shows that both the IndRNN and FFANN have high oscillatory tendencies. In fact, the FFANN has been observed, in literature, to suffer from overshooting and over-fitting during learning [21, 1]. Table 8 presents comparative performances of competing techniques on the Pearson Correlation Coefficient (PCC) metric.

Table 8. Performance of techniques on the PCC metric.

Dataset	Bosom-Lstm	FFANN	p.LSTM	GRU	IndRNN
Stanford	0.7287	0.5963	0.8145	0.6731	0.7086
Yahoo!	0.8861	0.6257	0.9256	0.7354	0.5648
Skytrax	0.9053	0.7283	0.8769	0.5786	0.7365
Average	0.8400	0.6492	0.8723	0.6624	0.6700
Epoch	10000	10000	10000	10000	10000
Runs	30	30	30	30	30

In this set of experiments, the PCC is used to assess the extent to which a technique is able to predict the target output during the testing phase. In Table 8, the best PCC value in any of the 30 runs per technique is recorded. On average the peephole LSTM has the best predictive power compared to all other competing techniques, at 0.8723. This performance implies that the peephole LSTM has approximately 87.23% correct prediction rate, compared to the BOSOM-LSTM at 84.00% prediction rate. The FFANN is the least performer on this metric, at 0.6492.

The performances of the IndRNN and the GRU network are considerably good despite the fact that they are lower than both the peephole LSTM and the BOSOM-LSTM technique.

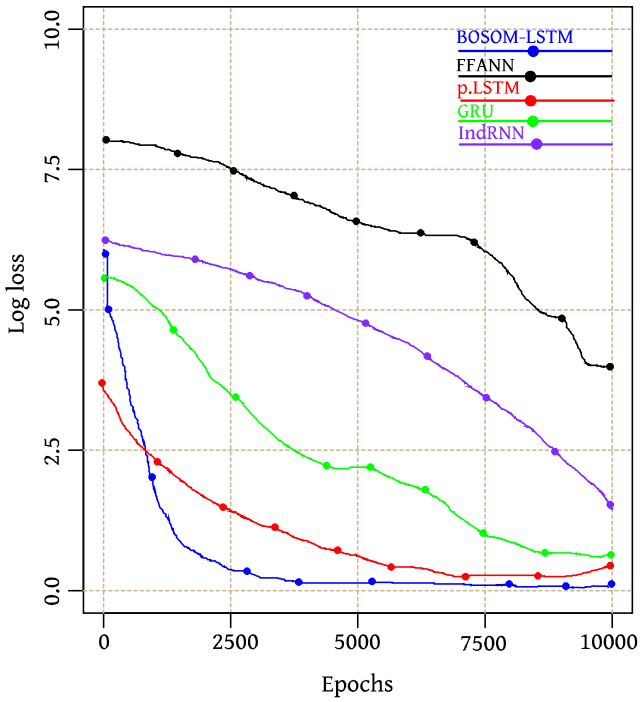

Figure 4. Cross Entropy performance on Stanford dataset.

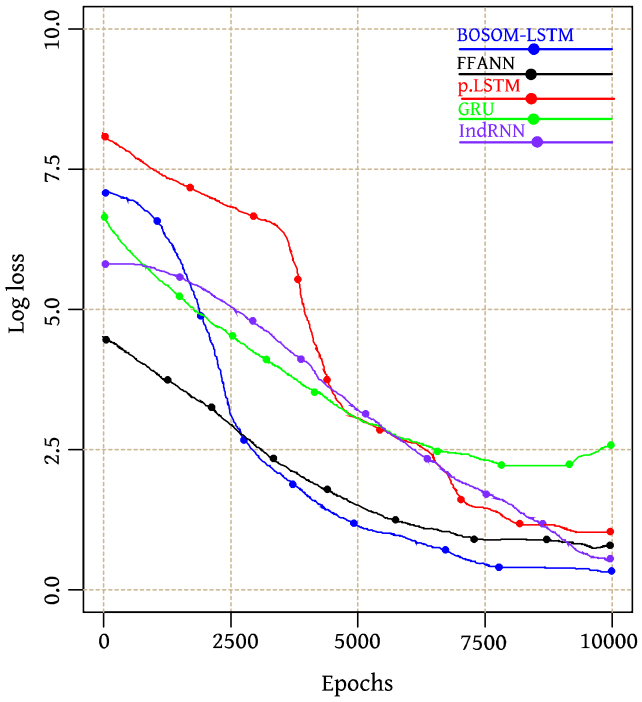

Figure 5. Cross Entropy performance on Yahoo! Dataset.

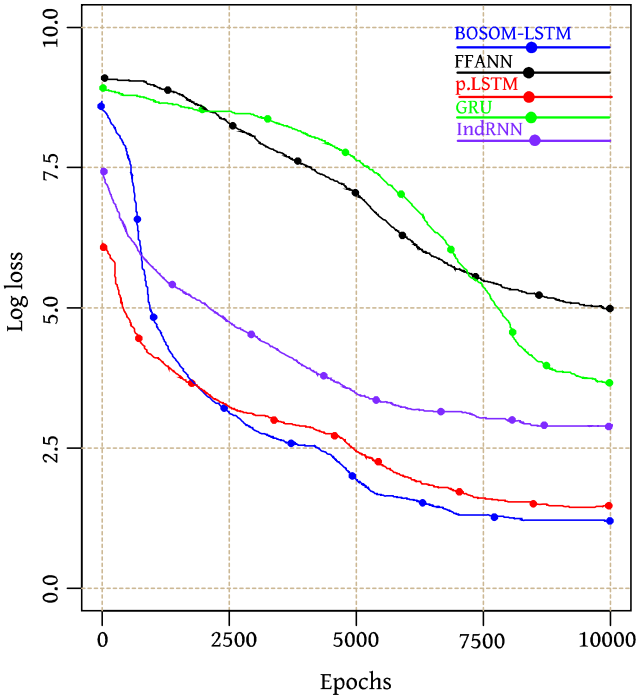

Figure 6. Cross Entropy performance on Skytrax dataset.

The performance of competing techniques for the considered datasets, on the cross entropy loss metric is shown in Figures 4, 5, and 6. It is evident that all techniques exhibit consistent behaviour, that is, at the beginning of a run there are high rates of CE loss and the rate of loss decreases as a technique learns in the process. The major difference lies in a technique's ability to reduce the CE loss significantly towards zero. Further, it appears, there are datasets wherein generally all techniques are able to reduce the CE loss significantly (for example, the Stanford dataset) to near zero values while in other datasets all techniques have generally higher loss levels (for example, the Skytrax dataset). Despite this behaviour, it is clear that the BOSOM-LSTM technique has the least loss on the CE loss metric. This performance suggests that the BOSOM-LSTM is a reliable technique for application areas that require high precision and high reliability. Similarly, the peephole LSTM has significantly good performance on the CE loss metric for all datasets. The performance put up by the peephole LSTM corroborates results reported in several studies [17, 15, 1]. In a way, the performance of the peephole LSTM, in these experiments, acts as a control measure for the newly proposed BOSOM-LSTM technique.

The GRU network and the IndRNN are average performers on the CE loss metric. These two techniques are relatively competitive; however, they have high loss levels. These results have not yet been reported in literature. The major metric on which the GRU and IndRNN have been tested is the MSE metric. On the MSE metric, results in this paper corroborate results reported by several researchers [2, 11],

however, there are several deviations that have been reported by other researchers [18]. The FFANN is the least performer on the CE loss metric. In fact, the FFANN has been the least performer on all the metrics used in this experimentation process. It may thus be put forth that non-recurrent Artificial Neural Networks do not perform well on data analysis problems that require review of long sequences in input vectors. Figures 7, 8, and 9 present the performances of the techniques on Precision Recall Graphs.

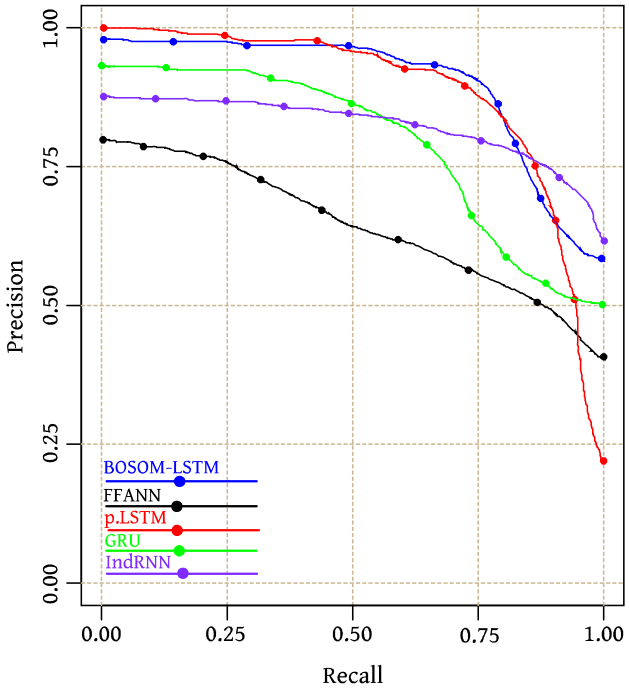

Figure 7. Precision Recall performance on Stanford dataset.

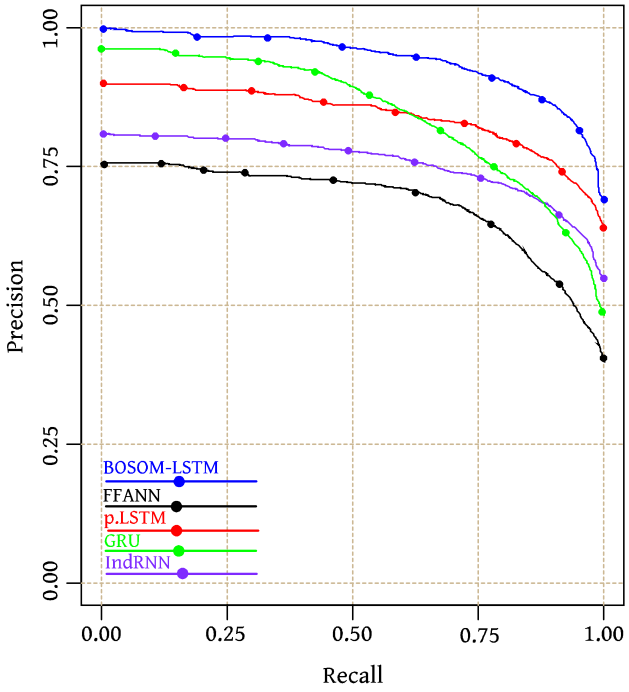

Figure 8. Precision Recall performance on Yahoo! Dataset.

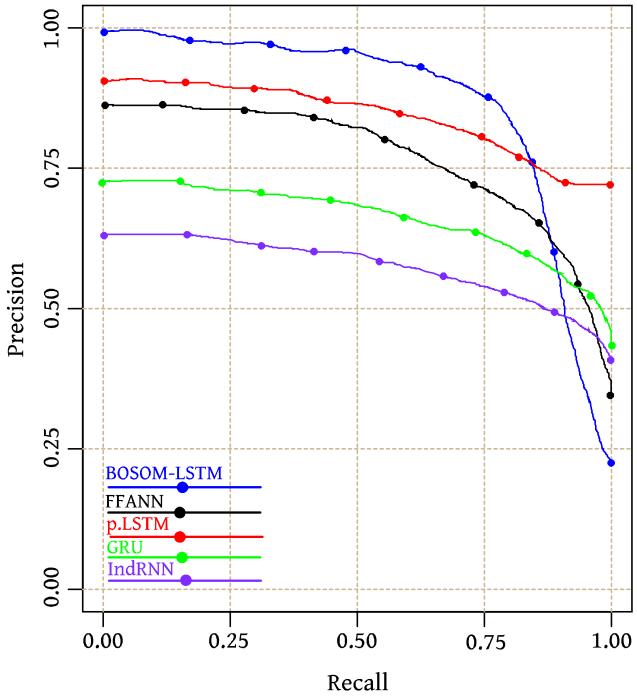

Figure 9. Precision Recall performance on Skytrax dataset.

By observing the curves in Figures 7, 8, and 9, it can be concluded that all the competing techniques have very good recall performance. It can be observed that all the techniques involved are able to reach 100% recall rate, meaning that all the techniques can be relied upon. Significant performance differences can be observed on the Precision axis, where the precision rates appear to decrease across runs. By calculating the average area under each curve, these differences become more apparent. Table 9 shows the average area under the Precision Recall curve for each technique.

Table 9. Average area under Precision Recall curves.

Technique	Area under curve (u^2)
BOSOM-LSTM	0.7528
FFANN	0.5735
p.LSTM	0.7083
GRU	0.6751
IndRNN	0.5962

As shown in Table 9, the BOSOM-LSTM technique has the best performance on the Precision Recall curve. However, it is worth noting that all techniques are above the 0.5000 mark. This suggests that all techniques are performing above average. Figures 10 and 11 present the performances of the techniques on space complexity metric and time complexity metric, respectively. The time and space complexity metrics are important because they measure a technique's inherent demand for computation power and storage space in respect to its ability to solve a problem at hand. In other words, time and space complexity measure how a technique consumes computing resources. In many application areas, the availability of resources is a critical factor in choosing an

appropriate technique to solve a problem at hand.

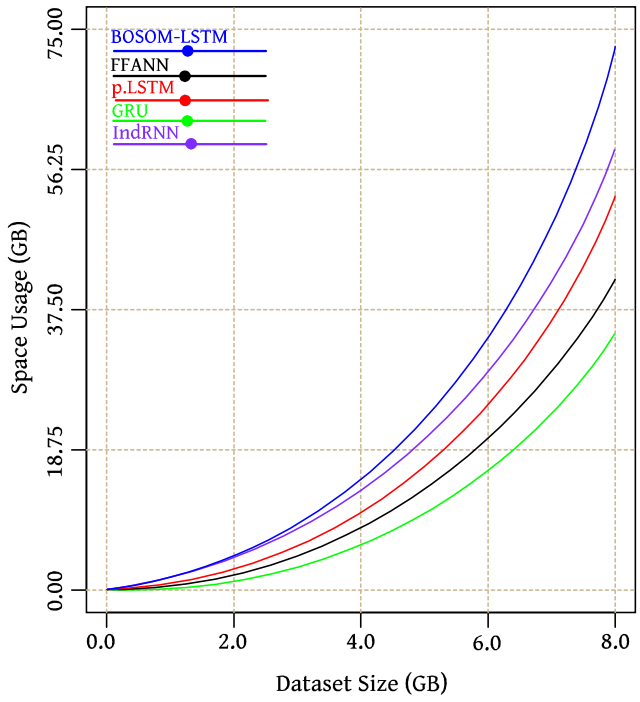

Figure 10. Space complexity behaviour.

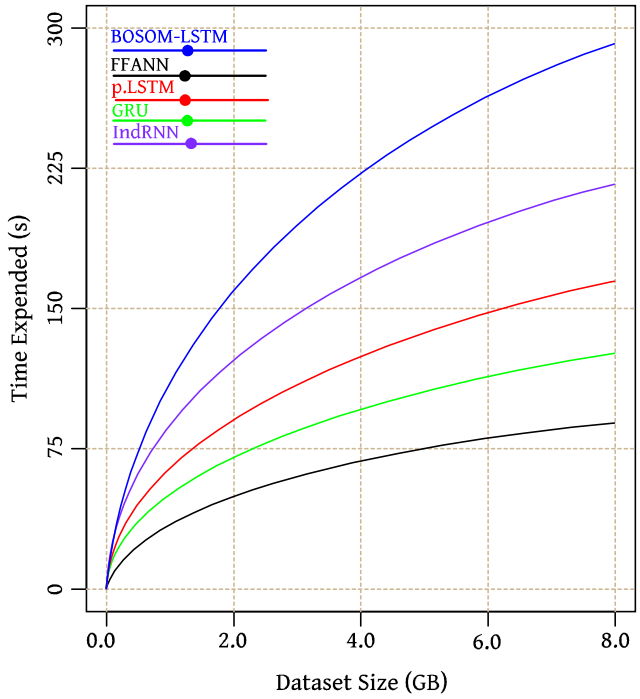

Figure 11. Time complexity behaviour.

As shown in Figure 10, there is a significant difference on how much the various techniques demand for space resources. It is clear in Figure 10 that the BOSOM-LSTM hybrid technique is the least performer. The BOSOM-LSTM technique has high demand for space resources. It, thus, may be put forth that the BOSOM-LSTM is only applicable in environments where there are ample amounts of space resources. The GRU network has the least space requirements and appears to be best suited in space constrained applications. The peephole LSTM appears to have average demands for space requirements, and thus is applicable in space flexible environments. In Figure 11, there are significant differences in regard to time expended for a given input dataset size. It is apparent that the BOSOM-LSTM hybrid technique is the least performer on the time complexity metric. It is clear that the BOSOM-LSTM technique is a very slow technique compared to all other competing techniques. It may be put forth that the BOSOM-LSTM technique must be applied in application areas that are not highly time constrained. In contrast, the peephole LSTM has better time requirements.

7. Conclusion

This paper has successfully proposed a hybrid technique of BOSOM and the LSTM. It has been shown in this paper that the BOSOM-LSTM hybrid technique is a very capable technique, particularly on accuracy and reliability in data analysis. The paper has further noted that the BOSOM-LSTM hybrid technique is a highly resource demanding technique, and this may limit its applicability in resource constrained application areas. This demand for resources may be attributed to the fact that the BOSOM-LSTM hybrid technique makes use of a buffering mechanism. It appears that this buffering mechanism increases time demands and space demands on the technique. Further research is needed in order to improve the buffering mechanism.

References

[1] Chaudhary, R. and Patel, H. (2015). A Survey on the Back-Propagation Algorithm for Neural Networks. International Journal for Technological Research in Engineering, 2, pp. 729-733.

[2] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. [Online] arXiv preprint arXiv:1406.1078.

[3] Demuth, H. B., Beale, M. H, De Jess, O. and Hagan, M. T. (2014). Neural Network Design. 2nd ed., Martin Hagan, Oklahoma, USA: Oklahoma State University.

[4] Fausett, L. V. (1994). Fundamentals of Neural Networks: Architectures, Algorithms and Applications. Prentice-Hall International Editions, New Jersey, USA: Prentice-Hall.

[5] Gers, F. A., Schmidhuber, J. and Cummins, F. (2000). Learning to Forget: Continual Prediction with LSTM. Neural Computation, 12 (10), pp. 2451-2471.

[6] Gers, F. A. and Schmidhuber, J. (2001). LSTM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages. IEEE Transactions on Neural Networks, 12 (6), pp. 1333-1340.

[7] Gers, F. A., Schraudolph, N. N. and Schmidhuber, J. (2002). Learning Precise Timing with LSTM Recurrent Networks. Journal of Machine Learning Research, 3, pp. 115-143.

[8] Gurney, K. (2004). An Introduction to Neural Networks. London: Taylor and Francis e-Library, University College London (UCL).

[9] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9 (8), pp. 1735-1780.

[10] Kröse, B. J. and van der Smagt P. (1996). An Introduction to Neural Networks. 8th ed. Department of Computer Systems, University of Amsterdam, Netherlands.

[11] Li, S., Li, W., Crook, C., Zhu, C. and Yanbo, G. (2018). Independently Recurrent Neural Network (IndRNN): Building A Longer and Deeper RNN. [Online] arXiv preprint arXiv:1803.04831v3.

[12] Mzelikahle, K., Mapuma, D. J., Hlatywayo, D. J. and Trimble, J. (2017). Optimisation of Self Organising Maps using the Bat Algorithm. American Journal of Information Science and Computer Engineering, 3 (6), pp. 77-83.

[13] Pang, B. and Lee, L. (2005). Seeing Stars: Exploiting Class Relationships for Sentiment Categorisation with Respect to Rating Scales. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, pp. 115-124.

[14] Pérezgonzález, J. D. and Gilbey, A. (2011). Predicting Skytrax Airport Rankings from Customer Reviews. Journal of Airport Management, 5 (4), pp. 335-339.

[15] Sak, H., Senior, A. and Beaufays, F. (2014). Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. In: Fifteenth Annual Conference of the International Speech Communication Association.

[16] Schleif, F. M. (2014). Discriminative Fast Soft Competitive Learning. In: Proceedings of the International Conference on Artificial Neural Networks, September 2014. pp. 81-88, Springer, Cham.

[17] Schmidhuber, J. (2015). Deep Learning in Neural Networks: An Overview. Neural Networks, Elsevier, 61, pp. 85-117.

[18] Shalev-Shwartz, S. and Ben-David, S. (2014). Understanding Machine Learning: From Theory to Algorithms. United Kingdom: Cambridge University Press.

[19] Spanakis, G. and Weiss, G. (2016). AMSOM: Adaptive Moving Self-Organizing Map for Clustering and Visualization. [Online] arXiv preprint arXiv:1605.06047.

[20] Taheri J. and Zomaya A. Y. (2006). Artificial Neural Networks. In: Zomaya A. Y. (eds.) Handbook of Nature-Inspired and Innovative Computing: Integrating Classical Models with Emerging Technologies. New York, USA: Springer, pp. 147–185.

[21] Yang, X. S. (2010). Nature-Inspired Metaheuristic Algorithms. University of Cambridge: Luniver Press.