Applied Soft Computing 122 (2022) 108861

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Multilayer extreme learning machines and their modeling performance on dynamical systems

Gizem Atac Kale, Cihan Karakuzu *

Bilecik Seyh Edebali University, Turkey

ARTICLE INFO

Article history:
Received 17 September 2020
Received in revised form 6 April 2022
Accepted 13 April 2022
Available online 21 April 2022

Keywords:
Extreme learning machines (ELM)
Multilayer ELM
Improved Multilayer ELM
System modeling
Dynamic systems

ABSTRACT

In this paper, two novel Multilayer Extreme Learning Machine (ML-ELM) networks are presented. We call them Improved Multilayer Extreme Learning Machines (IML-ELM). The proposed network architectures use neuron activations both during and after the training. In the first IML-ELM (IML-ELM1) network, each layer has connection weights assigned randomly as orthonormal. On the other hand, the second IML-ELM (IML-ELM2) has connection weights assigned randomly as orthonormal only in the first layer. Its following layers' connection weights are taken from the previous layer's output weight matrix. This assignment strategy made in the IML-ELM2 decreases the computation time even more. The networks' modeling performances on seven benchmark dynamic systems are investigated and it is shown that the proposed IML-ELM1 and IML-ELM2 perform better modeling than the ML-ELM. They have better modeling performance of more than 70% for both training and test data sets compared to ML-ELM for some systems studied. For instance, using 100 nodes, ML-ELM, IML-ELM1 and IML-ELM2 gave average testing root mean square error results of 0.627977, 0.104272 (83%) and 0.092683 (85%) respectively for BDS 7. In addition, it has been experimentally determined that the developed networks provide improvements in terms of average training time, and this improvement exceeds 60% in some cases. These achievements clearly prove that the proposed improved multilayer extreme learning machines are efficient tools for system modeling applications.

Code metadata

Permanent link to reproducible Capsule: https://doi.org/10.24433/CO.0688672.v1.

1. Introduction

Neural networks are widely used for solution of many engineering and scientific problems and are popular general modeling and calculation methods. These structures are usually trained by iterative algorithms. In order to overcome slowness of iterative algorithms, the concept of Extreme Learning Machine (ELM) has brought significant development [1]. Basically, the ELM is a data-driven learning algorithm for a Single Layer Feedforward Neural Network (SLFN) introduced by Huang et al. It is getting to be a widely used method in machine learning and deep learning applications. Another noteworthy development is about network architectures. In this context, structures such as Meta-ELM, which consist of the association of SLFNs, and transition from single layer to multilayered structures are remarkable [2]. The original ELM and its variants mainly focus on classification problems. Since feature learning is often required before classification in many applications, multilayer solutions are usually needed [3]. Kasun et al. [4] defined ELM-based autoencoder concept and developed a multilayer representational learning architecture. This was the first Multilayer ELM (ML-ELM) structure. In this architecture, external inputs are applied to the first layer, and its outputs are used as inputs of the next layer. In this flow, each layer is called as Autoencoder (AE). By means of AE, a hierarchical stacked knowledge of the inputs is passed to the last layer by encoding or processing. At the last layer the supervised least mean square optimization is operated. The architecture that allows this hierarchical information flow is presented in [4] with a more complete training algorithm under the name ML-ELM.

Due to the widespread use of multilayer ELMs and the advantage it offers, it has been observed that ELM structures are also used in deep learning recently. When looking at deep ELM structures in general, it will be seen that ML-ELM structures are used. Although the word "deep" is preferred, the structure used is generally not different from the structures mentioned above. It

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail addresses: gzm.atac@gmail.com (G.A. Kale), cihan.karakuzu@bilecik.edu.tr (C. Karakuzu).

https://doi.org/10.1016/j.asoc.2022.108861
1568-4946/

Nomenclature

Acronyms

ELM	Extreme Learning Machine
ML-ELM	Multilayer Extreme Learning Machine
IML-ELM	Improved Multilayer Extreme Learning Machine
BDS	Benchmark Dynamic Systems
RMSE	Root Mean Square Error
AE	Autoencoder
SLFN	Single Layer Feedforward Neural Network
ANFIS	Adaptive Neuro Fuzzy Inference System
CNN	Convolutional Neural Network

Variables

L	Number of nodes or neurons
P	Number of samples
M	Number of layers
N	Number of external inputs
q	Number of outputs
u	External control signal
g	Neuron activation function
β_i	Output weights for $i = 1, \ldots, L$
$\mathbf{w}_i$	Input weights for $i = 1, \ldots, L$
$\mathbf{x}_j$	Inputs for $j = 1, \ldots, P$
b_i	Bias for $i = 1, \ldots, L$
y	System output
$\mathbf{O}$	The output matrix
$\mathbf{o}_j$	Elements of the output matrix $\mathbf{O}$ for $j = 1, \ldots, P$
$\mathbf{T}$	The training data target matrix
$\mathbf{t}_j$	Elements of the training data target matrix $\mathbf{T}$ for $j = 1, \ldots, P$
$\mathbf{H}$	The output matrix of the layer
λ	Regulation factor

can be easily understood that the phrase "deep ELM" is preferred when more layers and/or cells are used with big data. In this context, several important research was made in this field in the near past. Vidnerova and Neruda designed deep networks with RBF layers [5]. Vong et al. made another research on representational learning using empirical kernel map based ML-ELM [6]. Chen et al. came up with a human activity recognition scheme for smart health using ML-ELM [7]. Wong et al. also worked on the representational learning using kernel based ML-ELM [8]. Kim et al. developed a fast learning method for Convolutional Neural Networks (CNN) using ELM and its application to lane detection [9]. Yousefi-Azar and McDonnell designed a semi-supervised convolutional extreme learning machine [10]. Altan and Kutlu studied in the area of deep learning where they have developed a Hessenberg ELM autoencoder [11]. Tissera and McDonnell designed a deep extreme learning machines as supervised autoencoding architecture for classification [12]. Zhang et al. came up with a non-iterative and a fast deep learning scheme using ML-ELM [13]. Ezzati Khatab et al. developed an autoencoder based semi-supervised deep extreme learning machine [14].

In order to combine the advantages of fuzzy logic and ELM learning concepts, fuzzy network structures in which ELM learning is adapted have also been studied in the recent past. For example; Golestaneh et al. studied in the area of fuzzy wavelet ELM [15]. Jagtap developed and ELM technique using Adaptive Network Fuzzy Inference System (ANFIS) [16]. Rong et al. also designed a fuzzy ELM method [17]. Zhang and Ji developed another fuzzy ELM technique [18]. On the other hand, studies on approaches in which the concept of fuzzy logic predominantly continue in the field of system modeling. While ELM was considered as a new machine learning tool with classical ones in [19], a network using the fuzzy logic-based clustering approach called DENFIS [20] was developed and used in modeling by a group of authors. Similarly eTS [21], FBeM [22], TEDA [23], and PANFIS [24] structures were developed. Also an economic Long Short-Term Memory (LSTM) approach for Recurrent Neural Networks (RNN) can be added as a good achievement [25].

In the ML-ELM, there are training and testing phases which have different structures. Also, ML-ELM architecture has autoencoder structure. Because of its feature extraction capabilities, it is widely used on classification problems. Our motivation is to use this fast learning architecture on system modeling applications by replacing the autoencoders with activation functions.

In this research, with the motivation given above, we introduce two Improved ML-ELM (IML-ELM) network architectures in order to use identical structures for training/testing phases and to speed up the computation time. In the first IML-ELM (IML-ELM1) structure, we have changed the representational input calculation method. In this scheme, training and testing phases use identical structures. Here, the representational input of a layer is calculated in a manner where the previous layer's $\mathbf{H}$ matrix is multiplied with the output layer weight parameters of the previous layer.

In the traditional ML-ELM and the improved IML-ELM1, weight and bias parameters for each layer are assigned randomly. In the second IML-ELM (IML-ELM2) structure, we have changed weight parameter assignment strategy. In contrast to the traditional ML-ELM and IML-ELM1, in our IML-ELM2 scheme, the input weight parameters are assigned randomly only in the first layer. As the input connection weight parameters, other layers take the transpose of the output weight parameters obtained in the previous layer. Like IML-ELM1 scheme, the IML-ELM2 has identical training and testing structures.

In this study, performance of IML-ELMs are investigated on the problem of highly nonlinear dynamic system identification. For this purpose, benchmark problems selected from the literature are used. For each system, a training and a test data sets were prepared to use in this investigation. Using training sets, IML-ELMs and their counterpart identification models have been obtained with the related training procedure for each benchmark system. And then, a generalization performance of each obtained models has been examined using test data sets. Many experiments have been done by repeating the just mentioned training and testing transactions with different parameters to obtain general statistical performance panorama of modeling approximations. The results have shown that as faster methods, the IML-ELM1 and IML-ELM2 perform better than the ML-ELM.

In addition to the design of two IML-ELM architectures, it can be stated that, the usage of ELM on system modeling applications is rare and this research fully focuses on system modeling applications. Hence, the novelty of this research is the design of two IML-ELM architectures and their performance comparison with the existing ML-ELM on Benchmark Dynamic System (BDS) modeling applications.

The paper starts with the definition of traditional ELM. Following traditional ELM, ML-ELM is described. In Section 4, the proposed IML-ELM structures are introduced. In this section, the authors describe what they contribute. Section 5 describes the dynamic systems used in this study. Section 6 gives the test results which were trained on 7 different BDSs. Section 7 gives related

discussion and finally Section 8 concludes the paper. Source codes of ML-ELM, IML-ELM1 and IML-ELM2 are publicly available on Code Ocean.[1]

2. Extreme Learning Machine (ELM)

The ELM is based on the SLFN whose input connection weight parameters (w) are assigned randomly and they are not updated. On the other hand, output weight parameters (β) between the hidden layer and output layer are determined using a linear model in an analytical and fast way. The mathematical structure of the ELM for P distinct samples $(\mathbf{x}_j, \mathbf{o}_j)$ with $\mathbf{x}_j = [x_{j1}, \ldots, x_{jN}]^T$ and $\mathbf{o}_j = [o_{j1}, \ldots, o_{jq}]^T$ is shown in Eq. (1).

$$\sum_{i=1}^{L} \beta_i g(\mathbf{w}_i \mathbf{x}_j + b_i) = \mathbf{o}_j, j = 1, \ldots, P \tag{1}$$

In this model, β_i are the output weights, $g(.)$ is neuron activation function, $\mathbf{w}_i$ are the input weights, and $\mathbf{x}_j$ are the inputs for $i = 1, \ldots, L$ and $j = 1, \ldots, P$. Here L is the number of nodes or neurons. Traditionally in SLFN, β_i weights have to be updated and computed. However, Huang et al. devised the ELM technique that would compute these weight parameters at once and they have proven that their solution is unique [1].

In the case of $L \leq P$, the P samples can be approximated with zero error, in other words $\sum_{j=1}^{P} \|\mathbf{t}_j - \mathbf{o}_j\| = 0$ or $\mathbf{o}_j$ is approximated to the training data target $\mathbf{t}_j$. In this case, there exists β_i, $\mathbf{w}_i$, $\mathbf{x}_j$, b_i for $i = 1, \ldots, L$ and $j = 1, \ldots, P$ where Eq. (2) is proven.

$$\sum_{i=1}^{L} \beta_i g(\mathbf{w}_i \mathbf{x}_j + b_i) = \mathbf{t}_j, j = 1, \ldots, P \tag{2}$$

In Eq. (2), as mentioned before, $\mathbf{t}_j$ stands for the elements of the training data target matrix $\mathbf{T}$. The P equations described in Eq. (2) can be summarized as

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{3}$$

where

$$\mathbf{H} = \begin{bmatrix} g(\mathbf{w}_1.\mathbf{x}_1 + b_1) & \ldots & g(\mathbf{w}_L.\mathbf{x}_1 + b_L) \\ \vdots & & \vdots \\ g(\mathbf{w}_1.\mathbf{x}_P + b_1) & \ldots & g(\mathbf{w}_L.\mathbf{x}_P + b_L) \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_L \end{bmatrix}, \mathbf{T} = \begin{bmatrix} \mathbf{t}_1 \\ \vdots \\ \mathbf{t}_P \end{bmatrix}, \tag{4}$$

In Eq. (4), $\mathbf{H}$ stands for the output matrix of the hidden layer. The output of the ith hidden node is the ith column of the matrix $\mathbf{H}$. Here, $\mathbf{T}$ is the training data target matrix.

The Moore–Penrose inverse of the matrix $\mathbf{H}$ is calculated (which is $(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$ if $\mathbf{H}^T\mathbf{H}$ is nonsingular) and assigned as $\mathbf{H}^\dagger$. Following the computation of $\mathbf{H}^\dagger$, ELM learns the $\boldsymbol{\beta}$ using Eq. (5).

$$\boldsymbol{\beta} = \mathbf{H}^\dagger \mathbf{T} \tag{5}$$

The output matrix $\mathbf{O}$ is calculated using Eq. (6)

$$\mathbf{O} = \mathbf{H}\boldsymbol{\beta} \tag{6}$$

In this manner, the ELM works. In the next section, the traditional ML-ELM is described.

[1] https://doi.org/10.24433/CO.0688672.v1.

3. Multilayer Extreme Learning Machine (ML-ELM)

This section is the brief review of the ML-ELM technique. This time, instead of a single layer ELM, there are multiple layers. Except the last layer of the architecture, all layers are autoencoders. The Moore–Penrose inverse of $\mathbf{H}$ is $\mathbf{H}^\dagger = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$ when $\mathbf{H}^T\mathbf{H}$ is nonsingular, as mentioned before. The Moore–Penrose inverse is also defined as $\mathbf{H}^\dagger = \mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1}$ if $\mathbf{H}\mathbf{H}^T$ is nonsingular. Using the ridge regression theory, $(1/\lambda)$ values are added to the diagonal elements of $\mathbf{H}\mathbf{H}^T$. Hence, for a single layer, the learning equation of an ML-ELM becomes

$$\boldsymbol{\beta} = \mathbf{H}^T \left(\frac{\mathbf{I}}{\lambda} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T} \tag{7}$$

In the ML-ELM, there are input representations for each hidden layer k for $k = 2, \ldots, M$. The input representation of the kth layer is shown as $\mathbf{X}^{(k)} = [\mathbf{x}_1^{(k)}, \ldots, \mathbf{x}_N^{(k)}]$ for N external inputs. In the scheme, $\boldsymbol{\beta}^{(k)} = [\boldsymbol{\beta}_1^{(k)}, \ldots, \boldsymbol{\beta}_L^{(k)}]$ is the transformation matrix for the kth layer with $\boldsymbol{\beta}_j^{(k)}$ is the transformation vector for representation learning for $j = 1, \ldots, L$. In this manner, the calculation of the input representation becomes

$$\mathbf{X}^{(k)} = \mathbf{H}^{(k)}\boldsymbol{\beta}^{(k)} \tag{8}$$

Using Eq. (8), the $\boldsymbol{\beta}^{(k)}$ is learned through Eq. (9) [3].

$$\boldsymbol{\beta}^{(k)} = \left(\mathbf{H}^{(k)}\right)^T \left(\frac{\mathbf{I}}{\lambda} + \mathbf{H}^{(k)} \left(\mathbf{H}^{(k)}\right)^T \right)^{-1} \mathbf{X}^{(k)} \tag{9}$$

For M layers, the final layer output matrix is described as $\mathbf{H}^{(M)}$. Using this layer output matrix, the last layer's output weight matrix $\boldsymbol{\beta}$ is learned by

$$\boldsymbol{\beta} = \left(\mathbf{H}^{(M)}\right)^T \left(\frac{\mathbf{I}}{\lambda} + \mathbf{H}^{(M)} \left(\mathbf{H}^{(M)}\right)^T \right)^{-1} \mathbf{T} \tag{10}$$

ML-ELM has two phases which are training and testing. The flow chart of the ML-ELM training and testing structures are shown in Fig. 1. Note that the training and testing structures are different regarding the ML-ELM case. In the ML-ELM structure, the $\boldsymbol{\beta}$ values that are learned in the training phase must be stored in order to be used in the testing phase. Finally, the weight parameters of the last layer of the training phase are used in the testing phase.

This ends the description of the traditional ML-ELM. In the next section, two new structures for Improved Multilayer Extreme Learning Machine (IML-ELM) are introduced for the first time.

4. Two proposed Improved Multilayer Extreme Learning Machines (IML-elms)

In this section, two IML-ELM structures are designed by the authors in a way to improve the modeling performance and to decrease the computation time of the traditional ML-ELM. Another motivation to develop these IML-ELM schemes can be summarized as to use the same structure for training and testing phases.

In the structure of the first IML-ELM (IML-ELM1), the representational input is calculated by multiplying the previous layer's output matrix $\mathbf{H}$ with the previous layer's output weight parameters. Whereas the ML-ELM uses two different structures for training and testing phases, IML-ELM1 uses the same structure for training and testing phases. The structure used for training and testing phases of IML-ELM1 is shown in Fig. 2.

The contribution of IML-ELM1 over ML-ELM is that it uses the same structure shown in Fig. 2 for training and testing phases. In the structure of IML-ELM1, it must be noted that, for each

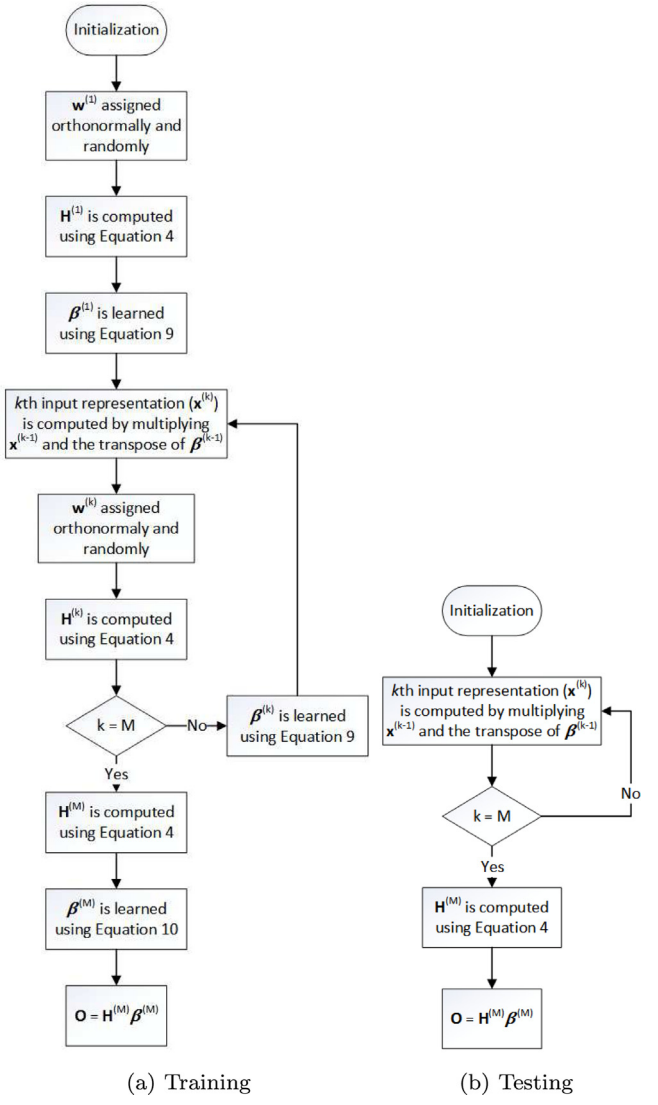


Fig. 1. ML-ELM scheme for M hidden layers.

layer, weight parameters **w** and biases are assigned randomly as orthonormal and this situation led us for a further advancement which was achieved in the other IML-ELM (namely IML-ELM2) introduced in this paper.

By calculating the input representation of the current layer and using the same structure for training and testing phases, IML-ELM1 decreases the computation time and simplifies the structure. The algorithm of the IML-ELM1 (for training phase) is shown in Algorithm 1.

For testing phase, the same structure is used. The only difference of the testing phase is that it only uses the input weight parameters **w**, biases and output weight parameters $\boldsymbol{\beta}$ for forward computation in the training phase.

As can be understood from Algorithm 1; for the learning phase, computation is the same as with that of the ML-ELM. But usage after learning is different from it. IML-ELM1 uses neurons or nodes after learning. ML-ELM uses a transformation matrix founded at the learning phase.

The structure of the second IML-ELM (IML-ELM2) uses IML-ELM1 as a basis. IML-ELM2 also uses the same structure for training and testing phases. In the structure of IML-ELM2, at the first layer, the input connection weight parameters are randomly assigned with the constraint that they are orthonormal to each other. At the following layers, the weight parameter matrix of a layer is the transpose of the previous layer's output weight parameter matrix. In this manner, computation time is decreased even more since there is no random assignment operations for the weight parameters. The network architecture is shown in Fig. 3.

Algorithm 1: Algorithm of the proposed IML-ELM1 (for training phase).

Initialization;
For the first layer, weight parameters $\mathbf{w}^{(1)}$ and $Bias^{(1)}$ are assigned randomly.
Compute $\mathbf{H}^{(1)}$ using Equation 4.
Then, output weight parameter $\boldsymbol{\beta}^{(1)}$ is learned using Equation (9).
for k=2:M **do**
  Current layer' s input representation is computed by multiplying the previous layer' s output matrix $\mathbf{H}^{(k-1)}$ with the previous layer' s output weight parameters $\boldsymbol{\beta}^{(k-1)}$.
  For the current layer, weight parameters $\mathbf{w}^{(k)}$ and $Bias^{(k)}$ are assigned randomly.
  Compute $\mathbf{H}^{(k)}$ using Equation 4.
  Then, output weight parameter $\boldsymbol{\beta}^{(k)}$ is learned using Equation (9) (If k = M using Equation (10)).
end

Table 1
Benchmark dynamic systems used for system identification with IML-ELM.

Number	Benchmark dynamic systems
1	$y(k)=\frac{y(k-1)y(k-2)(y(k-1)+2.5)}{1+y^2(k-1)+y^2(k-2)}+u(k)$, [26] $u(k)_{train}=cos\frac{2\pi k}{100}$, $u(k)_{test}=sin\frac{2\pi k}{25}$
2	$y(k+1)=\frac{y(k)}{y^2(k)+1}+1+u^3$, [26] $u(k)_{train}=cos(2\pi k/100)$, $u(k)_{test}=sin(2\pi k/25)$
3	$y(k+1)=y(k)+u(k)e^{-3\lvert y(k)\rvert}$, [27] $u(k)_{train}=\underset{\text{in } [-1,1]}{\texttt{Random amplitude}}$, $u(k)_{test}=\underset{\text{in } [-1,1]}{\texttt{Random amplitude}}$
4	$y(k+1)=\frac{24+y(k)}{30}y(k)-0.8\frac{u^2(k)}{1+u^2(k)}y(k-1)+0.5u(k)$, [28] $u(k)_{train}=\underset{\text{in } [-5,5]}{\texttt{Random amplitude}}$, $u(k)_{test}=\underset{\text{in } [-5,5]}{\texttt{Random amplitude}}$
5	$y(k+1)=0.5\left(\frac{y(k)}{1+y^2(k)}+(1+u(k))\,u(k)\,(1-u(k))\right)$, [29] $u(k)_{train}=\underset{\text{in } [-2,2]}{\texttt{Random amplitude}}$, $u(k)_{test}=\underset{\text{in } [-2,2]}{\texttt{Random amplitude}}$
6	$y(k+1)=\frac{y(k)y(k+1)y(k-2)u(k-1)(y(k-2)-1)+u(k)}{1+y(k-1)^2+y(k-2)^2}$, [30] $u(k)_{train}=\begin{cases}sin\frac{\pi k}{25} & ,k<250\\ 1 & ,250\le k<500\\ -1 & ,500\le k<750\\ 0.3sin\frac{\pi k}{25}+0.1sin\frac{\pi k}{32}+0.6sin\frac{\pi k}{10} & ,k\ge 750\end{cases}$ $u(k)_{test}=\begin{cases}-0.15cos\frac{\pi k}{50}+0.5cos\frac{\pi k}{16}-0.3cos\frac{\pi k}{20} & ,k<250\\ -0.65 & ,250\le k<500\\ 0.35 & ,500\le k<750\\ cos\frac{\pi k}{50} & ,k\ge 750\end{cases}$
7	$y(k+1)=0.3y(k)+0.6y(k-1)+f(k)$ $f(k)=sin\pi u(k)+0.3sin3\pi u(k)+0.1sin5\pi u(k)$, [31] $u(k)_{train}=\begin{cases}sin\frac{2\pi k}{250} & ,k<500\\ 0.5sin\frac{2\pi k}{250}+0.5sin\frac{2\pi k}{25} & ,k\ge 500\end{cases}$ $u(k)_{test}=\begin{cases}-0.15cos\frac{\pi k}{50}+0.5sin\frac{\pi k}{16}-0.3cos\frac{\pi k}{20} & ,k<234\\ cos\frac{2\pi k}{125} & ,234\le k<467\\ 0.3sin\frac{2\pi k}{250}+0.1cos\frac{\pi k}{64}+0.6sin\frac{\pi k}{20} & ,k\ge 467\end{cases}$

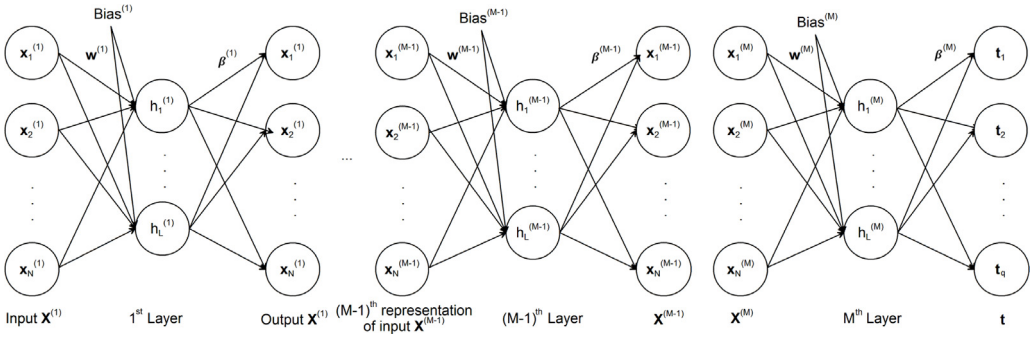


Fig. 2. The IML-ELM1 scheme for M hidden layers.

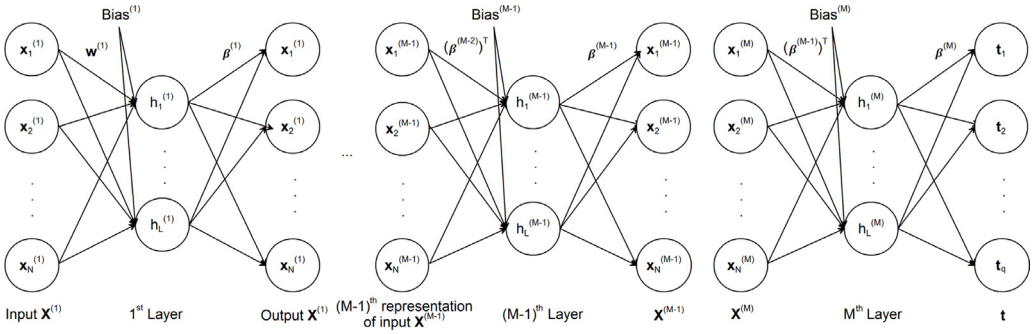


Fig. 3. The IML-ELM2 scheme for M hidden layers.

As mentioned in the above paragraph, the modification makes the structure simpler. There are no repetitive random assignments. Random assignment is only made at the first layer. Excluding the final layer's output weight parameter, each output weight parameter is used twice, once as an output weight parameter and once as an input weight parameter at the next layer. The algorithm of the IML-ELM2 is provided in Algorithm 2.

Algorithm 2: Algorithm of the proposed IML-ELM2 (for training phase).

Initialization;
For the first layer, weight parameters $\mathbf{w}^{(1)}$ and $Bias^{(1)}$ are assigned randomly.
Compute $\mathbf{H}^{(1)}$ using Equation 4.
Then, output weight parameter $\boldsymbol{\beta}^{(1)}$ is learned using Equation (9).
for *k=2:M* **do**
  Current layer' s input representation is computed by multiplying the previous layer' s output matrix $\mathbf{H}^{(k-1)}$ with the previous layer' s output weight parameters $\boldsymbol{\beta}^{(k-1)}$.
  Input connection weight parameters ($\mathbf{w}^{(k)}$) are picked as the transpose of the previous layer' s output weight parameters $(\boldsymbol{\beta}^{(k-1)})^T$. $Bias^{(k)}$ are assigned randomly.
  Compute $\mathbf{H}^{(k)}$ using Equation 4.
  $\boldsymbol{\beta}^{(k)}$ is learned using Equation (9) (If k = M Equation (10)).
end

Again, in IML-ELM2 architecture, the testing phase uses the same structure. Only difference of the testing phase is that the input weight parameters $\mathbf{w}$ and output weight parameters $\boldsymbol{\beta}$ computed in the training phase are used.

It is worth mentioning again: As with IML-ELM1, IML-ELM2 uses neurons both in training and after training to calculate. With this feature, they are separated from the ML-ELM and have gained a more adaptive calculation ability by not using the representative matrix obtained during training as in ML-ELM. A better depiction of the algorithms is shown in the flow charts displayed in Fig. 4. Here in our IML-ELM schemes, training and testing structures are the same for each scheme (except the input weight parameters $\mathbf{w}$ and output weight parameters $\boldsymbol{\beta}$ computed in the training phase are used in the testing phase), hence we have two structures for two schemes.

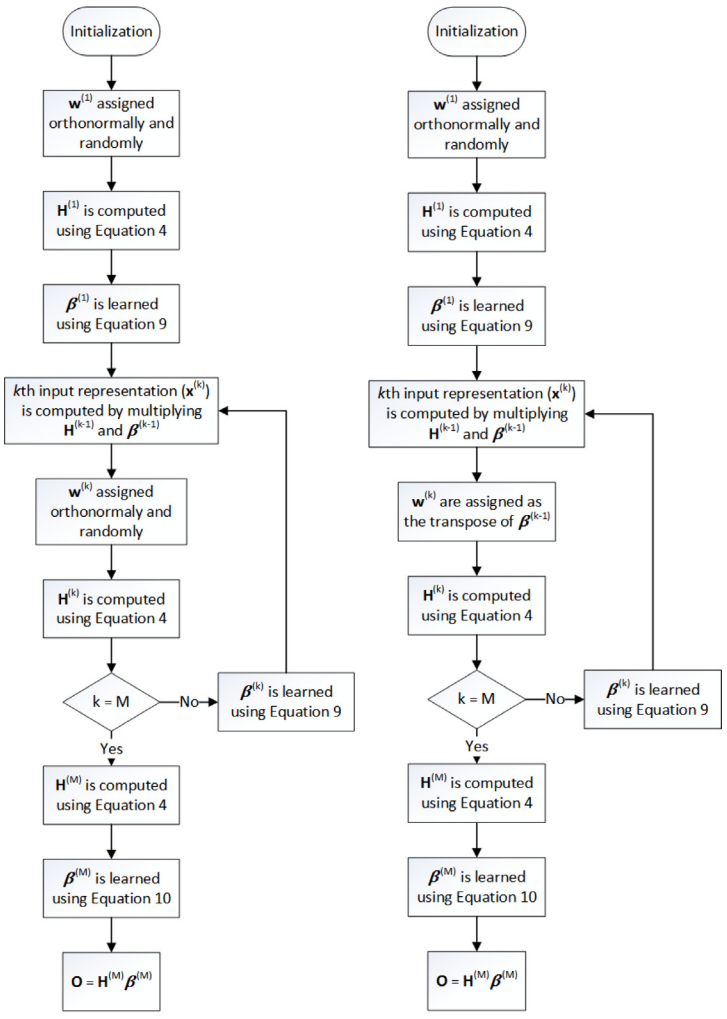
Limitations of these IML-ELM architectures: In each layer, the number of neurons are equal to each other. Also, the user picks the number of neurons and type of the activation function.

As can be seen in this section, the IML-ELMs are feasible, they use the same structure for training and testing phases, they are faster (especially IML-ELM2 which uses the transpose of the previous output parameters as input weight parameters), and the results will show that they perform better on dynamic system modeling applications. Finally, it must be underlined that IML-ELMs use activation functions instead of autoencoders in order to achieve better system modeling performance.

5. System modeling problems

Benchmark Dynamic Systems (BDS) used for system modeling are listed in Table 1. In this research, modeling performance of ML-ELM, IML-ELM1 and IML-ELM2 were tested on seven different BDS.

In these benchmark dynamic systems, $u(k)$ is the external control signal. For each BDS, $u(k)$ is used to prepare the training and testing data sets. The dynamic system modeling experiments on BDS 1, BDS 2, …, BDS 5 use 100 samples for training and 100 samples for testing. In this study, the same training and testing data sets in [32] are used. The experiment on BDS 6 uses 1000 samples for training and 1000 samples for testing.



(a) IML-ELM1 (b) IML-ELM2

Fig. 4. Computation flow chart of IML-ELM network architectures for M hidden layers.

Table 2
Input compositions for the system identification.

BDS	Input composition
BDS 1	$y(k-1)$, $y(k-2)$, $u(k)$
BDS 2	$y(k)$, $u(k)$
BDS 3	$y(k)$, $u(k)$
BDS 4	$y(k)$, $y(k-1)$, $u(k)$
BDS 5	$y(k)$, $u(k)$
BDS 6	$y(k)$, $y(k-1)$, $y(k-2)$, $u(k)$, $u(k-1)$
BDS 7	$y(k)$, $y(k-1)$, $u(k)$

Finally, the experiment on BDS 7 uses 700 samples for training and 700 samples for testing. BDS 1, BDS 2, …, BDS 7 are modeled using ML-ELM, IML-ELM1 and IML-ELM2 schemes with the input compositions given in Table 2.

In this research, as activation function for all ML-ELM and IML-ELMs, logarithmic sigmoid transfer function (logsig) was used.

6. Numerical results

Experimental setup was achieved on a laptop computer with Intel (R) Core (TM) i7-4510U CPU @ 2 GHz, 8.00 GB RAM, 64 bit Windows 8.1 operating system. For each BDS, each ELM model (in our case ML-ELM, IML-ELM1 and IML-ELM2) was run for 100 times. Experiments were carried out using different numbers of nodes.

Performance analysis for each BDS was examined for the number of nodes in the {10, 15, 20, 25, 30, 40, 50, 100}, provided that the number of layers is 3 and the number of nodes in each layer is equal.

In the Table 3, analysis results for BDS 1 to 7 are given in three categories: Training time in terms of elapsed second, performance in terms of the Root Mean Square Error (RMSE) for training data set and performance in terms of RMSE for testing data set.

The average of all experimental results obtained from the experiments are given in Table 3. The percentages given under the IML-ELM's results denote the difference rates from the original ML-ELM. If the percentage is negative, it means that the IML-ELM provides a better result. Thus, it can be observed from the Table 3 that, IML-ELMs perform better than the original ML-ELM except only for BDS 3 provided that testing RMSE mean results are taken under consideration. Except BDS 3, the training RMSE and test RMSE results of IML-ELM2 is better than the ML-ELM for other BDSs. On the other hand, ML-ELM provides better mean training RMSE than the IML-ELM1 on BDS 2, 3, and 5. However, still ML-ELM fails to achieve a better mean testing RMSE than that of IML-ELM1 on BDS 2 and 5.

It must be noted that BDS 3 is a problem that does not have details and sharp transitions. That is the reason for the better

Table 3
Performance analysis metrics for all BDS.

BDS	Scheme	Training time				Training RMSE				Testing RMSE			
		Mean	Best	Worst	Std	Mean	Best	Worst	Std	Mean	Best	Worst	Std
1	ML-ELM	0.007754	0.005295	0.031980	0.003166	0.691660	0.374657	0.897934	0.107618	0.939327	0.513410	1.656150	0.198839
	IML-ELM1	0.006968 (−10%)	0.005487 (4%)	0.062573 (96%)	0.005900 (86%)	0.158001 (−77%)	0.137317 (−63%)	0.182700 (−80%)	0.008671 (−92%)	0.254354 (−73%)	0.217501 (−58%)	0.304714 (−82%)	0.016661 (−92%)
	IML-ELM2	0.005465 (−30%)	0,003312 (−37%)	0.040225 (26%)	0.005721 (81%)	0.195109 (−72%)	0.092293 (−75%)	0.333822 (−63%)	0.050543 (−53%)	0.313705 (−67%)	0.159599 (−69%)	0.576434 (−65%)	0.081994 (−59%)
2	ML-ELM	0.008257	0.005756	0.030778	0.002972	0.111888	0.043965	0.200160	0.033147	0.255077	0.118665	0.502613	0.077562
	IML-ELM1	0.008302 (1%)	0.005494 (−5%)	0.136396 (343%)	0.013360 (350%)	0.113521 (1%)	0.109538 (149%)	0.119640 (−40%)	0.001948 (−94%)	0.213350 (−16%)	0.205675 (73%)	0.222668 (−56%)	0.003305 (−96%)
	IML-ELM2	0.004934 (−40%)	0.003307 (−43%)	0.027449 (−11%)	0.003094 (4%)	0.098776 (−12%)	0.084419 (92%)	0.114241 (−43%)	0.005869 (−82%)	0.162801 (−36%)	0.131467 (11%)	0.205828 (−59%)	0.013406 (−83%)
3	ML-ELM	0.006757	0.004935	0.022812	0.002049	0.043788	0.016991	0.086269	0.013856	0.116940	0.040256	0.261525	0.042354
	IML-ELM1	0.006503 (−4%)	0.005479 (11%)	0.029863 (31%)	0.002820 (38%)	0.096702 (121%)	0.090577 (433%)	0.103280 (20%)	0.002374 (−83%)	0.137063 (17%)	0.113111 (181%)	0.169053 (−35%)	0.011942 (−72%)
	IML-ELM2	0.005088 (−25%)	0.003296 (−33%)	0.035670 (56%)	0.004138 (102%)	0.084991 (94%)	0.073025 (330%)	0.097113 (13%)	0.004944 (−64%)	0.138689 (19%)	0.102517 (155%)	0.204633 (−22%)	0.020060 (−53%)
4	ML-ELM	0.008639	0.006764	0.032403	0.003101	0.457440	0.368015	0.559396	0.040371	0.880225	0.740992	1.028787	0.054026
	IML-ELM1	0.006466 (−25%)	0.005499 (−19%)	0.027912 (−14%)	0.002500 (−19%)	0.197119 (−57%)	0.162733 (−56%)	0.230555 (−59%)	0.013488 (−67%)	0.445663 (−49%)	0.389518 (−47%)	0.506575 (−51%)	0.024916 (−54%)
	IML-ELM2	0.004304 (−50%)	0.003295 (−51%)	0.013222 (−59%)	0.001491 (−52%)	0.205383 (−55%)	0.128052 (−65%)	0.289463 (−48%)	0.033586 (−17%)	0.553100 (−37%)	0.419298 (−43%)	0.692630 (−33%)	0.053157 (−2%)
5	ML-ELM	0.006823	0.005046	0.025599	0.002349	0.382042	0.241470	0.573292	0.066938	0.577983	0.324601	0.909941	0.116529
	IML-ELM1	0.006308 (−8%)	0.005480 (9%)	0.018392 (−28%)	0.001679 (−29%)	0.547743 (43%)	0.493925 (105%)	0.581726 (1%)	0.017752 (−73%)	0.523209 (−9%)	0.473988 (46%)	0.550691 (−39%)	0.015721 (−87%)
	IML-ELM2	0.005463 (−20%)	0.003380 (−33%)	0.022324 (13%)	0.002654 (13%)	0.344570 (−10%)	0.241380 (0%)	0.472900 (−18%)	0.048500 (−28%)	0.317759 (−45%)	0.208029 (−36%)	0.469107 (−48%)	0.052907 (−55%)
6	ML-ELM	0.499755	0.458654	0.805358	0.047345	0.093329	0.046668	0.149796	0.021389	0.130810	0.078266	0.190190	0.025004
	IML-ELM1	0.379242 (−24%)	0.320060 (−30%)	0.517136 (−36%)	0.035066 (−26%)	0.034227 (−63%)	0.029672 (−36%)	0.040873 (−73%)	0.002423 (−89%)	0.029657 (−77%)	0.023857 (−70%)	0.039111 (−79%)	0.003263 (−87%)
	IML-ELM2	0.206773 (−59%)	0.175776 (−62%)	0.310811 (−61%)	0.031368 (−34%)	0.032556 (−65%)	0.024729 (−47%)	0.044372 (−70%)	0.003631 (−83%)	0.030308 (−77%)	0.020966 (−73%)	0.049632 (−74%)	0.005343 (−79%)
7	ML-ELM	0.201866	0.177324	0.288117	0.017631	1.129291	0.769299	1.301125	0.110983	1.399192	0.867911	2.120719	0.211331
	IML-ELM1	0.154516 (−23%)	0.128529 (−28%)	0.313913 (9%)	0.029553 (68%)	0.109072 (−90%)	0.095379 (−88%)	0.133171 (−90%)	0.006761 (−94%)	0.121563 (−91%)	0.109850 (−87%)	0.143644 (−93%)	0.006111 (−97%)
	IML-ELM2	0.074915 (−63%)	0.063109 (−64%)	0.130063 (−55%)	0.012709 (−28%)	0.367961 (−67%)	0.165305 (−79%)	0.672807 (−48%)	0.112064 (1%)	0.413887 (−70%)	0.178144 (−79%)	0.802708 (−62%)	0.132409 (−37%)

mean training RMSE and mean testing RMSE results of ML-ELM compared the proposed IML-ELMs. Finally, it must be stated that for all BDSs, the proposed IML-ELMs have a much shorter computation time than that of the ML-ELM which makes IML-ELMs faster than the ML-ELM.

For BDS 5 and BDS 7, graphs involving mean training time, mean training RMSE and mean testing RMSE variations with respect to different nodes are given in Figs. 5 and 6, respectively.

Regarding test results for BDS 5, ML-ELM achieved the best performance using 100 nodes whereas IML-ELM1 gave its best performance using 15 nodes. Best performance was achieved by IML-ELM2 using 30 nodes. ML-ELM achieved a mean testing RMSE of 0.323780 and a training time of 0.016405 using 100 nodes. On the other hand, IML-ELM2 resulted a mean testing RMSE of 0.226226 and a training time of 0.004907 using 30 nodes. For BDS 5, not only ML-ELM performs worse than the IML-ELM2, but also the ML-ELM is 70% slower than the IML-ELM2.

If we look at the computation time in Fig. 6, there are differences between 3 models. BDS 7 modeling performance results are also similar to the BDS 5 modeling performance results. The average training time of ML-ELM, IML-ELM1 and IML-ELM2 are 0.201866, 0.154516 and 0.074915 respectively. Clearly IML-ELMs are faster than the ML-ELM with IML-ELM1 is 23% faster whereas IML-ELM2 is 63% faster. According to Fig. 6, using all nodes, IML-ELM1 and IML-ELM2 performed better than ML-ELM, regarding mean testing RMSE results. ML-ELM provided the best testing RMSE value with 100 nodes. With 100 nodes, ML-ELM, IML-ELM1 and IML-ELM2 achieved mean testing RMSE results of 0.627977, 0.104272 and 0.092683 respectively. Compared to the ML-ELM, the proposed IML-ELM1 performed 83% better whereas another design the IML-ELM2 performed 85% better. If the reader wants to observe the computation speed with 100 nodes, IML-ELM is 13% faster than the traditional ML-ELM whereas IML-ELM2 is 63% faster than the ML-ELM. Finally, compared to ML-ELM with 100 nodes, IML-ELM1 and IML-ELM2 achieved better test results on every occasion except using 10/15 nodes for IML-ELM2. Clearly IML-ELMs surpass ML-ELM while modeling BDS 7.

Below in Figs. 7 and 8, the best and worst system modeling performances using all schemes (ML-ELM, proposed IML-ELM1 and proposed IML-ELM2) are given based on the BDS 7 modeling task. From the figures, it can be clearly seen that the proposed IML-ELM2 performed better than the proposed IML-ELM1 which also performed better than the ML-ELM.

According to Fig. 7, in detailed and sharp transitions, the proposed IML-ELMs perform better modeling than the traditional ML-ELM scheme. If compared, IML-ELM2 scheme provides better modeling performance than the IML-ELM1 scheme.

Fig. 8 displays the worst modeling performance of all schemes and it can be seen that ML-ELM' s worst modeling defects are much more visible than the worst modeling defects of the proposed IML-ELMs. It is apparent that even in their worst, the proposed IML-ELM1 and IML-ELM2 made a good modeling.

7. Discussion

Regarding Table 3, the designed IML-ELM architectures provide better results than the ML-ELM on the modeling of dynamic systems that have sharp transitions (BDS 1, BDS 2, BDS 4-7). However, the IML-ELM architectures does not perform better than the

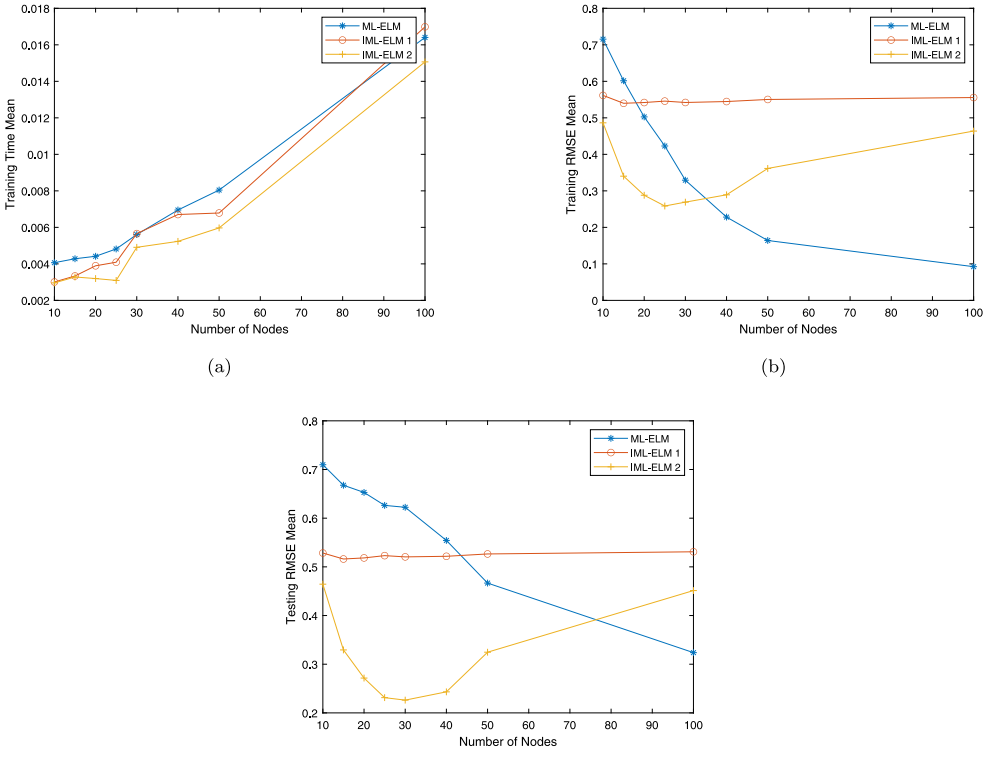


Fig. 5. Performance results in terms of training time (a), RMSE for training data set (b), RMSE for testing data set (c) with respect to node number variations for BDS 5.

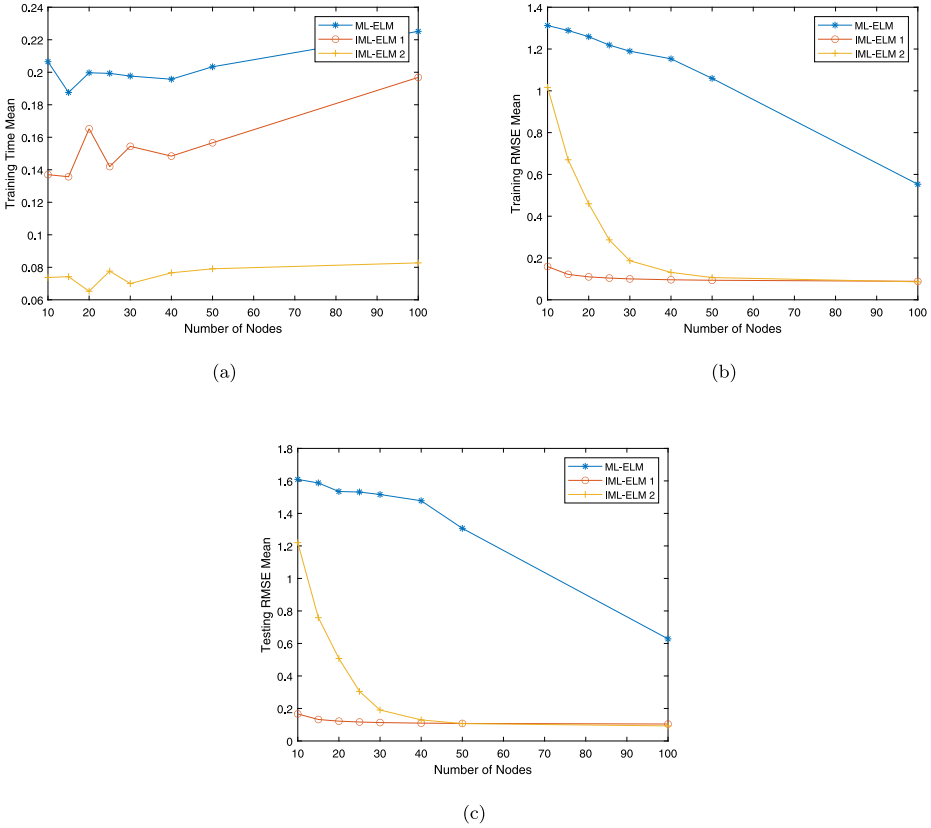


Fig. 6. Performance results in terms of training time (a), RMSE for training data set (b), RMSE for testing data set (c) with respect to node number variations for BDS 7.

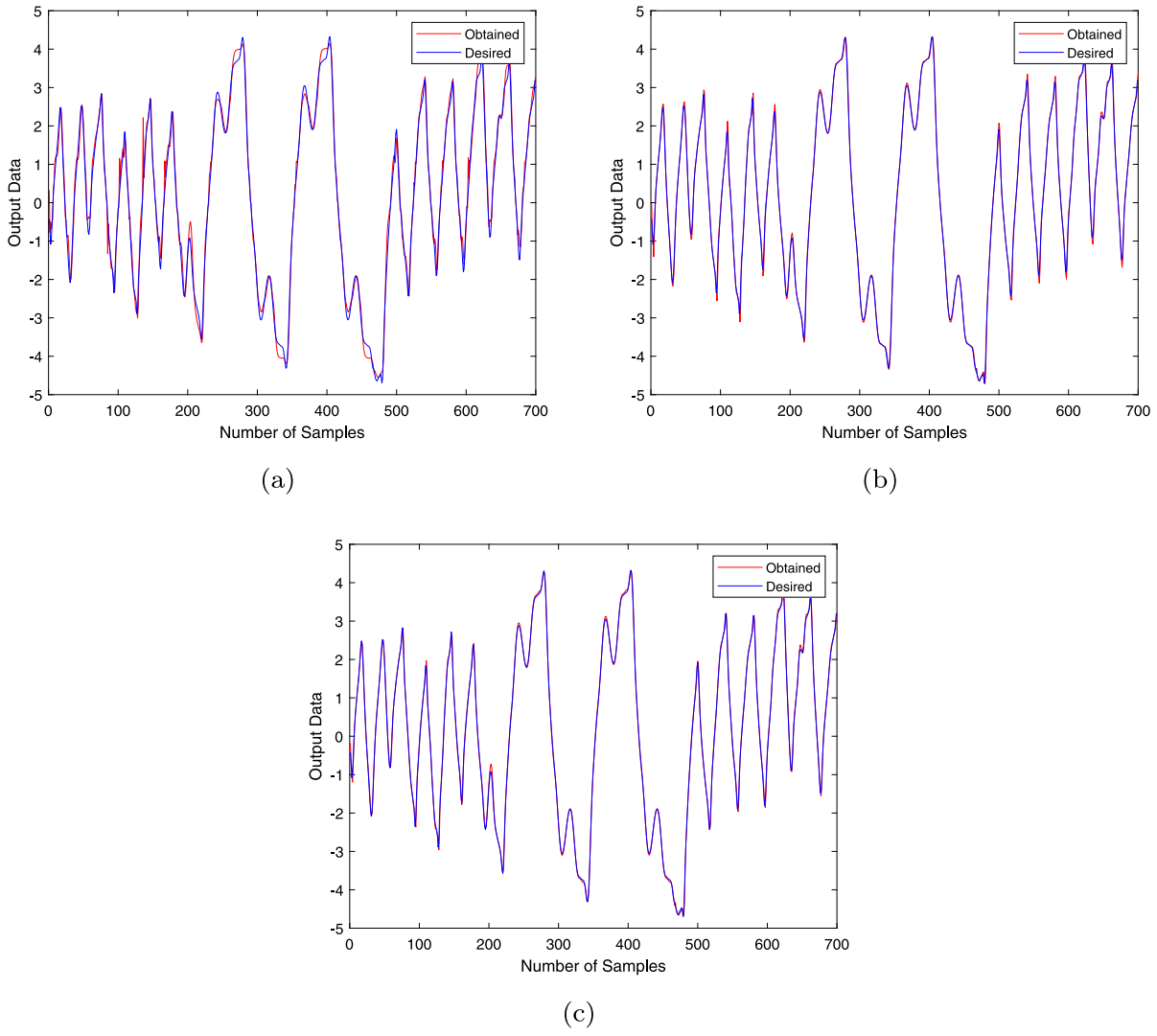


Fig. 7. The best system modeling performance on BDS 7 for ML-ELM (a), IML-ELM1 (b), and IML-ELM2 (c) structures.

ML-ELM on dynamic systems that have smooth transitions (BDS 3). Because of the novel input representation calculations that has been introduced in this research, IML-ELM1 is faster than the ML-ELM according to the results given in Table 3. Only on BDS 2, the ML-ELM has a faster training time than IML-ELM1. On the other hand, IML-ELM2 is much faster than both ML-ELM and IML-ELM1 on all cases. It can be summarized that IML-ELMs might not be more successful than the ML-ELM on modeling dynamic systems that have smooth transitions; however, IML-ELMs perform better on modeling dynamic systems that have sharp transitions. Finally, it can be underlined that IML-ELM1 is faster than the ML-ELM and IML-ELM2 is much faster than the traditional ML-ELM.

8. Conclusions

In terms of system modeling, no study has been accomplished in the context of the given one in this research. Regarding the training time, the proposed IML-ELM1 is close to but a little faster than the original ML-ELM whereas IML-ELM2 is much faster than the both, because the structure uses random assignments only at the first layer (it must be mentioned that IML-ELM2 structure also computes current layer' s representation in a manner the IML-MLM1 does).

During the modeling of systems that does not have sharp transitions and details, the proposed IML-ELMs does not perform better than ML-ELM. On the other hand, proposed IML-ELMs perform much better than the original ML-ELM on modeling systems that have details and sharp transitions.

On all BDSs, ML-ELM achieved its best performance using 100 nodes and this affects the computation time which makes the scheme slower. Except BDS 3, all IML-ELMs perform better than the ML-ELM for system modeling according to the Table 3. So not only the proposed IML-ELMs are a better tool, they are also faster (IML-ELM1 is close to but a little faster, IML-ELM2 is much faster) than the traditional ML-ELM. These facts prove that the proposed IML-ELMs are an efficient and feasible system modeling scheme.

CRediT authorship contribution statement

Gizem Atac Kale: Conceptualization, Methodology, Software (for IML-ELM2). **Cihan Karakuzu:** Conceptualization, Methodology, Software (for IML-ELM1).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

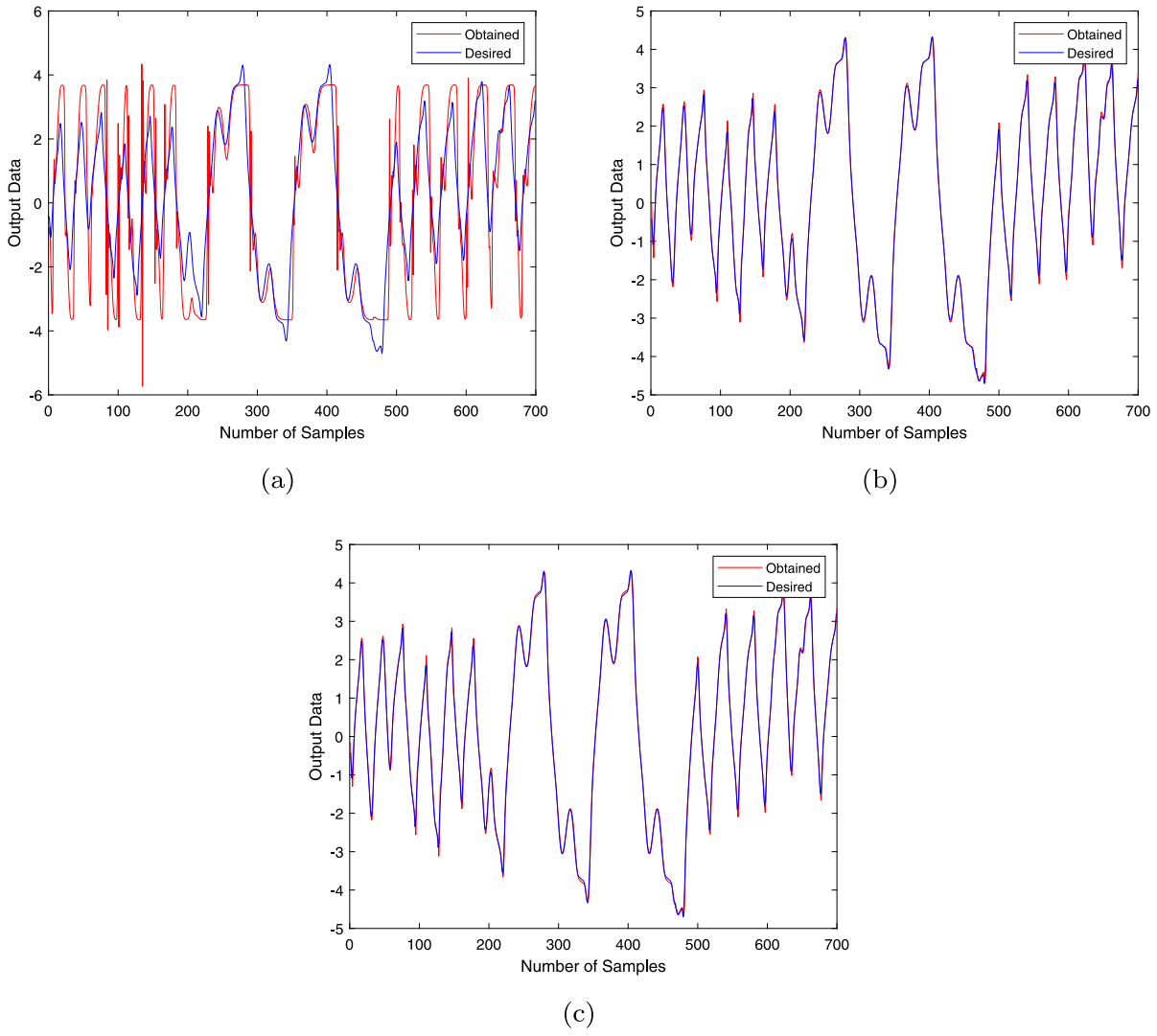


Fig. 8. The worst system modeling performance on BDS 7 for ML-ELM (a), IML-ELM1 (b), and IML-ELM2 (c) structures.

References

[1] G.-B. Huang, Q.-Y. Zhu, C.K. Siew, Extreme learning machine: A new learning scheme of feedforward neural networks, in: 2004 IEEE International Joint Conference on Neural Networks, 2004, http://dx.doi.org/10.1109/IJCNN.2004.1380068.

[2] S. Liao, C. Feng, Meta ELM: ELM with ELM hidden nodes, Neurocomputing 128 (2014) 81–87, http://dx.doi.org/10.1016/j.neucom.2013.01.060.

[3] J. Tang, C. Deng, G.-B. Huang, Extreme learning machine for multilayer perceptron, IEEE Trans. Neural Netw. Learn. Syst. 27 (4) (2016) 809–821, http://dx.doi.org/10.1109/TNNLS.2015.2424995.

[4] L.L.C. Kasun, H. Zhou, G.-B. Huang, C.M. Vong, Representational learning with extreme learning machine for big data, IEEE Intell. Syst. 28 (2013) 31–34.

[5] P. Vidnerova, R. Neruda, Deep networks with RBF layers to prevent adversarial examples, in: International Conference on Artificial Intelligence and Soft Computing, Springer, 2018, pp. 257–266.

[6] C.-M. Vong, C. Chen, P.-K. Wong, Empirical kernel map-based multilayer extreme learning machines for representation learning, Neurocomputing 310 (2018) http://dx.doi.org/10.1016/j.neucom.2018.05.032.

[7] M. Chen, Y. Li, X. Luo, W. Wang, L. Wang, W. Zhao, A novel human activity recognition scheme for smart health using multilayer extreme learning machine, IEEE Internet Things J. 6 (2019) 1410–1418, http://dx.doi.org/10.1109/JIOT.2018.2856241.

[8] C.M. Wong, C.M. Vong, P.K. Wong, J. Cao, Kernel-based multilayer extreme learning machines for representation learning, IEEE Trans. Neural Netw. Learn. Syst. 29 (3) (2018) 757–762, http://dx.doi.org/10.1109/TNNLS.2016.2636834.

[9] J. Kim, J. Kim, G.-J. Jang, M. Lee, Fast learning method for convolutional neural networks using extreme learning machine and its application to lane detection, Neural Netw. 87 (2017) 109–121, http://dx.doi.org/10.1016/j.neunet.2016.12.002.

[10] M. Yousefi-Azar, M.D. McDonnell, Semi-supervised convolutional extreme learning machine, in: Neural Networks (IJCNN), 2017 International Joint Conference on, IEEE, 2017, pp. 1968–1974.

[11] G. Altan, Y. Kutlu, Hessenberg ELM autoencoder kernel for deep learning, J. Eng. Technol. Appl. Sci. 3 (2) (2018) 141–151, http://dx.doi.org/10.30931/jetas.450252.

[12] M. Tissera, M. McDonnell, Deep extreme learning machines: supervised autoencoding architecture for classification, Neurocomputing 174 (2016) 42–49, http://dx.doi.org/10.1016/j.neucom.2015.03.110.

[13] J. Zhang, Y. Li, W. Xiao, Z. Zhang, Non-iterative and fast deep learning: Multilayer extreme learning machines, J. Franklin Inst. B 357 (13) (2020) 8925–8955, http://dx.doi.org/10.1016/j.jfranklin.2020.04.033.

[14] Z.E. Khatab, A.H. Gazestani, S. Ghorashi, M. Ghavami, A fingerprint technique for indoor localization using autoencoder based semi-supervised deep extreme learning machine, Signal Process. 181 (2021) 107915, http://dx.doi.org/10.1016/j.sigpro.2020.107915.

[15] P. Golestaneh, M. Zekri, F. Sheikholeslam, Fuzzy wavelet extreme learning machine, Fuzzy Sets and Systems 342 (2018) 90–108, http://dx.doi.org/10.1155/2011/753572.

[16] P. Jagtap, P. Raut, G.N. Pillai, F. Kazi, N.M. Singh, Extreme-ANFIS: A novel learning approach for inverse model control of nonlinear dynamical systems, in: 2015 International Conference on Industrial Instruments and Control, 2015, http://dx.doi.org/10.1109/IIC.2015.7150836.

[17] H.-J. Rong, G.-B. Huang, N. Sundararajan, P. Saratchandran, Online sequential fuzzy extreme learning machine for function approximation and classification problems, IEEE Trans. Syst. Man Cybern. 39 (2009) 1067–1072, http://dx.doi.org/10.1109/TSMCB.2008.2010506.

[18] W. Zhang, H. Ji, Fuzzy extreme learning machine for classification, Electron. Lett. 49 (2013) 448–450, http://dx.doi.org/10.1049/el.2012.3642.
[19] R.M. Adnan, M. Zounemat-Kermani, A. Kuriqi, O. Kisi, Machine learning method in prediction streamflow considering periodicity component, in: R. Deo, P. Samui, O. Kisi, Z. Yaseen (Eds.), Intelligent Data Analytics for Decision-Support Systems in Hazard Mitigation: Theory and Practice of Hazard Mitigation, Springer Singapore, Singapore, 2021, pp. 383–403, http://dx.doi.org/10.1007/978-981-15-5772-9_18.
[20] R.M. Adnan, Z. Chen, X. Yuan, O. Kisi, A. El-Shafie, A. Kuriqi, M. Ikram, Reference evapotranspiration modeling using new heuristic methods, Entropy 22 (5) (2020) 547, http://dx.doi.org/10.3390/e22050547.
[21] P. Angelov, D. Filev, An approach to online identification of Takagi-Sugeno fuzzy models, IEEE Trans. Syst. Man Cybern. -B: Cybern. 34 (1) (2004) 484–498, http://dx.doi.org/10.1109/TSMCB.2003.817053.
[22] D. Leite, R. Ballini, P. Costa, F. Gomide, Evolving fuzzy granular modeling from nonstationary fuzzy data streams, Evol. Syst. (3) (2012) 65–79, http://dx.doi.org/10.1007/s12530-012-9050-9.
[23] D. Kangin, P. Angelov, Evolving clustering, classification and regression with TEDA, in: 2015 International Joint Conference on Neural Networks (IJCNN), 2015, pp. 1–8, http://dx.doi.org/10.1109/IJCNN.2015.7280528.
[24] M. Pratama, S.G. Anavatti, P.P. Angelov, E. Lughofer, PANFIS: A novel incremental learning machine, IEEE Trans. Neural Netw. Learn. Syst. 25 (1) (2014) 55–68, http://dx.doi.org/10.1109/TNNLS.2013.2271933.
[25] K. Khalil, O. Eldash, A. Kumar, M. Bayoumi, Economic LSTM approach for recurrent neural networks, IEEE Trans. Circuits Syst. II: Express Briefs 66 (11) (2019) 1885–1889, http://dx.doi.org/10.1109/TCSII.2019.2924663.
[26] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Netw. 1 (1990) 4–27, http://dx.doi.org/10.1109/72.80202.
[27] R. Babuska, Fuzzy systems, modeling and identification, 1998.
[28] Y. Oussar, I. Rivals, L. Personnaz, G. Dreyfus, Training wavelet networks for nonlinear dynamic input-output modeling, Neurocomputing 20 (1998) 173–188, http://dx.doi.org/10.1016/S0925-2312(98)00010-1.
[29] P. Sastry, G. Santharam, K. Unnikrishnan, Memory neuron networks for identification and control of dynamical systems, IEEE Trans. Neural Netw. 5 (1994) 306–319, http://dx.doi.org/10.1109/72.279193.
[30] C.-F. Juang, A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms, IEEE Trans. Fuzzy Syst. 10 (2002) 155–170, http://dx.doi.org/10.1109/91.995118.
[31] J.-S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Trans. Syst. Man Cybern. 23 (1993) 665–685, http://dx.doi.org/10.1109/21.256541.
[32] C. Karakuzu, On the performance of newsworthy meta-heuristic algorithms based on the point of view fuzzy modelling, Turk. J. Electr. Eng. Comput. Sci. 25 (2017) 4706–4721, http://dx.doi.org/10.3906/elk-1705-337.