Extreme Learning Machine for Multilayer Perceptron

Jiexiong Tang, *Student Member, IEEE*, Chenwei Deng, *Senior Member, IEEE*, and Guang-Bin Huang, *Senior Member, IEEE*

***Abstract*— Extreme learning machine (ELM) is an emerging learning algorithm for the generalized single hidden layer feedforward neural networks, of which the hidden node parameters are randomly generated and the output weights are analytically computed. However, due to its shallow architecture, feature learning using ELM may not be effective for natural signals (e.g., images/videos), even with a large number of hidden nodes. To address this issue, in this paper, a new ELM-based hierarchical learning framework is proposed for multilayer perceptron. The proposed architecture is divided into two main components: 1) self-taught feature extraction followed by supervised feature classification and 2) they are bridged by random initialized hidden weights. The novelties of this paper are as follows: 1) unsupervised multilayer encoding is conducted for feature extraction, and an ELM-based sparse autoencoder is developed via ℓ_1 constraint. By doing so, it achieves more compact and meaningful feature representations than the original ELM; 2) by exploiting the advantages of ELM random feature mapping, the hierarchically encoded outputs are randomly projected before final decision making, which leads to a better generalization with faster learning speed; and 3) unlike the greedy layerwise training of deep learning (DL), the hidden layers of the proposed framework are trained in a forward manner. Once the previous layer is established, the weights of the current layer are fixed without fine-tuning. Therefore, it has much better learning efficiency than the DL. Extensive experiments on various widely used classification data sets show that the proposed algorithm achieves better and faster convergence than the existing state-of-the-art hierarchical learning methods. Furthermore, multiple applications in computer vision further confirm the generality and capability of the proposed learning scheme.**

***Index Terms*— Deep learning (DL), deep neural network (DNN), extreme learning machine (ELM), multilayer perceptron (MLP), random feature mapping.**

I. Introduction

DURING the past years, extreme learning machine (ELM) [1] has been becoming an increasingly significant research topic for machine learning and artificial intelligence, due to its unique characteristics, i.e., extremely fast training, good generalization, and universal approximation/classification capability. ELM is an effective solution for the single hidden layer feedforward networks (SLFNs), and has been demonstrated to have excellent learning accuracy/speed in various applications, such as face classification [2], image segmentation [3], and human action recognition [4].

Unlike the other traditional learning algorithms, e.g., back propagation (BP)-based neural networks (NNs), or support vector machine (SVM), the parameters of hidden layers of the ELM are randomly generated and need not to be tuned, thus the hidden nodes could be established before the training samples are acquired. Theoretically, Huang *et al.* [5], [6] have proved that the SLFNs with randomly generated hidden neurons and the output weights tuned by regularized least square maintain its universal approximation capability, even without updating the parameters of hidden layers. In addition, solving the regularized least squares problem in ELM is faster than that of the quadratic programming problem in standard SVM or gradient method in traditional BP-based NNs. Thus, ELM tends to achieve faster and better generalization performance than those of NNs and SVM [1], [5], [7].

Recently, ELM has been extensively studied, and remarkable contributions have been made both in theories and applications. The universal capability of ELM has been extended into kernel learning, and it turns out that the ELM is suitable for a wide type of feature mappings, rather than the classical ones [1]. An incremental ELM was proposed in [5], where the hidden nodes are added incrementally and the output weights are determined analytically. ELM has also been transformed in an online sequential version [8], and the original data can be input one-by-one or chunk-by-chunk (a block of data) with fixed or varying chunk size. Huang *et al.* [9] extend the ELM for both semisupervised and unsupervised tasks based on the manifold regularization, and the unlabeled or partially labeled samples are clustered using ELM.

However, the aforementioned works still face some issues when dealing with natural scenes (e.g., visual and acoustic signals), or practical applications (e.g., image classification, voice recognition, and so on). That is, the original ELM and/or its variants mainly focus on classification. However, feature learning is often required before classification is conducted in many applications. Thus, multilayer solutions are usually needed. Kasun *et al.* [10] attempt to develop a multilayer

Manuscript received September 2, 2014; revised April 19, 2015; accepted April 19, 2015. Date of publication May 7, 2015; date of current version March 15, 2016. This work was supported in part by the Excellent Young Scholars Research Fund of Beijing Institute of Technology under Grant 2013YR0508 and in part by the National Natural Science Foundation of China under Grant 61301090. *(Corresponding author: Chenwei Deng.)*

J. Tang and C. Deng are with the School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China (e-mail: jiexiongtang@bit.edu.cn; cwdeng@bit.edu.cn).

G.-B. Huang is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: egbhuang@ntu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2015.2424995

learning architecture using ELM-based autoencoder as its building block. The original inputs are decomposed into multiple hidden layers, and the outputs of the previous layer are used as the inputs of the current one. The autoencoder in [10] is simply stacked layer by layer in the hierarchical structure. Before the supervised least mean square optimization, the encoded outputs are directly fed into the last layer for decision making, without random feature mapping. This framework has not well exploited the advantages of ELM theories, as Huang *et al.* [5] have demonstrated that the universal approximation capability of ELM cannot be guaranteed without random projections of the inputs. Therefore, it is obvious that the potential in multilayer implementation of the ELM has not been fully deployed, and further research on an ELM-based multilayer perceptron (MLP) is badly in need, toward faster training speed and better feature learning and classification performance.

Meanwhile, another leading trend for hierarchical learning is called deep learning (DL) [11], or deep NNs (DNNs) [12]. Similarly, the deep architecture extracts features by a multilayer feature representation framework, and the higher layers represent more abstract information than those from the lower ones. From concept point of view [11]–[15], DL is a BP learning for multilayer networks with unsupervised initialization instead of the conventional random initialization. In other words, DL considers multilayer as a whole with unsupervised initialization, and after such initialization the entire network will be trained by BP-based NNs, and all of the layers are hard coded together. It should be mentioned that all the hidden parameters in DL framework need to be fine-tuned multiple times, for the entire system. And thus the training of DL is too cumbersome and time consuming.

Seen from the above analysis, the existing learning algorithms (including the ELM-based and DL-based ones) for MLPs cannot achieve excellent generalization performance with fast learning speed. In this paper, inspired by the MLP theories, we extend the ELM and propose a hierarchical ELM (H-ELM) framework for MLPs. The proposed H-ELM further improves the learning performance of the original ELM, while maintaining its advantages of training efficiency. The contributions of the proposed work are as follows.

1) A new ELM-based autoencoder is developed via ℓ_1-norm optimization. Unlike the existing autoencoders used in DL, i.e., BP-based NNs, the input weights and hidden node biases of the proposed ELM autoencoder are established by searching the path back from a random space. The ELM theory in [5] has proved that random feature mapping (with almost any nonlinear piecewise activation function) can provide universal approximation capability, and by doing so, more important information can be exploited for hidden layer feature representation. As compared with the ℓ_2-norm ELM autoencoder in [10], here the ℓ_1 penalty is applied to generate more sparse and meaningful hidden features.
2) A new H-ELM framework is proposed for an effective and efficient MLP. The proposed H-ELM consists of two main components: a) unsupervised multilayer feature encoding and b) supervised feature classification. For feature representation or extraction, the proposed ELM sparse autoencoder is utilized, and each layer in the stack architecture can be considered as a separate part (or an autonomous subsystem/submodule); for feature classification, the obtained high-level features are first scattered by a random matrix, and then the original ELM is applied for final decision making. Note that based on ELM theory [5], high dimensional nonlinear transform of extracted features can further improve feature classification precision. Moreover, different from the greedy layerwise learning in DL framework, feature extraction and classification are two separate autonomous parts in the proposed H-ELM framework, and parameter fine-tuning is not required for the entire system combined by feature extraction and classification, and thus the training speed can be much faster than the traditional BP-based DL.
3) To demonstrate the advantages of the proposed H-ELM framework, several H-ELM-based feature extraction and classification algorithms are developed for practical computer vision applications, such as object detection, recognition, and tracking. The obtained results are quite promising, and further confirm the generality and capability of the H-ELM.

The remainder of this paper is organized as follows. Section II introduces the related works, including the fundamental concepts and theories of ELM. Section III describes the proposed H-ELM framework and its related ELM sparse autoencoder. Section IV compares the performance of H-ELM with those of ELM and other relevant state-of-the-art MLP learning algorithms on various testing data sets. Section V presents several real-world applications incorporating the proposed H-ELM, including car detection, gesture recognition, and real-time object tracking. Finally, the conclusion is drawn in Section VI.

II. Related Works

In this paper, ELM is to be extended for MLPs. To facilitate the understanding of the proposed algorithm, this section briefly reviews the related concepts/theories of ELM, including the fundamental ideas and capabilities of the original ELM and ELM-based autoencoder.

A. ELM Theory

Suppose that SLFNs with L hidden nodes can be represented by the following equation:

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} G_i(\mathbf{x}, \mathbf{a}_i, b_i) \cdot \boldsymbol{\beta}_i, \quad \mathbf{a}_i \in \mathbf{R}^d, b_i, \boldsymbol{\beta}_i \in \mathbf{R} \quad (1)$$

where $G_i(\cdot)$ denotes the ith hidden node activation function, $\mathbf{a}_i$ is the input weight vector connecting the input layer to the ith hidden layer, b_i is the bias weight of the ith hidden layer, and β_i is the output weight. For additive nodes with activation function g, G_i is defined as follows:

$$G_i(\mathbf{x}, \mathbf{a}_i, b_i) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i) \quad (2)$$

and for radial basis function (RBF) nodes with activation function g, G_i is defined as

$$G_i(\mathbf{x}, \mathbf{a}_i, b_i) = g(b_i||\mathbf{x} - \mathbf{a}_i||). \tag{3}$$

Huang *et al.* [5] have proved that the SLFNs are able to approximate any continuous target functions over any compact subset $X \in R^d$ with above random initialized adaptive or RBF nodes. Let $L^2(X)$ be a space of functions f on a compact subset X in the d-dimensional Euclidean space $\mathbf{R}^d$ such that $|f|^2$ is integrable, that is, $\int_X |f(\mathbf{x})|^2 d\mathbf{x} < \infty$. For $u, v \in L^2(X)$, the inner product $\langle u, v\rangle$ is defined by

$$\langle u, v\rangle = \int_X u(\mathbf{x})v(\mathbf{x})d\mathbf{x}. \tag{4}$$

The norm in $L^2(X)$ space is denoted as $||\cdot||$, and the closeness between network function f_n and the target function f is measured by the $L^2(X)$ distance

$$||f_L - f|| = \left(\int_X |f_n(\mathbf{x}) - f(\mathbf{x})|^2 d\mathbf{x}\right)^2. \tag{5}$$

Theorem 2.1: Given any bounded nonconstant piecewise continuous function g: $\mathbf{R} \to \mathbf{R}$, if span $\{G(\mathbf{a}, b, \mathbf{x}) : (\mathbf{a}, b) \in \mathbf{R}^d \times R\}$ is dense in L^2, for any target function f and any function sequence $g_L(\mathbf{x}) = G(\mathbf{a}_L, b_L, x)$ randomly generated based on any continuous sampling distribution, $\lim_{n\to\infty}||f - f_n|| = 0$ holds with probability one if the output weights β_i are determined by ordinary least square to minimize $||f(\mathbf{x}) - \sum_{i=1}^{L} \beta_i g_i(\mathbf{x})||$.

The theorem above [5], [16], [17] shows that randomly generated networks with the outputs being solved by least mean square are able to maintain the universal approximation capability, if and only if the activation function g is nonconstant piecewise and span $\{G(\mathbf{a}, b, \mathbf{x}) : (\mathbf{a}, b) \in \mathbf{R}^d \times R\}$ is dense in L^2. Based on this theorem, ELM can be established for fast learning, which will be described in detail in the next section.

B. ELM Learning Algorithm

According to the Theorem 2.1, the ELM can be built with randomly initialized hidden nodes. Given a training set $\{(\mathbf{x}_i, \mathbf{t}_i)|\mathbf{x}_i \in \mathbf{R}^d, \mathbf{t}_i \in \mathbf{R}^m,\ i = 1, \ldots, N\}$, where $\mathbf{x}_i$ is the training data vector, $\mathbf{t}_i$ represents the target of each sample, and L denotes the number of hidden nodes.

From the learning point of view, unlike traditional learning algorithms (see the related works referred to in [7]), ELM theory aims to reach the smallest training error but also the smallest norm of output weights [1], [7]

$$\text{Minimize:}\quad \|\boldsymbol{\beta}\|_u^{\sigma_1} + \lambda\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_v^{\sigma_2} \tag{6}$$

where $\sigma_1 > 0,\ \sigma_2 > 0,\ u, v = 0, (1/2), 1, 2, \ldots, +\infty$, $\mathbf{H}$ is the hidden layer output matrix (randomized matrix)

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}_1) & \cdots & h_L(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ h_1(\mathbf{x}_N) & \vdots & h_L(\mathbf{x}_N) \end{bmatrix} \tag{7}$$

and $\mathbf{T}$ is the training data target matrix

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix} = \begin{bmatrix} t_{11} & \cdots & t_{1m} \\ \vdots & \vdots & \vdots \\ t_{N1} & \cdots & t_{Nm} \end{bmatrix}. \tag{8}$$

The ELM training algorithm can be summarized as follows [1].

1) Randomly assign the hidden node parameters, e.g., the input weights $\mathbf{a}_i$ and biases b_i for additive hidden nodes, $i = 1, \ldots, L$.
2) Calculate the hidden layer output matrix $\mathbf{H}$.
3) Obtain the output weight vector

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{9}$$

where $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]^T$, $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalized inverse of matrix $\mathbf{H}$.

The orthogonal projection method can be efficiently used for the calculation of MP inverse: $\mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$, if $\mathbf{H}^T\mathbf{H}$ is nonsingular; or $\mathbf{H}^{\dagger} = \mathbf{H}^T(\mathbf{H}^T\mathbf{H})^{-1}$, if $\mathbf{H}\mathbf{H}^T$ is nonsingular. According to the ridge regression theory, it was suggested that a positive value $(1/\lambda)$ is added to the diagonal of $\mathbf{H}^T\mathbf{H}$ or $\mathbf{H}\mathbf{H}^T$ in the calculation of the output weights β. By doing so, according to [1] and [7], the resultant solution is equivalent to the ELM optimization solution with $\sigma_1 = \sigma_2 = u = v = 2$, which is more stable and has better generalization performance. That is, in order to improve the stability of ELM, we can have

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

and the corresponding output function of ELM is

$$f(\mathbf{x}) = h(\mathbf{x})\boldsymbol{\beta} = h(\mathbf{x})\mathbf{H}^T\left(\frac{1}{\lambda} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T} \tag{11}$$

or we can have

$$\boldsymbol{\beta} = \left(\frac{1}{\lambda} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{H}^T\mathbf{T} \tag{12}$$

and the corresponding output function of ELM is

$$f(\mathbf{x}) = h(\mathbf{x})\boldsymbol{\beta} = h(\mathbf{x})\left(\frac{1}{\lambda} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{H}^T\mathbf{T}. \tag{13}$$

C. ELM-Based Autoencoder

Apart from the ELM-based SLFNs, the ELM theory has also been applied to build an autoencoder for an MLP. Conceptwise, the autoencoder [15] functions as some sort of feature extractor in a multilayer learning framework. It uses the encoded outputs to approximate the original inputs by minimizing the reconstruction errors. Mathematically, the autoencoder maps the input data $\mathbf{x}$ to a higher level representation, and then uses latent representation $\mathbf{y}$ through a deterministic mapping $\mathbf{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g(\mathbf{A} \cdot \mathbf{x} + b)$, parameterized by $\boldsymbol{\theta} = \{\mathbf{A}, b\}$, where $g(\cdot)$ is the activation function, $\mathbf{A}$ is a $d' \times d$ weight matrix and $\mathbf{b}$ is a bias vector. The resulting latent representation $\mathbf{y}$ is then mapped back to a reconstructed

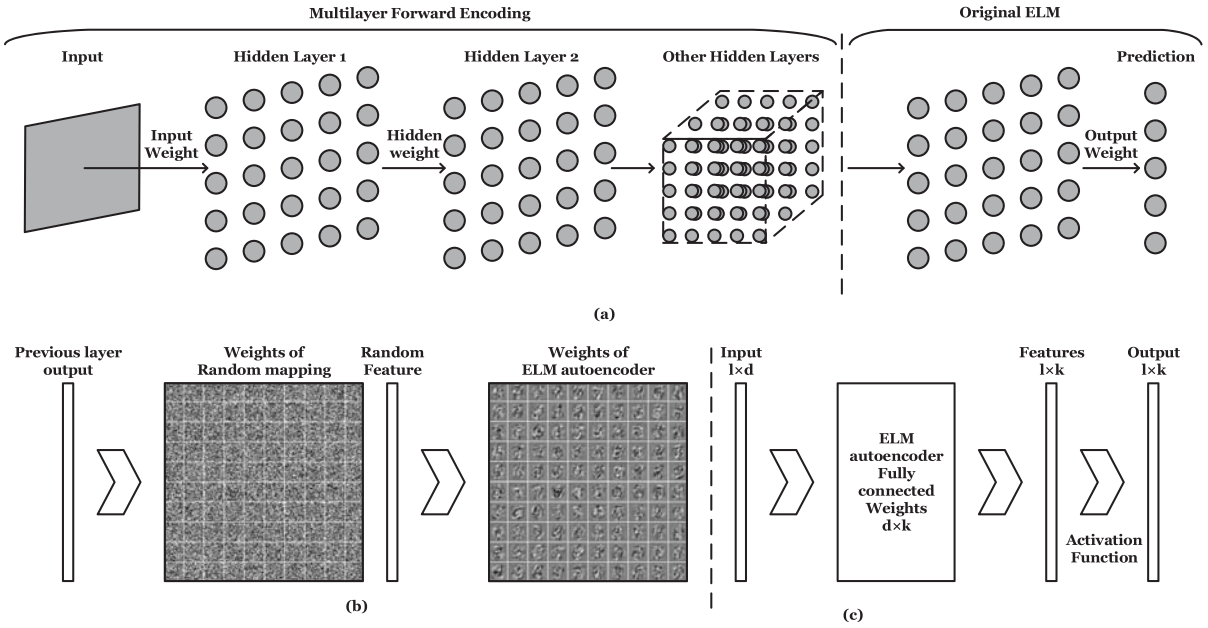

Fig. 1. Proposed H-ELM learning algorithm. (a) Overall framework of H-ELM, which is divided into two phases: multilayer forward encoding followed by the original ELM-based regression. (b) Implementation of ELM-autoencoder. (c) Layout of one single layer inside the H-ELM.

vector $\mathbf{z}$ in the input space $\mathbf{z} = h_{\boldsymbol{\theta}'}(\mathbf{y}) = g(\mathbf{A}' \cdot \mathbf{y} + b)$ with $\boldsymbol{\theta}' = \{\mathbf{A}', b'\}$.

Using randomly mapped outputs as latent representation $\mathbf{y}$, one can easily build the ELM-based autoencoder as shown in [10]. Then, the reconstruction of $\mathbf{x}$ can be regarded as an ELM learning problem, in which $\mathbf{A}'$ is obtained by solving regularized least mean square optimization. However, due to the use of ℓ_2 penalty in the original ELM, the extracted features by the ELM autoencoder in [10] tend to be dense and may have redundancy. In this case, a more sparse solution is preferred.

III. Proposed Learning Algorithm

In this section, we propose a new H-ELM framework for MLPs. The overall architecture of the proposed H-ELM is to be introduced in detail, and a new ELM sparse autoencoder is also presented, which is utilized as the basic elements of H-ELM.

A. H-ELM Framework

The proposed H-ELM is built in a multilayer manner, as shown in Fig. 1. Unlike the greedy layerwise training of the traditional DL frameworks [11], [13], one can see that the H-ELM training architecture is structurally divided into two separate phases: 1) unsupervised hierarchical feature representation and 2) supervised feature classification. For the former phase, a new ELM-based autoencoder is developed to extract multilayer sparse features of the input data, which is to be discussed in the next section; while for the latter one, the original ELM-based regression is performed for final decision making.

In the following, we will present a detailed description of H-ELM, as well as its advantages against the existing DL and multilayer ELM (ML-ELM) algorithms. Before unsupervised feature learning, the input raw data should be transformed into an ELM random feature space, which can help to exploit hidden information among training samples. Then, a N-layer unsupervised learning is performed to eventually obtain the high-level sparse features. Mathematically, the output of each hidden layer can be represented as

$$\mathbf{H}_i = g(\mathbf{H}_{i-1} \cdot \boldsymbol{\beta}) \tag{14}$$

where $\mathbf{H}_i$ is the output of the ith layer ($i \in [1, K]$), $\mathbf{H}_{i-1}$ is the output of the $(i-1)$th layer, $g(\cdot)$ denotes the activation function of the hidden layers, and $\boldsymbol{\beta}$ represents the output weights. Note that here each hidden layer of H-ELM is an independent module, and functions as a separated feature extractor. As the layers increasing, the resulting feature becomes more compact. Once the feature of the previous hidden layer is extracted, the weights or parameters of the current hidden layer will be fixed, and need not be fine-tuned. This is totally different from the existing DL frameworks [11]–[15], where all the hidden layers are put together as a whole system, with unsupervised initialization. The whole system needs to be retrained iteratively using BP-based NNs. Thus, the training of H-ELM would be much faster than that of the DL.

From Fig. 1(a), we can also see that after unsupervised hierarchical training in the H-ELM, the resultant outputs of the Kth layer, i.e., $\mathbf{H}_K$, are viewed as the high-level features extracted from the input data. When used for classification, they are randomly perturbed, and then utilized as the inputs of the supervised ELM-based regression to obtain the final results of the whole network. The reason for the perturbation of $\mathbf{H}_K$ is that random projection of the inputs is required to maintain the universal approximation capability of ELM.

Conceptually, to expedite the learning speed, the proposed H-ELM framework is developed based upon random feature mapping and fully exploits the universal approximation

capability of ELM, both in feature learning and feature classification. According to Theorem 2.1, using random mapped features as the inputs of the output weights, the hierarchical network is able to approximate or classify any input data in theory. On the other hand, it is also worth mentioning that H-ELM differentiates itself from the existing ML-ELM scheme [10] in threefold: 1) ML-ELM uses a simple stacked layer-by-layer architecture, and is just a straightforward replacement of DL autoencoder, with ELM autoencoder. In contrast, the proposed H-ELM considers MLP in a more comprehensive way, by dividing the whole network into two separated subsystems (i.e., unsupervised feature extraction/representation, and supervised feature classification, respectively), and using random projections of feature extraction results as the inputs of feature classification subsystem; 2) instead of ℓ_2-norm ELM autoencoder used in the ML-ELM, ℓ_1 penalty is applied in H-ELM, to obtain more compact and sparse hidden information; and 3) the orthogonal initialization of ML-ELM is avoided, since the orthogonal constraint is not reasonable when the number of the input nodes is different from that of the output ones.

B. ELM Sparse Autoencoder

As introduced in the previous section, the proposed H-ELM briefly consists of two separate parts: 1) unsupervised and 2) supervised training. Since the supervised training is implemented by the original ELM, in this section, we will focus on how to train the unsupervised building blocks (i.e., autoencoder) of the H-ELM architecture. From Section II-C, it is known that the autoencoder aims to learn a function $\mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}) \simeq \mathbf{x}$, where $\boldsymbol{\theta} = \{\mathbf{A}, b\}$, $\mathbf{A}$ are the hidden weights, and b is the bias. In other words, the autoencoder tries to approximate the input data to make the reconstructed outputs being similar to the inputs.

In this paper, the universal approximation capability of the ELM is exploited for the design of autoencoder, and moreover, sparse constraint is added upon the autoencoder optimization, and therefore, we term it as ELM sparse autoencoder. The implementation of ELM sparse autoencoder is demonstrated in Fig. 1(b). It can be seen that unlike the autoencoders (i.e., BP-based NNs) used in the traditional DL algorithms, the input weights of the proposed ELM sparse autoencoder are established by searching the path back from a random space. The ELM theory [10] has demonstrated that the training of ELM with random mapped input weights is efficient enough to approximate any input data. That is to say, if we train the autoencoder following the concept of ELM, once the autoencoder is initialized, the fine-tuning is not required.

In addition, in order to generate more sparse and compact features of the inputs, ℓ_1 optimization is performed for the establishment of ELM autoencoder, and this is different from the ELM autoencoder proposed in [10], where ℓ_2-norm singular values are calculated for feature representation. The optimization model of the proposed ELM sparse autoencoder can be denoted as the following equation:

$$O_{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\{||\mathbf{H}\boldsymbol{\beta} - \mathbf{X}||^2 + ||\boldsymbol{\beta}||_{\ell_1}\} \tag{15}$$

where $\mathbf{X}$ represents the input data, $\mathbf{H}$ denotes the random mapping output, and $\boldsymbol{\beta}$ is the hidden layer weight to be obtained. In the existing DL algorithms, $\mathbf{X}$ is usually the encoding outputs of the bases $\boldsymbol{\beta}$, which need to be adjusted during the iterations of optimization. However, in the proposed autoencoder, as we are to utilize random mapping for hidden layer feature representation, $\mathbf{X}$ is the original data and $\mathbf{H}$ is the random initialized output which need not to be optimized [1]. Furthermore, the experiments in the next section will show that it would not only help to improve the training time but also the learning accuracy.

Hereinafter, we will describe the optimization algorithm for the ℓ_1 optimization problem. For clear representation, we rewrite the object function in (12) as

$$O_{\boldsymbol{\beta}} = p(\boldsymbol{\beta}) + q(\boldsymbol{\beta}) \tag{16}$$

where $p(\boldsymbol{\beta}) = ||\mathbf{H}\boldsymbol{\beta} - \mathbf{X}||^2$, and $q(\boldsymbol{\beta}) = ||\boldsymbol{\beta}||_{\ell_1}$ is the ℓ_1 penalty term of the training model.

A fast iterative shrinkage-thresholding algorithm (FISTA) [18] is adopted to solve the problem in (13). The FISTA minimizes a smooth convex function with complexity of $O(1/j^2)$, where j denotes the iteration times. The implementation details of FISTA are as follows [18].

1) Calculate the Lipschitz constant γ of the gradient of smooth convex function ∇p.
2) Begin the iteration by taking $\mathbf{y}_1 = \boldsymbol{\beta}_0 \in \mathbf{R}^n$, $t_1 = 1$ as the initial points. Then, for $j (j \geq 1)$ the following holds.
   a) $\boldsymbol{\beta}_j = s_\gamma(\mathbf{y}_j)$, where s_γ is given by

$$s_\gamma = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \frac{\gamma}{2}||\boldsymbol{\beta} - (\boldsymbol{\beta}_{j-1} - \frac{1}{\gamma}\nabla p(\beta(j-1))||^2 + q(\boldsymbol{\beta}) \right\}. \tag{17}$$

   b) $t_{j+1} = \frac{1+\sqrt{1+4t_j{}^2}}{2}$.
   c) $\mathbf{y}_{j+1} = \boldsymbol{\beta}_j + \left(\frac{t_j - 1}{t_{j+1}}\right)(\boldsymbol{\beta}_j - \boldsymbol{\beta}_{j-1})$.

By computing the iterative steps above, we could manage to perfectly recover the data from the corrupted ones. Using the resultant bases $\boldsymbol{\beta}$ as the weights of the proposed autoencoder, the inner product of the inputs and learned features would reflect the compact representations of the original data. In addition, as the autoencoder is adopted as the building block of the proposed H-ELM, higher level feature representations can be generated by layerwise comparison. In addition, compared with the existing ELM autoencoder [10] which simply uses the inputs as outputs with the traditional least mean square method, the ℓ_1 optimization has been proved [18], [19] to be a better solution for data recovery and other applications. It will help to reduce the number of neural nodes and thus further improve the testing time of H-ELM.

IV. Performance Evaluation and Analysis

Extensive experiments are conducted to verify the effectiveness and efficiency of the proposed H-ELM framework, which is compared with the original ELM and other existing relevant state-of-the-art MLP algorithms.

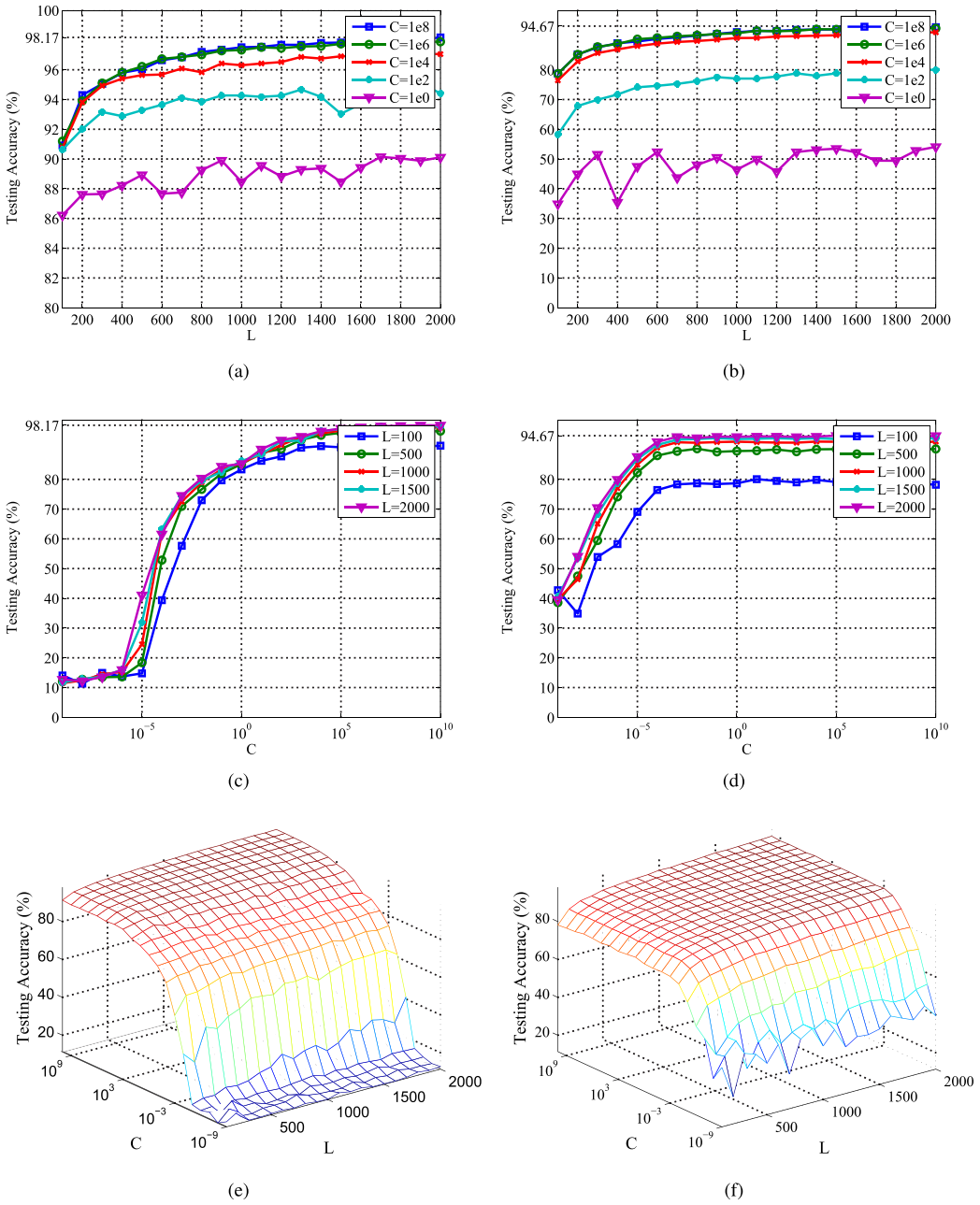

Fig. 2. Testing accuracy in (C, L) subspace for the H-ELM and ELM. (a) Accuracy of H-ELM in terms of L. (b) Accuracy of ELM in terms of L. (c) Accuracy of H-ELM in terms of C. (d) Accuracy of ELM in terms of C. (e) Accuracy curve of H-ELM in terms of (C, L). (f) Accuracy curve of ELM in terms of (C, L).

In all the simulations below, the testing hardware and software conditions are listed as follows: Laptop, Intel-i7 2.4G CPU, 16G DDR3 RAM, Windows 7, MATLAB R2013b.

A. Comparison Between H-ELM and ELM

1) Selection of Hyper Parameters: In this section, we will demonstrate the selection of hyperparameters in H-ELM and ELM schemes. Compared with the traditional NNs training algorithms, much less parameters need to be chosen in the training process of H-ELM and ELM. From Section II, we can see that two user specified parameters are required, i.e., the parameter C for the regularized least mean square calculation, and the number of hidden nodes L. In the simulations, C is set as $\{10^{-10}, 10^{-9}, \ldots, 10^{9}, 10^{10}\}$, and L is set as $\{100, 200, \ldots, 2000\}$.

Fig. 2(a) and (b) shows the learning accuracies of H-ELM and ELM in the L subspace, where the parameter C is prefixed. It can be seen that H-ELM seems to follow a similar convergence property of ELM but with higher testing accuracy, and the performances tend to be quite stable in a wide range of L. Meanwhile, in Fig. 2(c) and (d), the performance of H-ELM is more sensitive to the parameter C than that of the ELM. As C increases, the accuracy of H-ELM first slightly changes, and then increases rapidly until convergence to a better performance than ELM.

TABLE I
PERFORMANCE COMPARISON USING DIFFERENT METHODS OVER BINARY AND MULTIPLE CLASSED BENCHMARKS

Dataset	H-ELM (%)	Hidden Nodes	ELM (%)	Hidden Nodes
Binary Classes Datasets				
mushroom	100	N1=N2=200;N3=500;	100	N=500
liver	**72.17**	N1=N2=10;N3=200;	70.43	N=200
leu	**91.18**	N1=N2=200;N3=5000;	82.35	N=5000
diabetes	**80.47**	N1=N2=10;N3=200;	76.95	N=200
colon-cancer	85.71	N1=N2=100;N3=3000;	85.71	N=3000
australian	**86.96**	N1=N2=10;N3=50;	86.52	N=50
Multiple Classes Datasets				
satimage	**90.9**	N1=N2=50;N3=800;	89.7	N=800
DNA	**94.94**	N1=N2=200;N3=2000;	94.01	N=2000
IRIS	**100**	N1=N2=20;N3=200;	92	N=200
GLASS	**76.39**	N1=20;N2=20;N3=200;	66.67	N=200
Vowel	**64.94**	N1=10;N2=10;N3=3000;	51.52	N=3000
letter	95.82	N1=200;N2=200;N3=2000;	95.84	N=2000
wine	**100**	N1=N2=20;N3=500;	95.00	N=500
USPS	**96.76**	N1=N2=200;N3=3000;	94.57	N=3000

It should also be mentioned that the impacts of C and L on the performance of H-ELM are different. As shown in Fig. 2(a), a proper C will make the accuracy curve more smooth (with less jitter) as L increases. On the other hand, from Fig. 2(c), one can see that different values of L have different impacts on the learning performance, but the shapes of accuracy curves are quite similar with different values of L.

To facilitate understanding, the 3-D accuracy curves of H-ELM and ELM are further shown in Fig. 2(e) and (f), respectively. Note that the learning performance of H-ELM is better than that of ELM, and they all affected by the parameters C and L.

Practically, C needs to be selected carefully, while L should be large enough. We can use a small number of nodes to select C, as L will not affect the trend of the accuracy curve.

2) Performance of Learning Accuracy: As mentioned in Section III, the H-ELM briefly consists of two basic components: 1) unsupervised feature learning and 2) supervised feature classification. The major difference between H-ELM and the original ELM is that before ELM-based feature classification, H-ELM uses hierarchical training to obtain multilayer sparse representation of the input raw data; while in ELM scheme, the raw data is used for regression and classification. Generally speaking, the compact features can help to remove redundancy of the original inputs, and thus improve the overall learning performance.

The classification accuracies of the H-ELM and ELM are compared in Table I. For H-ELM, three-layer feature extraction is performed, and the resulting features are randomly projected before ELM-based classification; for ELM, the randomly mapped raw data is input for classification. For fair comparison, in the feature classification stage, the number of the hidden nodes of H-ELM is set to be equal to that of the ELM. In addition, both H-ELM and ELM are with the sigmoid activation function. The widely used binary-class and multiple-class data sets are tested, respectively. The experiments were repeated for 50 times, and the averaging results were obtained for comparison.

One can easily observe that, the proposed H-ELM has a remarkable improvement against the ELM with most of the testing data sets, in terms of learning accuracy. The results show that by hierarchical feature encoding, the classification performance of ELM has indeed been enhanced. The reason is that for H-ELM, the input of ELM-based feature classification is changed from the original data to a more compact and sparse one. The ELM sparse autoencoder of H-ELM helps to generate a better performance by providing more robust features extract from the data itself.

B. Comparison With State-of-the-Art MLP Algorithms

In this section, we used more complicated data sets with images patches to verify the learning performance of H-ELM over DL algorithms [including Stacked Auto Encoders (SAE) [14], stacked autoencoder (SDA) [15], Deep Belief Networks (DBN) [13], Deep Boltzmann Machines (DBM) [20], and MLP-BP [21]] and ML-ELM [10]. Since they are also fully connected MLP, we use the term MLPs to denote them. Note that in the experiments, the effects of data preprocessing techniques (e.g., data augmentation) are avoided, and we mainly focused on the verification of learning capability of different MLP training schemes. For BP-based MLP training algorithms (SAE, SDA, DBN, DBM, and MLP-BP), the initial learning rate is set as 0.1 with a decay rate 0.95 for each learning epoch. The pretraining and fine-tuning are set as 100 and 200 epochs, respectively. Besides, the input corruption rate of SDA is set at 0.5 with a dropout rate 0.2 for hidden layers. The ℓ_2 penalty parameters of the three-layer ML-ELM are set as 10^{-1}, 10^3, and 10^8, respectively.

1) MINST: The Mixed National Institute of Standards and Technology (MNIST) handwriting data set [22]

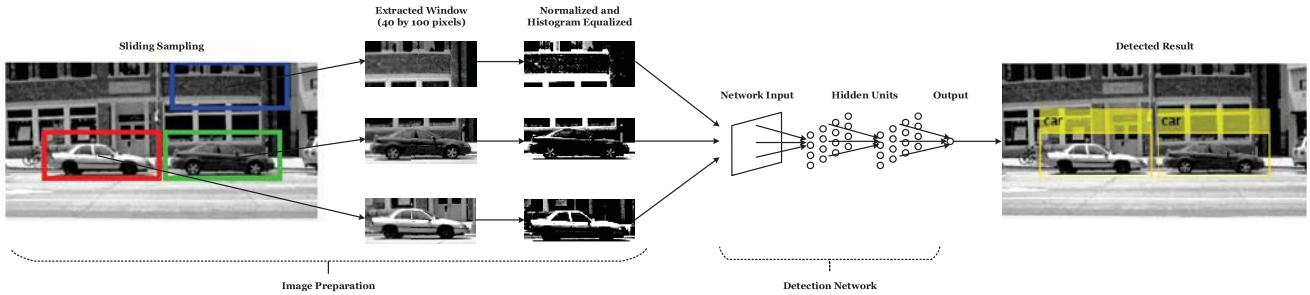

Fig. 3. H-ELM-based car detection approach.

TABLE II
COMPARISON OF LEARNING ACCURACY ON MNIST DATA SET

Method	Accuracy (%)	Training time (s)
SAE [14]	98.60	36448.40
SDA [15]	98.72	37786.03
DBN [13]	98.87	53219.77
DBM [20]	99.05	121455.69
MLP-BP [21]	97.39	21468.10
ML-ELM [10]	99.04	475.83
H-ELM	**99.13**	**281.37**

consists of 60 000 training images and 10 000 testing images, and the samples are digits 0–9 with 28 pixels in grayscale and have uniform backgrounds. As different digits have their unique shapes and different people write the number in their own ways, the MNIST is an ideal data set to test the effectiveness of the H-ELM. The original image patches are input into the training algorithms without preprocessing.

The testing accuracies of different MLP methods on MINST are shown in Table II. It can be seen that compared with other time consuming MLP training methods, the proposed H-ELM achieves 99.13% accuracy with hundreds times faster training time.

2) NORB: To further demonstrate the advantages of H-ELM, another experiment was conducted using NYU Object Recognition Benchmark (NORB) data set [23], which is more complicated than the MNIST. The NORB contains images of 50 different 3-D toy objects with ten objects in each of five generic classes: 1) cars; 2) trucks; 3) planes; 4) animals; and 5) humans. Each image is obtained with different viewpoints and various lighting conditions. The training set contains 24 300 stereo image pairs of 25 objects (five per class), while the testing set contains the remaining image pairs of 25 objects. The testing results are shown in Table III, and it can be found that the proposed H-ELM again achieves the highest accuracy and fastest training time among the state-of-the-art MLP training algorithms, which is consistent with the results on MNIST.

TABLE III
COMPARISON OF LEARNING ACCURACY ON NORB DATA SET

Method	Accuracy (%)	Training time (s)
SAE [14]	86.28	60504.34
SDA [15]	87.62	65747.69
DBN [13]	88.47	87280.42
DBM [20]	89.65	182183.53
MLP-BP [21]	84.20	34005.470
ML-ELM [10]	88.91	775.2850
H-ELM	**91.28**	**432.19**

C. Performance Analysis

Based on the experiments above, we can see that the H-ELM obviously outperforms the original single-layer ELM, in terms of learning and classification accuracy. Furthermore, compared with other MLP training methods, H-ELM leads to a promising better performance with extremely faster training speed. While most of the state-of-the-art MLP training algorithms take hours or days for training by hundreds of epoches of iteration even with high performance computers, the H-ELM can be easily utilized with a laptop in a few minutes. In addition, it should be mentioned that the implementation of H-ELM only uses the MATLAB scripts, and therefore, the efficiency could be improved in more than tens or hundreds times with object-oriented programming techniques, e.g., cpp codes on Graphics Processing Unit or server.

V. APPLICATIONS

To demonstrate the generality and capability of the H-ELM framework, in this section, three different computer vision applications (object detection, recognition, and tracking) are presented following the concept of H-ELM. In each application, H-ELM is used for feature extraction and classification, and the H-ELM-based methods are compared with state-of-the-art schemes, including the relevant DL-based SDA ones, where H-ELM is straightforwardly replaced by SDA. The testing hardware and software environments remain the same in Section IV.

A. Car Detection

Car detection is a classical application in computer vision. In our simulations, we used University of Illinois Urbana-Champaign car data set [24], [28] for training and testing.

TABLE IV
CAR DETECTION PERFORMANCE COMPARISON

Method	[24]	[25]	[26]	[27]	H-ELM
EER (%)	79	88.5	91	92.8	**95.5**

TABLE V
PERFORMANCE COMPARISON BETWEEN H-ELM- AND DL-BASED SDA FOR CAR DETECTION

Method	SDA [15]	H-ELM
Training Time (s)	3262.30	**46.78**
EER (%)	93.3	**95.5**

Various small training images of cars/backgrounds are used, and each of the testing images contains at least one car. The testing images uses the single-scale set where the cars to be detected are roughly the same size (100×40 pixels) like those in the training images.

The proposed H-ELM-based car detection algorithm is shown in Fig. 3, and one can see that a sliding window is used to extract a fixed-size image patch, which are grouped and input into the H-ELM for training and testing. The number of training samples is 1050, and before training, the images are normalized and histogram equalized to avoid the effects of uneven lighting and intensity. The H-ELM is set as the following network structure: three-layer feature learning with two-hidden-node ELM feature classification.

The testing results are shown in Tables IV and V, it can be seen that the recall at equal-error rates (EERs) (recall equals the precision) of the proposed H-ELM is 95.5%. Compared with the other existing methods shown in Table IV, the proposed car detection method achieves the best performance. It is worth noting that in order to remove the effects of background samples, the existing methods [25]–[27] adopted preprocessing before car detection. In contrast, the H-ELM-based detection is performed without any additional preprocessing, and achieves a pretty good performance. Furthermore, from Table V, we can see that the training time of the H-ELM-based method is much less than that of the DL-based SDA one, even with higher EER.

B. Gesture Recognition

Gesture recognition is also an important topic in computer vision due to its wide ranges of applications, such as human–computer interfaces, sign language interpretation, and visual surveillance. In the experiments, the Cambridge-gesture data set [29] is used. It consists of 900 image sequences of nine hand gesture classes, which are divided by three primitive hand shapes and three primitive motions, as is shown in Fig. 4(a). Each class contains 100 image sequences includes five different illumination backgrounds, and each of the sequence was recorded in front of a fixed camera which roughly isolated gestures in space and time.

In the testing, all the video sequences are initially resized to $60 \times 80 \times 10$ in pixels for fast implementation. Then, the

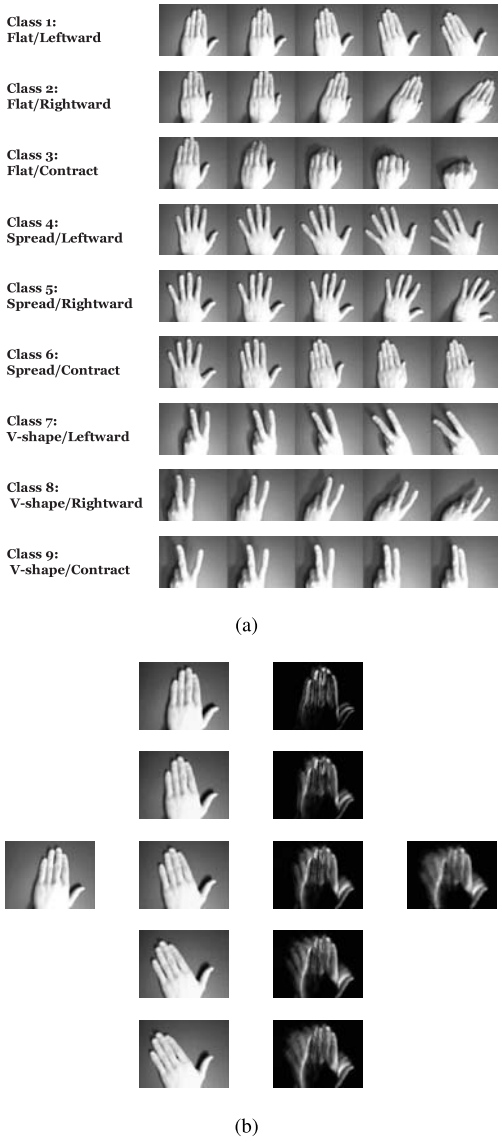

Fig. 4. Original gesture samples and difference images. (a) Cambridge gesture data set. (b) Difference images of flat/leftward gestures in Cambridge gesture data set.

first frame of each sequence is used for initialization, and the other frames subtract the corresponding value of the first frame, and the resultant differences are combined to one single fused image [shown in Fig. 4(b)]. By doing so, we obtain $900 \times 60 \times 80$ difference images which include the motion information of the hands and partially exclude the illumination effects.

The difference images are directly fed into the H-ELM-based classifier to recognize each of the gesture. Moreover, as some of the images are not at the centre, we further extend the data set with its random skewing version. In this case, each of the images is randomly moved with 3 and 5 pixels in horizontal and vertical directions (i.e., the data are augmented two times). This would help to make recognition more robust and invariant to small shifting in space.

Based on these settings, we use the proposed ELM sparse autoencoder for feature learning, and it is stacked twice to get

TABLE VI
GESTURE RECOGNITION PERFORMANCE COMPARISON

Method	[29]	[30]	[31]	H-ELM
Accuracy (%)	85	85.5	93.4	**99.4**

TABLE VII
PERFORMANCE COMPARISON BETWEEN H-ELM- AND DL-BASED SDA FOR GESTURE RECOGNITION

Method	SDA [15]	H-ELM
Training Time (s)	16035.79	**57.70**
Accuracy (%)	93.3	**99.4**

more deep representation. The output is then used to initialize the ELM classifier in the H-ELM, which has 5000 hidden nodes. In our simulations, we use the above-mentioned data set with fivefolds cross-validation. The H-ELM framework was performed for 20 times, and achieves an averaging performance with 99.4% testing accuracy. The performance of other methods is listed in Tables VI and VII. As can be seen, the proposed method achieves the best result over these traditional methods, where the handcrafted feature extractors or other data analysis methods are used. Unlike these methods, the H-ELM uses unsupervised hierarchical feature extractors to obtain higher level feature representations, which lead to a notable improvement and better testing accuracy. In addition, it should also be noted that the H-ELM outperforms DL-based SDA, in terms of training time, as well as learning accuracy.

C. Online Incremental Tracking

Online incremental tracking refers to the tracking method that aims to learn and adapt a representation to reflect the appearance changes of the target. The appearance changes of the specified target include pose variation and shape deformation. Apart from these, extrinsic illumination change, camera motion, viewpoint, and occlusions inevitably cause large appearance variation. Modeling such appearance variation is always one of the nature problems of tracking [32].

In this section, based on the proposed H-ELM, we build a novel incremental tracking method. Unlike the conventional online tracking method, more robust features are extracted using H-ELM, and the classification performance is far better compared with the probability models used in the conventional methods. In addition, as demonstrated in the previous section, the training time of H-ELM is much less than other MLP architectures (e.g., DL-based SDA), and this ensures that H-ELM has obvious advantages in the case of real-time tracking.

In the H-ELM-based tracking framework, as the training is incremental and need to be online updated, an ELM variant, i.e., online sequential ELM (OS-ELM) [8], is applied for feature classification and updating. The tracking algorithm is shown in Fig. 5, which can be summarized as follows.

1) Two sets of image patches are sampled around the target in the first frame, then the images with $d < r_1$ are used as positive training samples, while the images with $r_2 < d < r_3$ are the negative ones, where d denotes the Euclidean distance between the target and the sampling location, r_1, r_2, and r_3 are the parameters with the settings of 5, 10, and 20.
2) The H-ELM classifier is trained with the obtained image samples. For frame $n(n \geq 2)$, the following is applied.
   a) The object features are extracted by the ELM sparse autoencoder, and the feature classification is performed in frame (n), using OS-ELM.
   b) The location with the maximum response is used as the target location, and then go to Step 1).
   c) The training samples are updated, and the new samples are used for updating OS-ELM.

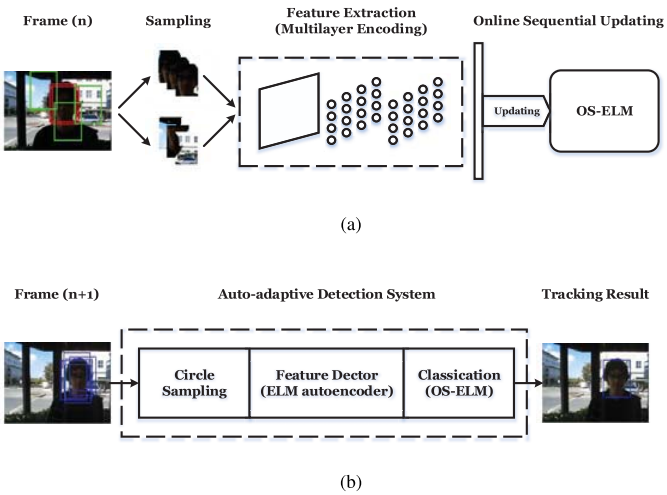

Fig. 5. H-ELM-based online tracking framework. (a) Updating with n-frame. (b) Tracking on $(n+1)$-frame.

Note that we adopted the same sampling scheme as that in compressive tracking (CT) [33], which is a popular real-time online tracking approach. For H-ELM tracking, the same initial position is labeled in the first frame, and the tracking results are obtained from the maximum values of the classification responses. In the simulations, we compared the feature extraction, feature updating, and classification performances of H-ELM, CT, and SDA, under the same conditions.

The well-used *David* and *Trellis* video sequences are tested. The tracking location errors of H-ELM, CT, and SDA are presented in Fig. 6, and one can see that the error of H-ELM is lower than those of CT and SDA ones, and the tracking results of H-ELM are with less fluctuations among different frames.

Moreover, the testing scenes are shown in Fig. 7, where Rows 1 and 4 are the results obtained from H-ELM, and the other rows are the results from CT and SDA. As can be seen in Rows 1–3 (*David Indoor* data set), the tracking locations of H-ELM is more accurate than those of CT and SDA. The tracking windows of CT and SDA more or less have some variations, while the window of H-ELM seems pretty stable.

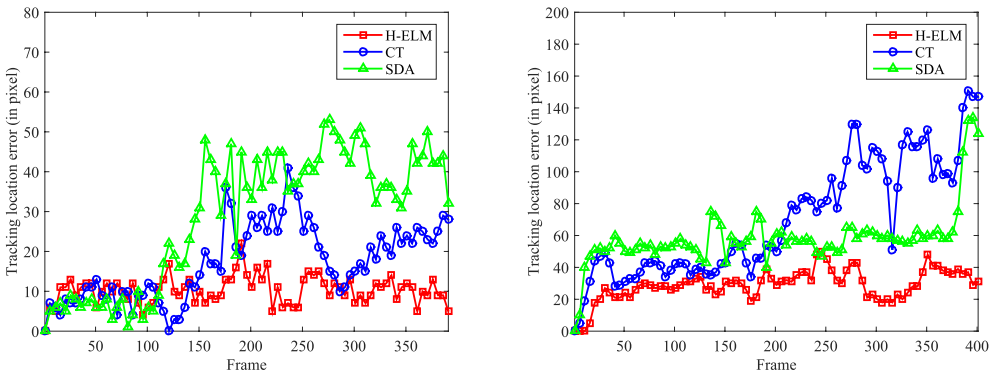

Fig. 6. Comparison of tracking location error using H-ELM, CT, and SDA on different data sets. (a) *David Indoor*. (b) *Trellis*.

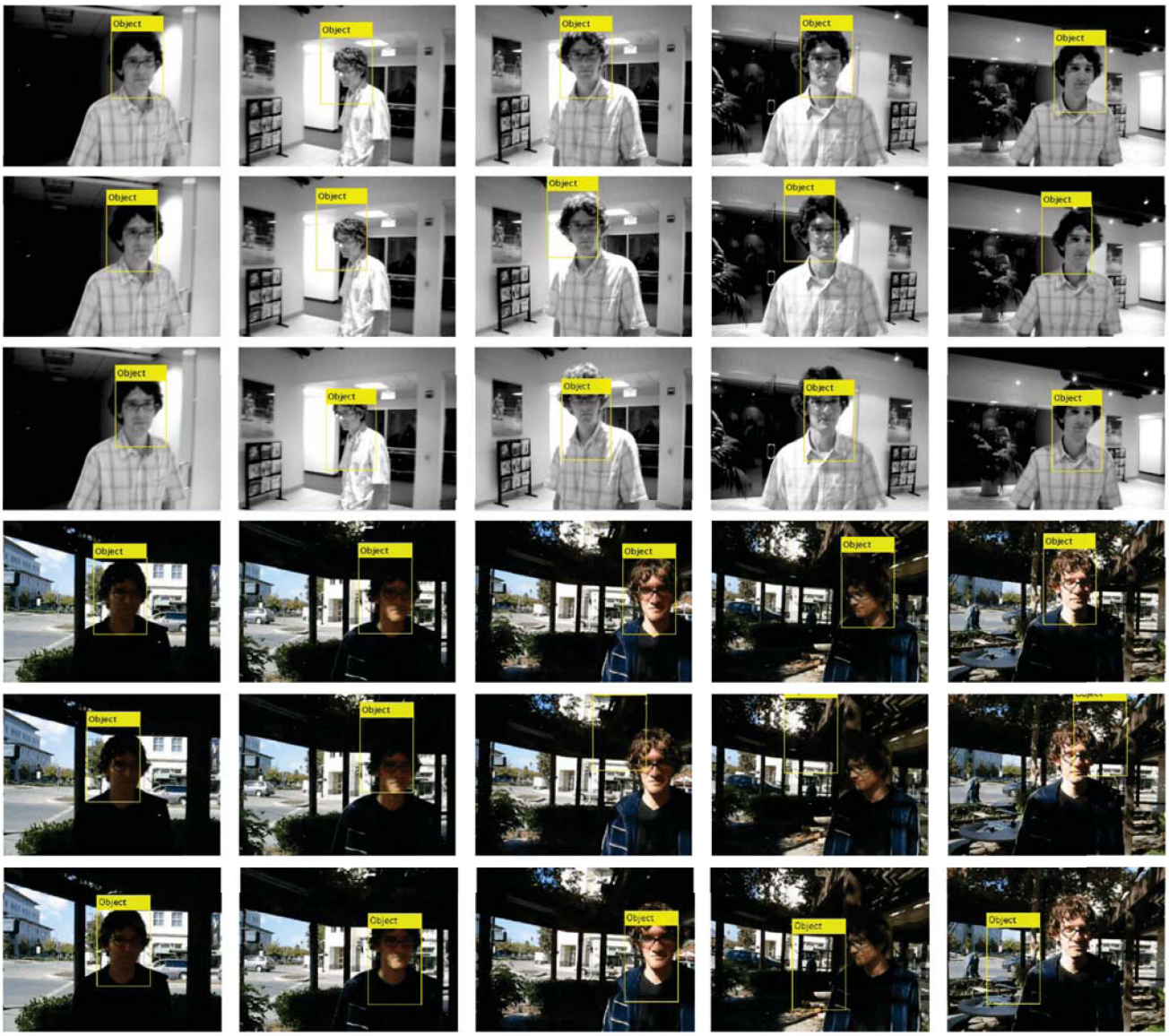

Fig. 7. Tracking performance comparison of H-ELM, CT, and SDA in *David Indoor* and *Trellis* data sets. Rows 1 and 4: results of H-ELM. Rows 2 and 5: results of CT. Rows 3 and 6: results of SDA.

For the *Trellis* video sequence, the CT and SDA lost tracking of the target when the illumination on face has obvious change, while the H-ELM still gives a robust tracking. Besides, by the advantages of fast training of H-ELM framework, H-ELM achieves real-time tracking (over 15 frames/s), while the result for SDA is only 1–2 frames/s.

VI. Conclusion

In this paper, we have proposed a novel MLP training scheme, which is based on the universal approximation capability of the original ELM. The proposed H-ELM achieves high level representation with layerwise encoding, and outperforms the original ELM in various simulations. Moreover, compared with other MLP training methods, the training of H-ELM is much faster and achieves higher learning accuracy. We also verified the generality and capability of H-ELM for practical computer vision applications. In theses applications, H-ELM functions as a feature extractor and classifier, and it achieves more robust and better performance than relevant state-of-the-art methods.

References

[1] 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.

[2] A. A. Mohammed, R. Minhas, Q. M. J. Wu, and M. A. Sid-Ahmed, "Human face recognition based on multidimensional PCA and extreme learning machine," *Pattern Recognit.*, vol. 44, nos. 10–11, pp. 2588–2597, 2011.

[3] C. Pan, D. S. Park, Y. Yang, and H. M. Yoo, "Leukocyte image segmentation by visual attention and extreme learning machine," *Neural Comput. Appl.*, vol. 21, no. 6, pp. 1217–1227, 2012.

[4] R. Minhas, A. Baradarani, S. Seifzadeh, and Q. M. J. Wu, "Human action recognition using extreme learning machine based on visual vocabularies," *Neurocomputing*, vol. 73, nos. 10–12, pp. 1906–1917, 2010.

[5] 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.

[6] 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.

[7] G.-B. Huang, "An insight into extreme learning machines: Random neurons, random features and kernels," *Cognit. Comput.*, vol. 6, no. 3, pp. 376–390, 2014.

[8] 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.

[9] G. Huang, S. Song, J. N. D. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2405–2417, Dec. 2014.

[10] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong, "Representational learning with extreme learning machine for big data," *IEEE Intell. Syst.*, vol. 28, no. 6, pp. 31–34, Nov. 2013.

[11] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[12] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[13] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[14] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[15] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn.*, Helsinki, Finland, Jul. 2008, pp. 1096–1103.

[16] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, nos. 16–18, pp. 3056–3062, 2007.

[17] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, nos. 16–18, pp. 3460–3468, 2008.

[18] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[19] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *IEEE Trans. Image Process.*, vol. 18, no. 11, pp. 2419–2434, Nov. 2009.

[20] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," in *Proc. 12th Int. Conf. Artif. Intell. Statist.*, Clearwater Beach, FL, USA, Jul. 2009, pp. 448–455.

[21] C. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[23] 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.*, Washington, DC, USA, Jun./Jul. 2004, pp. II-97–II-104.

[24] S. Agarwal and D. Roth, "Learning a sparse representation for object detection," in *Proc. 7th Eur. Conf. Comput. Vis.*, Copenhagen, Denmark, May 2002, pp. 113–127.

[25] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2. Madison, WI, USA, Jun. 2003, pp. II-264–II-271.

[26] B. Leibe, A. Leonardis, and B. Schiele, "Combined object categorization and segmentation with an implicit shape model," in *Proc. 8th Eur. Conf. Comput. Vis. (ECCV)*, Prague, Czech Republic, May 2004, vol. 2, no. 5, p. 7.

[27] J. Shotton, A. Blake, and R. Cipolla, "Contour-based learning for object detection," in *Proc. 10th IEEE Int. Conf. Comput. Vis.*, Beijing, China, Oct. 2005, pp. 503–510.

[28] S. Agarwal, A. Awan, and D. Roth, "Learning to detect objects in images via a sparse, part-based representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 11, pp. 1475–1490, Nov. 2004.

[29] T.-K. Kim, K.-Y. K. Wong, and R. Cipolla, "Tensor canonical correlation analysis for action classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Minneapolis, MN, USA, Jun. 2007, pp. 1–8.

[30] T.-K. Kim and R. Cipolla, "Gesture recognition under small sample size," in *Proc. 8th Asian Conf. Comput. Vis.*, Tokyo, Japan, Sep. 2007, pp. 335–344.

[31] Y. M. Lui, "Tangent bundles on special manifolds for action recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 6, pp. 930–942, Jun. 2012.

[32] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 125–141, 2008.

[33] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *Proc. 12th IEEE Eur. Conf. Comput. Vis.*, Florence, Italy, Oct. 2012, pp. 864–877.

Jiexiong Tang (S'13) received the B.Eng. degree from the Beijing Institute of Technology, Beijing, China, in 2013, where he is currently pursuing the M.S. degree with the Department of Electrical and Information Engineering.

His current research interests include remote sensing image processing, feature learning and extraction, extreme learning machines, deep neural networks, and applications in computer vision.

Chenwei Deng (M'09–SM'15) received the Ph.D. degree in signal and information processing from the Beijing Institute of Technology, Beijing, China, in 2009.

He was a Post-Doctoral Research Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore. He is currently an Associate Professor with the School of Information and Electronics, Beijing Institute of Technology. He has authored extensively in refereed international journals and conferences, and co-edited one book. His current research interests include machine learning, perceptual modeling, feature representation/fusion, object detection, recognition, tracking, image/video coding, and quality assessment.

Prof. Deng received the titles of the Beijing Excellent Talent and the Excellent Young Scholar of Beijing Institute of Technology in 2013. He was a Co-Organizer and/or Co-Chair of a series of special sessions in the IEEE International Conference on Multimedia Big Data in 2015, the International Conference on Connected Vehicles & Expo in 2014, IEEE International Conference on Granular Computing in 2013, and the IEEE International Symposium on Circuits and Systems in 2013. He serves as an Interesting Group Co-Chair of the IEEE Multimedia Communications Technical Committee.

Guang-Bin Huang (M'98–SM'04) received the B.Sc. degree in applied mathematics and the M.Eng. degree in computer engineering from Northeastern University, Shenyang, China, in 1991 and 1994, respectively, and the Ph.D. degree in electrical engineering from Nanyang Technological University, Singapore, in 1999.

He concurrently studied with the Department of Applied Mathematics and the Department of Wireless Communication, Northeastern University. He has been an Assistant Professor and Associate Professor (with tenure) with the School of Electrical and Electronic Engineering, Nanyang Technological University, since 2001. His current research interests include big data analytics, human–computer interface, brain–computer interface, image processing/understanding, machine learning theories and algorithms, extreme learning machine, and pattern recognition.

Dr. Huang was a recipient of the best paper award from the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS in 2013. He serves as an Associate Editor of the *Neurocomputing*, *Cognitive Computation*, *Neural Networks*, and the IEEE TRANSACTIONS ON CYBERNETICS. He received the Highly Cited Researcher Award, and listed in the 2014 The World's Most Influential Scientific Minds by Thomson Reuters. He was invited to give keynotes on numerous international conferences. One of his main works is to propose a new machine learning theory and learning techniques called Extreme Learning Machines. He is a Principal Investigator of several industrial sponsored research and development projects. He has led/implemented several key industrial projects (e.g., the Chief Architect/Designer and Technical Leader of the Singapore Changi Airport Cargo Terminal 5 Inventory Control System Upgrading Project).