Neural Networks 115 (2019) 11–22

Contents lists available at ScienceDirect

Neural Networks

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

Multilayer one-class extreme learning machine

Haozhen Dai [a,b], Jiuwen Cao [a,b,c,*], Tianlei Wang [a,b], Muqing Deng [a,b], Zhixin Yang [d]

[a] *School of Automation, Hangzhou Dianzi University, Zhejiang, 310018, China*
[b] *Artificial Intelligence Institute, Hangzhou Dianzi University, Zhejiang, 310018, China*
[c] *School of Electrical, Information and Media Engineering, University of Wuppertal, 42119 Wuppertal, Germany*
[d] *State Key Laboratory of Internet of Things for Smart City, Faculty of Science and Technology, University of Macau, Macau, China*

ARTICLE INFO

Article history:
Received 14 October 2018
Received in revised form 27 December 2018
Accepted 7 March 2019
Available online 19 March 2019

Keywords:
One-class classification
OC-ELM
ML-OCELM
Kernel learning
Outlier/anomaly detection

ABSTRACT

One-class classification has been found attractive in many applications for its effectiveness in anomaly or outlier detection. Representative one-class classification algorithms include the one-class support vector machine (SVM), Naive Parzen density estimation, autoencoder (AE), etc. Recently, the one-class extreme learning machine (OC-ELM) has been developed for learning acceleration and performance enhancement. But existing one-class algorithms are generally less effective in complex and multi-class classifications. To alleviate the deficiency, a multilayer neural network based one-class classification with ELM (in short, as ML-OCELM) is developed in this paper. The stacked AEs are employed in ML-OCELM to exploit an effective feature representation for complex data. The effective kernel based learning framework is also investigated in the stacked AEs of ML-OCELM, leading to a multilayer kernel based OC-ELM (in short, as MK-OCELM). The MK-OCELM has advantages of less human-intervention parameters and good generalization performance. Experiments on 13 benchmark UCI classification datasets and a real application on urban acoustic classification (UAC) are carried out to show the superiority of the proposed ML-OCELM/MK-OCELM over the OC-ELM and several state-of-the-art algorithms.

1. Introduction

One-class classification, also known as the anomaly or outlier detection, has attracted a lot of research interests in recent years (Juszczak, 2006; Tax, 2001). One-class algorithms aim to train the classifier with the target samples and reject the deviated data as outliers. In general, only the data of interests are well characterized while very few information on the outliers or negative data are available in the one-class classification. Comparing with conventional multi-class classifications, the one-class algorithm is more suitable for real applications, as practically, for the classifier learning, collecting data for the target of interests is more realistic and easier than collecting data for its opposites. A number of real world applications can be modeled as the one-class classification problem, such as the acoustic classification (Rabaoui, Davy, Rossignol, et al., 2007), fault detection (Lu & Wang, 2017), document classification (Alaei, Girard, Barrat, et al., 2014).

Comprehensive studies on one-class classification have been conducted in the past decades (Duin, 1976; Hu, Cao, Lai, & Liu, 2019; Khan & Madden, 2014; Parzen, 1962; Pekalska, Tax, & Duin, 2003; Scholkopf, Platt, Shawe, et al., 2001). The earliest attempt can be traced back to the Parzen density estimation method (Duin, 1976; Parzen, 1962), which approximates the probability density function (PDF) of the target and rejects the testing samples with a thresholding method. The drawback of Parzen density estimation is that an accurate estimation of the PDF relies on a large amount of labeled clean data. To achieve a maximum separation of the target data, a hyperplane has been constructed for classification in Scholkopf et al. (2001). An improved one-class linear programming classifier that minimizes the volume of prism by imposing the data dissimilarity constraints has been developed in Pekalska et al. (2003). A detailed survey on other popular one-class classifications can be found in Khan and Madden (2014) and references therein.

Recently, the artificial neural networks based one-class algorithms have been studied. The autoencoder (AE) based one-class classifier, that builds on a single hidden layer feedforward neural network (SLFN), has been developed in Gutoski, Ribeiro, Aquino, et al. (2017). The AE classifier is learnt to represent the data by reproducing the input patterns at the output layer and the training generally relies on the Backpropagation algorithm (BP). Other than the iterative updation methods, artificial neural networks assigned with random parameters and trained with the least-squares algorithm (LS) were also comprehensively investigated in the past, where the earliest can be traced back to Igelnik and

* Corresponding author at: School of Automation, Hangzhou Dianzi University, Zhejiang, 310018, China.
E-mail address: jwcao@hdu.edu.cn (J. Cao).

https://doi.org/10.1016/j.neunet.2019.03.004

Pao (1995). The recent extreme learning machine (ELM) (Huang, Chen, & Siew, 2006; Huang, Zhu, & Siew, 2004), which utilizes the random hidden node parameters and analytically solves the output weight with the LS method, became popular for its fast learning speed and good generalization performance (Cao, Zhang, Luo, Yin, & Lai, 2016; Cao, Zhang, Yong, Lai, Chen, and Lin, 2018; Kim, Kim, Jang, & Lee, 2017; Raghuwanshi & Shukla, 2018; Yang & Jonathan Wu, 2016; Yang, Wang, et al., 2012; Yang, Ye, Rong, & Chen, 2017). To alleviate the time-consuming issue of BP, the one-class extreme learning machine (OC-ELM), that takes the advantages of ELM (Huang et al., 2004), has been proposed in Leng, Qi, Miao, et al. (2015). Experiments on various benchmark datasets have shown the advantage of OC-ELM over many conventional one-class classifiers. A recently further enhancement OC-ELM, which exploits the geometric distribution of training data by introducing a graph Laplacian matrix into the cost function, has been presented in Iosifidis, Mygdalis, Tefas, et al. (2017).

However, the shallow structure of OC-ELMs may limit their capability in complex and high dimensional datasets learning, and thus lead to sub-optimal generalization performance. On the contrary, explosive developments on deep networks have been witnessed in the past one decade. The multilayer neural network based ELMs (ML-ELM) have also been investigated and the representative achievements include the stacked AEs based ML-ELM (Kasun, Zhou, Huang, et al., 2013), the deep weighted ELM (Wang, Cao, Lai, & Chen, 2018), the sparse representation based hierarchical ML-ELM (H-ELM) (Tang, Deng, & Huang, 2016), the kernel based MK-ELM (Wong, Vong, Wong, et al., 2018), etc., which provided us the inspirations of further improvement to OC-ELMs. In this paper, a novel multilayer network based one-class ELM (ML-OCELM) algorithm is developed. The stacked AEs are employed in ML-OCELM to exploit an effective feature representation for complex data. Meanwhile, the effective kernel based learning framework is adopted in the stacked AEs of ML-OCELM, leading to a novel multilayer kernel based OC-ELM (MK-OCELM). Benefiting from the kernel learning, the advantages of less human-intervention parameters in training and the exact inverse of kernel matrix in computing the transformation matrix of AEs bring MK-OCELM a fast learning speed and good generalization performance. Two experiments are conducted in the paper to show the advantages of the proposed algorithms. The first one studies the classification performance on 13 benchmark UCI datasets[1] and the second one considers the urban acoustic classification, where a large database containing 11 most frequently encountered urban acoustic signals recorded from the real environment is employed for performance evaluation. Several popular metrics, including the $\mathbf{F}_1$ score, overall accuracy, G-mean, precision and recall rate, are presented for the performance comparisons. Comparisons to many state-of-the-art one-class classification algorithms are also provided to show the superiority of the proposed algorithms.

The rest paper is organized in the following. Section 2 gives briefly, reviews of ELM and AE. Section 3 describes the proposed ML-OCELM and MK-OCELM algorithms. Section 4 presents the experiment results, comparisons and discussions. Section 5 concludes the paper with future work.

2. Preliminaries

2.1. ELM

Given a labeled training database $\left\{(\mathbf{x}_i, \mathbf{t}_i) \,|\mathbf{x}_i \in \mathbf{R}^d, \mathbf{t}_i \in \mathbf{R}^m, i = 1, 2, \ldots, N\right\}$, where $\mathbf{x}_i$ is the input vector, $\mathbf{t}_i$ is the target output, the input weights and hidden node bias are randomly generated and fixed in ELM. The training becomes optimizing the output weights by minimizing the error between the network outputs and the targets $\mathbf{T}$ as

$$\min_{\boldsymbol{\beta}} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_F^2 , \tag{1}$$

where $\boldsymbol{\beta}$ is the output weight matrix, $\mathbf{T}$ is the target output matrix, $\mathbf{H} = \left\{g_{ij}(\cdot)\right\}_{N\times L}$ with $g_{ij}(\cdot) = g\left(\mathbf{w}_j \cdot \mathbf{x}_i + b_j\right)$ is the hidden layer output matrix and $\|\cdot\|_F$ denotes the Frobenius norm. Here L is the number of hidden nodes, $g(\cdot)$ is the activation function, $\mathbf{w}_j$ and b_j are the input weight vector and bias of the jth node. The optimal output weight of (1) can be solved by LS, that is

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T}, \tag{2}$$

where $\mathbf{H}^{\dagger}$ is the generalized inverse of $\mathbf{H}$. The regularized ELM (RELM), that minimizes not only the training error but also the norm of output weights matrix, had been further developed in (Deng, Zheng, & Chen, 2009)

$$\min_{\boldsymbol{\beta}} \frac{1}{2} C \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_F^2 + \frac{1}{2}\|\boldsymbol{\beta}\|_F^2, \tag{3}$$

where C is the tradeoff between the training error and the norm of output weight. With the ridge regression theory (Hoerl & Kennard, 1970), (3) can be solved as

$$\boldsymbol{\beta} = \mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T}, \tag{4}$$

when the number of training data is much larger than the hidden node size, namely, $N \gg L$, a computational efficient solution of (3) is (Hoerl & Kennard, 1970)

$$\boldsymbol{\beta} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}. \tag{5}$$

2.2. OC-ELM

In one-class classifications, only the data of target class are available for the model training. The learning of OC-ELM thus changes to Leng et al. (2015) and Iosifidis et al. (2017)

$$\min_{\boldsymbol{\beta}} \frac{C}{2}\|\mathbf{H}\boldsymbol{\beta} - \mathbf{t}\|_2^2 + \frac{1}{2}\|\boldsymbol{\beta}\|_2^2, \tag{6}$$

where the training database is $\left\{(\mathbf{x}_i, t)\,|\mathbf{x}_i \in \mathbf{R}^d, i = 1, 2, \ldots, N\right\}$ and $\mathbf{t} = [t, \ldots, t]^T \in \mathbf{R}^N$ is the target output vector. In general, t can be set to any value. The output weight can be solved via (4) or (5) by replacing the matrix $\mathbf{T}$ with $\mathbf{t}$. To have a robust performance, OC-ELM rejects a small amount of samples through the error between the network output and its associated target, defined as

$$\varepsilon(\mathbf{x}_i) \triangleq |\xi_i| = \left|\sum_{j=1}^{L}\beta_j g\left(\mathbf{w}_j \cdot \mathbf{x}_i + b_j\right) - t\right|. \tag{7}$$

Ideally, the smaller the error is, the better the performance will be. A testing sample $\mathbf{x}^t$ with a large error is rejected as an anomaly or outlier, which is considered to be deviated from the target class. That is, when $\varepsilon\left(\mathbf{x}^t\right) \leqslant \theta$, $\mathbf{x}^t$ is classified as the target, otherwise, it is characterized as an outlier. Here, the error $\varepsilon\left(\mathbf{x}^t\right)$ is obtained by (7). In OC-ELM (Leng et al., 2015), the threshold θ is determined using the training errors where a certain percentage of training samples is considered as outliers or noises to ensure a good generalization performance. All the training errors are descendingly sorted as $\varepsilon_{(1)} \geqslant \cdots \geqslant \varepsilon_{(N)}$ and θ is then set to be $\varepsilon_{(i)}$ where the index i is determined by a manually selected percentage.

[1] Datasets available at https://archive.ics.uci.edu/ml/datasets.html.

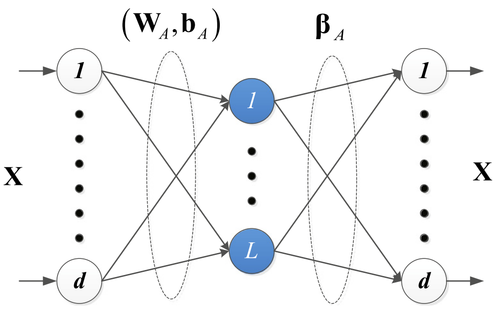


Fig. 1. ELM-AE.

3. The proposed multilayer one-class classifiers

3.1. ML-OCELM

Apparently, existing neural network based one-class classification algorithms generally build on a single hidden layer of neurons for data learning, which makes its less effective in characterizing complex and high dimensional datasets. An extension of the OC-ELM to multilayer network is thus developed to alleviate this deficiency in this section. The ELM based autoencoders (ELM-AE) are employed for the feature learning in the proposed ML-OCELM.

The stacked autoencoder (AE) has been applied to multilayer ELMs for feature extraction and learning (Kasun et al., 2013; Tang et al., 2016; Wong et al., 2018) and became popular for its good generalization performance and fast learning speed. The output weights in an AE are learnt to represent the data where the inputs are assigned as the outputs. In the proposed ML-OCELM, the stacked AEs are employed for feature learning. The general architecture of ELM-AE is shown in Fig. 1. After randomly generating the input weight matrix $\mathbf{W}_A$ and hidden bias $\mathbf{b}_A$ of an AE, the orthogonalization is operated on them, and then the output weight $\boldsymbol{\beta}_A$ is optimized as (Kasun et al., 2013)

$$\min \frac{C_A}{2}\left\|\mathbf{H}_A\boldsymbol{\beta}_A - \mathbf{X}^T\right\|_F^2 + \frac{1}{2}\left\|\boldsymbol{\beta}_A\right\|_F^2, \quad \text{s.t.} \quad \mathbf{W}_A\mathbf{W}_A^T = \mathbf{I}, \quad \mathbf{b}_A^T\mathbf{b}_A = \mathbf{I}. \tag{8}$$

Here, $\mathbf{H}_A$ denotes the hidden layer output matrix of the AE and $\mathbf{X}$ is the input data, which has been used as the output in AE for feature learning. Similarly, the optimal output weight of the AE (8) can be solved as

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

Particularly, when the number of training data is much larger than the hidden node size ($N \gg L$), a computationally more efficient calculation $\boldsymbol{\beta}_A = \left(\frac{\mathbf{I}}{C_A} + \mathbf{H}_A^T\mathbf{H}_A\right)^{-1}\mathbf{H}_A^T\mathbf{X}^T$ can be used. Then, the encoded output of the AE is

$$\mathbf{Y} = g\left(\boldsymbol{\beta}_A\mathbf{X}\right). \tag{10}$$

For stacked AEs, the encoded feature $\mathbf{Y}$ will be fed to the subsequent AE for feature learning. A brief summary of the ELM-AE is shown in Algorithm 1.

Algorithm 1 ELM-AE

Given:
Input data $\mathbf{X}$, regularization parameter C_A, the number of hidden nodes L.
Steps:

```
1. Generation and orthogonalization of the hidden parameters $\{\mathbf{W}_A, \mathbf{b}_A\}$
2. Calculation of the matrix $\mathbf{H}_A$ and the output weight $\boldsymbol{\beta}_A$ with (9)
3. Computation of the encoded result with (10)
```

In the proposed ML-OCELM algorithm, the stacked AEs are adopted for feature learning and the OC-ELM algorithm is adopted at the final stage for classification. Assuming K ELM-AEs are used in the feature extraction stage and letting $\mathbf{X}^{(k-1)}$ be the encoded feature of $(k-1)$th AE, the encoded feature of kth layer is

$$\mathbf{X}^{(k)} = g\left(\mathbf{X}^{(k-1)} \cdot \boldsymbol{\beta}_A^{(k-1)}\right), \tag{11}$$

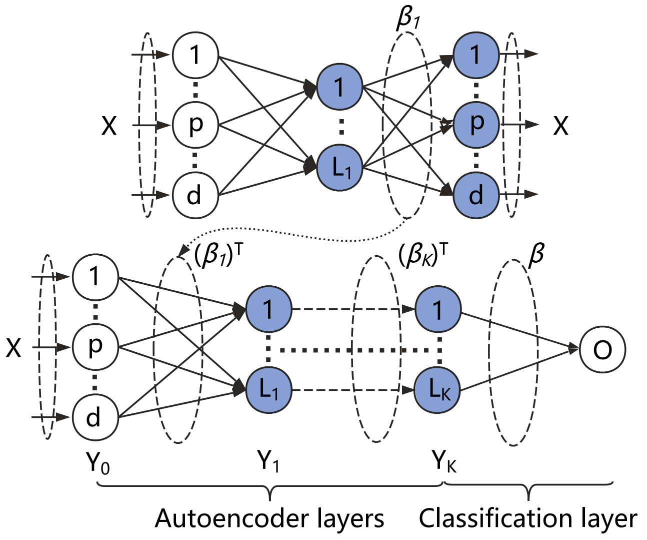


Fig. 2. ML-OCELM.

where $k = 1, \ldots, K$ and $\boldsymbol{\beta}_A^{(k)}$ is the learnt optimal weight. With the encoded feature, an OC-ELM is performed in the last stage for classification as

$$\min \frac{C}{2}\left\|\mathbf{H}_{\mathbf{X}^{(K)}}\boldsymbol{\beta} - \mathbf{t}\right\|_2^2 + \frac{1}{2}\left\|\boldsymbol{\beta}\right\|_2^2. \tag{12}$$

Here, $\mathbf{H}_{\mathbf{X}^{(K)}}$ is the hidden layer output matrix with the encoded feature $\mathbf{X}^{(K)}$ as the input feature. The optimal $\boldsymbol{\beta}$ can be solved via (4) or (5) by replacing the matrix $\mathbf{H}$ and $\mathbf{T}$ with $\mathbf{H}_{\mathbf{X}^{(K)}}$ and $\mathbf{t}$, respectively. Fig. 2 shows the structure of the proposed ML-OCELM.

With the trained ML-OCELM, the error for each sample can be calculated as

$$\varepsilon\left(\mathbf{x}_i^{(K)}\right) = \left|\mathbf{h}\left(\mathbf{x}_i^{(K)}\right) \cdot \boldsymbol{\beta} - t\right|, \tag{13}$$

where $i = 1, \ldots, N$ and $\mathbf{x}_i^{(K)}$ is the encoded feature of $\mathbf{x}_i$ by the stacked AEs. For all training samples, their errors are sorted in a descending order. A threshold θ is then assigned for the outlier detection. A brief summary of the proposed ML-OCELM is given in Algorithm 2.

Remark 1. It is noted that to reduce the redundant features and remove noises, the sparse feature encoding based AEs can be incorporated into the proposed ML-OCELM by imposing the ℓ_1-norm on the output weights in (12) (Tang et al., 2016). But for the sparse AE, there is no analytical solution for (12) and solving the optimal output weight has to resort to the ℓ_1-norm based optimization algorithms. Meanwhile, the ML-OCELM can also be

Algorithm 2 ML-OCELM

Given:

Training dataset $\mathbf{X}$, number of AEs K, number of hidden nodes $L_A^{(k)}$ and regularization parameter $C_A^{(k)}$ for each AE

Training:

1. For $k = 1 : K$, train the stacked AEs by Algorithm 1 and obtain the final encoded feature $\mathbf{X}^{(K)}$
2. Calculate the optimal output weight $\boldsymbol{\beta}$ through the model (12) and compute the network output for training dataset
3. Derive the errors of training dataset by (13) and determine the threshold θ

Testing:

1. Compute the encoded features $\mathbf{x}_t^{(K)}$ of the testing sample $\mathbf{x}_t$ with K AEs
2. Calculate the error $\varepsilon(\mathbf{x}_t^{(K)})$ by (13)
3. Perform the classification by:

$$\begin{cases} \varepsilon\left(\mathbf{x}_t^{(K)}\right) \leqslant \theta, & \mathbf{x}_t \text{ is a target} \\ \text{Else}, & \mathbf{x}_t \text{ is a outlier} \end{cases}$$

extended by using the kernel learning based AE, which has been testified to have good generalization performance in Wong et al. (2018).

3.2. MK-OCELM

The kernel learning framework is well recognized for its less human-intervention parameters and good generalization performance (Chen et al., 2018; Chen, Xing, Wang, Qin, & Zheng, 2018; Huang, Zhou, Ding, & Zhang, 2012; Steinwart & Christmann, 2008; Tang et al., 2016). The kernel based ELM (KELM) for a SLFN was first presented in Huang et al. (2012). The tuning of hidden node size as well as hidden parameters are not necessary in KELM, resultant a fast learning speed. Meanwhile, the exact invertible kernel matrix ensures better generalization performance in KELM with smaller reconstruction error than the pseudoinverse based ELMs (Tang et al., 2016). With (3) and (4), the output of ELM can be expressed as

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = \mathbf{h}(\mathbf{x})\mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T}, \tag{14}$$

where $\mathbf{h}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) & \cdots & g_L(\mathbf{x}) \end{bmatrix}$ is the hidden node output of $\mathbf{x}$. Applying the Mercer's condition, the kernel matrix $\boldsymbol{\Omega}$ using a kernel function κ was defined in ELM as $\boldsymbol{\Omega} = \mathbf{H}\mathbf{H}^T$, where $\Omega_{i,j} = \mathbf{h}(\mathbf{x}_i)\mathbf{h}(\mathbf{x}_j) = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. Replacing $\mathbf{h}(\mathbf{x})\mathbf{H}^T$ and $\mathbf{H}\mathbf{H}^T$ of (14) with the kernel function, the solution of KELM can be then represented as

$$f(\mathbf{x}) = \begin{bmatrix} \kappa(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ \kappa(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \boldsymbol{\Omega}\right)^{-1}\mathbf{T}. \tag{15}$$

The kernel method has been extended as a training algorithm for the feature learning of stacked AEs in ML-KELM (Wong et al., 2018). The kernel transformation matrix $\boldsymbol{\Omega}^{(k)}$ is adopted as the hidden layer outputs for feature learning. The feature learning can be then expressed as optimizing the output weight $\boldsymbol{\beta}^{(k)}$ by $\min_{\boldsymbol{\beta}^{(k)}} \left\|\boldsymbol{\Omega}^{(k)}\boldsymbol{\beta}^{(k)} - \mathbf{X}^{(k)}\right\|$. Here, $\mathbf{X}^{(k)}$ and $\boldsymbol{\Omega}^{(k)}$ denote the input data and the hidden layer output matrix (also referred to the kernel transformation matrix) of the kth AE. With (15), the optimal $\boldsymbol{\beta}^{(k)}$ can be derived as

$$\boldsymbol{\beta}^{(k)} = \left(\frac{\mathbf{I}}{C^{(k)}} + \boldsymbol{\Omega}^{(k)}\right)^{-1}\mathbf{X}^{(k)}, \tag{16}$$

where $C^{(k)}$ is the regularization parameter of the kth AE. The encoded feature of the kth AE, which becomes the input data of the subsequent AE, can be similarly calculated by (11) as $\mathbf{X}^{(k+1)} = g\left(\mathbf{X}^{(k)} \cdot \boldsymbol{\beta}^{(k)}\right)$.

In the proposed one-class multilayer kernel ELM (MK-OCELM), the output model in the last classification layer changes to $f(\mathbf{x}) = \begin{bmatrix} \kappa(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ \kappa(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \boldsymbol{\Omega}\right)^{-1}\mathbf{t}$, and the optimal output weight can be then derived with (16) by replacing $\boldsymbol{\Omega}^{(k)}$ and $\mathbf{X}^{(k)}$ with the one obtained by the transformation matrix $\boldsymbol{\Omega}^{\text{final}}$ using the final encoded features $\mathbf{X}^{\text{final}}$ and the target output $\mathbf{t}$, respectively. The error of each training sample $\mathbf{x}_i$ can be derived as

$$\varepsilon\left(\mathbf{x}_i^{(K)}\right) = \left\| \begin{bmatrix} \kappa(\mathbf{x}_i^{(K)}, \mathbf{x}_1) \\ \vdots \\ \kappa(\mathbf{x}_i^{(K)}, \mathbf{x}_N) \end{bmatrix}^T \boldsymbol{\beta} - t \right\|, \tag{17}$$

where $i = 1, \ldots, N$ and $\mathbf{x}_i^{(K)}$ are the encoded features of the K stacked ELM kernel AEs stated above (Wong et al., 2018). Similar to the OC-ELM and ML-OCELM, the anomaly/outlier detection of MK-OCELM is performed by a thresholding method based on the error. A brief summary of the MK-OCELM is given in Algorithm 3. Comparing to the traditional AE, there are no input weights and biases in the transformation matrix $\boldsymbol{\Omega}$ of the kernel based ELM AE. The number of hidden nodes is exactly the same as the training samples. Tuning the hidden node size is thus unnecessary in the kernel AE. Meanwhile, when the feature encoding activation function $g(\cdot)$ is a linear piecewise activation function, it is shown in Wong et al. (2018) that the transformation matrix $\boldsymbol{\beta}^{(k)}$ can be unified to a single one for all AEs ($k > 1$). As such, the computational burden can be dramatically reduced in the kernel ELM AEs, leading to a fast training speed for MK-OCELM.

Algorithm 3 MK-OCELM

Given:

Training dataset $\mathbf{X}$, number of AEs K, and regularization parameter $C^{(k)}$ for each AE

Training:

1. For $k = 1 : K$, calculate $\Omega^{(k)}$ and $\boldsymbol{\beta}^{(k)}$, compute the encoded feature $\mathbf{X}^{(k+1)}$
2. Calculate $\boldsymbol{\Omega}^{\text{final}}$ and $\boldsymbol{\beta} = \left(\frac{\mathbf{I}}{C} + \boldsymbol{\Omega}^{\text{final}}\right)^{-1}\mathbf{t}$
3. Derive the errors of training dataset by (17) and determine the threshold θ

Testing:

1. Compute the encoded features $\mathbf{x}_t^{(K)}$ of the testing sample $\mathbf{x}_t$
2. Calculate the error $\varepsilon(\mathbf{x}_t^{(K)})$ by (17)
3. Perform the classification by:

$$\begin{cases} \varepsilon\left(\mathbf{x}_t^{(K)}\right) \leqslant \theta, & \mathbf{x}_t \text{ is a target} \\ \text{Else}, & \mathbf{x}_t \text{ is a outlier} \end{cases}$$

Remark 2. Although the kernel learning method has the advantages of fast training speed and good generalization performance, it may not be suitable for the small sample dataset learning problems (also known as the few-shot learning problem) with a high feature dimension. When $d \gg N$, too much information will be filtered out in the kernel AEs during computing $\boldsymbol{\Omega}$ as the dimension of the kernel matrix is $N \times N$. The kernel AE has a similar capability of moderate dimension reduction to the principal component analysis (PCA), but an excessive compression may remove too much useful information and the resultant generalization performance may be poor. Thus, developing effective kernel learning methods for small sample dataset with high-dimensional features becomes an open problem and is worthy for further studies in the future.

4. Experiments

4.1. Experiments on UCI datasets

Experiments conducted on 13 UCI benchmark classification datasets, consisting of 8 small- and 5 medium-size datasets, respectively, are presented in this section for the performance comparison. The dataset specifications are shown in Table 1. The datasets with high-dimensional features or large training sample size are treated as the medium-size datasets. For small-size datasets, the target class is shown in Table 1 and for media-size datasets, the samples of the first class are set to be the targets and the rest are considered as the outliers. All the features are normalized into the range [0, 1]. For all datasets, the samples from the target class are equally partitioned into the training and testing datasets, respectively. The testing dataset combining with the outliers are then adopted for the algorithm testing. To determine the threshold θ for OC-ELM, ML-OCELM and MK-OCELM, 10% of training samples are set as the outliers during the training phase.

Similar to Leng et al. (2015), the overall accuracy, precision ($\mathbf{P}$), recall rate ($\mathbf{R}$), $\mathbf{F}_1$ score and G-mean are adopted as the measurements for performance assessment in this paper. The definitions of these assessments are shown below,

$$\text{Overall accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{18}$$

$$\mathbf{P} = \frac{TP}{TP + FP} \tag{19}$$

$$\mathbf{R} = \frac{TP}{TP + FN} \tag{20}$$

$$\mathbf{F}_1 = \frac{2\mathbf{R} \cdot \mathbf{P}}{\mathbf{P} + \mathbf{R}} \tag{21}$$

$$\text{G-mean} = \sqrt{\mathbf{R} \cdot \frac{TN}{TN + FP}} \tag{22}$$

where TP, TN, FP, and FN denote the true positive, true negative, false positive, and false negative, respectively. The overall accuracy reflects the proportion of all predictions that are correct. The precision $\mathbf{P}$ shows the proportion of all positive predictions that are correct. It measures the amount of positive predictions among all actual positive observations. The recall rate $\mathbf{R}$ indicates how many actual positive observations are correctly predicted. The $\mathbf{F_1}$ score is the harmonic mean of the precision and recall rate. It is an appropriate way to reflect the average ratio. The G-mean measures the geometric mean of sensitivity and specificity. For all the small-size datasets, the performance are compared with many state-of-the-art one-class classification algorithms given in Leng et al. (2015), including Parzen density estimation (Parzen) (Parzen, 1962), Naive Parzen density estimation (Naive Parzen) (Duin, 1976), k-means (Jiang, Tseng, & Su, 2001), k-centers (Hochbaum & Shmoys, 1985), 1-Nearest Neighbor (1-NN) (Tax & Duin, 2000), k-Nearest Neighbor (k-NN) (Knorr, Ng, & Tucakov, 2000), Autoencoder neural network(AE) (Tax, 2001), principal component analysis (PCA) (Bishop, 1995), a minimum spanning tree based one-class classifier (MST) (Juszczak, Tax, Pkalska, et al., 2009), minimax probability machine (MPM) (Lanckriet, Ghaoui, Jordan, et al., 2003), super vector data description (SVDD) (Tax & Duin, 2004), linear programming dissimilarity-data description (LPDD) (Pekalska et al., 2003), and support vector machine (SVM) (Scholkopf et al., 2001), where the results of all these algorithms are directly taken from (Leng et al., 2015) for comparison. To have a fair comparison, the dataset set-ups of OC-ELM (Leng et al., 2015), and the proposed ML-OCELM and MK-OCELM, are the same as Leng et al. (2015).

Table 1
UCI dataset specifications.

Datasets	Target class	Target	Outlier	Features
Small-size:				
Spectfheart	0	55	212	44
Arrhythmia	Normal	245	207	279
Sonar	Mines	111	97	60
Liver	Healthy	145	200	6
Ecoli	Periplasm	52	284	7
Diabetes	Present	500	268	8
Breast	Benign	241	458	9
Abalone	Classes1–8	1 407	2 770	8
Medium-size:				
Letter	1	10 000	10 000	16
Pendigits	1	7 494	3 498	16
Optical digits	1	3 823	1 797	64
Magic	1	10 000	9 020	10
Satimage	1	3 217	3 218	36

All the experiments are carried out in the Matlab R2017b environment running in a PC with the Intel Core i5-6500T 2.5 GHz CPU, 4 GB RAM. For each algorithm, twenty trials are conducted for each dataset. That is, the target dataset is randomly partitioned into the training and testing with an equal size in each trial. The average results are then reported in this paper. The kernel function $\kappa^k(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|}{2\sigma_k^2}}$ is adopted in MK-OCELM, where σ_k denotes the kernel size in the kth kernel matrix. Three layers of stacked AEs are used in ML-OCELM and MK-OCELM, including two stacked AEs and one classification layer. For ML-OCELM, the numbers of hidden nodes of the two AEs and the last classification layer are optimized within the range {100,700,1000} and {500, 1000, 2000}, respectively. The corresponding tradeoff parameter C for the AEs and the last classification layer are chosen from the range $\left\{10^{-5}, 10^{-3}, 0, 10^{3}, 10^{5}\right\}$ and $\left\{10^{-5}, 10^{-3}, 0, 10^{3}, 10^{5}\right\}$, respectively. For MK-OCELM, it is not necessary to tune the number of hidden nodes. Only the tradeoff parameter C and the kernel size σ need to be optimized, which are both searched within the ranges $\{10^{-3}, 10^{-1}, \ldots, 10^{3}\}$.

It is well-known that for most applications, the precision $\mathbf{P}$ and recall rate $\mathbf{R}$ are not the best metric in performance assessment. For some applications, the recall rate and precision are unlikely to be the best of both worlds. The $\mathbf{F}_1$ score, which is a harmonic mean of $\mathbf{P}$ and $\mathbf{R}$, is a more reasonable measure. With this objective, the $\mathbf{F_1}$ scores of the 8 small-size datasets are reported in Table 2. The best results are highlighted with the bold font in the table. The proposed ML-OCELM and MK-OCELM algorithms obtain the highest $\mathbf{F_1}$ score on 6 datasets while the Parzen and k-means algorithms win the rest 2 datasets, respectively. Among the 6 datasets, the best of ML-OCELM/MK-OCELM offers 1.7% $\sim$ 7.9% increments on the $\mathbf{F_1}$ score over the highest results obtained by

Table 2
Comparisons of $\mathbf{F_1}$ score on small-size datasets.

Dataset classifier	Spectfheart	Arrhythmia	Sonar	Liver	Ecoli	Diabetes	Breast	Abalone
Naive Parzen	41.7	61.8	46.8	41.5	71.7	68.7	82.4	51.8
Parzen	39.3	63.7	49.8	40.7	**75.1**	67.7	80.1	49.0
k-means	38.3	63.7	53.2	41.7	54.6	**68.9**	58.8	45.2
1-NN	31.8	59.2	60.4	41.3	21.2	64.8	35.3	35.7
k-NN	34.7	62.4	55.3	42.0	43.9	68.8	34.9	49.8
Autoencoder	39.3	64.8	50.6	42.2	53.5	66.9	37.9	48.7
PCA	NaN	26.3	37.2	41.1	33.7	65.5	31.1	46.0
MST	33.7	62.4	56.7	42.1	36.3	67.5	34.4	47.3
k-centers	36.4	62.8	53.3	41.6	38.8	67.7	49.4	42.5
SVDD	38.9	60.5	51.2	40.6	50.9	61.1	68.0	44.5
MPM	31.1	51.9	44.6	40.7	38.8	63.5	71.7	44.9
LPDD	38.3	63.8	52.2	40.7	67.8	66.6	79.6	45.8
SVM	38.1	63.4	53.6	40.5	57.2	66.6	83.1	46.2
OCELM	52.1	76.8	69.6	55.5	52.9	67.0	90.7	68.1
ML-OCELM	54.0	78.3	72.6	**59.3**	57.6	67.7	**95.7**	68.4
MK-OCELM	**58.7**	**78.7**	**77.5**	58.5	70.8	68.1	94.4	**71.4**

the rest of the compared algorithms. In addition, for the two proposed multilayer one-class algorithms, the MK-OCELM generally outperforms ML-OCELM.

Besides the $\mathbf{F_1}$ score, Fig. 3 draws the overall accuracy, G-mean, precision **P**, and recall rate **R** obtained by the OC-ELM and the proposed ML-OCELM/MK-OCELM on all 8 small-size datasets. It is readily found that the proposed ML-OCELM/MK-OCELM algorithms generally achieve better performance than OC-ELM (Leng et al., 2015) on the overall accuracy, G-mean, and precision **P**. Particularly, the ML-OCELM/MK-OCELM win the highest overall accuracy on 7 out of 8 datasets and the OC-ELM only offers the best overall accuracy on the Arrhythmia dataset. Similar observations can be found in the assessments of G-mean and precision **P**. While for the recall rate **R**, the proposed ML-OCELM offers the best results on 5 datasets and the original OC-ELM wins on 2 datasets. It is consistent with the previous statement that when the precision is high, the corresponding recall rate is usually low. For instance, the proposed MK-OCELM algorithm offers the highest precision on the Liver, Abalone, Ecoli, Spectfhear, Diabetes, and Sonar datasets, the associated recall rates on these datasets are also relatively low. On the contrary, for the Arrhythmia dataset, the MK-OCELM provides the smallest precision but achieves a highest recall rate.

For the medium-size datasets, similar experiments are conducted and the performance comparisons on the original OC-ELM and the proposed ML-OCELM/MK-OCELM are presented. Table 3 shows the obtained average $\mathbf{F_1}$ scores. The results are consistent with the observations of aforementioned small-size datasets. As expected, the multilayer network based ML-OCELM and MK-OCELM outperform the original OC-ELM on all datasets. Particularly, the MK-OCELM achieves more than 6.42% and 45.81% increments of the $\mathbf{F_1}$ score over the original OC-ELM on the Pendigits and Letter datasets, respectively. Meanwhile, employing the kernel transformation in MK-OCELM helps to improve the classification performance. One can readily see from the table that MK-OCELM offers higher $\mathbf{F_1}$ score than ML-OCELM on 4 out of 5 datasets. Further, the overall accuracy, G-mean, precision **P**, and recall rate **R** of the medium-size datasets are presented in Fig. 4, respectively. As observed in the figure, the proposed ML-OCELM/MK-OCELM generally outperforms the original OC-ELM under all metrics. For instances, MK-OCELM achieves the highest overall accuracy on 3 datasets and ML-OCELM wins on the rest 2 datasets, and using the G-mean assessment, MK-OCELM and ML-OCELM offer the best performance on 4 and 1 datasets, respectively. It is noteworthy that the original OC-ELM fails in the Letter dataset, while the proposed MK-OCELM offers more than 44% increments of the $\mathbf{F_1}$ score over OC-ELM.

The parameter sensitive of the proposed ML-OCELM/MK-OCELM is also studied in this experiment. We present the $\mathbf{F_1}$ score

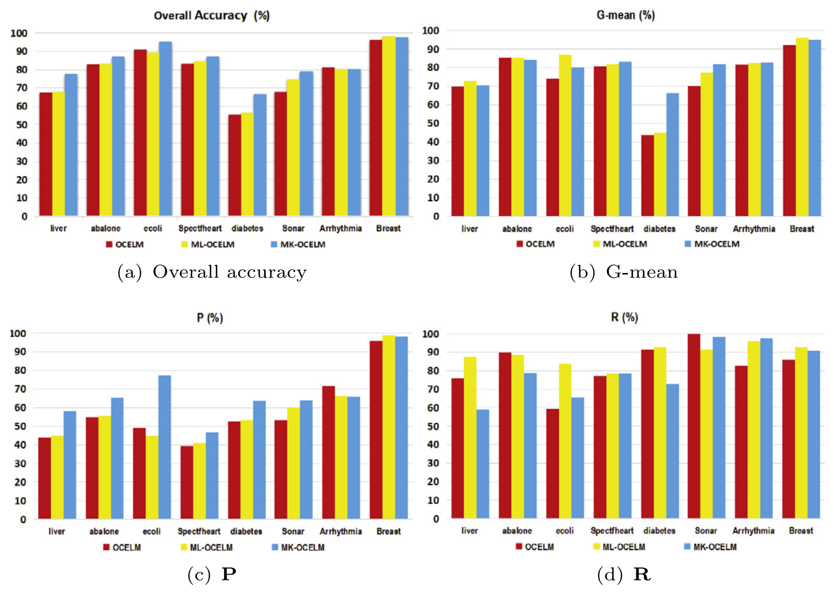
Table 3
Comparisons of $\mathbf{F_1}$ score on medium-size datasets.

Classifier datasets	OCELM	ML-OCELM	MK-OCELM
Letter	43.45	61.35	**89.26**
Pendigits	86.82	88.10	**93.24**
Optical digits	94.42	95.12	**95.35**
Magic	83.81	85.05	**86.22**
Satimage	93.31	**95.56**	94.12

obtained on the Abalone and Satimage by using different tradeoff parameters in ML-OCELM, and different tradeoff parameters and kernel sizes in MK-OCELM in Fig. 5, respectively. For ML-OCELM, C in y-axis denotes the tradeoff used in all AEs while for ML-OCELM and MK-OCELM, C in the x-axis represents the tradeoff adopted in the last classification layer. One can observe from these plots that ML-OCELM/MK-OCELM are generally robust the network tradeoff parameter and kernel size, besides a very small tradeoff is employed in the classification layer of ML-OCELM for Satimage and the combination of a very large σ and a very small C is adopted in MK-OCELM for Abalone and Satimage. Similar performance can be found for the other datasets, while due to the page limitation, we will omit them in the paper.

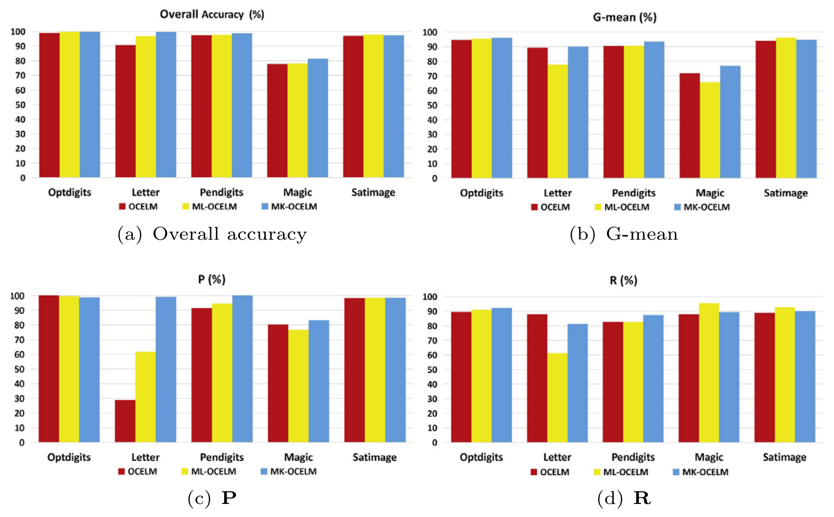
In addition, we know that the decision criterion of one-class algorithms is decided by setting a percentage of training samples as noises to enhance its capability. To test the performance of the one-class algorithm w.r.t. different thresholds, we conduct the experiments by setting three different percentages of training samples as outliers, namely mu $= 0.01, 0.05, 0.1$, respectively. The classification performance of OC-ELM, ML-OCELM and MK-OCELM are compared in all 13 UCI datasets, where the $\mathbf{F_1}$ scores are collected for comparisons. Fig. 6 draws the $\mathbf{F_1}$ scores w.r.t. the three different percentages for all 13 datasets, respectively. We can clearly see that when mu$=0.05$, the OC-ELM algorithm wins 6 highest $\mathbf{F_1}$ scores of 13 datasets, while for mu $= 0.01$ and 0.1, it reduces to 4 and 3 datasets, respectively. The ML-OCELM achieves highest $\mathbf{F_1}$ score on 9, 1, and 3 datasets when mu $= 0.01, 0.05, 0.1$, respectively. For MK-OCELM, the numbers of highest $\mathbf{F_1}$ score datasets are 7, 5, and 1. In general, one can find that using a small mu can guarantee a high $\mathbf{F_1}$ score. But when a small percentage is used, less training samples will be considered as noises, which may enhance the performance of the model. But it also has a risk of increasing the over-fitting, leading to a poor generalization performance, and vice versa. One can also find that for all three different percentages of outliers, the proposed ML-OCELM/MK-OCELM algorithms generally outperforms OC-ELM.

Computational complexity is another important index for the algorithm assessment. In this paper, the CPU time spend in the



(a) Overall accuracy (b) G-mean (c) **P** (d) **R**

Fig. 3. Comparisons of (a) overall accuracy (b) G-mean (c) **P** and (d) **R** on small-size datasets.



(a) Overall accuracy (b) G-mean (c) **P** (d) **R**

Fig. 4. Comparisons of (a) overall accuracy (b) G-mean (c) **P** and (d) **R** on medium-size datasets.

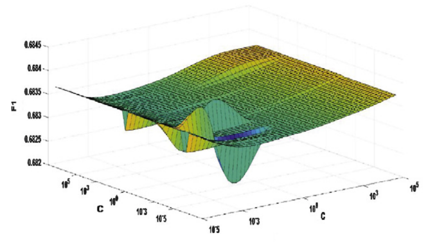
training phase by the original OC-ELM and the proposed ML-OCELM/MK-OCELM are recorded for comparison in Table 4. Due to the usage of multilayer structure and the increased tuning parameters, the training time costed by ML-OCELM is longer than the one by OC-ELM and MK-OCELM. It is also worth to point out that employing the unify transformation matrix in the kernel and without the step of optimizing the number of hidden neurons, MK-OCELM has fast training speed in most datasets. As highlighted in Table 4, MK-OCELM provides the lowest training time on 9 datasets while the original OC-ELM wins on the rest 4 datasets.

Table 4
Training time (s) on the UCI datasets.

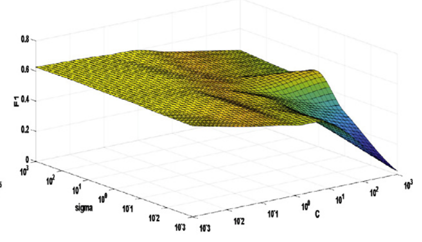
Datasets	OCELM	ML-OCELM	MK-OCELM
Spectfheart	0.0446	7.1055	**0.0203**
Arrhythmia	0.1081	14.0371	**0.0164**
Sonar	0.0367	24.0031	**0.0051**
Liver	0.2075	24.3454	**0.0054**
Ecoli	**0.0010**	7.4804	0.0271
Diabetes	**0.0879**	1.9006	0.1236
Breast	0.0667	21.1637	**0.0058**
Abalone	0.1729	2.9762	**0.0244**
Letter	0.1202	4.9638	**0.0454**
Pendigits	0.2941	7.2322	**0.1833**
Optical digits	0.1203	4.8635	**0.0277**
Magic	**0.1031**	31.0864	19.9821
Satimage	**0.0661**	7.2343	0.1177

4.2. Experiments on urban acoustic classification

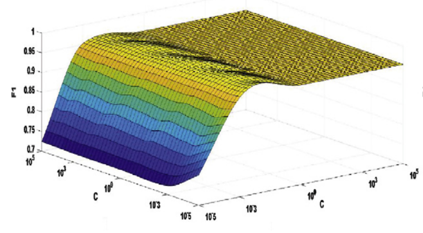
Urban acoustic classification (UAC) aims at the identification of the class of frequently encountered acoustical streams in the urban environment. UAC is vital to the smart city engineering (Cao, Cao, Wang, Yin, Wang, and Vidal, 2018; Piczak, 2015), urban security (Cao, Shang, Wang, Vong, Yin, and Cheng, 2018; Cao, Wang, Wang, & Wang, 2017), noise pollution analyses (Agha, Ranjan, & Gan, 2016; Torija & Ruiz, 2016), etc., but is also a challenging task due to the complex and changeable environment as well as due to the various kinds of background noises. Thus, comparing



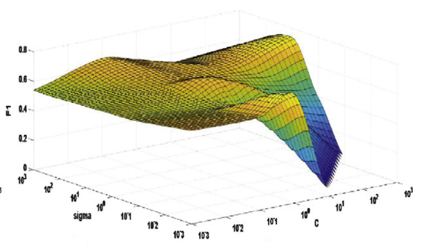
(a) Abalone (ML-OCELM)



(b) Abalone (MK-OCELM)

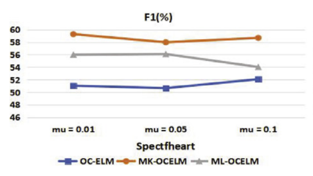


(c) Satimage (ML-OCELM)

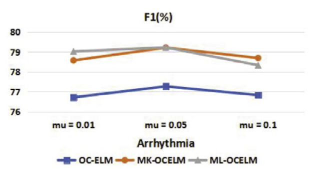


(d) Satimage (MK-OCELM)

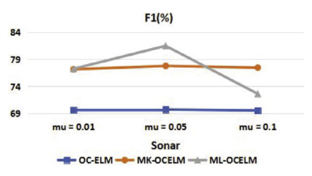
Fig. 5. Performance on different network parameters (tradeoff C and kernel size σ).



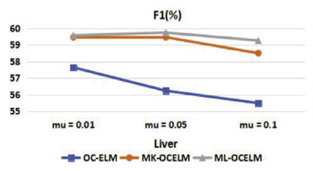
(a) Spectfheart



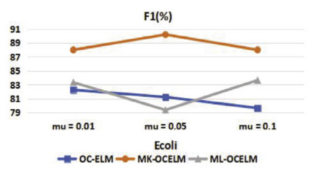
(b) Arrhythmia



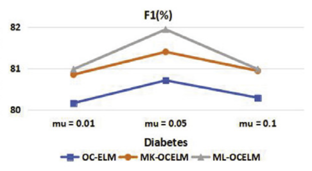
(c) Sonar



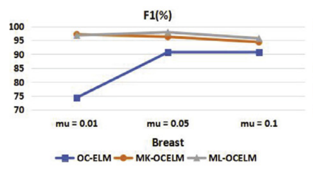
(d) Liver



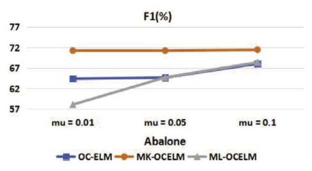
(e) Ecoli



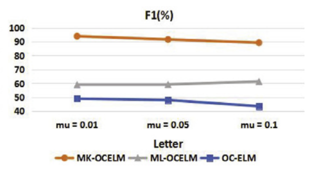
(f) Diabetes



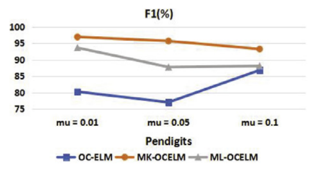
(g) Breast



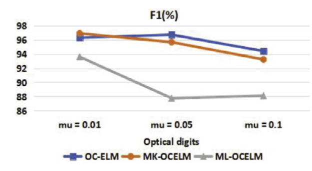
(h) Abalone



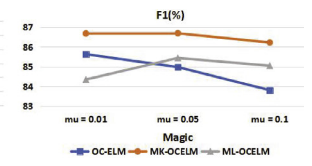
(i) Letter



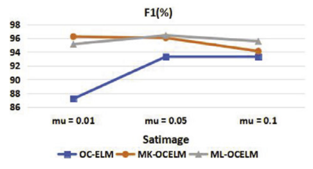
(j) Pendigits



(k) Opticaldigits



(l) Magic



(m) Satimage

Fig. 6. Comparisons of $\mathbf{F_1}$ score on different percentages of outliers.

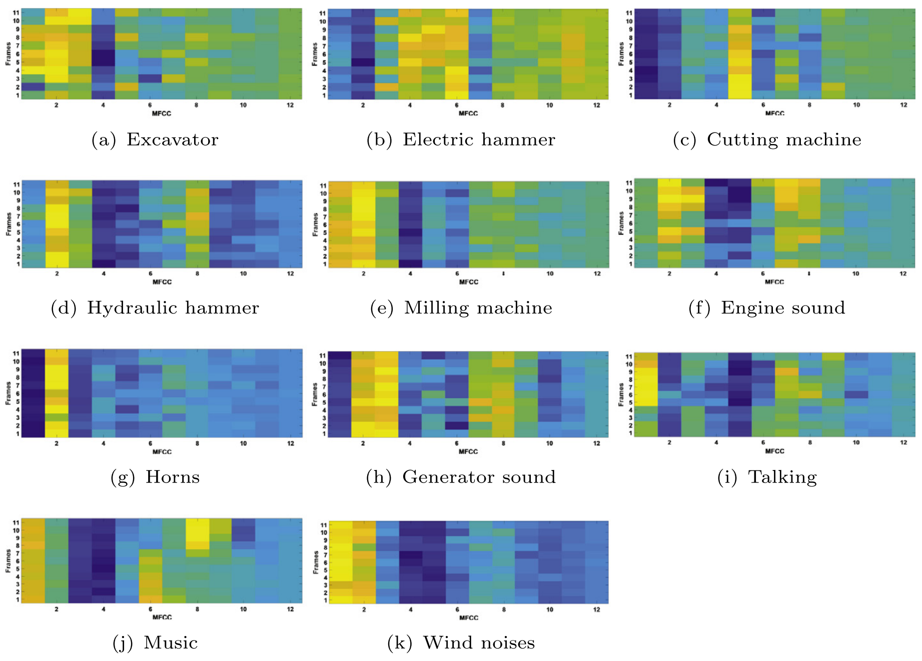


Fig. 7. MFCCs of urban acoustic waveforms.

with the conventional multi-class classification algorithms, the one-class classification methods are more practical in UAC. It is more reasonable to build a specific one-class classifier for each category of interested urban acoustical signals and make rejection to acoustic streams from other categories as the outliers.

In this section, an experiment conducted on a real recorded acoustic dataset consisting of 11 representative urban acoustic streams is presented for performance comparison. These acoustic signals are the most frequently encountered ones in the urban environment, including the engine sound of passing vehicles, horns of vehicles, sound of high-rating generator, human talking, music of cellphones, acoustics generated by five frequently used construction machines (including the excavator, electric hammer, cutting machine, hydraulic hammer, and pavement milling machine), and wind noises (Cao et al., 2018). All the acoustic streams are recorded in the urban environment with a cross-layer MEMS (type: SPH0611LR5H) microphone sensor. The sampling frequency is set to be 20 kHz. To have a well characterization of the time-frequency property of urban acoustics, the concatenated Mel-Frequency Cepstral Coefficients (MFCC) (Davis & Mermelstein, 1980) obtained from multiple frames of a segment acoustic stream are extracted, where the MFCC is defined as

$$c(n)=\sum_{m=0}^{M-1}S(m)\cos\left[\frac{\pi n(m+1/2)}{M}\right]. \tag{23}$$

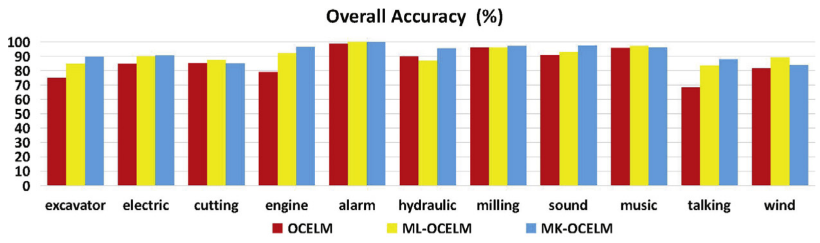
Here, n is the sample index, $m=1,\dots,M$ represents the Mel filters, and $S(m)$ is the logarithmic output of the mth Mel filter bank on the spectrum (Cao et al., 2018), which is generally obtained by the discrete Fourier transform (DFT) on a short frame of acoustic signals. MFCC was shown effective in automatic speech recognition due to its good capability in describing the shape of sound channels through extracting the envelop of the short-time power spectrum.

Similar to Cao et al. (2018), to preserve the time-frequency information of urban acoustics, multiple consecutive frames extracted from an acoustic stream are used to calculate a set of MFCC features, which are then concatenated for classifier learning. In the experiment, 11 frames with 50% overlaps are used and the frame length is set to be 1024 points. The number of Mel filters is 12 and the resultant feature dimensions for each sample are 132. Fig. 7 shows the visualized MFCCs for all 11 urban acoustic waveforms, where for each image, the horizontal axis draws the 12-ordered coefficients obtained by Mel filters on each frame and the vertical axis indicates the number of frames. The intensity of the image along with the horizontal axis can roughly indicate the spectra distribution from low to high frequencies. It is obvious that the power spectra distributions are generally consistent with each other among the acoustic frames of a same class. Moreover, one can readily find that the spectra distribution varies among different urban acoustic classes. For instance, the power spectra of the acoustic waveforms generated by electric hammer distributes evenly from low to high frequency bands, while for the horns of passing vehicles and wind noises, the energy mainly concentrates in the low frequency bands.

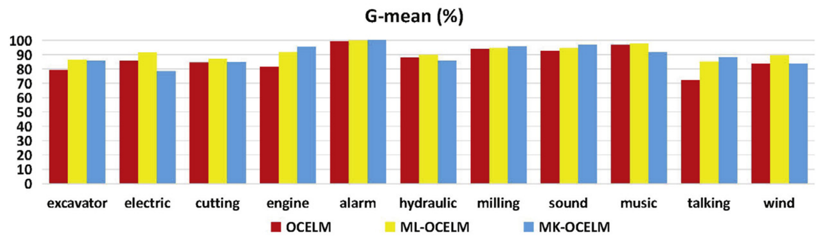
Table 5
UAC databases Specifications.

Datasets	Training (Target)	Testing (Target+Outliers)
Excavator	2536	3025 (665+2360)
Electric hammer	2931	2832 (462+2370)
Cutting machine	4455	4588 (2310+2278)
Engine sound	4000	3499 (1164+2335)
Horns	5000	6184 (3990+2194)
Hydraulic hammer	5000	12 189 (10311+1878)
Milling machine	4000	3205 (854+2351)
Generator sound	5000	8322 (6241+2081)
Music	5000	18 205 (16644+1561)
Talking	5000	3461 (1124+2337)
Wind noises	5000	6346 (4164+2275)

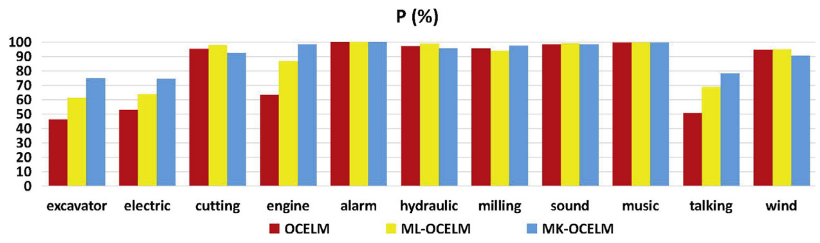
Table 5 shows the specifications of the UAC databases used in this experiment, where each sample contains 132 features. For each acoustic dataset, a one-class classifier is trained on the training dataset and the classifier performance is validated by the testing dataset. It is noteworthy that for each acoustic source, the testing dataset includes two categories of samples, that is, one is from the target category and the other is set to be the outliers to the acoustic source, specified in Table 5 as "Target + Outliers". For each database, 5% testing samples of all the rest databases are taken to construct the outlier dataset to validate the robustness



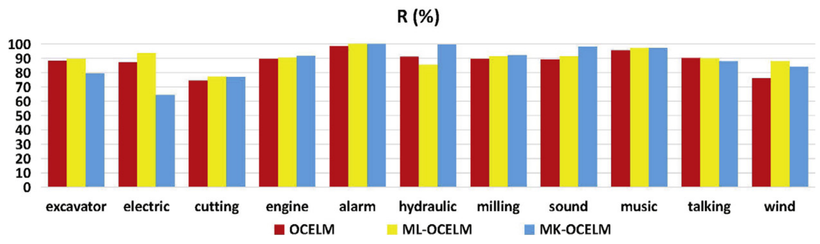
(a) Overall accuracy



(b) G-mean



(c) **P**



(d) **R**

Fig. 8. Comparisons of (a) overall accuracy (b) G-mean (c) **P** and (d) **R** of UAC databases.

of the algorithms. For instance, for the excavator database, there exist 3025 samples in the testing dataset, including 665 targets and 2360 outliers, where the outlier dataset is built by randomly taking 5% samples from the testing datasets of all the rest 10 databases other than the excavator. For the classifier training, the range settings for parameters optimizations are the same to the previous experiment. Similarly, for performance evaluation and comparison, the $\mathbf{F_1}$ scores obtained by the original OC-ELM and the proposed ML-OCELM and MK-OCELM are listed in Table 6. These metrics are obtained by averaging the results of twenty trials of experiment. As highlighted, ML-OCELM/MK-OCELM outperform the original OC-ELM on all urban acoustic datasets. Particularly, MK-OCELM offers the highest $\mathbf{F_1}$ score on 7 datasets and ML-OCELM wins the rest 4. For some acoustic datasets, the improvement obtained by ML-OCELM/MK-OCELM is significant. For the datasets of excavator, engine sound and talking, MK-OCELM provides 16.32%, 20.7%, and 17.92% increments of the $\mathbf{F_1}$ score over OC-ELM, respectively. Meanwhile, for the datasets of electric hammer and wind noises, ML-OCELM offers 10.2% and 7.03% improvements over OC-ELM, respectively.

Similar to the previous experiment, the overall accuracy, G-mean, precision **P** and recall rate **R** of the 11 urban acoustic

Table 6
Comparisons of $\mathbf{F}_1$ score on UAC databases.

Datasets	OCELM	ML-OCELM	MK-OCELM
Excavator	60.87	72.57	**77.19**
Electric hammer	65.51	**75.71**	69.09
Cutting machine	83.64	**86.30**	83.90
Engine sound	74.27	88.57	**94.97**
Horns	99.20	99.93	**99.96**
Hydraulic hammer	93.99	91.71	**97.45**
Milling machine	92.59	92.58	**94.76**
Generator sound	93.46	95.14	**98.24**
Music	97.63	**98.49**	98.37
Talking	64.99	78.09	**82.91**
Wind noises	84.35	**91.38**	87.21

datasets are drawn in Fig. 8 for performance comparison, respectively. As shown in the figure, the performance is consistent with the observation of the first experiment. That is, ML-OCELM/MK-OCELM are superior to the original OC-ELM under all these assessments. It can be readily seen that ML-OCELM/MK-OCELM obtains the highest overall accuracy on all 11 urban acoustic datasets. The same results can be found when using the

Table 7
Comparisons of $\mathbf{F}_1$ score on single frame and multiple frames.

Datasets	OC-ELM		ML-OCELM		MK-OCELM	
	Single	Multiple	Single	Multiple	Single	Multiple
Excavator	54.49	**60.87**	63.36	**72.57**	62.39	**77.19**
Electric hammer	61.67	**65.51**	71.62	**75.71**	61.26	**69.09**
Cutting machine	83.01	**83.64**	85.94	**86.30**	82.74	**83.90**
Engine sound	**77.03**	74.27	86.13	**88.57**	88.64	**94.97**
Horns	97.21	**99.2**	**99.97**	99.93	99.85	**99.96**
Hydraulic hammer	**94.99**	93.99	**96.28**	91.71	**97.57**	97.45
Milling machine	84.01	**92.59**	88.85	**92.58**	92.12	**94.76**
Generator sound	92.85	**93.46**	**98.31**	95.14	**98.59**	98.24
Music	**97.7**	97.63	**98.53**	98.49	**98.52**	98.37
Talking	63.26	**64.99**	67.78	**78.09**	67.97	**82.91**
Wind noises	**86.26**	84.35	**91.43**	91.38	**87.68**	87.21

G-mean metric. While for the precision **P** and recall rate **R**, ML-OCELM/MK-OCELM provides the highest values on 10 out of 11 datasets.

To test the effectiveness of using concatenated MFCCs from multiple acoustic frames, an experimental comparison to the performance of UAC using the MFCC vector of a single frame is provided. Table 7 lists the $\mathbf{F_1}$ scores obtained from single and multiple acoustic frames, respectively. As highlighted in boldface, adopting the concatenated MFCC vectors of multiple acoustic frames generally offer better performance than single frame. It is also noteworthy that the concatenated MFCC vectors of multiple frames provide a convincing performance on those datasets with a relatively low $\mathbf{F_1}$ score. A detailed comparison can be found from the Excavator, Electric hammer, and Talking datasets. Using multiple frames achieves a significant increment of the $\mathbf{F_1}$ score. For instance, using concatenated MFCCs, the OC-ELM, ML-OCELM, and MK-OCELM offer 6.38%, 9.21%, and 14.8% increments of the $\mathbf{F_1}$ score over the ones using single frames, respectively.

5. Conclusions

The paper proposes two multilayer neural network based one-class classification algorithms for outlier/anomaly detection. The multilayer extreme learning machine (ELM) algorithm and the multilayer kernel ELM are exploited for one-class classification to improve the capability of the original OC-ELM. Two experiments are conducted to demonstrate the superiority of the developed ML-OCELM and MK-OCELM, where the first one tests on several benchmark classification databases and the second focuses on the real urban noises classification. The results on these experiments have shown that: (1) the multilayer network based one-class classifiers are more effective than OC-ELM and several related state-of-the-art algorithms; (2) benefitting from the unity transformation matrix and the tuning-free of neuron sizes in the kernel autoencoders, MK-OCELM has a low computational complexity in classifier training, which generally learns faster than or comparable to OC-ELM. However, the existing one-class classifications are mostly based on the mean square error (MSE) cost function, which are generally shown to be sensitive to outliers or non-Gaussian noises. Therefore, other than MSE, developing a more suitable cost function which is robust to outliers/non-Gaussian noises is worth for the further research.

Acknowledgments

This work was supported by the NNSF of China (61503104, U15092051) and by the K. C. Wong Education Foundation, DAAD, Germany.

References

Agha, A., Ranjan, R., & Gan, W. (2016). Noisy vehicle surveillance camera: a system to deter noisy vehicle in smart city. *Applied Acoustics*, *117*, 236–245.

Alaei, F., Girard, N., Barrat, S., et al. (2014). A new one-class classification method based on symbolic representation: application to document classification. In *IAPR international workshop on document analysis systems* (pp. 272–276).

Bishop, C. (1995). *Neural networks for pattern recognition*. Walton Street: Oxford University Press.

Cao, J., Cao, M., Wang, J., Yin, C., Wang, D., & Vidal, P.-P. (2018). Urban noise recognition with convolutional neural network. *Multimedia Tools and Applications*, http://dx.doi.org/10.1007/s11042-018-6295-8.

Cao, J., Shang, L., Wang, J., Vong, C., Yin, C., Cheng, Y., et al. (2018). A novel distance estimation algorithm for periodic surface vibrations based on frequency band energy percentage feature. *Mechanical Systems and Signal Processing*, *113*, 222–236.

Cao, J., Wang, W., Wang, J., & Wang, R. (2017). Excavation equipment recognition based on novel acoustic statistical features. *IEEE Transactions on Cybernetics*, *47*(12), 4392–4404.

Cao, J., Zhang, K., Luo, M., Yin, C., & Lai, X. (2016). Extreme learning machine and adaptive sparse representation for image classification. *Neural Networks*, *81*, 91–102.

Cao, J., Zhang, K., Yong, H., Lai, X., Chen, B., & Lin, Z. (2018). Extreme learning machine with affine transformation inputs in an activation function. *IEEE Transactions on Neural Networks and Learning Systems*, http://dx.doi.org/10.1109/TNNLS.2018.2877468.

Chen, B., Wang, X., Lu, N., Wang, S., Cao, J., & Qin, J. (2018). Mixture correntropy for robust learning. *Pattern Recognition*, *79*, 318–327.

Chen, B., Xing, L., Wang, X., Qin, J., & Zheng, N. (2018). Robust learning with Kernel mean *p*-power error loss. *IEEE Transactions on Cybernetics*, *48*(7), 2101–2113.

Davis, B., & Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, *28*(4), 357–366.

Deng, W., Zheng, Q., & Chen, L. (2009). Regularized extreme learning machine. In *IEEE symposium on computational intelligence and data mining* (pp. 389–395).

Duin, R. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. *IEEE Transactions on Computers*, *25*(11), 1175–1179.

Gutoski, M., Ribeiro, M., Aquino, N., et al. (2017). A clustering-based deep autoencoder for one-class image classification. In *IEEE Latin American conference on computational intelligence* (pp. 1–6).

Hochbaum, D., & Shmoys, D. (1985). A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, *10*(2), 180–184.

Hoerl, A., & Kennard, R. (1970). Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55–67.

Hu, W., Cao, J., Lai, X., & Liu, J. (2019). Mean amplitude spectrum based epileptic state classification for seizure prediction using convolutional neural networks. *Journal of Ambient Intelligence and Humanized Computing*, http://dx.doi.org/10.1007/s12652-019-01220-6.

Huang, G., Chen, L., & Siew, C. (2006). Universal approximation using incremental feedforward networks with arbitrary input weights. *IEEE Transactions on Neural Networks*, *17*(4), 879–892.

Huang, G.-B., Zhou, H., Ding, X., & Zhang, R. (2012). Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man and Cybernetics B*, *42*(2), 513–529.

Huang, G., Zhu, Q., & Siew, C. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. In *IEEE international joint conference on neural networks, vol. 2* (pp. 985–990).

Igelnik, B., & Pao, Y. (1995). Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Transactions on Neural Networks*, *6*(6), 1320–1329.

Iosifidis, A., Mygdalis, V., Tefas, A., et al. (2017). One-class classification based on extreme learning and geometric class information. *Neural Processing Letters*, *45*, 577–592.

Jiang, M., Tseng, S., & Su, C. (2001). Two-phasee clustering process for outliers detection. *Pattern Recognition Letters*, *22*(6–7), 691–700.

Juszczak, P. (2006). *Learning to recognise: a study on one-class classification and active learning* (Ph.D. thesis), Delft University of Technology.

Juszczak, P., Tax, D., Pkalska, E., et al. (2009). Minimumspanning tree based one-class classifier. *Neurocomputing*, *72*(7-9).

Kasun, L., Zhou, H., Huang, G., et al. (2013). Representational learning with extreme learning machine for big data. *IEEE Intelligent Systems*, *28*(6), 31–34.

Khan, S., & Madden, M. (2014). One-class classification: taxonomy of study and review of techniques. *Knowledge Engineering Review*, *29*(3), 345–374.

Kim, J., Kim, J., Jang, G., & Lee, M. (2017). Fast learning method for convolutional neural networks using extreme learning machine and its application to lane detection. *Neural Networks*, *87*, 109–121.

Knorr, E., Ng, R., & Tucakov, V. (2000). Distance-based outliers: algorithms and applications. *The VLDB Journal*, *8*(3-4), 237–253.

Lanckriet, G., Ghaoui, L., Jordan, M., et al. (2003). Robust novelty detection with single-class MPM. In *Advances in neural information processing systems, vol. 15* (pp. 905–912).

Leng, Q., Qi, H., Miao, J., et al. (2015). One-class classification with extreme learning machine. *Mathematical Problems in Engineering*, *2015*.

Lu, S., & Wang, B. (2017). Ensemble one-class classification applied for anomaly detection in process control systems. In *Chinese control and decision conference* (pp. 6589–6592).

Parzen, E. (1962). On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, *33*(3), 1065–1076.

Pekalska, E., Tax, D., & Duin, R. (2003). One-class LP classifier for dissimilarity representations. *Neural Information Processing Systems*, 761–768.

Piczak, K. J. (2015). Environmental sound classification with convoltional neural networks. In *IEEE international workshop on machine learning for signal processing* (pp. 1–6).

Rabaoui, A., Davy, M., Rossignol, S., et al. (2007). Improved one-class SVM classifier for sounds classification. In *IEEE conference on advanced video and signal based surveillance* (pp. 117–122).

Raghuwanshi, B., & Shukla, S. (2018). Class-specific extreme learning machine for handling binary class imbalance problem. *Neural Networks*, *105*, 206–217.

Scholkopf, B., Platt, J., Shawe, J., et al. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, *13*(7), 1443–1471.

Steinwart, I., & Christmann, A. (2008). *Support vector machines*. New York, NY, USA: Springer.

Tang, J., Deng, C., & Huang, G. (2016). Extreme learning machine for multilayer perceptron. *IEEE Transactions on Neural Networks and Learning Systems*, *27*(4), 809–821.

Tax, D. (2001). *One-class classification* (Ph.D. thesis), Delft University of Technology.

Tax, D., & Duin, R. (2000). Data descriptions in subspaces. In *Proceedings of the international conference on pattern recognition, vol. 2* (pp. 672–675).

Tax, D., & Duin, R. (2004). Support vector data description. *Machine Learning*, *54*(1), 45–66.

Torija, A., & Ruiz, D. (2016). Automated classification of urban locations for environmental noise impact assessment on the basis of road-traffic content. *Expert Systems with Applications*, *53*, 1–13.

Wang, T., Cao, J., Lai, X., & Chen, B. (2018). Deep weighted extreme learning machine. *Cognitive Computation*, *10*(6), 890–907.

Wong, C., Vong, C., Wong, P., et al. (2018). Kernel-based multilayer extreme learning machines for representation learning. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(3), 757–762.

Yang, Y., & Jonathan Wu, Q. M. (2016). Multilayer extreme learning machine with subnetwork nodes for representation learning. *IEEE Transactions on Cybernetics*, *46*(11), 2570–2583.

Yang, Y., Wang, Y., et al. (2012). Bidirectional extreme learning machine for regression problem and its learning effectiveness. *IEEE Transactions on Neural Networks and Learning Systems*, *23*, 1498–1505.

Yang, J., Ye, F., Rong, H., & Chen, B. (2017). Recursive least mean p-power extreme learning machine. *Neural Networks*, *91*, 22–33.