IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 8, 2019, accepted June 20, 2019, date of publication June 24, 2019, date of current version September 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2924647

Sparse Deep Tensor Extreme Learning Machine for Pattern Classification

JIN ZHAO AND LICHENG JIAO, (Fellow, IEEE)
Key Laboratory of Intelligent Perception and Image Understanding, School of Artificial Intelligence, Ministry of Education, Xidian University, Xi'an 710071,China

Corresponding author: Jin Zhao (zhaojin154@163.com)

This work was supported in part by National Key R&D Program of China (No. 2018YFC0825305, No.2018YFC0825303), the State Key Program of National Natural Science of China (No. 61836009), the National Natural Science Foundation of China (No. 61501353, No. 61573267, No.61473215), the Natural Science Pre-Research Fund of Shaanxi University of Science and Technology (No. 2019BJ-11), in part by the Program for Cheung Kong Scholars and Innovative Research Team in University (No. IRT_15R53), in part by The Fund for Foreign Scholars in University Research and Teaching Programs (the 111 Project) (No. B07048).

ABSTRACT A novel deep architecture, the sparse deep tensor extreme learning machine (SDT-ELM), is presented as a tool for pattern classification. In extending the original ELM, the proposed SDT-ELM gains the theoretical advantage of effectively reducing the number of hidden-layer parameters by using tensor operations, and using a weight tensor to incorporate higher-order statistics of the hidden feature. In addition, the SDT-ELM gains the implementation advantage of enabling the random hidden nodes to be added block by block, with all blocks having the same hidden layer configuration. Moreover, an SDT-ELM without randomness can also achieve better learning accuracy. Extensive experiments with three widely used classification datasets demonstrate that the proposed algorithm achieves better generalization performance.

INDEX TERMS Extreme learning machine, deep learning, tensor, stacking, pattern classification.

I. INTRODUCTION

The extreme learning machine (ELM) has become an effective and efficient machine-learning technique in recent years, requiring the use of neither a back-propagation algorithm nor iterative techniques [1], [2]. It has been proved theoretically that single hidden layer feedforward networks with arbitrary hidden parameters and a continuous active function can universally approximate any continuous function. From an application perspective, the ELM has shown excellent generalization performance in various applications, including face classification, human action recognition and image segmentation. However, the ELM suffers from two major drawbacks [3]. First, the accuracy of the ELM is dramatically influenced by the number of hidden neurons. An excellently performing ELM model can easily contain hundreds or even thousands of hidden neurons for a practical application. Second, feature learning using an ELM may not be effective.

To address the first of these drawbacks, current approaches to finding the optimal number of hidden neurons are mainly based on incremental learning methodologies that aim to minimize the training error. Their weakness is that there might be many hidden neurons selected in the trained model following the minimization of the training error in ranking neurons, which will result in a high computational cost.

A variety of autoencoders (AEs) with different regular constraints based on the ELM have been proposed to address the second drawback. Furthermore, inspired by the deep stacked autoencoder network (DSAE) [4], [5], the hierarchical extreme learning machine (H-ELM) has been proposed [3]. Unlike the greedy layerwise training of the DSAE, H-ELM can achieve better generalization performance without requiring parameter fine-tuning of the entire system. However, the H-ELM encounters the same problems as the ELM. That is, for high generalization performance, large numbers of hidden nodes are used. The broad learning system (BLS) [6] has similar advantages to the ELM, in that incremental learning can rapidly update and remodel the system. However, the accuracy of the BLS is also drastically influenced by the number of neurons in the last hidden layer.

In recent years, DSAEs [7] and deep stacking network (DSN) [8], [9] have been trained by using layerwise learning and fine-tuning mechanisms [10], [11], which can improve efficiency and performance. However, for DSAEs, the AE uses an unsupervised-learning paradigm, which does

The associate editor coordinating the review of this article and approving it for publication was Chao Shen.

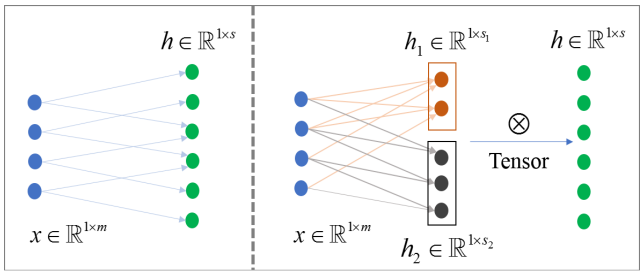


FIGURE 1. Left: Traditional feed-forward mode. Right: Feed-forward mode with tensor operations.

not utilize the category-prior information of the samples. For the DSN, each block can output an estimate of the final label class, and this estimate is concatenated with the input to form an augmented input for the next block. However, DSNs do not aim to discover transformed feature representations. Besides, the DSN is a scalable deep architecture amenable to parallel weight learning, which can be trained in a supervised, block-wise fashion, without the need for back-propagation over all blocks. Because the original input is retained for each higher block in the DSN, it is guaranteed to perform with better accuracy on the training set than the previous block.

To preserve the original data format and their respective correlations and to achieve more-compact representations, tensor operations are embedded into the DSN [9]. Building on this, the new deep architecture presented in this paper, which we call the sparse deep tensor extreme learning machine (SDT-ELM), improves and extends the DSN architecture in two significant ways. First, the linear and non-linear layers inherited from ELM makes the stacking up of deeper layers both simple and easy to implement. Second, the tensor operations can effectively reduce the number of hidden-layer parameters, as shown in Fig. 1. That is, if we assume $s \triangleq s_1 \cdot s_2$, we have $h = h_1 \otimes h_2 \in \mathbb{R}^s$, where '$\otimes$' denote the tensor operation. Clearly, from x to h, we will have $m \cdot s$ parameters in the traditional feed-forward mode, and $m\cdot(s_1+s_2)$ parameters for the feed-forward mode using tensor operations. In general, if s is relatively large, we will have $(s_1 + s_2) < s$. Extensive experiments on various widely used classification data sets show that the proposed SDT-ELM can achieve better and faster convergence than the existing state-of-the-art deep stacking learning methods. Furthermore, The motivation of extensive practical use of the theoretical results presented SDT-ELM mainly lies in the fact that tensors can maintain the structural information contained in data, and this advantage makes tensors have great potential in the high-dimensional data processing.

This paper is organized as follows. Section 2 introduces the related works, including the fundamental ideas and capabilities of ELM and the framework of DSN. Section 3 describes the SDT-ELM and gives details of its proposed algorithm. Section 4 presents our experimental results, comparing the performance of various deep systems. Finally, a discussion and our conclusions are given in Section 5.

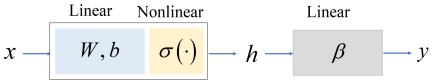


FIGURE 2. ELM: Single-hidden-layer feed-forward network.

II. RELATED WORK

To facilitate the understanding of the proposed the algorithm, we first review the related theory of the ELM and the framework of DSN.

A. ELM: A REVIEW

An ELM is a unified framework for a broad type of generalized single-hidden-layer feedforward networks (SLFN). It randomly chooses hidden-node parameters and analytically determines the output weights [1].

For an ELM (see Fig. 2), we have the mathematical model,

$$\begin{cases} h = \sigma\left(x\boldsymbol{W} + b\right) \\ y = h\boldsymbol{\beta}, \end{cases} \tag{1}$$

where $\boldsymbol{W}$ and b are randomly chosen parameters, $\boldsymbol{\beta}$ is the parameter to be learned, and the activation function $\sigma(\cdot)$ used in classical ELM is a simple sigmoid function or a infinitely differentiable function. The hidden-layer output vector h can remain unchanged after random values have been assigned to these parameters at the beginning of the learning process. For N arbitrary distinct samples $\{x_n, y_n\}$, where $x_n \in \mathbb{R}^{1\times m}$ and $y_n \in \mathbb{R}^{1\times c}$, we have

$$\begin{cases} H = \sigma\left(X\boldsymbol{W} + \boldsymbol{b}\right) \\ Y = H\boldsymbol{\beta}, \end{cases} \tag{2}$$

where $X \in \mathbb{R}^{N\times m}$ and $Y \in \mathbb{R}^{N\times c}$ are matrices composed of sample sets $\left\{x_n, y_n\right\}, n = 1, 2, \cdots, N$, the weights $\boldsymbol{W} \in \mathbb{R}^{m\times s}$, $\boldsymbol{\beta} \in \mathbb{R}^{s\times c}$, and the bias $\boldsymbol{b} \in \mathbb{R}$, where s is the number of hidden nodes. The optimization objective function is

$$\min_{\boldsymbol{\beta}} \|Y - H\boldsymbol{\beta}\|_F^2 + \lambda \|\boldsymbol{\beta}\|_F^2 . \tag{3}$$

The smallest-norm least-squares solution of Eq. (2) is

$$\boldsymbol{\beta} = H^{\dagger}Y = (H^T H + \lambda \boldsymbol{I})^{-1} H^T Y, \tag{4}$$

where $H^{\dagger}$ is the Moore–Penrose generalized inverse [12] of the matrix H, which is called the hidden-layer output matrix. In theory, an ELM tends to provide good generalization performance at an extremely fast learning speed.

At present, some mathematical theorems prove [13], [14] that randomly generated networks with the outputs being solved by least mean square is able to preserve the universal approximation capability, if and only if the activation function is non-constant piecewise and linear span of radial basis function with this activation function is dense in $L^2(R)$. Further, with the continuous development of research, more and more scientific research workers develop many improved networks based on ELM. A more detailed overview of the current ELM architecture (but not limited to) is shown in TABLE.1.

As can be seen from TABLE.1, with the continuous popularization of research on deep network architecture, people

TABLE 1. The overview of the current ELM architecture.

Model	Proposed Year	Architecture
ELM [1], [15]	2004∼2006	Shallow
Evolutionary ELM [16]	2005	Shallow
Fully complex ELM [17], [18]	2005∼2008	Shallow
Online sequential ELM [19]	2006	Shallow
Incremental ELM [13], [14]	2006∼2007	Shallow
Pruning ELM [20]	2008	Shallow
ELM ensembles [21], [22]	2009∼2011	Shallow
Bayesian ELM [23]	2011	Shallow
Bidirectional ELM [24]	2012	Shallow
Kernel based ELM [2]	2012	Shallow
Robust ELM [25]	2013	Shallow
Sparse ELM [26]	2014	Shallow
ELM Local Receptive Field [35]	2015	Shallow
ELM Sparse AutoEncoder [3]	2015	Shallow
ELM for Multilayer Perceptron [3]	2015	Hierarchical
Hierarchical ELM [27]	2015	Hierarchical
Stacked ELM [28]	2015	Hierarchical
Deep ELM [29]	2016	Hierarchical
Deep Convolutional ELM [30]	2016	Hierarchical
Deep Kernelized ELM [31]	2018	Hierarchical

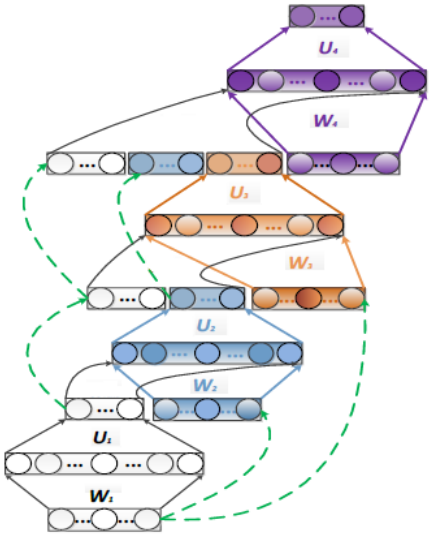


FIGURE 3. DSN: Deep stacking network.

are gradually inclined to combine ELM with deep mode to construct many new models with excellent generalization performance suitable for the pattern recognition task.

B. THE FRAMEWORK OF DSN

Different from the classical deep stacked autoencoder network, DSN architecture adopts the hierarchical stack form of supervised learning, that is, each module of the DSN network can effectively estimate the output label. Its network structure is shown in Fig. 3.

For the first module, the input is $x_1 \triangleq x$, then the output of the module is

$$y_1 = U_1^T \sigma(W_1^T x_1) \tag{5}$$

where W_1 and U_1 are the parameters to be learned in this module, respectively; and σ is the activation function usually takes the sigmoid function. y_1 is the approximation of the first module to the output label or target y.

For the second module, its input is $x_2 \triangleq [x_1, y_1]$, then the output of this module is

$$y_2 = U_2^T \sigma(W_2^T x_2) \tag{6}$$

where W_2 and U_2 are also the parameters to be learned in this module, respectively; and σ is the activation function. y_2 is also the approximation of the first module to the output label or target y. Similarly, we can build a deep stacking network for a given number of layers in this way.

How do we optimize these parameters in each module? Without loss of generality, we give the parameter optimization method of the first module. Suppose the training sets in the form of a matrix is X, and the corresponding output label is denoted by Y. According to the Eq. (5) of the first module, we have the following optimization objective function,

$$\min_{W,U} J \triangleq \left\| Y - U^T H \right\|_F = \left\| Y - U^T \sigma(W^T X) \right\|_F \tag{7}$$

where $H \triangleq \sigma(W^T X)$. The parameters are optimized by alternating iterations as follows. 1) when W is fixed, and H is known, then the parameter U can be given by the solution in the closed form,

$$U = H^{\dagger} Y^T \tag{8}$$

where $H^{\dagger}$ is the Moore–Penrose generalized inverse of the matrix H. 2) when U is fixed, the parameter W is given in the following iterative way,

$$W^{(k+1)} = W^{(k)} - \alpha \frac{\partial J}{\partial W}|_{W=W^{(k)}} \tag{9}$$

where α is the learning rate and k is the number of iterations. In particular,

$$\begin{aligned} \frac{\partial J}{\partial W} &= \frac{\partial Tr\left[(U^T H - Y)(U^T H - Y)^T\right]}{\partial W} \\ &= \frac{\partial Tr\left[YY^T - YH^T(HH^T)^{-1}HY^T\right]}{\partial W} \\ &= 2X\left[H^T \circ (1-H)^T V\right] \end{aligned} \tag{10}$$

where $V \triangleq [H^{\dagger}(HY^T)(YH^{\dagger}) - Y^T(YH^{\dagger})]$. By alternately optimizing and updating the parameters, we can solve the parameters U and W in the first module. The parameters of other modules can be solved similarly. Different from DSAE, DSN does not need to carry out error back propagation adjustment on all modules, but only needs to update parameters on each module.

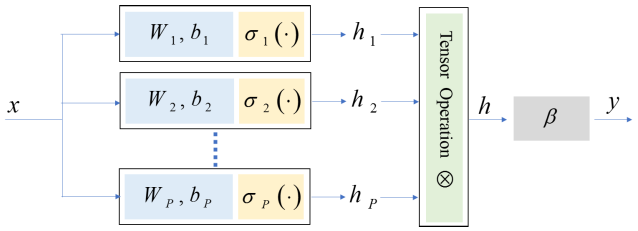


FIGURE 4. ST-ELM architecture.

III. SDT-ELM

In this section, we provide an overview of the SDT-ELM architecture and describe its fundamental properties. First, we design a shallow-architecture sparse tensor extreme learning machine (ST-ELM), in which sparsity is mainly reflected in the setting of the weight matrix and the active function. We extend this design to the SDT-ELM by stacking ST-ELM modules, for which the main parameter-estimation burden is shifted to a convex sub-problem with a closed-form solution for the ST-ELM.

A. ST-ELM

The structure of the proposed ST-ELM (see Fig. 4) is based on the original ELM. In an ST-ELM, the mathematical modelling from input $x \in \mathbb{R}^{1\times m}$ to output $y \in \mathbb{R}^{1\times c}$ is given by

$$\begin{cases} h_i = \sigma_i(x\boldsymbol{W}_i + b_i) \in \mathbb{R}^{1\times s_i}, \\ h = h_1 \otimes h_2 \otimes \cdots \otimes h_P \in \mathbb{R}^{1\times s}, \\ y = h\boldsymbol{\beta} \in \mathbb{R}^{1\times c}, \end{cases} \tag{11}$$

where, for $i = 1, 2, \cdots, P$, $\boldsymbol{W}_i \in \mathbb{R}^{m\times s_i}$, and $b_i \in \mathbb{R}^{1\times s_i}$ are randomly chosen parameters, and $\sigma_i(\cdot)$ is the active function usually takes the sigmoid function. '$\otimes$' denotes the Kronecker tensor product and $s \triangleq \prod_{i=1}^{P} s_i$. $\boldsymbol{\beta} \in \mathbb{R}^{s\times c}$ is the parameter to be learned. Note that $\boldsymbol{W}_i$ and b_i are randomly assigned with the sparsity degree of ρ $(0 < \rho \leq 1)$, giving

$$\begin{cases} Sparsity\,(\boldsymbol{W_i}) \triangleq \dfrac{\|\boldsymbol{W_i}\|_0}{m \times s_i} = \rho, \\ Sparsity\,(b_i) \triangleq \dfrac{\|b_i\|_0}{s_i} = \rho, \end{cases} \tag{12}$$

where $\|\cdot\|_0$ represents the number of zero elements in a matrix or vector.

For the training samples $\{x_n, y_n\}(n = 1, 2, \cdots, N)$, we have

$$\begin{cases} H_i = \sigma_i(X\boldsymbol{W}_i + b_i) \in \mathbb{R}^{N\times s_i}, \\ H = H_1 \odot H_2 \odot \cdots \odot H_P \in \mathbb{R}^{N\times s}, \\ Y = H\boldsymbol{\beta} \in \mathbb{R}^{1\times c}, \end{cases} \tag{13}$$

where the '$\odot$' operation is the Khatri-Rao product, which produces a columnwise Kronecker product [32].

We solve for the parameter $\boldsymbol{\beta}$ by using Eq.(3) and Eq.(4). Without loss of generality, the ST-ELM algorithm reverts to the classical ELM with respect to sparsity when satisfying $s_i = s$ and $s_j = 1 (j \neq i)$. To summarize, we have a generalization from the mapping of $\mathbb{R}^s \to \mathbb{R}^c$ in the ELM to the mapping of $\mathbb{R}^{s_1} \otimes \mathbb{R}^{s_2} \otimes \cdots \otimes \mathbb{R}^{s_p} \to \mathbb{R}^c$ in the ST-ELM.

B. SDT-ELM

In this subsection, we provide an overview of the SDT-ELM architecture, as shown in Fig. 5. For each ST-ELM sub-module of the SDT-ELM, its output vector y is not used for decision-making but is concatenated with its input vector to feed to the next sub-module. This leads to the mathematical model

$$\begin{cases} \widetilde{y}_l = h^{(l)}\boldsymbol{\beta}^{(l)} = \left(h_1^{(l)} \otimes h_2^{(l)} \otimes \cdots \otimes h_{P_l}^{(l)}\right)\boldsymbol{\beta}^{(l)}, \\ h_i^{(l)} = \sigma_i^{(l)}\left(\widetilde{x}_{l-1}\boldsymbol{W}_i^{(l)} + b_i^{(l)}\right), \\ \widetilde{x}_{l+1} = [\widetilde{x}_l, \widetilde{y}_l], \end{cases} \tag{14}$$

where P_l denotes the number of paths in the ST-ELM sub-module and L represents the number of simple modules (i.e., the depth of the SDT-ELM), for $i = 1, 2, \cdots, P_l$ and $l = 1, 2, \cdots, L$. Note that the output of the l-th ST-ELM sub-module $\widetilde{y}_l$ represents an approximation to the targets of classification. Clearly, the dimensionality of the augmented input $\widetilde{x}_l \in \mathbb{R}^{m_l}$ is a function of the module number, that is,

$$m_l = m + c \cdot (l - 1), \tag{15}$$

where m is the dimensionality of input $x \in \mathbb{R}^{1\times m}$, and c is the dimensionality of target $y \in \mathbb{R}^{1\times c}$.

In particular, $l = 0$ corresponds to the first module, and $\widetilde{x}_0$ corresponds to x. In a totally different approach from those of available DSAE frameworks, the SDT-ELM does not aim to discover a transformed feature representation. Instead, it makes full use of a simple layer-by-layer stacking architecture, which concatenates intermediate outputs with previous input maps to maintain more complete information.

There are five hyper-parameters of an SDT-ELM, namely the depth of the SDT-ELM (L), the sparsity of the weight matrix and bias (ρ), the active function (σ), the number of paths (P), and the number of hidden units $(s_i\ (i = 1, 2 \cdots, P))$ in the ST-ELM sub-modules. This can be denoted by the five-tuple

$$\boldsymbol{\zeta} \triangleq [L, \rho, \sigma, P, V], \tag{16}$$

where $V = [s_1, s_2, \cdots, s_P]$.

In addition, the parameters of the SDT-ELM are $\boldsymbol{\theta} = \left\{\boldsymbol{W}_i^{(l)}, \boldsymbol{b}_i^{(l)}, \boldsymbol{\beta}_i^{(l)}\right\}_{i,l}$, where $\boldsymbol{W}_i^{(l)} and \boldsymbol{b}_i^{(l)}$ are randomly chosen parameters. According to the predefined description, the parameters remain unchanged after being assigned their random values. Note that the pseudo-inverses of all the matrices involved can be computed by Eq.(4) and the parameters at each ST-ELM sub-module can be computed deterministically. Note also that the key advantage of the SDT-ELM is its novel ability to capture higher-order feature interactions via the Kronecker tensor product.

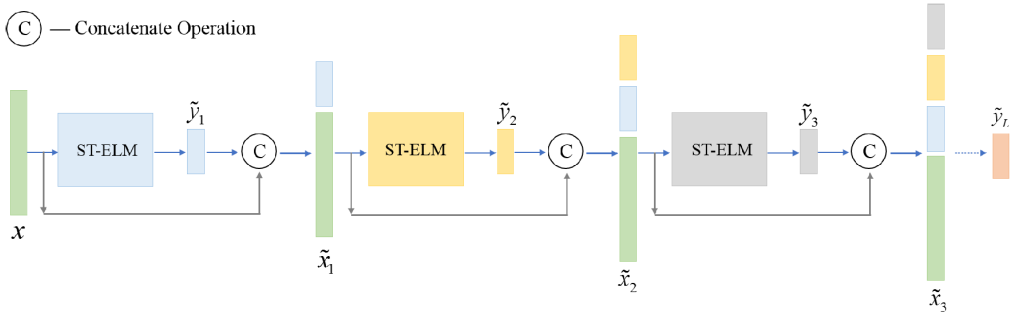


FIGURE 5. SDT-ELM architecture: All ST-ELM blocks have the same hidden-layer configuration $[\rho, \sigma, P, s]$.

C. REMOVE RANDOMNESS

In the SDT-ELM, one of the main mechanisms involves the randomness of the weight matrix $\boldsymbol{W}$ and the bias b for each ST-ELM sub-module, which is inherited from the ELM. The removal of randomness means that the parameter values for each module need to be updated. The advantages brought by randomization will be weakened by the independent update of each module and by the fine tuning of the whole network. In particular, the process of training becomes more complicated. We can identify two approaches to removing randomness. The first involves an ELM-based sparse AE, which can be regarded as an important tool for modest fine-tuning of the random features into sparse and compact features. The second involves the use of a gradient-descent algorithm for the objective function. Here, we briefly outline this second approach.

In an ST-ELM, we have the optimization objective function,

$$\min_{\boldsymbol{\beta},\boldsymbol{W},b} J = \|Y - H(X; \boldsymbol{W}, b)\boldsymbol{\beta}\|_F^2 + \lambda \|\boldsymbol{\beta}\|_F^2 . \tag{17}$$

where $H(X; \boldsymbol{W}, b)$ can be written as,

$$\begin{cases} H(X; \boldsymbol{W}, b) = H_1 \odot H_2 \odot \cdots \odot H_P \\ H_i = \sigma(X\boldsymbol{W}_i + b_i), i = 1, 2, \cdots, P. \end{cases} \tag{18}$$

Here, $\boldsymbol{W} \triangleq [\boldsymbol{W}_1, \boldsymbol{W}_2, \cdots, \boldsymbol{W}_P]$ and $b \triangleq [b_1, b_2, \cdots, b_P]$.

First, if we fix $\boldsymbol{W}$ and b, then Eq.(17) has the closed-form solution,

$$\boldsymbol{\beta} = (H^T H + \lambda \boldsymbol{I})^{-1} HY. \tag{19}$$

Second, if we fix $\boldsymbol{\beta}$ by the chain rule, then we have

$$\frac{\partial J}{\partial \boldsymbol{W}_i} = \frac{\partial H_i}{\partial \boldsymbol{W}_i} \cdot \frac{\partial H}{\partial H_i} \cdot \frac{\partial J}{\partial H}, \tag{20}$$

where $i = 1, 2, \cdots, P$. In particular, from the definition of the Khatri-Rao product, we obtain

$$\left[\frac{\partial H}{\partial H_i(j, r)}\right] = H_1 \odot \cdots \odot M_i^{(j,r)} \odot \cdots \odot H_P, \tag{21}$$

where symbol $H_i(j, r)$ denotes the (j, r)th element of matrix H_i and $M_i^{(j,r)}$ means that the mask of matrix H_i is 0 everywhere except for a value of 1 in the (j, r)th position. Furthermore, we have

$$\frac{\partial J}{\partial \boldsymbol{H}_i(j, r)} = < \Theta, \frac{\partial H}{\partial H_i(j, r)} >, \tag{22}$$

where '$<, >$' denotes matrix inner product and Θ denotes

$$\Theta = \frac{\partial J}{\partial H} = 2(Y - H\boldsymbol{\beta})\boldsymbol{\beta}^T. \tag{23}$$

We can also compute

$$\frac{\partial H_i}{\partial \boldsymbol{W}_i} = XT_i. \tag{24}$$

Here, T_i can be obtained by

$$T_i(u, v) = \begin{cases} 1, & \text{if } F_i(u, v) > 0, \\ 0, & \text{if } F_i(u, v) \le 0, \end{cases} \tag{25}$$

where $F_i = \boldsymbol{W}_i x + b_i$ and the active function is sigmoid function.

Finally, from Eq.(22) and Eq.(24), we can easily obtain the $\nabla_{\boldsymbol{W}_i} J$ required to update the weight matrix $\boldsymbol{W}$. Each parameter b_i can be updated similarly. Especially, in the training process, we have chosen the mini-batch gradient descent (MBGD) for updating parameters W_i and b_i, $i = 1, 2, \cdots, P$ for each ST-ELM.

D. REDUCING THE NUMBER OF HIDDEN PARAMETERS

For ELM, the number of nodes in the hidden layer is s in Eq.(2) and Eq.(4), and N is the number of distinct training samples. Generally, the upper bound of the required number of hidden nodes is less than the number of distinct training samples in most cases, that is, $s \le N$. In addition, the larger the s, the better generalization performance of ELM can be guaranteed [1], [42]–[44]. For the ELM without randomization, and the weights $W \in \mathbb{R}^{m \times s}$ and $b \in \mathbb{R}^{1 \times s}$ also needs to be optimized, then the number of hidden parameters to learn is $(m + 1) \times s$. We can embed tensor operation in ELM (ST-ELM) to reduce the number of hidden parameters W, b in ELM. Tensor manipulation brings two advantages. One is to preserve the structural information contained in the data; the other is the advantage of computing, that is, lower

computational complexity. And expressly, assume that we have P paths in Eq.(11), s can be written as,

$$s \triangleq \prod_{i=1}^{P} s_i \tag{26}$$

then the number of hidden parameters ($W_i \in \mathbb{R}^{m\times s_i}$, $b_i \in \mathbb{R}^{1\times s_i}$) for ith path to learn is $(m+1)\times s_i$. For ST-ELM without randomization, the number of hidden parameters is

$$(m+1)\times\left(\sum_{i=1}^{P} s_i\right) \tag{27}$$

In general, when s is large, we have,

$$\left(\sum_{i=1}^{P} s_i\right) \leq s \tag{28}$$

Therefore, on the premise that randomness is removed for ELM and ST-ELM, the number of implicit parameters that ST-ELM needs to learn is relatively small.

IV. EXPERIMENTAL RESULTS

In this section, we report on the extensive experiments conducted to evaluate the effectiveness and efficiency of the proposed SDT-ELM framework. Three well-known image databases for benchmarking were used, namely MNIST, NORB, and Fashion MNSIT.

In addition, the SDT-ELM was compared with other methods involving both shallow and deep networks, as listed below.

DSAE: a deep stacked AE network based on a sparse AE network [7], [33].

PCANet: a simple deep learning architecture, which is a cascaded network based on principal component analysis (PCA) and binary quantization [34].

DBN: a stacking architecture based on the restricted Boltzmann machine (RBM) [4].

ELM: the original ELM.

H-ELM: an ELM for multilayer perception based on a sparse AE.

ELM-LRF: a local receptive field (LRF) implementation based on the ELM [35].

BLS: an effective and efficient learning system that does not have a need for deep architecture.

CNN: in this paper, the classical LeNet5 [36] is included to enable a fair comparison.

All the simulations for these algorithms were carried out in the MATLAB2016b environment on a machine with an $Intel^{®}Xeon(R)CPUE5-2630v3@2.40\ GHZ \times 16$ running the Ubuntu 14.04 LTS with a 64-bit OS using *Gallium*0.4 on *llvmpipe*.

A. DATA DESCRIPTION

MNIST [37]: This dataset comprises 70, 000 handwritten digits partitioned into a training set of 60, 000 samples and a test set of 10, 000 samples. The task is to classify each 28×28 image into one of the 10 digits, as shown on the left of Fig. 6.

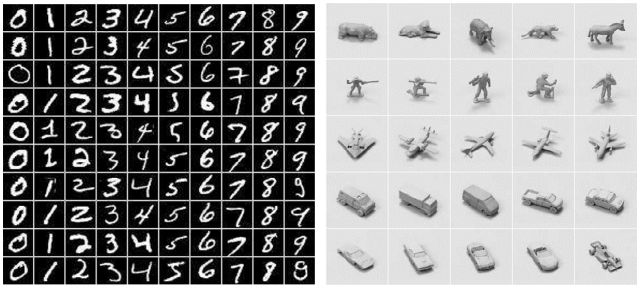


FIGURE 6. Examples from the MNIST (left) and NORB (right) datasets.

Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

FIGURE 7. Examples from the Fashion MNIST dataset.

NORB [38]: This dataset comprises images of 50 different $3D$ toy objects belonging to five disjoint categories. It comprises 48, 600 images partitioned into a training set of 24, 300 stereo images of 25 objects and a testing set of 24, 300 images of the remaining 25 objects. Each image has a $2\times 32\times 32$-voxel format, as shown on the right of Fig. 6.

Fashion MNIST [39]: This is a new dataset comprising 28×28 gray-scale images of 70, 000 fashion products from 10 categories, with 7, 000 images per category, as shown in Fig. 7. The training set contains 60, 000 images and the test set contains 10, 000 images.

B. SDT-ELM PERFORMANCE ANALYSIS

First, we evaluated the performance of the ST-ELM architecture with the MNIST database. The hidden-layer configuration of the ST-ELM can be described in terms of the four-tuple $[\rho, \sigma, P, V]$. Given the parameter values $\rho = 0.05$ and $\sigma(t) = 1/(1+\exp(-t))$, we can observe the impact of the relationship between P and V on network performance. The relationship between (P, V) and s is

$$s = \prod_{i=1}^{P} V(i) = \prod_{i=1}^{P} s_i, \tag{29}$$

where $V(i) = s_i$.

Without loss of generality, we investigated the three cases $s = 10,000$, 8, 100, and 1600. Clearly, the decomposition of

TABLE 2. ST-ELM ($L = 1$): Testing accuracy rate and training time for hidden-layer configuration s with MNIST.

P	V	Accuracy	Training Time
1	10000	97.67%	47.857 (s)
2	[100, 100]	97.45%	32.921 (s)
4	[10, 10, 10, 10]	93.43%	32.387 (s)
6	[2, 2, 5, 5, 10, 10]	90.17%	32.175 (s)
1	8100	97.57%	34.068 (s)
2	[90, 90]	97.34%	22.704 (s)
4	[9, 9, 10, 10]	92.72%	23.101 (s)
6	[3, 3, 3, 3, 10, 10]	89.90%	22.486 (s)
1	1600	94.95%	4.8874 (s)
2	[40, 40]	95.05%	2.1513 (s)
4	[5, 5, 8, 8]	85.41%	2.0397 (s)
6	[2, 2, 4, 4, 5, 5]	77.74%	2.3165 (s)

s can vary widely (from Eq.(29)), but we only experimented with the three cases given in TABLE.2. Note that, for the ST-ELM, the regularization parameter λ was 2^{-20} in Eq.(17). After averaging the results from 30 trial simulations for each case, we obtained the testing accuracies and training times shown in TABLE.2. Further, from the experimental results presented in TABLE.2, when P is larger and the elements in vector V are relatively smaller, the corresponding ST-ELM behaves with a significantly lower accuracy rate. In addition, when the performance for $P = 2$ is compared with that for $P = 1$, the generalization performance of the ST-ELM is almost unchanged, but the training time is significantly reduced.

In the second set of experiments, we set parameter values $P = 2$, $s_1 = s_2$ (i.e., $V(1) = V(2)$), $\rho = 0.05$, and $\lambda = 2^{-20}$, and observed the effect of parameter L on network performance.

Here, we investigated the three cases $s = 1,024$, 2, 304, and 4, 096. To enable comparison with the classical ELM, and considering the time-consuming training phase for the SDT-ELM, we chose the L values for the various s values given in TABLE.3. Note that the condition for an SDT-ELM to degenerate to an ELM with sparsity is $P = L = 1$. After averaging the results of 30 trial simulations for each case, the testing accuracies and training times are as shown in TABLE.3.

Clearly, with an increasing L, the generalization performance for the SDT-ELM gradually improves. In addition, although the ELM is a special case of the SDT-ELM, the SDT-ELM can effectively mitigate the first drawback of the ELM. That is, the accuracy of the ELM is dramatically influenced by the large number of hidden neurons. For example, the performance of an SDT-ELM with $[L, \rho, \sigma, P, V] = [15, 0.05, Sigmoid, 2, (48, 48)]$ is better than an ELM with $[L, \rho, \sigma, P, V] = [1, 0.05, Sigmoid, 1, 10000]$.

In addition, we analysed the convergence trend of the SDT-ELM with respect to parameter L. The results, as shown in in Fig. 8, indicate that, when a new ST-ELM sub-module

TABLE 3. SDT-ELM: Testing accuracy rate and training time for parameter L with MNIST.

	L	Accuracy	Training Time
$s = 1024$ $(s_1 = s_2 = 32)$	1	93.82%	1.5010 (s)
	5	95.37%	7.1948 (s)
	10	96.11%	14.043 (s)
	15	96.69%	22.209 (s)
	20	97.37%	27.566 (s)
	30	97.86%	38.377 (s)
	40	98.43%	60.410 (s)
$s = 2304$ $(s_1 = s_2 = 48)$	1	95.79%	3.4418 (s)
	5	97.11%	17.186 (s)
	10	97.59%	34.381 (s)
	15	97.97%	51.558 (s)
	20	98.68%	68.669 (s)
$s = 4096$ $(s_1 = s_2 = 64)$	1	96.65%	7.7980 (s)
	5	97.96%	39.979 (s)
	10	98.83%	80.832 (s)

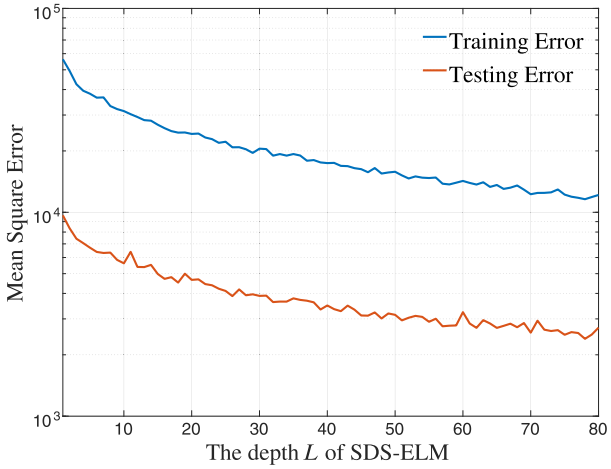


FIGURE 8. MNIST: The trend of the mean square error curve for increasing L.

is added to the SDT-ELM, the objective loss of the SDT-ELM with hidden-layer configuration $[L, \rho, \sigma, P, V] = [80, 0.05, Sigmoid, 2, (32, 32)]$ can appear in the convergence trend.

Finally, one convenient aid to setting the parameter V is illustrated in TABLE.4. For a given number of implicit hidden units s and a given value for L (in this example, $s = 1,024$ and $L = 80$), then using a 'symmetric' V is relatively better than using an 'asymmetric' V for $P = 2$. In particular, lower values in V resulted in poorer network performance.

C. COMPARISON WITH STATE-OF-THE-ART ALGORITHMS

For the third group of experiments, the hyper-parameters for the SDT-ELM were set as follows. The sparsity degree $\rho = 0.05$, the active function used $Sigmoid$, $P = 2$, $s_1 = s_2$ (to give a 'symmetric' V), and the depth L of the SDT-ELM network was obtained by searching for $L \in [1, 100]$. In addition, the randomly chosen parameters $\boldsymbol{W}$ and b followed a

TABLE 4. The influence of different configuration of V on SDT-ELM performance.

V	Accuracy	Training Time
$[32, 32]$	98.57%	124.66 (s)
$[16, 64]$	98.31%	129.53 (s)
$[64, 16]$	97.94%	130.60 (s)
$[8, 128]$	97.63%	147.31 (s)
$[128, 8]$	97.37%	149.09 (s)
$[4, 256]$	97.04%	157.08 (s)
$[256, 4]$	96.63%	157.20 (s)
$[2, 512]$	95.53%	198.69 (s)
$[512, 2]$	96.21%	196.31 (s)

TABLE 5. Comparison of learning accuracy with MNIST.

Methods	Accuracy	Training Time
DSAE	97.56 %	797.09 (s)
DSN	98.23 %	1029.1 (s)
PCANet	98.34 %	4066.5 (s)
DBN	98.37 %	2376.8 (s)
ELM	97.85 %	119.95 (s)
H-ELM	98.93 %	46.585 (s)
ELM-LRF	96.75 %	37.470 (s)
BLS	98.33 %	53.546 (s)
CNN	98.71 %	3341.1 (s)
ST-ELM	96.66 %	10.428 (s)
SDT-ELM(R)	99.04 %	239.12 (s)
SDT-ELM(D)	**99.25** %	687.56 (s)

sparse normal distribution, (i.e., *sprandn* in MATLAB could be used directly). Note that all ST-ELM sub-modules had the same hidden-layer configuration in all experiments in this group.

1) MNIST DATASET

For reference, the structures of DSAE, DBN, ELM, and H-ELM were $1,000 \rightarrow 500 \rightarrow 100$, $500 \rightarrow 500 \rightarrow 2,000$, $12,000$, and $300 \rightarrow 300 \rightarrow 12,000$, respectively. We used a state-vector-machine (SVM) classifier for PCANet, setting the filter size $k_1 = k_2 = 7$, and the number of PCA filters $L_1 = L_2 = 8$, with the overlapping region between blocks being half the block size. For the ELM-LRF, the parameters that required tuning included the size of the receptive field (5×5), the number of features maps (10), the pooling size (3), and the regularization parameter C (0.01). The BLS was structured as 100×10 feature nodes and $11,000$ enhancement nodes. For the DSN [8], the hidden-layer size was $1,000$.

The test results are shown in TABLE.5. If we use $s = 5,041$ (i.e., $V = [71, 71]$) and $L = 26$, then an SDT-ELM with randomness (SDT-ELM(R)) can achieve 99.04% accuracy. Furthermore, an SDT-ELM without randomness (SDT-ELM(D))[1] can achieve 99.25% accuracy, but the time consumption is nearly three times that for an SDT-ELM(R).

TABLE 6. Comparison of learning accuracy with NORB.

Methods	Accuracy	Training Time
DSAE	86.27 %	626.45 (s)
DSN	88.27 %	1687.3 (s)
PCANet	90.04 %	4337.8 (s)
DBN	88.27 %	5581.9 (s)
ELM	90.37 %	94.892 (s)
H-ELM	89.66 %	71.227 (s)
ELM-LRF	83.91 %	19.980 (s)
BLS	87.60 %	21.884 (s)
CNN	86.74 %	5597.8 (s)
ST-ELM	82.21 %	0.8626 (s)
SDT-ELM(R)	91.05 %	14.724 (s)
SDT-ELM(D)	**92.43** %	472.16 (s)

2) NORB DATASET

This database was used to further demonstrate the advantages of an SDT-ELM. For reference, the structures of the DSAE, DBN, ELM, and H-ELM were $1,000 \rightarrow 500 \rightarrow 100$, $4,000 \rightarrow 4,000 \rightarrow 4,000$, $15,000$, and $3,000 \rightarrow 3,000 \rightarrow 15,000$, respectively. We again used an SVM classifier for PCANet, setting the filter sizes as $k_1 = k_2 = 6$ and the number of PCA filters as $L_1 = L_2 = 8$, with the overlapping region between blocks being half the block size. For the ELM-LRF, parameters that required tuning included the size of receptive field (4×4), the number of feature maps (3), the pooling size (3), and the regularization parameter C (0.01). The BLS was structured as 100×10 feature nodes and $9,000$ enhancement nodes. For the DSN [8], the hidden-layer size was $2,000$.

The test results are shown in TABLE.6. If we use $s = 1,024$ (i.e. $V = [32, 32]$) and $L = 26$, then the SDT-ELM(D) achieves higher accuracy (92.43%) than any of the state-of-the-art training algorithms. Note also that the number of hidden nodes is again relatively small compared with those for the ELM, H-ELM, or BLS.

3) FASHION MNIST DATASETS

In the experiments with this dataset, the structures of the DSAE, DBN, ELM, and H-ELM were $1,000 \rightarrow 500 \rightarrow 100$, $500 \rightarrow 500 \rightarrow 2,000$, $10,000$, and $1000 \rightarrow 1000 \rightarrow 10000$, respectively. We again used an SVM classifier for PCANet, setting the filter size $k_1 = k_2 = 4$, and the number of PCA filters $L_1 = L_2 = 8$, with the overlapping region between blocks being half the block size. For the ELM-LRF, the parameters that required tuning included the size of receptive field (5×5), the number of features

[1] Here, the 'D' means deterministic, removing the randomness by gradient update.

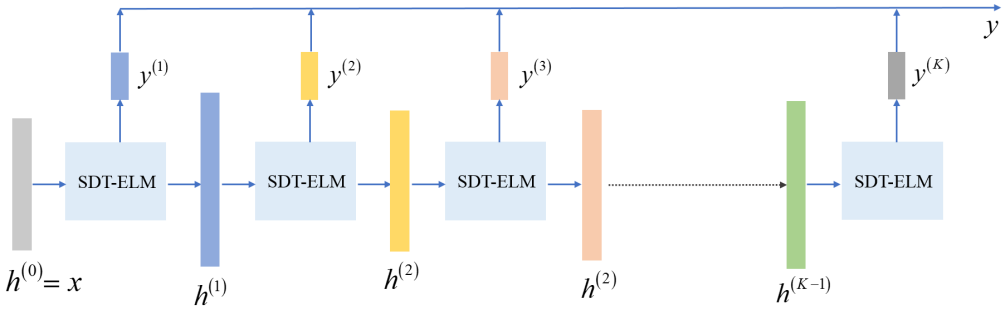


FIGURE 9. An improved architecture based on SDT-ELM.

TABLE 7. Comparison of learning accuracy with Fashion MNIST.

Methods	Accuracy	Training Time
DSAE	88.94 %	4151.2 (s)
DSN	89.24 %	6831.9 (s)
PCANet	90.01 %	3572.6 (s)
DBN	86.99 %	10487 (s)
ELM	88.87 %	24.449 (s)
H-ELM	88.55 %	46.472 (s)
ELM-LRF	88.14 %	9113.6 (s)
BLS	89.58 %	62.656 (s)
CNN	88.94 %	4724.7 (s)
SD-ELM	87.02 %	6.9035 (s)
SDT-ELM(R)	89.96 %	132.95 (s)
SDT-ELM(D)	**91.09** %	1472.4 (s)

maps (10), the pooling size (2), and the regularization parameter C (0.05). The BLS was structured as 10×10 feature nodes and $12,000$ enhancement nodes. For the DSN [8], the hidden-layer size was $2,000$.

The testing results are shown in TABLE.7. If we use $s = 4,096$ (i.e., $V = [64, 64]$) and $L = 18$, then the proposed SDT-ELM again achieves a relatively better performance. When compared with other training algorithms, the SDT-ELM leads to a promising set of principles for network design. In addition, it is important to note that that the number of hidden nodes is small, compared with those required for the ELM, H-ELM, or BLS.

D. ANALYSIS OF EXPERIMENTAL RESULTS

Experimental results in all above pattern classification tasks demonstrate the effectiveness of the SDT-ELM and the associated (remove randomness) learning methods in a consistent manner. In particular, compared with the classical model presented in this paper, the SDT-ELM with randomness removed can obtain the optimal test accuracy. Also, compared with the model with randomness, the model without randomness usually has higher training time consumption in parameter optimization.

SDT-ELM can obtain excellent generalization performance in the pattern classification task mainly for the following two advantages. The first advantage is that the main parameter estimation burden in SDT-ELM is shifted to a convex sub-problem with a closed form solution, which is different from the traditional deep neural network parameter optimization. The second advantage is that SDT-ELM is equipped with the new stacking mechanism where the hidden representations can be concatenated with the input data in stacking the SDT-ELM blocks, which is different from the traditional deep neural network stacked mechanism.

V. DISCUSSION AND CONCLUSION

In this paper, we have presented a novel deep-architecture the SDT-ELM for pattern classification tasks, based on the universal approximation capability of an ELM. The principle novelty is to split the original large hidden layer (in each block) into several smaller ones, and through their multiplicative outer product and the associated tensor weights, to create multiple non-linear models exploiting the higher-order covariance structure in hidden feature interactions. Further, It can offer an alternative approach to deep learning and supporting architectures.

To benefit from the original ELM, the proposed SDT-ELM is simple both in theory and implementation. Compared with other some deep learning techniques, a similar or better generalization performance is achievable. One advantage is that it can effectively reduce the number of hidden-layer parameters by using tensor operations, thereby addressing a significant drawback encountered by the classical ELM. Our proposed ST-ELM and SDT-ELM implementations were tested on three public image datasets, demonstrating the competitive performance of the SDT-ELM against the ELM, H-ELM, and BLS. A second advantage is that the SDT-ELM allows the random hidden nodes to be added block by block (via ST-ELM sub-modules), with all these elements having the same hidden-layer configuration. In particular, the SDT-ELM without randomness demonstrated better learning accuracy than any of the other competing systems. In summary, an SDT-ELM can produce a compact pattern classification model of high generalization with relatively fast training time. Moreover, the SDT-ELM is equipped with the new stacking mechanism where the more compact hidden representations can be concatenated with the input data in stacking the ST-ELM.

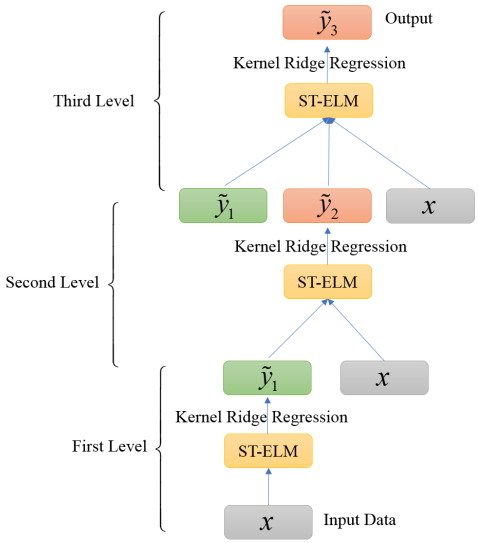


FIGURE 10. An improved kernel architecture with three layers based on ST-ELM.

Note that the SDT-ELM does not aim to discover transformed feature representations. However, encouraged by our recent results, we aim to explore the significant extension shown in Fig. 9, where we combine the hierarchical feature representation in classical deep learning (e.g., DSAE) with the computational advantages of an SDT-ELM.

Briefly, the output of each module of an SDT-ELM at level k would include two parts, namely a prediction label $y^{(k)}$ and a feature representation $h^{(k)}$, where $k = 1, 2, \cdots, K$. The ultimate prediction y could then be obtained by

$$y = \sum_{k=1}^{K} \lambda_k \cdot y^{(k)}, \tag{30}$$

where λ_k indicates the contribution of the lth level to the final target y. We hope that this extended architecture will be able to achieve better and more robust results than existing state-of-the-art methods.

Further, from the network architecture point of view, it deserves further consideration is that the parallelism in learning the SDT-ELM can be likely implemented either in a CPU cluster or in a GPU cluster. If SDT-ELM parallelized implementation can be achieved, then we expect meaningful improvements in pattern recognition tasks and robust automatic speech recognition tasks.

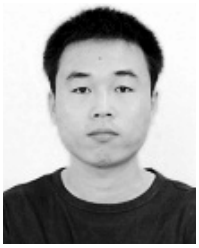
In addition, Kernel Deep Stacking Networks (KDSN) belong to the class of deep learning algorithms, which have emerged as powerful tools to discover complex non-linear relationships. Compared to traditional neural networks that relay on back-propagation algorithms, KDSN has the advantage that high-level feature is obtained efficiently by fitting a series of kernel ridge regression models. The direction of further research is to combine KDSN with ST-ELM to construct a new framework (see Fig.10) and algorithm for deep learning. We expect greater success with the use of SDT-ELM and its variants in some applications.

Finally, From the perspective of practical application, the proposed SDT-ELM can also be used for three different computer vision tasks (object detection, recognition, and tracking [40], [41]), This is one of the applications tasks we will be verifying in the future. In each application, SDT-ELM can be used for feature extraction and pattern classification.

REFERENCES

[1] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, 2006.
[2] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 513–529, Apr. 2012.
[3] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 809–821, Apr. 2015.
[4] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[5] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.
[6] C. L. P. Chen and Z. L. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 10–24, Jan. 2018.
[7] J. Gehring, Y. Miao, F. Metze, and A. Waibel, "Extracting deep bottleneck features using stacked auto-encoders," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, vol. 1, no. 1, pp. 3377–3381.
[8] L. Deng, B. Hutchinson, and D. Yu, "Parallel training of deep stacking networks," *Interspeech*, vol. 13, no. 4, pp. 52–61, 2012.
[9] B. Hutchinson, L. Deng, and D. Yu, "Tensor deep stacking networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1944–1957, Aug. 2013.
[10] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 1, no. 1. Cambridge, MA, USA: MIT Press, 2006, pp. 153–160.
[11] L. Jiao, J. Zhao, F. Liu, and S. Yang, "Deep learning," in *Optimization and Recognition*. Beijing, China: Tsinghua Univ. Press, 2017.
[12] G. S. Stiles and D.-L. Denq, "On the effect of noise on the Moore–Penrose generalized inverse associative memory," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-7, no. 3, pp. 358–360, May 1985.
[13] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.
[14] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, nos. 16–18, pp. 3056–3062, 2007.
[15] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2004, vol. 2, no. 1, pp. 985–990.
[16] Q.-Y. Zhu, A. K. Qin, P. N. Suganthan, and G.-B. Huang, "Evolutionary extreme learning machine," *Pattern Recognit.*, vol. 38, no. 10, pp. 1759–1763, Oct. 2005.
[17] M.-B. Li, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "Fully complex extreme learning machine," *Neurocomputing*, vol. 68, pp. 306–314, Oct. 2005.
[18] G.-B. Huang, M.-B. Li, L. Chen, and C.-K. Siew, "Incremental extreme learning machine with fully complex hidden nodes," *Neurocomputing*, vol. 71, nos. 4–6, pp. 576–583, 2008.
[19] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.
[20] H.-J. Rong, Y.-S. Ong, A.-H. Tan, and Z. Zhu, "A fast pruned-extreme learning machine for classification problem," *Neurocomputing*, vol. 72, no. 1, pp. 359–366, 2008.
[21] H. X. Tian and Z. Z. Mao, "An ensemble ELM based on modified AdaBoost.RT algorithm for predicting the temperature of molten steel in ladle furnace," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 1, pp. 73–80, Jan. 2010.

[22] M. van Heeswijk, Y. Miche, E. Oja, and A. Lendasse, "GPU-accelerated and parallelized ELM ensembles for large-scale regression," *Neurocomputing*, vol. 74, no. 16, pp. 2430–2437, 2011.
[23] E. Soria-Olivas, J. Gomez-Sanchis, J. D. Martin, J. Vila-Frances, M. Martinez, J. R. Magdalena, and A. J. Serrano, "BELM: Bayesian extreme learning machine," *IEEE Trans. Neural Netw.*, vol. 22, no. 3, pp. 505–509, Mar. 2011.
[24] Y. Yang, Y. Wang, and X. Yuan, "Bidirectional extreme learning machine for regression problem and its learning effectiveness," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 9, pp. 1498–1505, Sep. 2012.
[25] P. Horata, S. Chiewchanwattana, and K. Sunat, "Robust extreme learning machine," *Neurocomputing*, vol. 102, no. 1, pp. 31–44, Feb. 2013.
[26] Z. Bai, G.-B. Huang, D. Wang, H. Wang, and M. B. Westover, "Sparse extreme learning machine for classification," *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1858–1870, Oct. 2014.
[27] W. Zhu, J. Miao, L. Qing, and G.-B. Huang, "Hierarchical extreme learning machine for unsupervised representation learning," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2015, vol. 1, no. 1, pp. 1–8.
[28] H. Zhou, G.-B. Huang, Z. Lin, H. Wang, and Y. C. Soh, "Stacked extreme learning machines," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 2013–2025, Sep. 2015.
[29] M. D. Tissera and M. D. McDonnell, "Deep extreme learning machines: Supervised autoencoding architecture for classification," *Neurocomputing*, vol. 174, pp. 42–49, Jan. 2016.
[30] S. Pang and X. Yang, "Deep convolutional extreme learning machine and its application in handwritten digit classification," *Comput. Intell. Neurosci.*, vol. 2016, pp. 1–10, Jul. 2016.
[31] W. Ibrahim and M. S. Abadeh, "Protein fold recognition using deep kernelized extreme learning machine and linear discriminant analysis," *Neural Comput. Appl.*, vol. 5, no. 4, pp. 1–14, 2018.
[32] S. Liu, "Several inequalities involving Khatri–Rao products of positive semidefinite matrices," *Linear Algebra Appl.*, vol. 354, nos. 1–3, pp. 175–186, 2002.
[33] N. Elaraby, M. Elmogy, and S. Barakat, "Large scale sensor data processing based on deep stacked autoencoder network," *J. Theor. Appl. Inf. Technol.*, vol. 95, no. 21, pp. 5907–5923, 2017.
[34] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "PCANet: A simple deep learning baseline for image classification?" *IEEE Trans. Image Process*, vol. 24, no. 12, pp. 5017–5032, Dec. 2015.
[35] G.-B. Huang, Z. Bai, L. L. C. Kasun, and C. M. Vong, "Local receptive fields based extreme learning machine," *IEEE Comput. Intell. Mag.*, vol. 10, no. 2, pp. 18–29, May 2015.
[36] A. El-Sawy, H. El-Bakry, and M. Loey, "CNN for handwritten arabic digits recognition based on LeNet-5," in *Proc. Int. Conf. Adv. Intell. Syst. Inform.*, 2016, pp. 566–575.
[37] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the Web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.
[38] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2004, vol. 2, no. 11, pp. 97–104.
[39] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*. [Online]. Available: https://arxiv.org/abs/1708.07747
[40] Y. Zhao, Y. Shen, A. Bernard, C. Cachard, and H. Liebgott, "Evaluation and comparison of current biopsy needle localization and tracking methods using 3D ultrasound," *Ultrasonics*, vol. 73, pp. 206–220, Jan. 2017.
[41] G. Sun, L. Wu, Z. Kuang, Z. Ma, and J. Liu, "Practical tracking control of linear motor via fractional-order sliding mode," *Automatica*, vol. 94, no. 1, pp. 221–235, Aug. 2018.
[42] C. M. Wong, C. M. Vong, P. K. Wong, and J. Cao, "Kernel-based multilayer extreme learning machines for representation learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 3, pp. 757–762, Mar. 2018.
[43] H. Li, H. Zhao, and H. Li, "Neural-response-based extreme learning machine for image classification," *IEEE Trans. Neural Netw. Learn. Sys.*, vol. 30, no. 2, pp. 539–552, Feb. 2019.
[44] J. Hu, J. Zhang, C. Zhang, and J. Wang, "A new deep neural network based on a stack of single-hidden-layer feedforward neural networks with randomly fixed hidden neurons," *Neurocomputing*, vol. 171, pp. 63–72, Jan. 2016.



JIN ZHAO received the M.S. degree in mathematics and applied mathematics from Shaanxi Norm University, Xi'an, China, in 2012. He is currently pursuing the Ph.D. degree with the Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education of China, Xidian University, Xi'an. His research interests include sparse representation and modeling, deep learning algorithm design, and performance Analysis.

LICHENG JIAO (SM'89–F'18) received the B.S. degree from Shanghai Jiao Tong University, Shanghai, China, in 1982, and the M.S. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 1984 and 1990, respectively. He is currently a Professor with the School of Artificial Intelligence, Xidian University, Xi'an. His research interests include image processing, natural computation, machine learning, and intelligent information processing.

Dr. Jiao is a member of the IEEE Xi'an Section Execution Committee and a Committee Member of the Chinese Committee of Neural Networks. He is the Chairman of Awards and Recognition Committee and the Vice Board Chairperson of the Chinese Association of Artificial Intelligence. He is the Councilor of the Chinese Institute of Electronics and an Expert of the Academic Degrees Committee of the State Council.