Received: June 9, 2019

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Automotive Rank Based ELM Using Iterative Decomposition

Archana Nagelli[1]* **Ramesh Ragala[1]** **Badarudeen Saleena[1]**

[1]*School of Computing Science and Engineering, VIT University-Chennai Campus, India*
* Corresponding author's Email: archana.nagelli@gmail.com

Abstract: Nowadays, the Extreme learning machine (ELM) is playing a key role in machine intelligence and big data analytics due to its various advantages such as fast training rate, universal classification/regression and the capability of approximation. The standard ELM uses the Moore–Penrose generalized pseudo-inverse for solving the hidden layer activation matrix and also it identifies the output weights. Because of that, the standard ELM takes more time to train the features from the dataset. In ELM, scalability also considered as a one of the major concern while processing the large dataset. In order to overcome this concern, the Automotive Rank based ELM (AR-ELM) is proposed to obtain an effective tensor decomposition for diminishing the training time. Besides, the Bayesian approach is considered in this AR-ELM to remove the redundancy from the decomposed samples of the tensor. The major objective of this proposed AR-ELM is to process the large amount of dataset without depending on memory capacities. The recognition accuracy is improved by eliminating redundant information. The key idea of the AR-ELM is to reduce the training time while processing the huge dataset. The implementation and simulation of the AR-ELM is done in Spark Python 3.7. The performance of the AR-ELM is analysed in terms of accuracy, precision, recall and training time. The proposed methodology is compared with three existing methodologies such as basic ELM, ELM-TUCKER and ELM-PARAFAC. The recognition accuracy of the AR-ELM methodology with Hardlims activation function is 0.8879 for letter recognition dataset, it is high when compared to the basic ELM, ELM-TUCKER and ELM-PARAFAC that are 0.8102, 0.8375 and 0.834 respectively.

Keywords: Extreme learning machine, Scalability, Accuracy, Training time, Tensor decomposition.

1. Introduction

ELM is generally a single layer feed-forward neural network (SLFN) with significant features such as integrated solutions for regression, binary, and multi-class classification. The ELM performs effective classification in balanced datasets. But, the ELM provides less sensitivity while processing the imbalanced datasets [1]. The feed-forward neural networks are the example for inductive learning. This inductive learning is used to solve the major issues such as overfitting, convergence speed, and location of free parameters [2]. The issue over the Artificial Neural Network (ANN) learning speed is solved by developing the ELM in SLFN [3]. The gradient-based methods adjust the network parameters of hidden nodes to increase the learning speed. Instead of using this, the weights and bias values are generated randomly to increase the learning speed [4]. The SLFN is transformed into a linear system based on the random measurement of weights and bias operation. This operation also used for the output weight's analytical determination by using least-squares [5]. The ELM only updates the output weights among the output layer and the hidden layer. But, the values of the biases and input weights of the hidden layer are generated in random manner [6].

The learning speed of the ELM generalization performance is higher than the SVM without tuning any model parameters. Besides, the ELM has been developed to solve the drawbacks of multilayer feedforward neural networks [7]. The ELM has various advantages such as unification of multi-classification, fast learning speed, minimal human intervention, ease of implementation and regression [8, 9]. The ELM is computationally powerful single-

hidden-layer feed-forward neural network, which is widely utilized in the several real-world problems due to its remarkable efficiency, simplicity, and impressive generalization performance [10]. The ELM utilized in various applications such as object recognition [11], landmark recognition [12], identification of refractive index for ionic liquids [13], EEG signal classification [14], protein fold recognition [15], intrusion detection system [16] etc. The major contributions of this paper are stated as follows:

- The tensor decomposition is effectively performed by using two different decomposition methods. One is tucker decomposition and another one is automotive rank selection. In automotive rank selection, iterative low rank approximation is used for further decomposing of the tensor information.
- The iterative low rank approximation is used for compressing the tensor information to reduce the training time of ELM.
- Besides, the testing accuracy of the ELM is improved based on the elimination of redundant information from the decomposed values by using the Bayesian approach.

This research work is organized as follows: Section 2 provides the literature survey about the ELM. The detailed description about the proposed Automotive Rank based ELM is given in section 3. Section 4 presents the results and discussion about the proposed ELM methodology. Finally, the conclusion is made in section 5.

2. Literature survey

X. Su, S. Zhang, Y. Yin, and W. Xiao, [17] presented the combination of the multi-layer ELM (ML-ELM) and the Principal Component Analysis (PCA) method to develop a modified ML-ELM algorithm for the prediction of permeability index of the blast furnace. The prediction accuracy of the modified ML-ELM is enhanced by solving the multicollinearity problem which is present in the last hidden layer of the ML-ELM algorithm. The problem due to the multicollinearity is resolved by using the PCA and it enhances the precision and generalized performance. The bias and the weights of every hidden layer and input layer are randomly produced, respectively.

A. Mishra, A. Rajpal, and R. Bala, [18] introduced the Bi-directional ELM (B-ELM) for obtaining the watermarking of the JPEG images and this B-ELM has the capacity of fast training with less number of hidden neurons. The principle behind the B-ELM is to optimize the two parameters which is need in the hidden layer represented as (a_i, b_i). Where, the weight vector is a_i, and bias of the hidden node is b_i. The optimization of these two parameters results in the decrease of residual error of SLFNs as fast as possible. The feature extraction performed using B-ELM is semi blind in nature.

J. Tang, C. Deng, and G.B. Huang [19] presented the Hierarchical ELM (H-ELM) based theories of multilayer perceptron. The training architecture of the H-ELM is divided into two separate phases such as unsupervised hierarchical feature representation and supervised feature classification. The high-level sparse features are obtained by using the N-layer unsupervised learning. The features are randomly perturbed during the classification process and these features are used as the input to the supervised ELM-based regression for obtaining the final classification results. Here, the features are not directly extracted by the H-ELM. It uses the one more technique called l_1 optimization to establish the ELM encoder for extracting the sparse and compact features from the input.

X. Li, W. Mao, and W. Jiang [20] introduced the Multiple-Kernel Learning (MKL) ELM for learning the optimal combination of multiple large-scale data sets. In this MKL-ELM, two different formulations of multiple-kernel classifiers are introduced. The first formulation mainly depends on the convex combination of the base kernels and second formulation utilizes the convex combination of the equivalent kernels. Additionally, the MKL-ELM optimizes the regularization parameter at unified framework along with the kernels. The optimization of the regularization parameter creates the learning system more automatic. The training time of the MKL-ELM is high when compared to the single kernel ELM and ELM.

N.K. Nair and S. Asharaf, [21] presented the tensor decomposition based ELM by considering two different decomposition methods such as TUCKER and PARAFAC (parallel factor analysis). The TUCKER decomposition is a generalized version of the canonical decomposition. Here the ELM is trained based on the factor matrices from the ELM-PARAFAC and core tensor from the ELM-TUCKER. The training time of the standard ELM is more when compared to the ELM-TUCKER

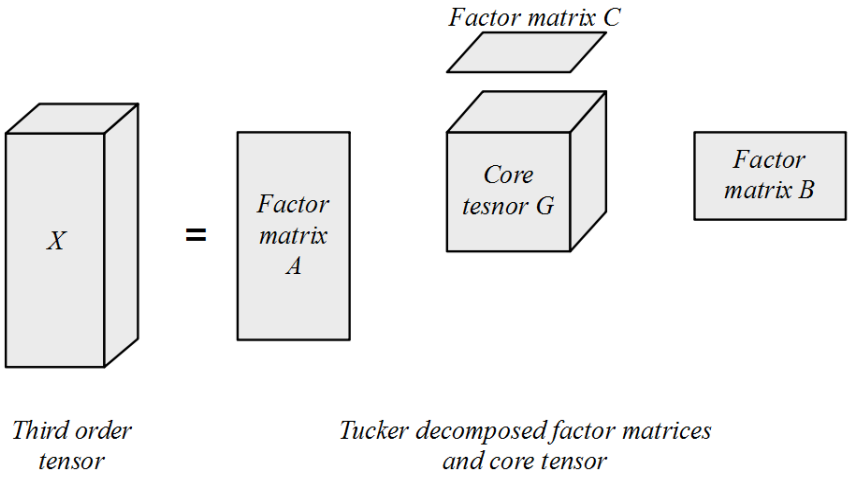

Figure.1 Tucker decomposition of the tensor in the three ways case, the tucker decomposition

and ELM-PARAFAC. Because, the basic ELM requires more time while processing the huge dataset.

The existing ELM learning algorithm requires more time to process the large dataset. Moreover, some existing ELM researches carried out only one stage decomposition. But, in this AR-ELM methodology, two levels of decomposition have been considered to reduce the tensor information. Furthermore, the training time for processing the huge dataset reduced by using the iterative low rank approximation.

3. AR-ELM methodology for an effective learning of tensors

The automotive rank based ELM is developed for reducing the training time and increasing the accuracy while processing the large dataset. The AR-ELM has two different stages. In the first stage, the HOSVD based tucker decomposition is performed for decomposing the tensor into the core tensor and multiple matrices. Then the automotive rank selection is used in the second stage for the further decomposition of the HOSVD decomposed samples. The detailed explanation about the AR-ELM is given below:

3.1 Tucker decomposition on tensors

The tucker decomposition typically decomposes a tensor into the core tensor and multiple matrices. The multiple matrices from the tucker decomposition are related to the various core scaling along with each node. So, the tucker decomposition is considered as a high order PCA. Here, the Higher Order generalization of Singular Value Decomposition (HOSVD) is used as the optimization method to compute the tucker decomposition. The Tucker decomposition used in the higher order sensors is shown in the following Fig 1.

In the three ways case, the tucker decomposition of $x \in \mathbb{R}^{I \times j \times k}$ is expressed in the following Eq. (1).

$$X = G \times A_1 \times B_2 \times C_3 = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{s=1}^{S} G_{pqs} a_p \circ b_q \circ c_s = [G; A, B, C] \quad (1)$$

Where, the factor matrices are $A \in \mathbb{R}^{I \times P}$, $B \in \mathbb{R}^{J \times Q}$ and $C \in \mathbb{R}^{K \times S}$. Then the core tensor is represented as $G \in \mathbb{R}^{P \times Q \times S}$ and third order tensor is represented as X. This core tensor displays the interaction level among the various components. The decomposed values from the TUCKER decomposition is applied to the automotive rank selection to obtain the selective values from the set of decomposed values.

3.2 Automotive rank selection

In automotive rank selection, two different scenarios are developed for low-rank approximation of θ by automatically selecting the gradual decreasing ranks (i.e., $R_1 > R_2 > ..$). The two different scenarios are Bayesian approach and constant compression rate. The redundancy present in the weight tensor is eliminated by using the Bayesian approach based rank estimation. Subsequently, the parameter reduction rate is obtained by using the constant compression rate.

3.2.1. Bayesian approach

Bayesian approach has two notations for the ease of redundancy elimination such as extreme rank and a weakened rank. The redundancy present in the tensor is removed after decomposition at extreme rank value. But, in the weakened rank only a suitable amount of redundancy is kept in tensor. Initially, the Global Analytic Solution of Empirical Variational Bayesian Matrix Factorization (GAS of EVBMF) is used in the Bayesian approach for obtaining the value of extreme rank R_{extr}. The weakened rank is identified after the estimation of extreme rank. The matrix rank is automatically calculated by using the Bayesian matrix in the GAS of EVBMF. The utilized Bayesian interference gives the suboptimal solution. The decomposition values are applied to the unfoldings of the weight tensor which is related to the channel dimensions. The unfolding is performed in the iteration of $k+1$ over the matrices of sizes $R_{in}^{k} \times d^2 R_{out}^{k}$ and $R_{out}^{k} \times d^2 R_{in}^{k}$.

The value of the weakened rank R_{weak} is linearly based on extreme rank and it preserves a high amount of redundancy in the decomposed tensor. The fine tuning is enabled and a compression step with better accuracy is achieved by fixing $R = R_{weak}$. The weakened rank is expressed in Eq. (2).

$$R_{weak} = R_{init} - w(R_{init} - R_{extr}) \quad (2)$$

Where, the hyper parameter is denoted as w ($0 < w < 1$), which is called as weakening factor and R_{init} describes the initial rank value.

3.2.2. Constant compression rate

The parameter reduction rate is used for selecting the tensor approximation rank. The rate of parameter reduction is calculated in each compression step. The speed-up of each convolutional layer is controlled by selecting the rank.

The following Eq. (3) is derived based on the parameters in the decomposed layer ($R_{in}C_{in} + R_{out}R_{in}d^2 + R_{out}C_{out}$) and also it assumes multilinear rank form that is $(\beta R, R), \beta > 0$.

$$R \leq \frac{-\frac{C_{in}+\beta C_{out}}{\beta d^2} + \sqrt{\frac{(C_{in}+\beta C_{out})^2}{\beta^2 d^2} + \frac{4C_{in}C_{out}}{\beta \alpha}}}{2} \quad (3)$$

Where, C_{in} and C_{out} are the input and output channel respectively, and β is the output weights.

The ranks are chosen according to the inequality given in Eq. (3) for obtaining the times α parameters reduction by using the Tucker-2 tensor approximation.

3.2.3. Iterative low rank approximation algorithm

The steps present in the low rank approximation is given as follows:

1. At first, the layer is compressed with weight tensor θ by solving the concern over the minimization of the Frobenius norm for given rank R that is expressed in Eq. (4).

$$\min_{\theta_1^R,\ldots,\theta_N^R} \left\|\theta - \hat{\theta}^R\right\|,$$
$$F_{fact}(\hat{\theta}^R) = (\theta_1^R,..,\theta_N^R), \quad (4)$$

Where, the tensor's factorized form of components is denoted as $\theta_1^R,..,\theta_N^R$. The weights of the N layers is defined in the initial layer when the decomposition of initial layer during the factorization of rank-R.

2. The fine-tuned weights $\{\theta_n^R\}_{n=1}^N$ of decomposed layer is updated for further compression. Specifically, the rank values $R' < R$ solves the minimization problem given in Eq. (5).

$$\min_{\theta_1^{R'},\ldots,\theta_N^{R'}} \left\|F_{full}(\theta_{fact}) - \hat{\theta}^{R'}\right\|,$$
$$\theta_{fact} = (\theta_1^R,..,\theta_N^R) \quad (5)$$
$$F_{fact}(\hat{\theta}^{R'}) = (\theta_1^{R'},..,\theta_N^{R'})$$

Where, the operators are represented as F_{full} and F_{fact}, and the rank of factorized weights are represented as θ_{fact}. The updated weights of the decomposed layer are $\theta_1^{R'},..,\theta_N^{R'}$ that also defined as factor matrices.

3. The loss function l of the training data $\{(X_j, Y_j)\}_{j=1}^J$ is reduced at the fine tuning step. Where, input sample is X_j and related target value is Y_j. The following optimization problem Eq. (6) is used for fine tuning process.

$$L(\theta) \rightarrow \min_{\theta \epsilon \ominus_{fact}^R}, \quad s.t. L(\theta) =$$
$$\sum_{j=1}^J l(f^R(X_j, \theta), Y_j) \quad (6)$$

Where, the compressed architecture is f^R and the set of all possible model parameters are $\ominus_{fact}^R$.

This low rank approximation is used instead of the compression and fine tuning steps with weight approximation. Besides, the weight approximation is occurred by automatically selecting ranks. If the layers are not compressed, the optimization problem of Eq. (4) is used else the Eq. (5) based layer compression is used. The fine tuning step is similar for all iterations.

Algorithm 1 – Iterative low rank approximation

Input: The original pre-trained model, M
Output: Compressed fine tuned model M^*

1. $M^* \leftarrow M$
2. **while** - the wanted compression rate is not achieved or **do** - when the automatically selected ranks do not have stability
3. The rank (R) are automatically choose for the approximation of low rank tensor of typical and fully connected weight tensors.
4. Additionally, the layer weights with its rank R tensor approximations are replaced with the compressed model from the M.
5. M^* ←fine tuned model $\widehat{M}$.
6. **end while**

3.3 Automotive rank selection based ELM

The typical machine learning techniques are failed to process a high amount of data. Besides, those methods do not have the capacity to load the entire memory at once. Because the data from the real-time applications are extremely large. Thus the standard ELM techniques cannot process the immense data in efficient manner. Additionally, the time consumption of the standard ELM is high due to its calculation of computation of matrix Moore–Penrose pseudoinverse.

In order to overcome the problem due to training time, the AR-ELM technique has two levels of decomposition process. In first level the decomposition, has the tucker decomposition to effectively decompose the tensor into a core tensor and multiple matrices. In second level decomposition, the Bayesian approach and calculation of constant compression rate are used to eliminate the redundancy from the compressed values.

3.3.1. Implementation of automotive rank selection based ELM

The following Fig 2 illustrates the process of the automotive rank selection based ELM. In this section, the working process of the training and

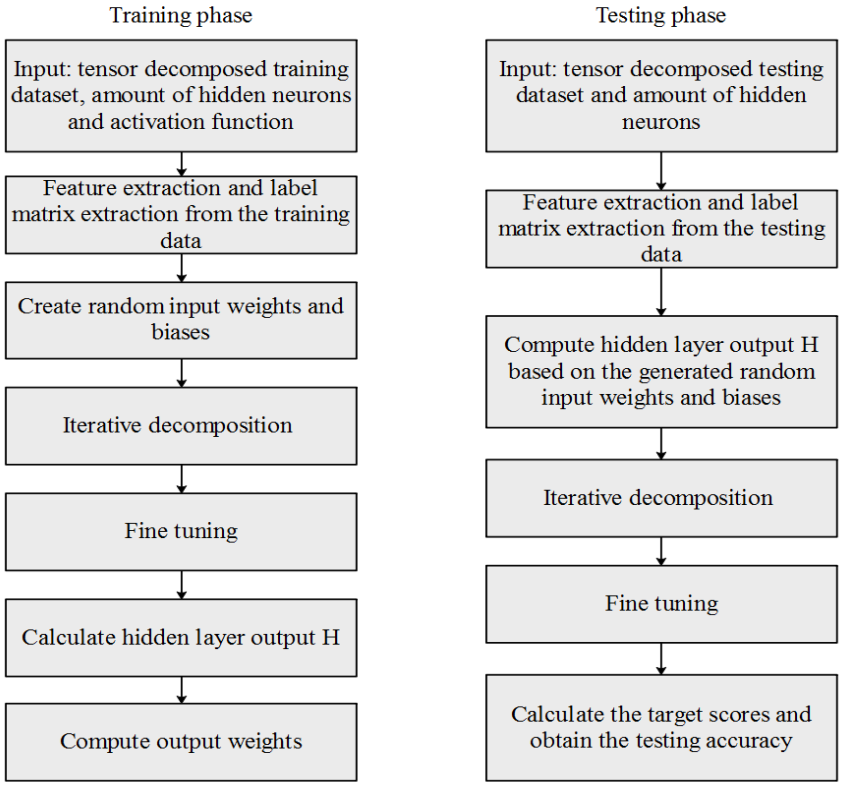

Figure.2 Automotive rank selection based ELM

testing process is clearly described. Initially, the tensor-based training dataset is given as the input to the ELM. ELM receives three different inputs such as training dataset samples, number of hidden neurons (L) and activation function (G). From the training dataset, the feature and label matrixes are extracted. Then the input weights (W) and biases (b) are randomly generated. Besides the output matrix of the hidden layer is calculated by considering various parameters such as input weight matrix, biases, feature matrix and specified activation function such sigmoid, tanh, hardlims, tribas or sine. The Moore–Penrose inverse of the hidden layer matrix is used for calculating the output weights.

Similarly, the testing dataset is used for a testing phase and in that testing phase the predictable labels and feature matrix are obtained by processing the testing dataset. In order to find the actual labels, the target scores are computed in this testing phase. The actual labels are compared with the expected labels to obtain the overall accuracy.

The training algorithm developed in the automotive rank selection based ELM is given below:

Algorithm 2-Training algorithm for decomposition based ELM

Input: Training dataset $S = [(x_i, t_i)|x_i \epsilon R, i = 1,2,3, \dots N]$, L- amount of hidden neurons of the ELM and the $G(w, b, x)$- output function of hidden node.

Output: output weights and output matrix

1. Random allocation of input weights and biases of ELM
2. Decomposition over the large of tensor data and this data given as the input to the ELM.
3. Calculation of hidden layer output matrix H
4. Calculation of output weights (β) using the constant compression rate from the iterative low rank approximation.

3.3.2. Decomposition technique on ELM

The measurement of decomposition of sensors are carried out by using the HOSVD based tucker decomposition and automotive rank selection. Here, the Singular Value Decomposition (SVD) takes place on each n-mode matricized component. The tensor decomposition is written in the following Eq. (7).

$$A = S \times U \times 2V \times W \tag{7}$$

Where, S belongs to the $R^{I \times J \times K}$, U belongs to the $R^{I \times I}$, V belongs to the $R^{J \times J}$ and W belongs to the $R^{K \times K}$. The I, J and K are the indices of the tuned model M from the iterative low rank approximation. This HOSVD is also applicable for the higher order tensors and it is obtained by using the SVD of each flattening matrices.

4. Results and discussion

The proposed ELM is implemented and simulated in Spark python 3.7. The proposed ELM implementation has two stages. In first stage, the tucker decomposition utilized for decomposing the tensors. The redundancy present in the tensor decomposition values is eliminated by using the iterative low-rank approximation. The performance of the proposed ELM is analyzed in four different parameters such as accuracy, precision, recall and training time. Besides, this proposed methodology has analyzed in different kind of datasets.

4.1 Dataset description

In this proposed ELM, there are six different datasets are analysed such as MNIST handwritten dataset, KDD Cup 1999 dataset, Statlog (Landsat Satellite) dataset, Statlog (Shuttle) dataset, Letter Recognition dataset and Mushroom dataset.

a. MNIST handwritten dataset

The MNIST dataset is used for the digit recognition process. This dataset has 42000 training samples and 784 features along with labels.

b. KDD Cup 1999 dataset

The KDD Cup 1999 dataset is an intrusion detection dataset which contains 60000 training samples and 41 features along with one label. The label of the KDD Cup 1999 dataset specifies normal or malicious attacks.

c. Statlog (Landsat Satellite) dataset

This satellite dataset has multispectral values of pixels of the satellite image. This dataset has 7 decision classes and 4435 samples with 36 attributes. The various classes included in this statlog dataset (Landsat) are grey soil, soil with vegetation stubble, red soil, damp grey soil, mixture class (all types present) and very damp grey soil.

d. Statlog (Shuttle) dataset

This shuttle dataset has 43500 training samples with 9 attributes. The various classes included in this shuttle dataset are Fpv Close, Fpv Open, Rad Flow, Bpv Close, Bpv Open and Bypass.

e. Letter Recognition dataset

The letter recognition dataset classifies huge amount of pixel displays which belongs to one of the 26 capital letters in the English alphabet. This dataset has 20000 samples with 16 attributes.

f. Mushroom dataset

The mushroom dataset contains descriptions of 23 species of grilled mushrooms of Lepiota and Agaricus Family. This dataset has 8124 training samples and 22 attributes.

4.2 Performance metrics

The performance of the proposed ELM is analyzed by four different parameters such as accuracy, precision, recall and training time (in seconds).

Accuracy

Accuracy (ACC) is defined as the ratio of correct predictions over the total amount of iterations assessed in the ELM. The accuracy is expressed in Eq. (8).

$$\mathrm{ACC} = \frac{TP+TN}{TP+FP+TN+FN} \quad (8)$$

Where, TP is true positive, TN is true negative, FP is false positive and FN is false negative.

Precision

Precision (P) is defined as the measurement of positive patterns which are correctly identified from the total predicted patterns. The Eq. (9) describes the precision.

$$\mathrm{P} = \frac{TP}{TP+FP} \quad (9)$$

Recall

Recall (R) is defined as the measure of the positive patterns which are correctly classified by the ELM. The following Eq. (10) specifies the specificity.

$$\mathrm{R} = \frac{TP}{TP+TN} \quad (10)$$

Table 1. Training time comparison for MNIST handwritten dataset

Activation function	Basic ELM [21]	ELM-TUCKER [21]	ELM-PARAFAC [21]	AR-ELM
Sine	12332.981	10643.794	10555.316	10423.46
Tribas	12296.731	10474.998	10414.283	10369.13
Sigmoid	12298.065	10484.763	10421.326	10398.457

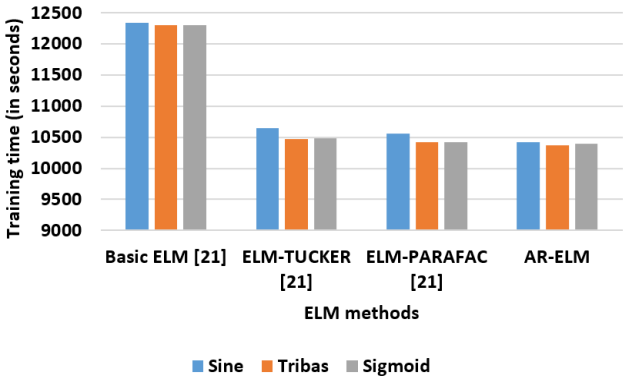

Figure.3 Comparative analysis of training time for MNIST handwritten dataset

4.2.1. Performance analysis of MNIST handwritten dataset

The following Table 1 and Fig 3 provides the comparative analysis of the MNIST handwritten dataset for the AR-ELM with existing ELM methods such as basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. This MNIST handwritten dataset is analyzed in three different datasets such as sine, tribas and sigmoid.

Table 2 and Fig 3 shows that the AR-ELM training time for MNIST handwritten dataset is less when compared to the existing methods such as basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. The training time of the Basic ELM [21] is high than the other methods. Because, the basic ELM takes more time for processing the huge datasets. Besides, the AR-ELM training time is lesser than the ELM-TUCKER [21] and ELM-PARAFAC [21], due to its fine-tuning and higher level of decomposition over the tensors.

The testing performance of the MNIST handwritten dataset is shown in the Table 2. The AR-ELM method has improved performance interms of accuracy, precision and recall than the other methods basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. The AR-ELM has high accuracy, because of its optimal learning rate.

Table 2. Comparison of testing performance for MNIST handwritten dataset

Activation function	Basic ELM [21]			ELM-TUCKER [21]			ELM-PARAFAC [21]			AR-ELM		
	ACC	P	R	ACC	P	R	ACC	P	R	ACC	P	R
Sine	0.0991	0.10	0.10	0.2154	0.21	0.22	0.227	0.22	0.23	0.248	0.25	0.262
Tribas	0.25011	0.23	0.25	0.4527	0.46	0.45	0.4780	0.48	0.48	0.4910	0.52	0.505
Sigmoid	0.8595	0.86	0.86	0.8732	0.88	0.87	0.8790	0.88	0.88	0.8812	0.91	0.89

Table 3. Training time comparison for KDD Cup 1999 dataset

Activation function	Basic ELM [21]	ELM-TUCKER [21]	ELM-PARAFAC [21]	AR-ELM
Tanh	14198.517	12258.568	12182.942	12102.56
Sigmoid	14190.885	12280.145	12165.450	12143.59
Hardlims	14185.134	11677.938	11592.199	11512.81
Tribas	14176.046	11669.836	11488.908	11412.68

Table 4. Comparison of testing performance for KDD Cup 1999 dataset

Activation function	Basic ELM [21]			ELM-TUCKER [21]			ELM-PARAFAC [21]			AR-ELM		
	ACC	P	R	ACC	P	R	ACC	P	R	ACC	P	R
Tanh	0.9954	0.99	0.98	0.9964	0.99	0.99	0.9964	0.99	0.99	0.9971	0.99	0.98
Sigmoid	0.9959	0.97	0.95	0.9959	0.99	0.99	0.9962	0.98	0.98	0.9968	0.99	0.99
Hardlims	0.9629	0.95	0.96	0.9976	0.97	0.97	0.9982	0.99	0.99	0.9989	0.99	0.99
Tribas	0.9968	0.98	0.97	0.9972	0.99	0.99	0.9987	0.98	0.99	0.9989	0.98	0.99

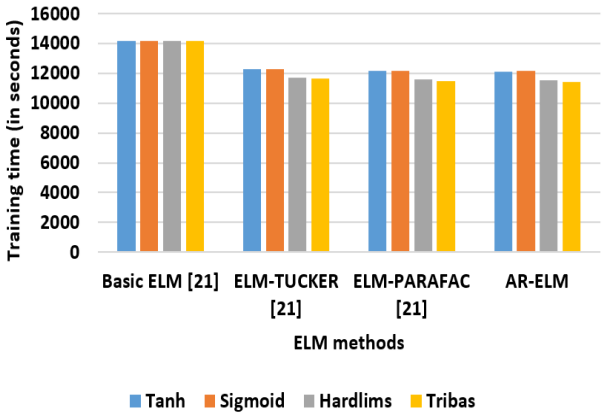

Figure.4 Comparative analysis of training time for KDD Cup 1999 dataset

4.2.2. Performance analysis of KDD Cup 1999 dataset

The following Table 3 and Fig 4 provides the comparative analysis of the KDD Cup 1999 dataset for the AR-ELM with existing ELM methods such as basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. These ELM methods are developed in four different activation functions such as sigmoid, tanh, hardlims and tribas.

The Table 3 and Fig 4 shows that the AR-ELM training time for the KDD Cup 1999 dataset is less when compared to the existing methods such as basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. Due to the high amount of samples of the KDDCup99 data (i.e., above 60000), the training process of the ELM [21] takes more time. Similarly, the training time of the ELM-TUCKER [21] and ELM-PARAFAC [21] also higher than the AR-ELM.

The testing performance of the KDD Cup 1999 dataset is shown in the Table 4. The AR-ELM method has improved performance in terms of accuracy, precision and recall than the other methods basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. The learning rate of the AR-ELM makes the classification process more accurate than the existing methods.

4.2.3. Performance analysis of satellite dataset

Table 5 and Fig 5 provides comparison of training time (in seconds) for two different activation functions such as hardlims and tribas. This training time comparison is made for AR-ELM with three different existing methodologies such as basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21].

Table 5. Training time comparison for satellite dataset

Activation function	Basic ELM [21]	ELM-TUCKER [21]	ELM-PARAFAC [21]	AR-ELM
Hardlims	124.215	116.012	108.011	98.28
Tribas	124.228	102.624	98.927	91.05

Table 6. Comparison of testing performance for satellite dataset

Activation function	Basic ELM [21]			ELM-TUCKER [21]			ELM-PARAFAC [21]			AR-ELM		
	ACC	P	R	ACC	P	R	ACC	P	R	ACC	P	R
Hardlims	0.9825	0.96	0.94	0.9932	0.99	0.99	0.9982	0.99	0.98	0.9986	0.99	0.99
Tribas	0.9943	0.98	0.97	0.9976	0.99	0.98	0.9987	0.98	0.99	0.9989	0.99	0.99

Table 7. Training time comparison for shuttle dataset

Activation function	Basic ELM [21]	ELM-TUCKER [21]	ELM-PARAFAC [21]	AR-ELM
Tribas	6716.550	6376.865	6315.289	6286.254
Tanh	6725.434	6680.117	6617.240	6595.207
Hardlims	6706.932	6629.858	6584.301	6501.125

Table 8. Comparison of testing performance for shuttle dataset

Activation function	Basic ELM [21]			ELM-TUCKER [21]			ELM-PARAFAC [21]			AR-ELM		
	ACC	P	R	ACC	P	R	ACC	P	R	ACC	P	R
Tribas	0.9568	0.95	0.94	0.9752	0.97	0.98	0.9787	0.98	0.97	0.9816	0.97	0.99
Tanh	0.9653	0.95	0.93	0.9812	0.97	0.98	0.9964	0.99	0.99	0.9972	0.99	0.99
Hardlims	0.9629	0.95	0.96	0.9676	0.96	0.97	0.9782	0.98	0.97	0.9941	0.99	0.98

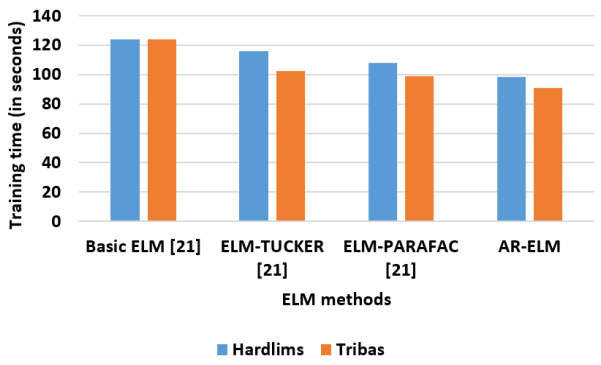

Figure.5 Comparative analysis of training time for satellite dataset

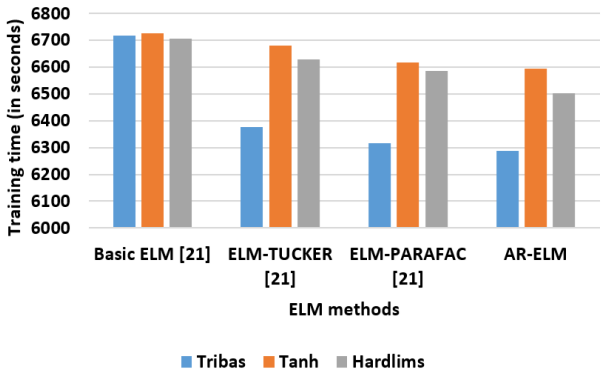

Figure.6 Comparative analysis of training time for shuttle dataset

From the Table 5 and Fig 5 conclude that the training time of the satellite dataset is less when compared to the existing methods basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. The training time of the basic ELM [21] is high, because it spends more time in the hidden layer output matrix computation. The fine tuning of the parameters from the tensors helps to reduce the training time of the AR-ELM than the existing methods.

The testing performance of the satellite dataset is shown in the Table 4. The AR-ELM method has improved performance in terms of accuracy, precision and recall than the other methods such as basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. The testing accuracy of the AR-ELM is improved due to its learning rate and elimination of redundant information from the decomposed tensor samples.

4.2.4. Performance analysis of shuttle dataset

The following Table 7 and Fig 6 provides the comparative analysis of the shuttle dataset for the AR-ELM training time with existing ELM methods such as basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. These ELM methods are developed in four different activation functions such as sigmoid, tanh, hardlims and tribas.

From the Table 7 and Fig 6 shows that the AR-ELM training time for the shuttle dataset is less when compared to the existing methods b basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. Because, the AR-ELM has two levels of tensor decomposition such as TUCKER decomposition and low rank iterative approximation.

The testing performance of the shuttle dataset is shown in the Table 8. The AR-ELM method has improved performance in terms of accuracy,

Table 9. Training time comparison for Letter Recognition dataset

Activation function	Basic ELM [21]	ELM-TUCKER [21]	ELM-PARAFAC [21]	AR-ELM
Tribas	2502.307	2413.627	2210.872	2186.56
Tanh	2507.960	2401.842	2198.813	2095.27
Sigmoid	2504.618	2478.125	2173.671	1956.23
Hardlims	2501.627	2465.783	2178.354	1945.35

Table 10. Comparison of testing performance for Letter Recognition dataset

Activation function	Basic ELM [21]			ELM-TUCKER [21]			ELM-PARAFAC [21]			AR-ELM		
	ACC	P	R	ACC	P	R	ACC	P	R	ACC	P	R
Tribas	0.6857	0.68	0.71	0.7832	0.78	0.80	0.8012	0.80	0.79	0.8572	0.85	0.82
Tanh	0.8590	0.85	0.73	0.9023	0.91	0.90	0.9254	0.92	0.91	0.9425	0.94	0.94
Sigmoid	0.8830	0.85	0.86	0.9034	0.90	0.92	0.9251	0.92	0.91	0.9456	0.93	0.93
Hardlims	0.8102	0.80	0.79	0.8375	0.83	0.83	0.834	0.84	0.83	0.8879	0.87	0.89

Table 11. Training time comparison for Mushroom dataset

Activation function	Basic ELM [21]	ELM-TUCKER [21]	ELM-PARAFAC [21]	AR-ELM
Tribas	412.883	388.762	385.128	356.48
Tanh	414.840	386.351	362.901	347.15
Sigmoid	413.797	381.147	376.561	338.451
Hardlims	412.699	382.347	378.903	319.58

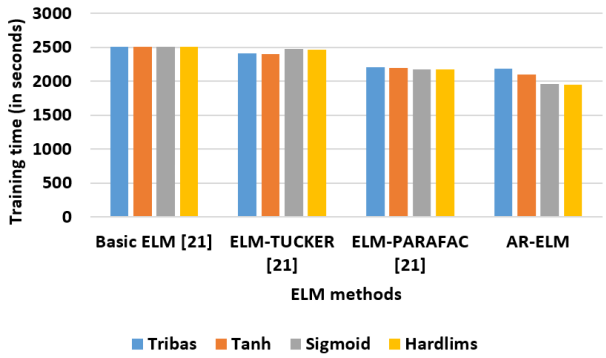

Figure.7 Comparative analysis of training time for Letter Recognition dataset

precision and recall than the other methods basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. The elimination of redundant information from the decomposed tensor samples is leads to increase the testing accuracy.

4.2.5. Performance analysis of Letter Recognition dataset

The following Table 9 and Fig 7 provides the comparative analysis of the Letter Recognition dataset for the AR-ELM with existing ELM methods such as basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. This Letter Recognition dataset is analysed in four different datasets such as tanh, tribas, hardlims and sigmoid.

The Table 9 and Fig 7 shows that the AR-ELM training time for the Letter Recognition dataset is less when compared to the existing methods basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. The training time of the basic ELM [21] is high than the other methods. Because, the basic ELM takes more time for processing the huge datasets. Besides, the AR-ELM training time is lesser than the ELM-TUCKER [21] and ELM-PARAFAC [21], due to its fine tuning and higher level of decomposition over the tensors.

The testing performance of the Letter Recognition dataset is shown in the Table 10. The AR-ELM method has improved performance in terms of accuracy, precision and recall than the other methods such basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. The learning rate of the AR-ELM improves the testing accuracy during the recognition.

4.2.6. Performance analysis of Mushroom dataset

The Table 11 and Fig 8 provides comparison of Mushroom dataset training time (in seconds) four different datasets such as tanh, tribas, hardlims and sigmoid. This training time comparison is made for AR-ELM with three different existing methodologies such as basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21].

Table 12. Comparison of testing performance for Mushroom dataset

Activation function	Basic ELM [21]			ELM-TUCKER [21]			ELM-PARAFAC [21]			AR-ELM		
	ACC	P	R	ACC	P	R	ACC	P	R	ACC	P	R
Tribas	0.9621	0.96	0.96	0.9712	0.97	0.98	0.9787	0.97	0.97	0.9813	0.98	0.99
Tanh	0.9676	0.97	0.96	0.9812	0.97	0.98	0.9946	0.99	0.98	0.9956	0.99	0.99
Sigmoid	0.9619	0.95	0.96	0.9686	0.96	0.97	0.9782	0.98	0.97	0.9867	0.99	0.98
Hardlims	0.9629	0.96	0.96	0.9676	0.96	0.97	0.9782	0.98	0.97	0.9789	0.99	0.99

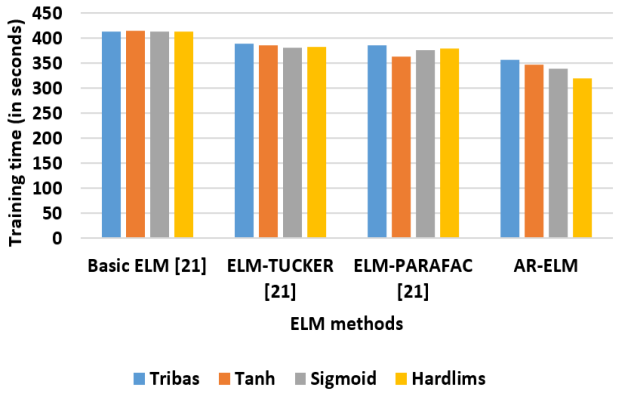

Figure.8 Comparative analysis of training time for Mushroom dataset

Table 11 and Fig 8 conclude that the training time of the Mushroom dataset is less when compared to the existing methods such as basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. Due to the high amount of samples of the mushroom dataset, the training process of the basic ELM [21] takes more time. Similarly, the training time of the ELM-TUCKER [21] and ELM-PARAFAC [21] also higher than the AR-ELM.

The testing performance of the Mushroom dataset is shown in Table 12. The AR-ELM method has improved performance in terms of accuracy, precision and recall than the other methods basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. Generally, the mushroom dataset has high amount of training samples as 8124 with 22 attributes. Hence, the testing accuracy is enhanced due to the determination of the optimal learning rate.

The proposed AR-ELM was analysed in six different datasets. From the analysis, concluded that the AR-ELM gives better performance than the basic ELM [21], ELM-TUCKER [21] and ELM-PARAFAC [21]. Because, the existing ELM approach performs only one level of tensor decomposition to decrease the tensor information. But in this AR-ELM better performance is achieved by using two different stages. At first, the tensor was decomposed by using the HOSVD based TUCKER. Then the Bayesian approach and constant compression rate used at second stage of AR-ELM. The Bayesian approach was used for removing the redundant information from the tensor and constant compression rate was used to achieve parameter reduction rate.

5. Conclusion

In this paper, an AR-ELM is introduced for a better reduction in training time and improvement in testing accuracy. This AR-ELM has two different stages for obtaining better tensor decomposition and eliminating the redundant information from the decomposed samples. The HOSVD based TUCKER decomposition is utilized in first stage to decompose the tensor. In the second stage, the automotive rank selection is used to eliminate the redundancy from the decomposed samples by using the Bayesian approach. Besides, output weights from the constant compression rate is utilized for better training of the ELM. The proposed AR-ELM is analyzed by using six different datasets such as MNIST handwritten dataset, KDD Cup 1999 dataset, Satellite dataset, Shuttle dataset, Letter Recognition dataset and Mushroom dataset. The AR-LVM is compared with some of the existing methodologies such as basic ELM, ELM-TUCKER and ELM-PARAFAC. The AR-LVM shows better performance when compared to the existing methodologies. The training time of the AR-ELM methodology with Tribas activation function is 356.48 for mushroom dataset, it is less than the basic ELM, ELM-TUCKER and ELM-PARAFAC that are 412.883, 388.762 and 385.128 respectively. In the future, the training time and testing accuracy can be improved by increasing the variety of matrix/tensor decompositions in the compression step.

References

[1] X. Tang and L. Chen, “Artificial bee colony optimization-based weighted extreme learning machine for imbalanced data learning”, *Cluster Computing*, pp.1-16, 2018.

[2] Chaturvedi, E. Ragusa, P. Gastaldo, R. Zunino, and E. Cambria, “Bayesian network based extreme learning machine for subjectivity

detection", *Journal of The Franklin Institute,* Vol.355, No.4, pp.1780-1797, 2018.

[3] A. Samat, P. Gamba, P. Du, and J. Luo, "Active extreme learning machines for quad-polarimetric SAR imagery classification", *International Journal of Applied Earth Observation and Geoinformation,* Vol.35, pp.305-319, 2015.

[4] N. Liu and H. Wang, "Ensemble based extreme learning machine", *IEEE Signal Processing Letters,* Vol.17, No.8, pp.754-757, 2010.

[5] R. Taormina and K.W. Chau, "Data-driven input variable selection for rainfall–runoff modeling using binary-coded particle swarm optimization and Extreme Learning Machines", *Journal of Hydrology,* Vol.529, pp.1617-1632, 2015.

[6] G. Huang, S. Song, J. N. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines", *IEEE Transactions on Cybernetics,* Vol.44, No.12, pp.2405-2417, 2014.

[7] S.X. Xia, F.R. Meng, B. Liu, and Y. Zhou, "A kernel clustering-based possibilistic fuzzy extreme learning machine for class imbalance learning", *Cognitive Computation,* Vol.7, No.1, pp.74-85, 2015.

[8] Z. Liouane, T. Lemlouma, P. Roose, F. Weis, and H. Messaoud, "An improved extreme learning machine model for the prediction of human scenarios in smart homes", *Applied Intelligence*, Vol. 48, No.8, pp.2017-2030, 2018.

[9] Q. Shen, X. Ban, R. Liu, and Y. Wang, "Decay-weighted extreme learning machine for balance and optimization learning", *Machine Vision and Applications,* Vol.28, No.7, pp.743-753, 2017.

[10] X. Tang and L. Chen, "A self-adaptive evolutionary weighted extreme learning machine for binary imbalance learning", *Progress in Artificial Intelligence,* Vol.7, No. 2, pp.95-118, 2018.

[11] H. Liu, F. Li, X. Xu, and F. Sun, "Active object recognition using hierarchical local-receptive-field-based extreme learning machine", *Memetic Computing,* Vol.10, No.2, pp.233-241, 2018.

[12] J. Cao, Y. Zhao, X. Lai, M. E. H. Ong, C. Yin, Z. X. Koh, and N. Liu, "Landmark recognition with sparse representation classification and extreme learning machine", *Journal of the Franklin Institute,* Vol.352, No.10, pp.4528-4545, 2015.

[13] X. Kang, Y. Zhao, and J. Li, "Predicting refractive index of ionic liquids based on the extreme learning machine (ELM) intelligence algorithm", *Journal of Molecular Liquids*, Vol. 250, pp.44-49, 2018.

[14] Y. Song and J. Zhang, "Discriminating preictal and interictal brain states in intracranial EEG by sample entropy and extreme learning machine", *Journal of neuroscience methods*, Vol.257, pp.45-54, 2016.

[15] W. Ibrahim and M.S. Abadeh, "Extracting features from protein sequences to improve deep extreme learning machine for protein fold recognition", *Journal of theoretical biology*, Vol.421, pp.1-15, 2017.

[16] S. Roshan, Y. Miche, A. Akusok, and A. Lendasse, "Adaptive and online network intrusion detection system using clustering and Extreme Learning Machines", *Journal of the Franklin Institute,* Vol.355, No.4, pp.1752-1779, 2018.

[17] X. Su, S. Zhang, Y. Yin, and W. Xiao, "Prediction model of permeability index for blast furnace based on the improved multi-layer extreme learning machine and wavelet transform", *Journal of the Franklin Institute,* Vol.355, No.4, pp.1663-1691, 2018.

[18] A. Mishra, A. Rajpal, and R. Bala, "Bi-directional extreme learning machine for semi-blind watermarking of compressed images", *Journal of Information Security and Applications,* Vol.38, pp.71-84, 2018.

[19] J. Tang, C. Deng, and G. B. Huang, "Extreme learning machine for multilayer perceptron", *IEEE Transactions on Neural Networks and Learning Systems,* Vol.27, No.4, pp.809-821, 2015.

[20] X. Li, W. Mao, and W. Jiang, "Multiple-kernel-learning-based extreme learning machine for classification design", *Neural Computing and Applications,* Vol.27, No.1, pp.175-184, 2016.

[21] N.K. Nair, and S. Asharaf, "Tensor Decomposition Based Approach for Training Extreme Learning Machines", *Big Data Research,* Vol.10, pp.8-20, 2017.