7-7-2020

Parallel Multistage Wide Neural Network

Jiangbo Xi
Chang'an University, xijiangbo@chd.edu.cn

Okan K. Ersoy
Purdue University

Jianwu Fang
Chang'an University, fangjianwu@chd.edu.cn

Tianjun Wu
Chang'an University, tjwu@chd.edu.cn

Xin Wei
Xi'an Institute of Optics and Precision Mechanics of CAS, weixin@opt.cn

See next page for additional authors

Parallel Multistage Wide Neural Network

Jiangbo Xi, Okan K. Ersoy, Jianwu Fang, Tianjun Wu, Xin Wei, and Chaoying Zhao,

Abstract

Deep learning networks have achieved great success in many areas such as in large scale image processing. They usually need large computing resources and time, and process easy and hard samples inefficiently in the same way. Another undesirable problem is that the network generally needs to be retrained to learn new incoming data. Efforts have been made to reduce the computing resources and realize incremental learning by adjusting architectures, such as scalable effort classifiers, multi-grained cascade forest (gc forest), conditional deep learning (CDL), tree CNN, decision tree structure with knowledge transfer (ERDK), forest of decision trees with RBF networks and knowledge transfer (FDRK). In this paper, a parallel multistage wide neural network (PMWNN) is presented. It is composed of multiple stages to classify different parts of data. First, a wide radial basis function (WRBF) network is designed to learn features efficiently in the wide direction. It can work on both vector and image instances, and be trained fast in one epoch using subsampling and least squares (LS). Secondly, successive stages of WRBF networks are combined to make up the PMWNN. Each stage focuses on the misclassified samples of the previous stage. It can stop growing at an early stage, and a stage can be added incrementally when new training data is acquired. Finally, the stages of the PMWNN can be tested in parallel, thus speeding up the testing process. To sum up, the proposed PMWNN network has the advantages of (1) fast training, (2) optimized computing resources, (3) incremental learning, and (4) parallel testing with stages. The experimental results with the MNIST, a number of large hyperspectral remote sensing data, CVL single digits, SVHN datasets, and audio signal datasets show that the WRBF and PMWNN have the competitive accuracy compared to learning models such as stacked auto encoders, deep belief nets, SVM, MLP, LeNet-5, RBF network, recently proposed CDL, broad learning, gc forest etc. In fact, the PMWNN has often the best classification performance.

Index Terms

Multistage wide learning, Parallel testing, Incremental learning, Ensemble learning.

I. Introduction

LEARNING models are becoming very complex as in the case of deep learning [1]. These complex models usually need high dimensional big data to train, and take large computing resources and long computational times. Meanwhile, data is usually not homogeneous, meaning some samples are easy, and some are hard to be classified. Most errors happen when inputs are hard to be classified due to reasons such as imbalanced distribution of samples, abnormally acquired samples, and samples close to the decision boundaries between classes or linearly nonseparable samples. This is shown in Fig. 1. Usually, the easy and hard samples are processed in the same way, thereby causing inefficient use of computing resources. Another difficulty is that when a deep network, say, a convolutional neural network is designed with many layers, samples need to be processed through all the layers of the model, which is time consuming during testing.

Some desirable properties of a neural network to be considered are the following:

- incremental learning: the network can learn features incrementally without retraining the whole network and recalculating all the parameters when new input samples become available;
- efficient learning and testing: training the neural network fast and highly efficiently, and testing the samples in parallel stages;
- optimized architecture: adaptive structure growth to optimize the computing resources and to reduce parameters of the network;
- robustness and generalization: achieving high accuracy during testing.

A number of learning algorithms including deep learning have been developed. However, there are few works using classifiers to handle all the issues discussed above as a whole. Some early related works were proposed by Ersoy et al. [2], [3]. They designed the parallel, self-organizing, hierarchical neural networks (PSHNN), which are composed of a number of stages, in which each stage is a neural network. In most types of PSHNNs, the current stage only receives the rejected samples after a nonlinear transformation from the previous stage. These rejected samples are transformed to another space where they are classified more easily in the current stage. Sabzali and Ersoy [4] proposed an optimal adaptive multistage image transform coding for bit allocation, in which the error signal from the quantization in the previous stage is sent to the succeeding stage, and each stage corrects the errors from the previous stage. Benediktsson and Ersoy et al. [5] proposed a parallel consensual neural network (PCNN), which introduced statistical consensus theory and stage neural networks with transformed inputs. The

This work was supported in part by the National Key R&D Program of China under Grant 2018YFC1504805, in part by the National Natural Science Foundation of China under Grant 61806022, Grant 61603057, Grant 41601437, and Grant 41874005, and in part by the China Scholarship Council (CSC) under Scholarship 201404910404. *(Corresponding author: Jiangbo Xi, and Okan K. Ersoy.)*

J. Xi is with the School of Geology Engineering and Geomatics, Chang'an University, Xian 710054, China (e-mail:xijiangbo@chd.edu.cn).

Okan K. Ersoy is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA (e-mail: ersoy@purdue.edu).

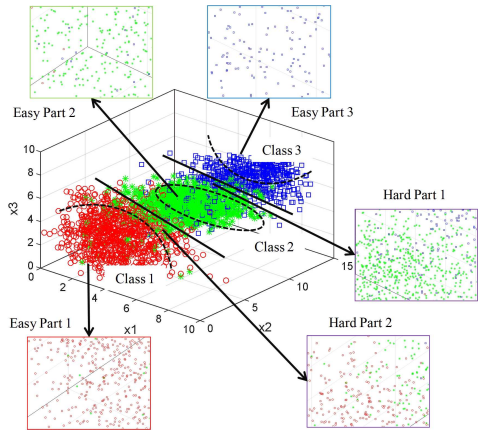

Fig. 1. Different difficulties of samples. Data usually includes easy parts and hard parts to be classified because it is not homogeneous. Easy parts are usually far away from the decision boundary, while hard parts are close to it.

outputs coming from the stage neural networks are weighted and combined using a consensual decision. Viola and Jones [6] proposed a robust real-time face detection method, which combines classifiers in a cascade. During each stage, if an image part has a particular feature pattern, it is sent to the next stage, otherwise it is rejected in the current stage.

In recent years, Venkataramani et al. [7], [8] proposed scalable-effort classifiers, which adjusted the computational effort according to the difficulty of the inputs. The method used a chain of classifiers with increasing architectural complexity of classifiers. Scalable effort was implemented by adjusting the number of stages needed to classify a sample. Panda et al. [9] proposed a conditional deep learning (CDL) method in which the convolutional layer features were used to classify the samples, and the deeper layers were activated if the input is hard to classify. Therefore, CDL can adjust the computing resources depending on the difficulty of the inputs. Zhou and Feng [10] proposed multi-Grained Cascade Forest (gcForest), a deep forest method, which is a decision tree ensemble, and can be implemented in parallel, and can even work well on small-scale training data. Wang et al. proposed stochastic configuration networks (SCNs) [11], which can be generated incrementally by stochastic configuration, and extended into stochastic configuration networks ensemble (SCNE) [12] using heterogeneous features and negative correlation learning (NCL). Another robust stochastic configuration networks (RSCNs) [13] is also proposed using kernal density estimation. Chen and Liu [14] proposed a broad learning system (BLS) to learn incrementally and effectively without a deep learning structure. Its weights can be computed using least squares, which reduces much training time. Roy and Panda et al. [15] proposed a Tree-CNN, which is a hierarchical deep neural network growing in a tree-like manner for lifelong learning. Another method using focal loss to process unbalanced data in dense object detection was proposed by Lin et al. [16]. It is observed that researchers have begun to put emphasis on minimizing the disadvantages of deep neural networks with new methods and models, especially using ensemble methods.

Ensemble learning is composed of multiple base learning models, and the final output is given by voting or stacking the results of base methods. The first approach is to construct ensemble independently, like Bagging [17], which generates a number of subsets by sampling uniformly from the original training set. Each subset is used to train a model independently. The second approach is to construct ensembles incrementally, like boosting, especially the Adaboost [18], which incrementally adds hypothesis one after another. The bagging method can be used for multi-class classification directly, but the number of basis models is not optimized. Adaboost can learn incrementally with the combination of week classifiers, but it is not used for multi-class classification directly. The recently proposed multi-column deep neural networks (MCDNN) [19] is an independent ensemble with 35 columns of deep convolutional neural networks. A decision tree structure with knowledge transfer (ERDK) [20] for high-dimensional data classification uses a decision to sort the features and uses RBF networks as the nodes based on these features. Each level contains a leaf node that assigns a class label to samples, which is determined by the RBF network at the node. The model of the forest of decision trees with RBF networks and knowledge transfer (FDRK) [21] for high-dimensional data classification is also a tree with nodes consisting of RBF networks, while hybrid feature clustering is used to construct clusters with low redundancy and highly accurate features, which are sorted by mutual information for all the clusters. These sorted features of each cluster are then used to define RBF nodes of the neural tree. Also a mix gating network using majority vote and cheat method (MMCF) is applied to aggregate the classification results. A comparison of recently proposed related works is summarized in Table S1 in the supplementary file.

On incremental learning, Kirkpatrick et al. [22] proposed elastic weight consolidation (EWC), and Lee et al. [23] proposed incremental moment matching (IMM), to overcome catastrophic forgetting in neural networks. Schwarz et al. [24] proposed a scalable framework for continuous learning. Wu et al. [25] proposed a bias correction (BiC) method to realize large scale incremental learning.

A particular neural network to reduce the training time is the radial basis function (RBF) network [26] with two layers.

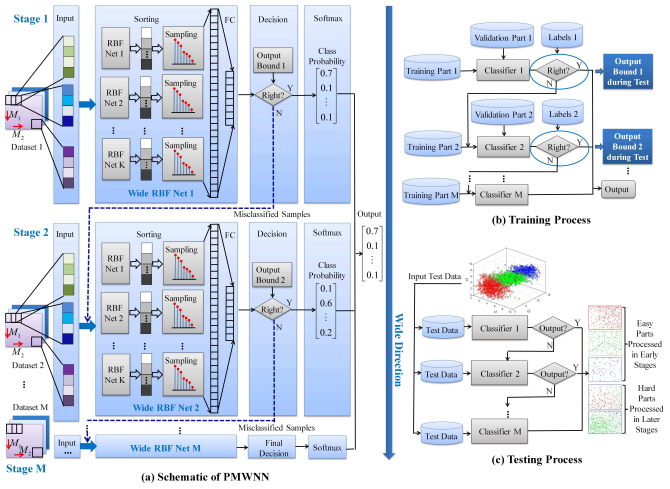

Fig. 2. The schematic of the PMWNN, training process, and testing process.

Gaussian functions in the hidden layer are used to convert input vectors to new vectors in another space. Then, the output layer combines these converted vectors as a linear combiner in the new space. The parameters, which are the weights of the linear combiner, are determined by the least squares (LS) method, which is a great advantage of the RBF network to save training time [26], [27].

This paper proposes a parallel multistage wide neural network (PMWNN), with multiple stages of wide RBF (WRBF) networks to classify different parts of data, and to be implemented in parallel during testing. The wide direction of networks is the direction along which the hidden units grow. First, a WRBF network is designed to learn features efficiently in the wide direction. The LS method and randomized singular value decomposition (randomized SVD) is used to train the WRBF network, and to compute the weights fast and efficiently. Secondly, multiple WRBF networks are combined as a multistage ensemble. Each stage focuses on a different partition of the samples, and a new stage can be added incrementally when new training data is acquired. During testing, the network stages can be implemented in parallel, thus speeding up the testing process. To sum up, the proposed PMWNN has the advantages of (1) fast training, (2) optimized computing resources, (3) incremental learning, and (4) parallel testing with stages. The structure of the proposed PMWNN is shown in Fig. 2. Its main features are as follows:

(1) it is composed of stages of well designed WRBF networks. Each WRBF is a strong classifier by itself with pretty good performance such as fast training and good testing accuracy. The WRBF uses sliding window to receive local features, and these local connections highly reduce the number of connected RBF neurons. Then, subsampling is introduced to reduce the number of linear weights.

(2) the proposed PMWNN can grow and be extended incrementally in the wide direction, depending on the characteristics of the given data and incoming new data. Meanwhile, each stage is focused on misclassified data from previous stages, and the stages can be run in parallel during testing. Although other tree-like models, such as tree CNN, ERDK, and FRDK, have similar properties, they are implemented differently.

The rest of this paper is organized as follows. Section II introduces the architecture of the PMWNN in detail. In section III, the training and testing methods, and the properties of the PMWNN are described, including incremental learning and parallel testing. Section IV compares the performance of the PMWNN in the classification of the MNIST data, a number of large hyperspectral remote sensing data, CVL single digit data, SVHN dataset, and audio signal datasets in comparison to other learning models. Section V provides the conclusions.

II. WRBF AND PMWNN

In this section, the WRBF network and the acceleration of computing the weights of WRBF are first presented. Then, the detailed architecture of PMWNN and how to choose the output bounds of PMWNN are given. Finally, the data preparation for training and validation is provided.

A. Wide RBF network

A very important property of deep convolutional networks is that they use local receptive fields to perceive features in the image. The pixels in a small area near the local receptive field usually have a strong correlation whereas the pixels far away from it usually have a weak correlation. The features in the local area are computed using sliding windows. The higher order features are abstracted in the succeeding layers. The same approach using small sliding window on the input image is to be used to design the WRBF network.

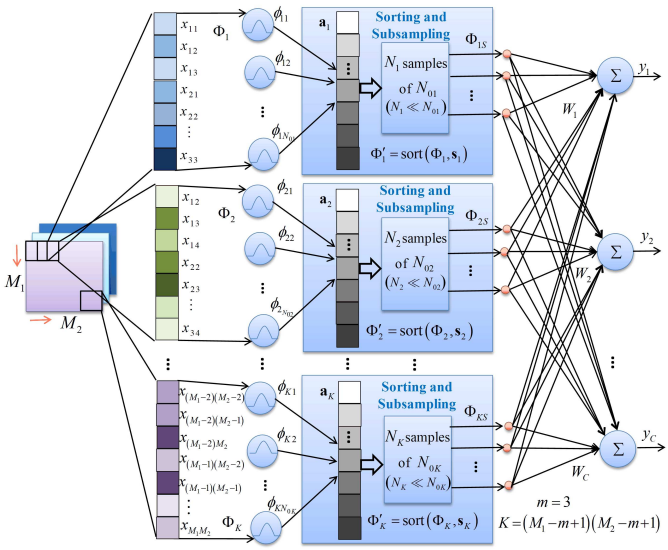

Fig. 3. The architecture of the WRBF network. The sliding window size can be of size, say, 3×3, and for each window, a RBF network is generated. Ordered output subsampling is used to reduce the number of outputs, and a WRBF network is obtained.

When the input dimension increases, the required number of fully connected hidden units of RBF network also increases, making it difficult to compute the weights with limited computational resources. In order to use the advantage of the original RBF network efficiently, two approaches are proposed. (1) Sliding windows are used to restrict the number of hidden units, and for each window, a RBF network is generated resulting in a WRBF network. (2) Ordered output subsampling method discussed below is used to further reduce the number of outputs of each RBF network. After reordering, the responses of all the Gaussian response functions are sampled in equal intervals. The resulting architecture of the WRBF network is shown in Fig.3.

Assume that the input image set is $\mathbf{X} \in \mathbf{R}^{M_1 \times M_2 \times N}$, where $M_1 \times M_2$ and N are the size (weight*height) and number of images, respectively. A sliding window is used to choose the input pixels within the image. Assume the size of the sliding window is set as $r = m \times m$. The sliding direction is from left to right, and from top to bottom. During the kth sliding time $(1 \le k \le K = (M_1 - m + 1) \times (M_2 - m + 1))$, the chosen input cube of sub-image is flattened to a matrix $\mathbf{x}_k \in \mathbf{R}^{r \times N}$, which is then fed into a RBF network characterized by N_{0k} Gaussian response functions $\{\phi_{k1}, \phi_{k2}, \cdots, \phi_{kN_{0k}}\}$. The outputs coming from Gaussian response functions of the single kth RBF are denoted as

$$\mathbf{\Phi}_k = \quad \phi_{k1} \ \mathbf{x}_k^T \ , \quad \phi_{k2} \ \mathbf{x}_k^T \ , \quad \ldots, \quad \phi_{kN_{0k}} \ \mathbf{x}_k^T \quad , \tag{1}$$

where $\phi_{ki} \ \mathbf{x}_k^T \ (1 \le i \le N_{0k})$ is a column vector with N elements.

In the end, K RBF networks are combined together. We introduce reordering and subsampling for each RBF network. The summation of $\mathbf{\Phi}_k$ along column direction is computed and sorted in descending direction as

$$\mathbf{a}_k = \mathrm{sort} \quad \sum_{col=1}^{N} \mathbf{\Phi}_k \downarrow \quad . \tag{2}$$

Suppose the sorting indices of $\mathbf{a}_k$ are $\mathbf{s}_k$. The outputs are sorted using these indices as

$$\mathbf{\Phi}_k^{'} = \mathrm{sort}\left(\mathbf{\Phi}_k, \mathbf{s}_k\right). \tag{3}$$

Then, they are subsampled, and the number of outputs after subsampling is given by

$$N_k = \frac{N_{0k}}{N_{kS}}, \tag{4}$$

where N_{kS} is the subsampling interval, and the total number of outputs of the WRBF is

$$N_S = \sum_{k=1}^{K} N_k. \tag{5}$$

The sampled outputs are denoted by

$$\mathbf{\Phi}_{kS} = \mathrm{subsample} \quad \mathbf{\Phi}_k^{'}, N_{kS} \quad . \tag{6}$$

All the vectors from all the Gaussian response functions are placed in a matrix $\mathbf{\Phi} \in \mathbf{R}^{N \times N_S}$ given by

$$\mathbf{\Phi} = \quad \mathbf{\Phi}_{1S}, \quad \mathbf{\Phi}_{2S}, \quad \cdots, \quad \mathbf{\Phi}_{KS} \quad . \tag{7}$$

The subsampled outputs of the WRBF network are combined together with linear weights

$$\mathbf{W} = \quad \mathbf{W}_1, \quad \mathbf{W}_2, \quad \ldots, \quad \mathbf{W}_C \quad . \tag{8}$$

$$\mathbf{W}_j = [w_{j1}, w_{j2}, \cdots, w_{jN_1}, w_{jN_1+1}, w_{jN_1+2}, \cdots, w_{jN_1+N_2}, \cdots, w_{jN_{K-1}+1}, w_{jN_{K-1}+2}, \cdots, w_{jN_{K-1}+N_K}]^T \quad , \tag{9}$$

where $1 \leq j \leq C$, and C is the number of classes.
The final outputs $\mathbf{Y} \in \mathbf{R}^{N \times C}$ of the WRBF network are

$$\mathbf{Y} = \mathbf{\Phi}\mathbf{W}, \tag{10}$$

$$\mathbf{Y} = \quad \mathbf{Y}_1, \quad \mathbf{Y}_2, \quad \cdots, \quad \mathbf{Y}_C \quad . \tag{11}$$

The desired outputs $\mathbf{D} \in \mathbf{R}^{N \times C}$ are

$$\mathbf{D} = \quad \mathbf{D}_1, \quad \mathbf{D}_2, \quad \cdots, \quad \mathbf{D}_C \quad . \tag{12}$$

The least squares estimation of $\hat{\mathbf{W}}$ is obtained by

$$\hat{\mathbf{W}} = \arg \min_{\mathbf{W}} \|\mathbf{\Phi}\mathbf{W} - \mathbf{D}\|^2 . \tag{13}$$

The pseudoinverse $\mathbf{\Phi}^+$ of $\mathbf{\Phi}$ is used to calculate $\hat{\mathbf{W}}$ [4]:

$$\hat{\mathbf{W}} = \mathbf{\Phi}^+\mathbf{D} = \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{D}. \tag{14}$$

The outputs of the WRBF network are

$$\mathbf{Y} = \mathbf{\Phi}\mathbf{\Phi}^+\mathbf{D} = \mathbf{\Phi}\left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{D}. \tag{15}$$

To control the risk of over-fitting of the WRBF network, an adaptive regularization [28] can be used with an over-fitting parameter $r_{over\_fitting}$. After training of the WRBF network using LS method, a validation set $\mathbf{X}_v$ can be used to compute the performance. Suppose the mean square error of training and validation data are E_{tr} and E_v, respectively. The over-fitting parameter is defined as

$$r_{over\_fitting} = \frac{E_v}{E_{tr}}. \tag{16}$$

(1) Overfitting: When $r_{over\_fitting}$ is large and E_{tr} is small, E_v is larger than E_{tr}. That means the WRBF is over-fitting. Therefore, the sliding window size can be increased to reduce the generated RBF networks, and the number of hidden neurons of the composed RBF networks can be reduced. The subsampling number can also be reduced. The complexity of the model can be reduced.

(2) Underfitting: When $r_{over\_fitting}$ is small and close to 1, E_{tr} and E_v are close to each other. If E_{tr} and E_v are both small, that means the WRBF learns training data well without overfitting. If both of them are big, that means the WRBF does not have enough complexity to learn from the current dataset. Therefore, the sliding window size can be reduced to increase the number of generated RBF networks. The number of hidden neurons and the number of subsampling can be increased to learn more features. The complexity of the model can be increased.

This subsection further describes the WRBF network, which is referenced on line 2 in Algorithm 2, and line 3 in Algorithm 3.

B. Acceleration of Learning in the Wide RBF Network

The singular value decomposition (SVD) is an effective method to find the pseudoinverse $\mathbf{\Phi}^+$ of the matrix $\mathbf{\Phi}$, and thereby to compute the weights $\hat{\mathbf{W}}$ of the WRBF network. For small datasets with low or moderate dimensions, the weights computing process of WRBF using SVD is quick enough and acceptable. When the dataset has a large number of images, the sliding window is used and a large number of RBFs are generated to create WRBF. Although sorting and subsampling is introduced to reduce the number of weights, it is still a big number. When there are a big number of training samples, it is still time consuming to compute the weights although SVD is used. The randomized SVD proposed by Halko et al. [29]–[31] can accelerate the SVD decomposition process of the matrix $\mathbf{\Phi}$, and thus accelerate the computation of the weights of the WRBF network. A low-rank approximation of a matrix $\mathbf{\Phi} \in \mathbf{R}^{K_1 \times K_2}$ using the truncated SVD is expressed as

$$\mathbf{\Phi} \approx \mathbf{\Psi}_{K_1 \times l}\mathbf{A}_{l \times K_2}, \tag{17}$$

where the dimension l is the numerical rank of the matrix $\mathbf{\Phi}$. For the WRBF, $K_1 = N$ and $K_2 = N_S$.

Algorithm 1 Randomized Singular Value Decomposition (Randomized SVD)

Input: Matrix $\mathbf{\Phi} \in \mathbf{R}^{K_1 \times K_2}$, desired rank l, oversampling parameter p, exponent q.
Output: $\mathbf{U}$, $\mathbf{\Sigma}$, $\mathbf{V}$.

1: Generate Gaussian random distribution matrix $\mathbf{\Omega}_{K_2 \times (l+p)}$;
2: Calculate the matrix $\mathbf{Y} = (\mathbf{\Phi}\mathbf{\Phi}^T)^q \mathbf{\Phi}\mathbf{\Omega}$ ($q = 1$ or $q = 2$);
3: Generate a matrix $\mathbf{Q}$ with orthonormal basis as columns coming from the matrix $\mathbf{Y}$;
4: Compute $\mathbf{\Psi} = \mathbf{Q}^T\mathbf{\Phi}$;
5: Compute a SVD of the new small matrix: $\mathbf{\Psi} = \widetilde{\mathbf{U}}\mathbf{\Sigma}\mathbf{V}^T$;
6: Compute $\mathbf{\Phi} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U} = \mathbf{Q}\widetilde{\mathbf{U}}$.

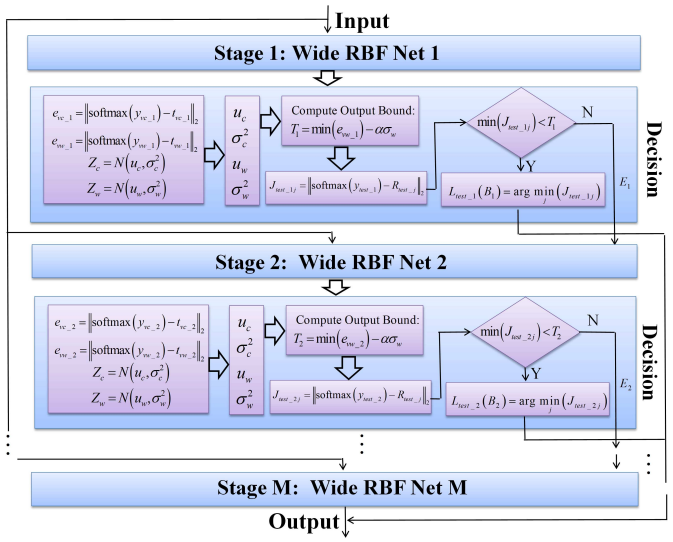

Fig. 4. The architecture of the PMWNN.

If l is much smaller than K_1 or K_2, but good enough to approximate $\mathbf{\Phi}$, $\mathbf{\Phi}$ is much easier to be stored in memory and to be computed rapidly. The randomized SVD includes two stages. In the first stage, a matrix $\mathbf{Q} \in \mathbf{R}^{K_1 \times l}$ with as few orthonormal columns as possible is required, satisfying

$$\mathbf{\Phi} \approx \mathbf{Q}\mathbf{Q}^T\mathbf{\Phi}. \tag{18}$$

Consequently, a low-dimensional subspace is constructed. In the second stage, the matrix $\mathbf{Q}$ is used to obtain the decomposition $\mathbf{\Phi} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ by computing matrices $\mathbf{U}$ and $\mathbf{V}$ with orthonormal columns, and $\mathbf{\Sigma}$, which is a nonnegative, diagonal matrix. The steps are as follows:

(1) Use $\mathbf{\Psi} = \mathbf{Q}^T\mathbf{\Phi}$ as the projection matrix, and compute a SVD of the new small matrix: $\mathbf{\Psi} = \widetilde{\mathbf{U}}\mathbf{\Sigma}\mathbf{V}^T$.
(2) Compute $\mathbf{\Phi} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U} = \mathbf{Q}\widetilde{\mathbf{U}}$.

The method is summarized in Algorithm 1. In this way, the computation of the weights can be accelerated, and the memory required for computations can be saved.

C. Parallel Multistage Wide Neural Network

The WRBF network can be used as a basic classifier to design learning models. In many applications, a large portion of input vectors are easy to be classified, whereas a small portion of input vectors are difficult to be classified. Inspired by PSHNN [2], scalable-effort classifier ensembles [7], and Adaboost [18], multiple stages of WRBF networks are combined together as basic classifiers in PMWNN to specialize on different parts of features at each stage. The detailed structure of the proposed network is shown in Fig. 4. The stages in the figure are shown growing from left to right (the deep direction) to conserve space. It is actually growing in the vertical direction (the wide direction) as in Fig. 2, since the input of each stage consists of subsets of original input vectors and their associated data sets from the previous stage.

Suppose M WRBF networks are adopted, a. k. a., Net_1, Net_2,..., Net_M. The first WRBF network Net_1 is used to classify the first training dataset. After training, there are some training samples which are misclassified. These training samples are sent to the next stage WRBF network after being augmented as described later. Hence, the next stage focuses on the misclassified samples of the previous WRBF network. When the training accuracy is high, there may not be enough misclassified training samples to train the WRBF network at the next stage. The first validation dataset at the current stage (separated from the

training set or generated from the original training set using augmentation) is used to perform validation at stage 1. After validation, the misclassified validation samples together with misclassified training samples are sent to the next stage.

The reference matrix discussed below is used to calculate the output. Suppose there are C classes and N_{tr} training samples at stage n_{stg}, and the training and validation set at each stage has the same size. The reference matrices for training and validation at each stage are $\mathbf{R}_j (1 \leq j \leq C)$, in which the values of all elements in the jth row are equal to 1, and 0 for the other elements. There are C reference matrices, and each has the size of $C \times N_{tr}$.

The class of an input sample is determined by the row number of minimum error of outputs and corresponding label. The succeeding stages are set up using the same method.

There are still two key problems. The first one is how to get the output bounds or thresholds for acceptance/rejection during training and how to decide the number of stages. This will be discussed in Section II-D. The second one is how to add a new WRBF network as a new stage if there are new incoming samples and the network needs to be extended. This is discussed in Section III-B-1).

The detailed implementation during training and testing is in Algorithms 2 and 3.

D. Choosing Thresholds at Each Stage

The difficult part of the proposed network is how to determine the decision threshold at each stage during training to decide which stage is chosen for each sample during testing. After training and validation, the correctly classified and misclassified validation samples are detected by comparing the outputs with their labels, as shown in detail in Section III-A. Suppose the correctly classified and misclassified validation outputs at stage n_{stg} are $\mathbf{y}_{vc\_n_{stg}} \in \mathbf{R}^{N_{vc} \times C}$ and $\mathbf{y}_{vw\_n_{stg}} \in \mathbf{R}^{N_{vw} \times C}$. The number of the two types of samples are $N_{vc\_n_{stg}}$ and $N_{vw\_n_{stg}}$, and $N_{vc\_n_{stg}} + N_{vw\_n_{stg}} = N_{val}$. The two types of errors are computed by

$$\mathbf{e}_{vc\_n_{stg}} = \text{ softmax } \mathbf{y}_{\text{vc\_n}_{\text{stg}}} - \mathbf{t}_{\text{vc\_n}_{\text{stg}}} {}_2 , \tag{19}$$

$$\mathbf{e}_{vw\_n_{stg}} = \text{ softmax } \mathbf{y}_{\text{vw\_n}_{\text{stg}}} - \mathbf{t}_{\text{vw\_n}_{\text{stg}}} {}_2 , \tag{20}$$

where $\mathbf{t}_{vc\_n_{stg}}$ and $\mathbf{t}_{vw\_n_{stg}}$ are the true labels of correct and misclassified validation samples at stage n_{stg}, respectively. The statistical parameters means μ_c, μ_w, and standard deviations σ_c, σ_w of errors $\mathbf{e}_{vc\_n_{stg}}$ $\mathbf{e}_{vw_n_{stg}}$are estimated, and using these parameters, two Gaussian distributions are generated as

$$Z_c \sim N \ \mu_c, \sigma_c^2 \ , \tag{21}$$

$$Z_w \sim N \ \mu_w, \sigma_w^2 \ . \tag{22}$$

The two corresponding Gaussian probability density functions (pdf) are

$$f_c(z) = \frac{1}{\sqrt{2\pi}\sigma_c} \exp \ \frac{(z-\mu_c)^2}{2\sigma_c^2} \ , \tag{23}$$

$$f_w(z) = \frac{1}{\sqrt{2\pi}\sigma_w} \exp \ \frac{(z-\mu_w)^2}{2\sigma_w^2} \ . \tag{24}$$

The validation errors of both correctly classified and misclassified samples, and their Gaussian pdfs are shown in Fig. 5. The decision threshold is computed by

$$T_{n_{stg}} = \min \ \mathbf{e}_{vw\_n_{stg}} - \alpha\sigma_w, \tag{25}$$

where α is the constant deciding the margin level to assure all misclassified validation samples are rejected at current stage.

Choosing the proper constant α depends on two factors. First, the distribution of errors of correctly classified validation samples described by μ_c and σ_c. Secondly, the number of stages of PMWNN. If there are adequate number of stages, α can be large to assure that each stage only outputs the correctly classified testing samples. If the number of stages is limited to meet a required accuracy, a balance needs to be considered between all the stages, and smaller α can be chosen for each stage. The interval for α can be 3 to 5, to make sure a small proportion of misclassified samples cannot pass the current stage. The computing method of decision threshold, which is referenced on lines 9 to 11 in Algorithm 2.

The proposed network can grow adaptively according to the training and validation process. The WRBF network at each stage is trained independently. When the training and validation accuracy is high enough, the growing process is stopped, and the final structure of the PMWNN is established. Assuming the number of misclassified samples at stage n_{stg} is $N_{w\_n_{stg}}$, the growing process stops when

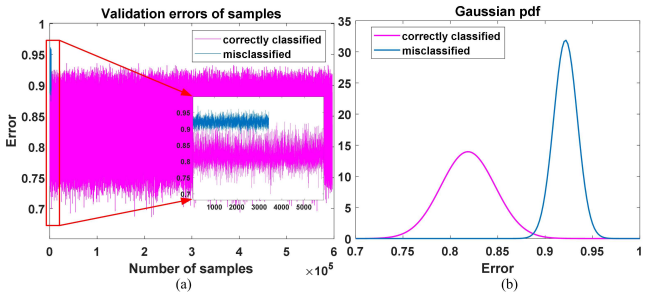

Fig. 5. The errors of validation samples and the Gaussian pdfs validation errors, including both correctly classified and misclassified samples. The validation samples are from the augmented training set. (a) The errors of validation samples. (b) The Gaussian pdfs of validation errors.

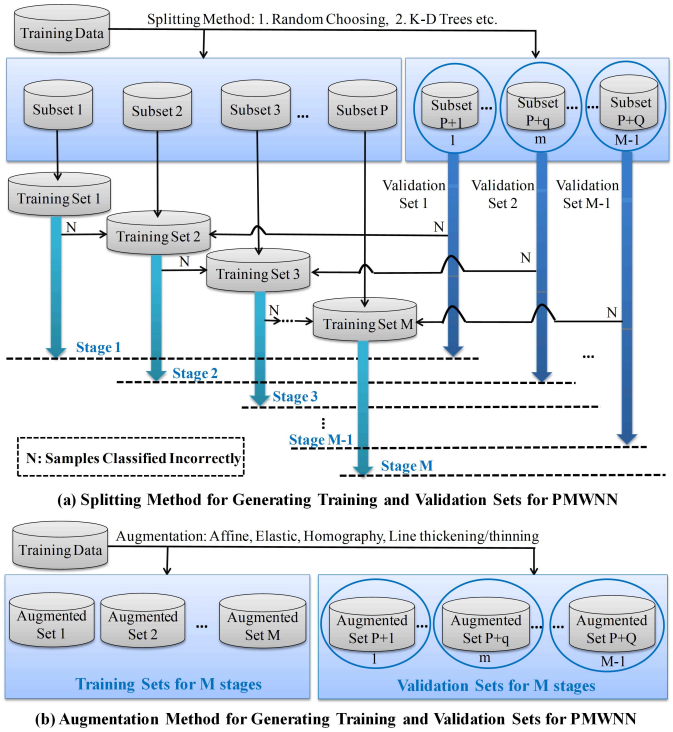

Fig. 6. The training and validation sets for PMWNN. (a) Splitting method. (b) Augmentation method.

$$N_{w\_n_{stg}} \leq T_v, \tag{26}$$

where T_v is chosen according to the properties of the dataset and learning tasks.

E. Data separation and Data Augmentation

For PMWNN, there maybe many stages composed of WRBF networks. Each stage needs a training set and a validation set, and each stage puts more emphasis on the misclassified samples from the previous stage.

If the training set is big and sufficient, it can be split into small subsets as training sets and validation sets for all the stages of the PMWNN network. One method is to get these subsets randomly. There are also other methods such as using k-d tree [32]. The splitting method is shown in Fig. 6 (a).

For small and insufficient training dataset. The data augmentation method can be used on the whole training set to generate the same number of training and validation sets as in the situation of big training set. Four types of augmentations can be used to generate training, validation, and testing samples [33], [34] as shown in Fig. 6 (b). These augmentation methods for images are: (1) Affine, (2) Elastic, (3) Homography: (4) Line thickening/thinning. The morphological image processing technology including erosion and dilation can be used to deform the image.

The detailed description of this subsection are also presented in Section S-III in the supplementary file. In the next sections, augmented data sets are assumed.

Algorithm 2 Training Procedure for PMWNN

Input: Original training dataset: $\mathbf{x}_{aug\_1} = \mathbf{x}_{tr\_ori}$; $N_{stg} - 1$ augmented training datasets: $\mathbf{x}_{aug\_2}$, $\mathbf{x}_{aug\_3}$, ... , $\mathbf{x}_{aug\_N_{stg}}$; $N_{stg} - 1$ augmented validation datasets $\mathbf{x}_{v\_1}$, $\mathbf{x}_{v\_2}$, ..., $\mathbf{x}_{v\_N_{stg}-1}$, original PMWNN Net_{PMW}; and N_{stg} WRBF network: Net_1, Net_2, ..., $Net_{N_{stg}}$.

Output: Trained PMWNN Net_{PMM}, and decision boundaries of each stage.

for currentStage $i = 1 : N_{stg}$ **do**
  Train Net_i using x_{tr_i}, sorting and subsampling is implemented in the WRBF network at this stage. Do validation using $\mathbf{x}_{v\_i}$. These samples are predicted directly with the sorted and subsampled outputs, and the trained weights of the WRBF network at this stage, the positions of which are the same as in training;
  for class $j = 1 : C$ **do**
    Calculate L2 norm of differences between validation outputs $\mathbf{y}_{v\_i}$ and reference matrix $\mathbf{R}_j$:
    $\mathbf{J}_{v\_ij} = \left\| \mathrm{softmax}\left(\mathbf{y}_{v\_i}\right) - \mathbf{R}_j \right\|_2$;
  end for
  Determine the class labels of validation samples:
  $\mathbf{y}_{v\_ind\_i} = \arg\ \min_j \left(\mathbf{J}_{v\_ij}\right), 1 \leq\ j \leq C$;
  Calculate mean μ_c and μ_w, standard deviation σ_c, σ_w of errors of correctly classified and misclassified validation samples, respectively;
  Use the above statistic parameters to compute the decision threshold:
  $T_i = \min\left(\mathbf{e}_{vw\_i}\right) - \alpha\delta_w$;
  if $i < N_{stg}$ **then**
    Prepare training data x_{tr_i+1} for the next stage:
    Generate γ_{tr_stg} times and γ_{v_stg} times augmented training samples using misclassified samples in x_{tr_i} and x_{v_i} , respectively;
    Generate supplementary training samples using original training samples in $\mathbf{x}_{tr\_ori}$.
  end if
end for

III. Training and Testing Methodology, and the properties of PMWNN

A. Training and Testing Methodology

In this subsection, we describe the detailed method for training and testing the PMWNN.

1) Training PMWNN: It is shown in Algorithm 2. Assume the PMWNN has N_{stg} stages. The training procedure starts with original training dataset $\mathbf{x}_{tr\_ori}$, and augmented training datasets $\mathbf{x}_{aug\_2}$, $\mathbf{x}_{aug\_3}$, ..., $\mathbf{x}_{aug\_N_{stg}}$, augmented validation datasets $\mathbf{x}_{v\_1}$, $\mathbf{x}_{v\_2}$, ..., $\mathbf{x}_{v\_N_{stg}}$, original network Net_{PMW} and number of classes C as inputs. It generates the trained network Net_{PMW} and decision boundaries at each stage.

The first stage Net_1 of Net_{PMW} is trained using original training data $\mathbf{x}_{tr\_ori}$, which is also denoted as $\mathbf{x}_{aug\_1}$. During this process, sorting and subsampling is implemented in WRBF network at this stage, and the LS method is used to compute the weights of the first stage, which is described in detail in Section II-A).

Then, the augmented validation data $\mathbf{x}_{v\_1}$ is used to do validation. The validation samples are predicted directly with the sorted and subsampled outputs, and the trained weights of WRBF network at this stage, the position of which are the same as in training. After that, the L2 norm of differences between validation outputs and $\mathbf{R}_j$ are calculated as

$$\mathbf{J}_{v\_1j} = \left\| \mathrm{softmax}\left(\mathbf{y}_{v\_1}\right) - \mathbf{R}_j \right\|_2, 1 \leq\ j \leq C, \tag{27}$$

where the size of the validation outputs $\mathbf{y}_{v\_1}$ is $C \times N_{val}$.

Then, the labels of the validation data are determined by

$$\mathbf{y}_{v\_ind\_1} = \arg\ \min_j \left(\mathbf{J}_{v\_1j}\right), 1 \leq\ j \leq C, \tag{28}$$

where $\mathbf{y}_{v\_ind\_1}$ is a vector with the size $1 \times N_{val}$.

After that, the mean values and the standard deviations of errors of correctly classified and misclassified validation samples are calculated. The decision boundary T_1 of the first stage is determined by using Eq.(25). The following stages are trained similarly until reaching the final stage.

The training data for the next stage is composed of three parts:

(1) deformed samples of N_{tr_mis} misclassified training samples in the current stage increased by a factor γ_{tr_1};

(2) deformed samples of N_{v_mis} misclassified validation samples in the current stage increased by a factor γ_{v_1};

(3) supplementary training samples randomly chosen from the original training data. The number of chosen samples is N_{tr_sump}.

Algorithm 3 Testing Procedure for PMWNN

Input: Testing dataset $\mathbf{x}_{test}$, trained N_{stg} stages PMWNN Net_{PMW}, decision thresholds T_{stg_1}, T_{stg_1}, ..., $T_{stg\_M_{stg}}$.
Output: Class labels $\mathbf{L}_{test}$.

for currentStage $i = 1 : N_{stg}$ **do**
  Test ith stage of Net_{PMM} with $\mathbf{x}_{test}$. These samples are predicted directly with the sorted and subsampled outputs, and the trained weights of the WRBF network at corresponding stages, the positions of which are the same as in training. Obtain the outputs $\mathbf{y}_{test\_i}$;
  for class $j = 1 : C$ **do**
    Calculate L2 norm of differences between output $\mathbf{y}_{test\_i}$ and reference matrix:
    $\mathbf{J}_{test\_j} = \|\mathrm{softmax}\left(\mathbf{y}_{\mathrm{test\_i}}\right) - \mathbf{R}_{\mathrm{test\_j}}\|_2$;
  end for
  if $i < N_{stg}$ **then**
    if $\min\limits_{j\in(1,C)} \left(\mathbf{J}_{test\_ij}\right)\ of those the samples < T_i$ **then**
      Calculate the class labels of these samples:
      $\mathbf{L}_{test\_i}\left(\mathbf{B}_i\right) = \arg\ \min\limits_{j} \left(\mathbf{J}_{test\_ij}\right), 1 \le\ j \le C$;
      Output these samples at current stage using the output binary mask $\mathbf{B}_i$;
    end if
  else
    Calculate the class labels of remaining samples:
    $\mathbf{L}_{test\_Nstg} = \arg\ \min\limits_{j} \left(\mathbf{J}_{test\_ij}\right), 1 \le j \le C$;
  end if
end for
$\mathbf{L}_{test} = \mathbf{L}_{test\_1} + \mathbf{L}_{test\_2} + \cdots + \mathbf{L}_{test\_N_{stg}}$.

To ensure that population size does not change at the next stage, we have

$$\gamma_{tr\_1} \times N_{tr\_mis} + \gamma_{v\_1} \times N_{v\_mis} + N_{tr\_sump} = N_{tr}. \tag{29}$$

The detailed discussion of choosing γ is given in Section S-IV in supplementary file.

2) Testing PMWNN: The testing procedure is shown in Algorithm 3. The testing data $\mathbf{x}_{test}$, the network Net_{PMW}, and decision thresholds T_1, T_2, ..., $T_{N_{stg}}$ are the inputs of the testing procedure. The class labels $\mathbf{L}_{test}$ of testing samples are also provided as outputs.

These testing data are sent to all the stages at the same time. After all the instances are tested, the decision thresholds obtained during training are used to determine the class labels and which samples are classified at which stage. During this process, a binary mask is generated in cascade stage by stage and used for the above purpose. The binary mask at the first stage is defined as

$$\mathbf{B}_1\left(i\right) = \begin{cases} 1, & \min\limits_{j\in(1,C)} \left(\mathbf{J}_{test\_1j}\left(i\right)\right) \le T_1 \\ 0, & \min\limits_{j\in(1,C)} \left(\mathbf{J}_{test\_1j}\left(i\right)\right) > T_1 \end{cases}, \tag{30}$$

where, $1 \le i \le N_{test\_samples}$, and $N_{test\_samples}$ is the number of test samples.

An exclusive binary mask is obtained by

$$\mathbf{E}_1 = 1 - \mathbf{B}_1. \tag{31}$$

For the second stage, a temporary binary mask $\mathbf{B}_{2temp}$ is computed by using T_2. A temporary exclusive binary mask $\mathbf{E}_{2temp}$ is computed similarly as in stage 1.

Then, the binary mask for the second stage is computed by

$$\mathbf{B}_2 = \mathbf{B}_{2temp}\mathbf{E}_1. \tag{32}$$

The exclusive binary mask $\mathbf{E}_2$ is obtained by

$$\mathbf{E}_2 = \mathbf{E}_1\mathbf{E}_{2temp}. \tag{33}$$

The succeeding binary masks for each stage are computed recursively in the same way. If no classifier is activated at stages from 1 to $M-1$, the output is given at stage M.

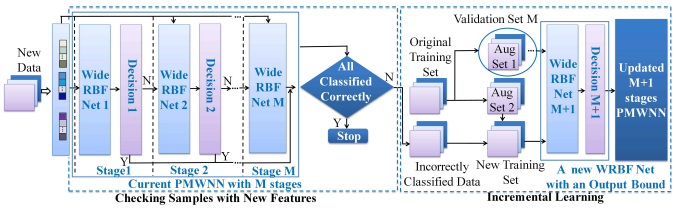

Fig. 7. Incremental learning of PMWNN with new incoming data.

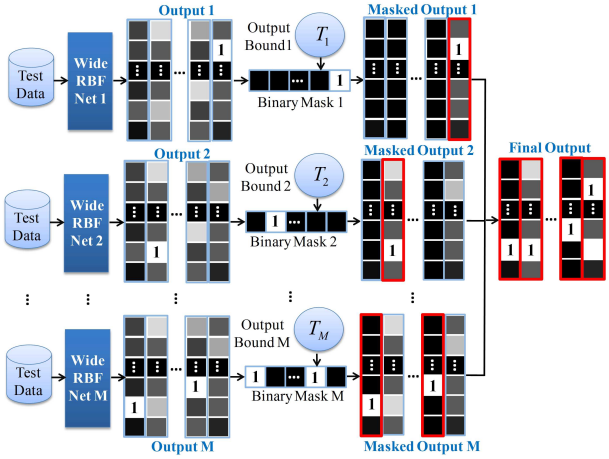

Fig. 8. The parallel structure of PMWNN during testing. Testing data are given to all the stages in parallel. The output bound at each stage is used to generate a binary mask to decide which stage gives the output.

B. *Properties of PMWNN*

1) Incremental Property during Learning: When there are new incoming data, we can add another new WRBF network to learn new features without retraining the entire PMWNN, meaning the network can learn new knowledge without forgetting the old knowledge. The incremental learning process is shown in Fig. 7. The new input training data is sent to the current PMWNN network with M stages. If there are misclassified samples, they are used to set up a new training dataset together with the new augmented original training dataset. Meanwhile, a new validation dataset is also generated to obtain the output bound. Then, a new WRBF network is trained and added to the current PMWNN as the $M+1th$ stage.

2) Fast Learning Properties and Testing Time Analysis: The proposed PMWNN is composed of stages of WRBF networks, which uses local sliding window, subsampling, and LS method to train it in one epoch. Therefore, the training process of the WRBF network is fast. The testing time mainly depends on the testing time of single WRBF network.

The comparative computation times of PMWNN and CNN in terms of number of multiplications and the time analysis of testing are given in Section S-V of the supplementary file.

3) Parallel Property during Testing: When the structure of the proposed method is determined and the training process is finished, the stages can be tested in parallel. This can be seen in Fig. 8. All the testing samples are sent to all the WRBF networks. The decision threshold decides which WRBF network is assigned to the output. This process does not need to wait for the classification results from the other stages. Therefore, during testing, the WRBF networks give outputs in parallel with high efficiency.

IV. Experiments

A. *Experiment Setup and Model Fine Tuning*

The experiments were implemented using a Dell Work Station with Intel-i7-8700K CPU @ 3.7GHz, 32G memory. The influence of different parameters of WRBF on MNIST dataset is given in S-VI of the supplementary file. Other than that, the proposed models are fine tuned using validation sets together with random search of parameters including the sliding window scale, the numbers of hidden units and subsampling outputs.

The validation set for the MNIST data is the augmented training dataset using elastic deformation. The validation set for the hyperspectral remote sensing data set is 5% of the total labeled pixels. (13% for Indian Pines because of the imbalance of training classes). For CVL, SVHN, and audio signal datasets, the validation set is 10% of randomly chosen training samples.

The fine tuning of compared models on different datasets is mainly based on the validation set and grid search. For MLP, RBF, and SAE, the hyper parameter to be fine tuned is the number of hidden units. For CNN, they are the size and the number

TABLE I
TESTING RESULTS OF 3 STAGES PMWNN AT EACH STAGE ON MNIST DATASET WITH TRAINING AND TESTING DATA AUGMENTATION (50 SUBSAMPLES)

Stage	1000 hidden units		300 hidden units	
	Acc. (%)	Output NO.	Acc. (%)	Output NO.
1	99.73	9599	100	8291
2	90.65	214	97.59	1412
3	68.54	187	73.4	297
Total Acc. (%)	98.95		98.87	

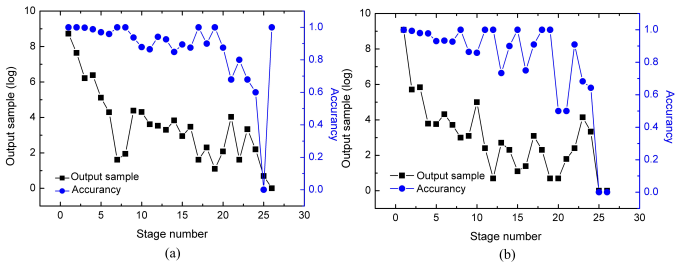

Fig. 9. Changes of number of output samples and accuracy during testing. (a)Number of hidden units=1000 (b) Number of hidden units=300

of convolutional kernels, and the number of convolutional layers, which are fine tuned manually. The related hyper parameters of RBFE and CNNE are chosen the same as the single RBF and CNN, respectively.

B. MNIST Data

The MNIST data [35] includes 60 000 training samples and 10 000 testing samples. Each sample is a gray-scale image with size 28×28.The elastic deformation is used to implement data augmentation. Simard et al. [33] proved that the best accuracy can be achieved using elastic deformation instead of affine deformation.The parameters of the WRBF network can be considered in two groups. One is the parameters including the size of the sliding window m, the number of hidden units N_{0k}, and the number of sampling outputs N_{kS}. The other is the parameters of the RBF kernel including the centers $\mathbf{c}_i$ and the standard deviation σ_i. The centers of the basis function are randomly chosen from the training dataset in the current sliding window, and the number of centers is the number of chosen centers of the hidden units of the RBF network. We chose $N_{0k} = 1000$, $N_{kS} = 50$, 13×13 as the window size, and 1×10 as the widths of basis functions.

1) Classification Performance of PMWNN with Augmented Dataset: The augmented dataset was generated for each training stage. In the experiments, both 3 stages and 26 stages were used to get enough accuracy. 250 additional augmented datasets were generated using elastic deformation. The displacement scale factor was $\eta = 34$, and the standard deviation of the convolved Gaussian function was $\sigma = 4$ [33]. The proposed network was trained stage by stage, and 10 augmented datasets were used as validation sets at each stage. After training,the testing dataset was used at each stage.

The experiments were performed using PMWNNs with 1000 and 300 hidden units at each stage, and 3 and 26 stages, respectively. The testing results are shown in Tables I and II. The last 2 results are also observed in Fig. 9.

In Table I, the testing accuracy and the number of accepted testing samples with 3 stage are given at each stage. The testing accuracies were improved to 98.95% and 98.87% with 1000 and 300 hidden units, respectively, compared to 98.76% of the WRBF with the same parameters in Table S3 of the supplementary file.

In Table II, the testing accuracy and the number of accepted testing samples with 26 stages are given at each stage. We see that the best total accuracy of the PMWNN was 99.12%. The testing accuracy is pretty high for the first 10 stages, then it changes a lot and is not very high in final stages. That is because most of the testing samples are output at previous stages, and there are not many testing samples left at later stages. The number of output samples and testing accuracy at each stage are determined by the output threshold of that stage.

The PMWNN was compared mainly with two kinds of models: one is the classical learning models with equivalent complexity, and another is recent learning models focused on some good properties such as incremental, or newly developed models without using CNN as the base architectures. These models are stacked auto encoders [36], deep belief net [37], deep Boltzmann machines [38], another stacked autoencoder [39], a standard random forest [10], SVM [10], multilayer perceptron [40], LeNet-5 [10], an implemented RBF network with 10000 hidden units and a RBF ensemble, an implemented CNN (6c-2s-12c-2s) and a CNN ensemble, recently proposed conditional deep learning [9] , broad learning system (BLS) [14], multi-Grained Cascade Forest [10], Elastic weight consolidation (EWC) [22], and Incremental moment matching (IMM) [23]. The results are shown in Table III. We see that the proposed PMWNN has the second best testing accuracy among the compared learning models in the experiment, and even the WRBF network (2000 hidden units) has the accuracy 98.80%. Actually, the testing accuracy of the PMWNN could be made higher than 99.12% by adding stages with more computing resources. Recently related models with PMWNN also includes scalable-effort classifiers, tree-CNN, ERDK, and FDRK as

TABLE II
TESTING RESULTS OF 26 STAGES PMWNN AT EACH STAGE ON MNIST DATASET WITH TRAINING DATA AUGMENTATION (50 SUBSAMPLES)

Stage	1000 hidden units		300 hidden units	
	Acc. (%)	Output NO.	Acc. (%)	Output NO.
1	100.00	6140	99.94	8773
2	100.00	2079	99.34	302
3	99.80	499	97.95	342
4	98.81	590	97.73	44
5	96.99	166	93.02	43
6	95.89	73	93.33	75
7	100.00	5	92.68	41
8	100.00	7	100.00	20
9	93.75	80	86.36	22
10	87.84	74	85.81	148
11	86.48	37	100.00	11
12	94.12	34	100.00	2
13	92.59	27	73.33	15
14	84.87	46	90.00	10
15	89.47	19	100.00	3
16	87.50	32	75.00	4
17	100.00	5	90.91	22
18	90.00	10	100.00	10
19	100.00	3	100.00	2
20	87.50	8	50.00	2
21	67.86	56	50.00	6
22	80.00	5	90.91	11
23	67.86	28	68.25	63
24	60.00	9	64.29	28
25	0.00	2	0.00	1
26	100.00	1	0.00	0
Total Acc. (%)	**99.12**		**99.06**	

TABLE III
TESTING ACCURACIES OF DIFFERENT LEARNING MODELS ON MNIST DATASET

Method	Accuracy (%)
Stacked auto encoders [36]	98.6
Deep belief nets [37]	98.87
Deep boltzmann machines [38]	99.05
stacked autoencoder [39]	98.72
Random forest [10]	96.8
SVM (rbf kernel) [10]	98.6
Multilayer perceptron [40]	97.39
LeNet-5 [10]	99.05
Elastic weight consolidation (EWC) [22]	98.2
Incremental moment matching (IMM) [23]	98.30
Conditional deep learning [9]	98.92
Broad learning system [14]	98.74
gc Forest [10]	99.26
CNN (6c-2s-12c-2s)	98.63
RBF network (10 000 hidden units)	97.41
RBF ensemble (10 000 hidden units, 5 RBF Nets)	97.50
CNN ensemble (6c-2s-12c-2s, 5 CNNs)	98.81
WRBF network	**98.80**
PMWNN	**99.12**

described in the introduction section. These models show excellent performance on other different kinds of datasets with several similar properties to PMWNN, but being implemented in different ways.

C. Hyperspectral Remote Sensing Data

Three hyperspectral datasets [41] are used to test the performance. If all the spectral and spatial information are used, the computation load is demanding. The datsasets are:

(1) Pavia Center: It was acquired over Pavia, in northern Italy. The number of bands is 102, and the size of the image after discarding the pixels without information is 1096×715 for each band. There are 9 classes in the image ground truth.

(2) Pavia University: It was also acquired over Pavia. The number of bands is 102. It has size 610×340 for each band after discarding the pixels without information. There are 9 classes in the image ground truth.

(3) KSC (Kennedy Space Center): It has 176 bands after removing water absorption and low SNR bands. The size of image is 512×614 for each band, and there are 13 classes.

TABLE IV
TESTING ACCURACIES OF DIFFERENT LEARNING MODELS ON DIFFERENT HYPERSPECTRAL DATASETS

Method	Pavia Center/ (%)		Pavia Univ. / (%)		KSC/ (%)		Indiana Pines/ (%)		Salinas/ (%)	
	OA	AA	OA	AA	OA	AA	OA	AA	OA	AA
MLP	98.66	94.60	91.06	89.30	91.67	86.59	81.15	74.57	91.31	95.10
RBF	97.96	91.06	90.57	83.72	94.03	90.55	75.35	60.04	91.75	95.74
SAE	92.80	76.30	83.43	72.32	92.36	87.79	92.26	91.97	93.19	96.15
CNN	99.06	96.47	91.53	88.76	95.10	91.49	81.08	78.61	92.14	94.95
RBFE	98.00	91.15	91.62	85.92	94.34	91.08	76.55	62.25	95.34	97.34
CNNE	99.19	97.07	92.17	89.52	96.00	93.75	83.12	82.00	92.52	95.95
WRBF	**98.71**	**96.53**	**93.87**	**89.79**	**96.03**	**92.52**	**96.04**	**92.19**	**99.03**	**99.43**
PMWNN	**99.20**	**97.34**	**95.26**	**91.93**	**97.03**	**95.07**	**97.45**	**96.82**	**99.38**	**99.66**

TABLE V
TRAINING AND TESTING TIMES OF DIFFERENT LEARNING MODELS ON DIFFERENT HYPERSPECTRAL DATASETS

Method	Pavia Center/ (s)		Pavia Univ. / (s)		KSC/ (s)		Indiana Pines/ (s)		Salinas/ (s)	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
MLP	15.5	1.0	15.1	0.8	4.2	0.1	30.1	0.3	44.0	2.3
RBF	11.5	7.4	7.4	3.9	0.8	0.2	0.3	0.3	4.1	3.6
SAE	221.6	0.4	137.4	0.3	97.0	0.1	1236.2	0.4	1025.5	1.8
CNN	5700.5	17.0	1426.3	4.0	710.2	3.5	2981.5	16.5	3731.0	7.1
RBFE	43.1	43.2	39.9	25.3	6.8	8.9	1.2	2.0	20.6	20.9
CNNE	24028.0	191.9	7952.2	31.3	5609.5	88.49	15986.8	45.8	18204.8	56.9
WRBF	**10.9**	**98.5**	**4.6**	**30.1**	**2.3**	**2.0**	**78.9**	**114.6**	**4.0**	**41.5**
PMWNN	**446.6**	**102.4**	**178.6**	**30.4**	**83.2**	**2.0**	**2743.1**	**114.9**	**165.1**	**42.0**

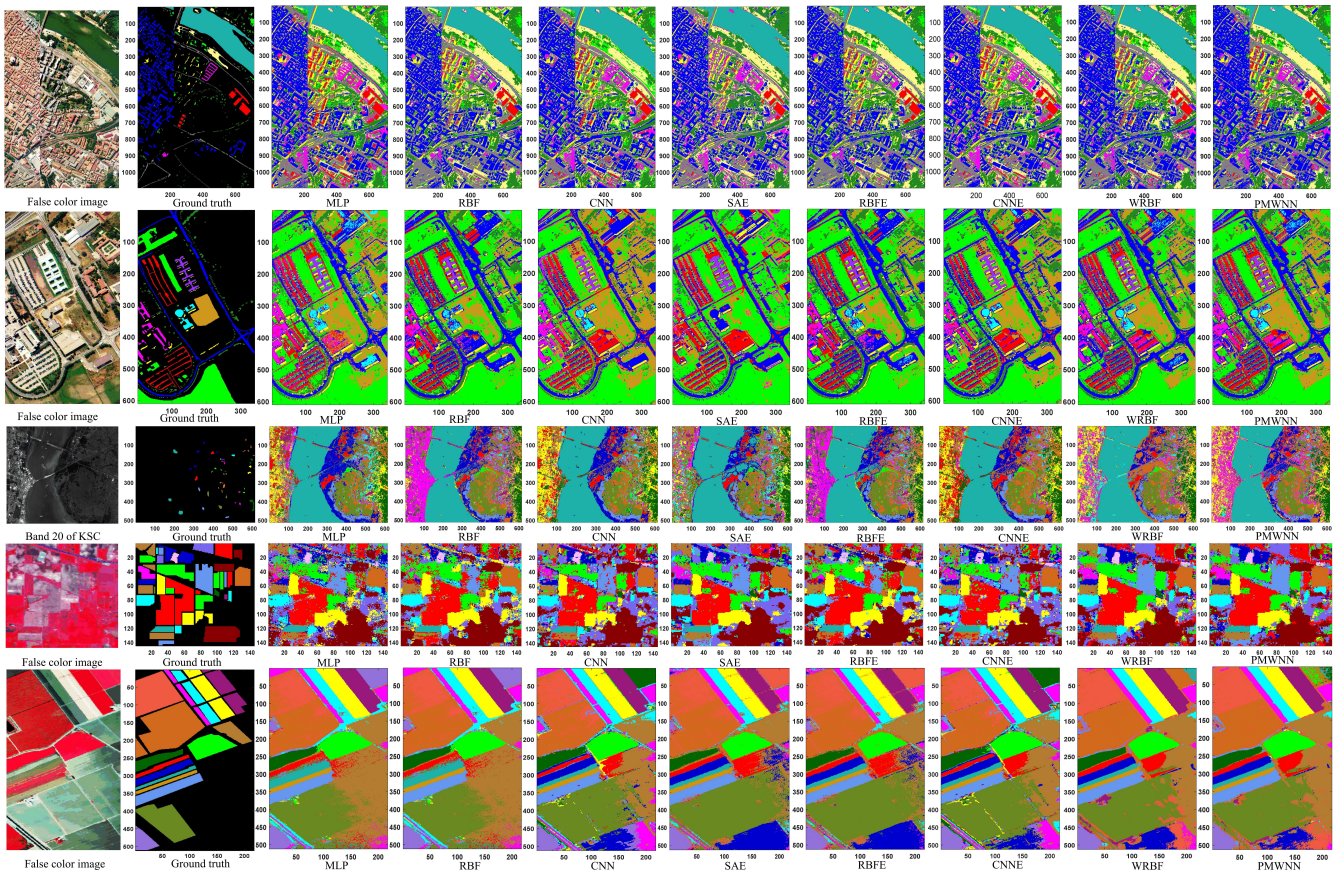

Fig. 10. Classification results with hyperspectral remote sensing datasets including Pavia Center, Pavia University, KSC, Indian Pines, and Salinas.

(4) Indian Pines: It was acquired in northwestern Indiana, which includes 200 bands after removing those bands of water absorption. The image size is145×145 pixels, and there are 16 classes including agriculture, forests, and vegetations etc.

(5) Salinas: It was gathered in Salinas valley, California including 204 bands after abandoning bands of water absorption. Its size is 512×217 pixels, and the class number is 16.

Both the spectral and spacial information (3×3 of each pixel as an instance is used for Pavia Center, Pavia University, and KSC. 7×7 and 9×9 is used for Indian Pines, and Salinas, respectively) are used for training and testing. The principal components analysis (PCA) was used to reduce the redundant spectral information, and the number of spectral features was reduced to 15 (40 for Indian Pines). The proportion of training , and validation sets is both 5% (For Indian Pines, it is 27%, 13%). The platform is the same as described in the beginning of the experiment section.

The overall accuracy (OA) and average accuracy (AA) are used to evaluate the performance. They are defined as

$$OA = N_{correct}/N_{total} \tag{34}$$

$$AA = (1/C)\sum_{i=1}^{C})N_{correct(i)} \quad N_{total(i)} \tag{35}$$

where $N_{correct}$, and N_{total} are the numbers of correctly classified and total number of testing samples, respectively. $N_{correct(i)}$ and $N_{total(i)}$ are the above two kinds of samples for class i, respectively.

The testing accuracy (or overall accuracy, OA), average accuracy(AA), together with training and testing time of the proposed PMWNN were compared with other learning models. The hyper parameters such as number of hidden units, the size and number of convolutional kernels were chosen using the validation set. The comparison methods include both shallow and deep models. The shallow models are MLP with 500 hidden units (1000 for Salinas), RBF with 2000 hidden units (500 for Indian Pines), SAE with 100 hidden units for encoder, and 50 hidden units for decoder (200 for both encoder and decoder on Indian Pines and Salinas), CNN (6c-2s-15c-2s) with 5×5 convolutional size, RBF ensemble composed of 5 RBF Nets with 2000 hidden units (500 for Indian Pines), and CNN ensemble (6c-2s-15c-2s, 5 CNNs). The results are shown in Tables IV and V. The predicted results are shown in Figure 10. It is seen that the proposed PMWNN has the best OA among all 5 datasets in Table IV. In Indian Pines and Salinas data, it is observed that the prediction results of WRBF and PMWNN are much smoother than the results of compared models.

For the computing time, it is observed in Table V that the training time of the proposed PMWNN is 446.6s, 178.6s, 83.2s, 2743.1s, and 165.1s, which is much shorter than that of CNN (5700.5s, 1426.3s, 710.2s, 2981.5, and 3731.5s), and comparable to that of SAE (221.6s, 137.4s 97.0s, 1236.2s, and 1025.5s). This is because the PMWNN has 26 stages of WRBF. It is seen that the WRBF network has much shorter training time (10.9s, 4.6s, 2.3s, 78.9s, and 4.0s) than SAE, while the OA is better than SAE. Shallow learning models MLP, RBF have less training time, but their OAs are not as good as those of WRBF and PMWNN. The testing time of the proposed WRBF and PMWNN are comparable and competitive with both shallow and deep learning models. Because the PMWNN can be naturally implemented in parallel, its testing time can be very close to the testing time of a single stage WRBF network. In the current work, the testing of the WRBF network is implemented in cascade. The testing time of both WRBF and PMWNN can be reduced further if they are implemented in parallel. This is a big advantage over deep learning models.

D. CVL Single Digit Dataset

The CVL single digit database [42] is a new freely available real world dataset collected from 303 writers, who are mostly students of the Vienna University of Technology, and an Austrian secondary school. This dataset includes 0-9 digits. The training and validation sets are both 7000 samples (700 per class) from 67 and 60 writers, respectively. The test set contains 21780 samples (2178 per class) from the remaining 176 writers.

All the samples are converted to gray scale images with the size of 28×28 following the method in SCN [11]. The training and validation sets are combined together as the training sets, and the data augmentation method is not used, which is the same as in SCN. Only the WRBF network is compared here. There are three kinds of configurations for WRBF networks with 1200, 2400, and 3600 hidden units, respectively. The number of RBF nodes of these WRBF networks is 600, and the window size is 11. The MLP, SAE (4000 hidden units for both encoder and decoder), CNN, and recently proposed SCN and its related fundamental model random vector functional link (RVFL) are compared with. The results are in Table VI. It is observed that the WRBF network has better training and test accuracies than the compared models.

E. SVHN Dataset

The Street View House Numbers (SVHN) dataset [43] is acquired from house numbers in Google Street View images. All digits are resized to a fixed resolution of 32-by-32 pixels. There are 10 classes, and the training and testing sets include 73275 and 26032 images, respectively.

The WRBF network was compared with ResNet using parameters setting in the same scale, and the same training data usage without data augmentation. The WRBF network has 1000 hidden units, the number of RBF nodes of these WRBF networks is 50, and the window size is 19. The total number of weights is 9800. The ResNet has 10035 weights in total, which includes 6 convolutional layers with 6 batch normalization, 2 skip connection layers, 1 pooliong layer, and 1 fully connection layer. The activation function was ReLu, and softmax was used before output. PMWNN is used in this experiment. The PMWNN had 26

(a) Samples of CVL data (b) Samples of SVHN images (c) Samples of SVHN dataset

Fig. 11. Samples of the (a) CVL Single Digit dataset ,and (b) (c) the SVHN dataset.

TABLE VI
ACCURACIES OF DIFFERENT LEARNING MODELS ON CVL DATASET.

Method	Train Acc. (%)	Test Acc. (%)
MLP (4000 hidden units)	97.4	85.87
CNN (6c-2s-12c-2s)	90.9	89.8
SAE (4000 hidden units)	79.3	77.4
RVFL (4000 hidden units) [11]	97.5	85.0
SCN (4000 hidden units) [11]	98.0	86.1
WRBF network-I	100	86.7
WRBF network-II	100	87.0
WRBF network-III	**100**	**91.4**

TABLE VII
ACCURACIES OF LEARNING MODELS ON AN4 AUDIO SIGNAL DATASET

Method	Train Acc. (%)	Test Acc. (%)
MLP (1000 hidden units)	82.40	71.09
RBF (1000 hidden units)	90.50	74.52
SAE (300 hidden units)	99.59	74.21
WRBF (1000 hidden units)	**88.56**	**74.80**
WRBF (2000 hidden units)	**88.68**	**76.40**

stages of WRBF networks with the same settings. For PMWNN, data augmentation was performed to generate more samples, which included image random rotation (from -20 to 20 degree), and random translation (3 pixels in both height and width directions). The test accuracy for ResNet was 76.81%. The test accuracies for the proposed WRBF network and PMWNN were 77.37% and 79.14%, respectively.

F. Audio Signal Dataset

(1) Sound Noise Dataset: Sound noise signals includes white noise, brown noise, and pink noise, which are shown in Fig. 12, and can be seen as time series. In this experiment, 1000 sound signals were generated equally for three classes of noise. The frequency of the sound signals was 44.1KHz, and the duration time is 0.5s. Therefore, there were 22050 data points for each sound signal instances. The amplitude of the white noise was uniformly distributed between -1 and 1. The system transfer function for the filter to generate the brown noise using white noise was $H(z) = 1/\ 1 - 0.99z^{-1}$. There were 800 training samples, and 200 testing samples for each class (No audio signal augmentation was performed). The centroid and slope of Mel spectrum [44] for each instance was computed, and the two feature vectors were combined together as one vector for classification. The hyper parameters were chosen using validation set (10% instances were randomly chosen from the training sets) and random search. The window size of the WRBF was 30, the number of hidden units was 50, and the number of subsampled RBF nodes was 5. Both the training and testing accuracy of WRBF could reach 100%, which demonstrated that the proposed WRBF can be used for non-image or time series classification tasks effectively.

(2) AN4 Audio Signal Dataset: It is also known as the Census dataset which was gathered by CMU Robust Speech Recognition Group. It includes audio signals from different speakers saying personal information, such as name, phone number, address, etc. 10 speakers were chosen from the whole dataset as 10 classes, and there were 125 related audio files. 80% of the audio files are chosen for training, and 20% were chosen for testing. Each audio file was split into frames of 30ms with an overlap ratio of 0.75. The pitch and 13 Mel Frequency Cepstrum and Coefficients (MFCC) [44] each frame were extracted, therefore, there were 14 feature values as a vector of a frame. No audio data augmentation was performed for the audio frames. MLP with 1000 hidden units, RBF with 1000 hidden units, SAE with 300 hidden units for both encoder and decoder were compared with WRBF with 1000, and 2000 hidden units, respectively. The CNN was not suitable for AN4 dataset because of very few number of the feature dimension. The classification results are shown in Table VII. It is observed that the proposed WRBF has the best classification performance among the compared models.

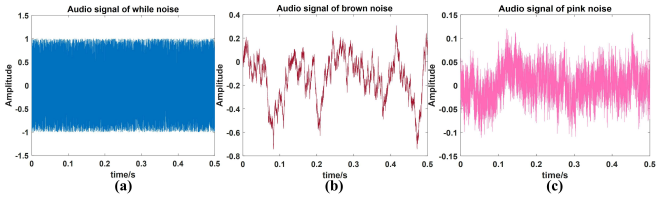

Fig. 12. Samples of three types of audio signals including: (a) white noise, (b) brown noise, and (c) pink noise.

V. Conclusions

A parallel multistage wide neural network (PWMNN) based on wide radial basis function (WRBF) network is proposed in this paper, which uses local receptive fields, and grows stages of WRBF networks also in the wide direction. Each WRBF network consists of a number of RBF networks. The subsampling method is introduced at the output of each RBF to reduce the number of outputs of each stage. The WRBF network can learn complex datasets without long training time because of using the LS method. Each stage of the proposed PMWNN network is specialized on a different part of (or difficulty level of) the given dataset and trained in cascade. Thus, the network structure is adaptive with the complexity of the dataset. Meanwhile, new stages can be added incrementally without retraining the previous stages if new input samples become available. Finally, the stages can be processed in parallel during testing, reducing the testing time to that of a single stage network. Experiments were implemented using the MNIST dataset, hyperspectral remote sensing data (Pavia Center, Pavia University, Kennedy Space Center, Indian Pines, and Salinas), CVL single digit data, SVHN dataset, and audio signal datasets (noise audio signal, and AN4 audio dataset) in comparison to other prominent networks. PMWNN always had competitive testing results. The proposed model also has potential to be used in new environments such as both GPU and embedded intellectual terminals [45], which emphasizes more on restricted processing time and computing sources, considering the properties of fast training, incremental learning to optimize the usage of computing sources, and the parallel testing of the PMWNN.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[2] O. K. Ersoy and D. Hong, "Parallel, self-organizing, hierarchical neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 2, pp. 167–178, 1990.

[3] O. K. Ersoy and S.-W. Deng, "Parallel, self-organizing, hierarchical neural networks with continuous inputs and outputs," *IEEE Transactions on Neural Networks*, vol. 6, no. 5, pp. 1037–1044, 1995.

[4] S. Aghagolzadeh and O. K. Ersoy, "Optimal adaptive multistage image transform coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 1, no. 4, pp. 308–317, 1991.

[5] J. A. Benediktsson, J. R. Sveinsson, O. K. Ersoy, and P. H. Swain, "Parallel consensual neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 54–64, 1997.

[6] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[7] S. Venkataramani, A. Raghunathan, J. Liu, and M. Shoaib, "Scalable-effort classifiers for energy-efficient machine learning," in *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015, p. 67.

[8] P. Panda, S. Venkataramani, A. Sengupta, A. Raghunathan, and K. Roy, "Energy-efficient object detection using semantic decomposition," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 9, pp. 2673–2677, 2017.

[9] P. Panda, A. Sengupta, and K. Roy, "Conditional deep learning for energy-efficient and enhanced pattern recognition," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2016, pp. 475–480.

[10] Z. Zhou and J. Feng, "Deep forest: Towards an alternative to deep neural networks," *CoRR*, vol. abs/1702.08835, 2017.

[11] D. Wang and M. Li, "Stochastic configuration networks: Fundamentals and algorithms," *IEEE Transactions on Cybernetics*, vol. 47, no. 10, pp. 3466–3479, 2017.

[12] D. Wang and C. Cui, "Stochastic configuration networks ensemble with heterogeneous features for large-scale data analytics," *Information Sciences*, vol. 417, pp. 55–71, 2017.

[13] D. Wang and M. Li, "Robust stochastic configuration networks with kernel density estimation for uncertain data regression," *Information Sciences*, vol. 412, pp. 210–222, 2017.

[14] C. L. P. Chen and Z. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 1, pp. 10–24, 2018.

[15] D. Roy, P. Panda, and K. Roy, "Tree-cnn: A deep convolutional neural network for lifelong learning," *CoRR*, vol. abs/1802.05800, 2018. [Online]. Available: http://arxiv.org/abs/1802.05800

[16] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[17] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[18] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[19] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3642–3649.

[20] S. Abpeykar and M. Ghatee, "An ensemble of rbf neural networks in decision tree structure with knowledge transferring to accelerate multi-classification," *Neural Computing and Applications*, 2018.

[21] S. Abpeykar, M. Ghatee, and H. Zare, "Ensemble decision forest of rbf networks via hybrid feature clustering approach for high-dimensional data classification," *Computational Statistics & Data Analysis*, vol. 131, pp. 12–36, 2019.

[22] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.
[23] S. Lee, J. Kim, J. Ha, and B. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," *CoRR*, vol. abs/1703.08475, 2017.
[24] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, "Progress & compress: A scalable framework for continual learning," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmssan, Stockholm Sweden: PMLR, 2018, pp. 4528–4537.
[25] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
[26] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 302–309, 1991.
[27] Y. W. Wong, K. P. Seng, and L. M. Ang, "Radial basis function neural network with incremental learning for face recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 4, pp. 940–949, 2011.
[28] M. M. Bejani and M. Ghatee, "Convolutional neural network with adaptive regularization to classify driving styles on smartphones," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2019.
[29] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.
[30] M. Li, W. Bi, J. T. Kwok, and B. Lu, "Large-scale nystrm kernel matrix approximation using randomized svd," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 1, pp. 152–164, 2015.
[31] G. Martinsson, A. Gillman, E. Liberty, N. Halko, V. Rokhlin, S. Hao, Y. Shkolnisky, P. Young, J. Tropp, M. Tygert *et al.*, "Randomized methods for computing the singular value decomposition (svd) of very large matrices," *Works. on Alg. for Modern Mass. Data Sets, Palo Alto*, 2010.
[32] A. O. Hoori and Y. Motai, "Multicolumn rbf network," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 4, pp. 766–778, 2018.
[33] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, 2003, pp. 958–963.
[34] I. Sato, H. Nishimura, and K. Yokoi, "APAC: augmented pattern classification with neural networks," *CoRR*, vol. abs/1505.03229, 2015.
[35] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[36] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.
[37] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
[38] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, vol. 5, 2009, pp. 448–455.
[39] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.
[40] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 2006.
[41] "Hyperspectral remote sensing scenes," http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes#Indian_Pines.
[42] M. Diem, S. Fiel, A. Garz, M. Keglevic, F. Kleber, and R. Sablatnig, "ICDAR2013 Handwritten Digit and Digit String Recognition Competition," 2018.
[43] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.
[44] L. Rabiner and R. Schafer, *Theory and applications of digital speech processing*. Prentice Hall Press, 2010.
[45] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song *et al.*, "Going deeper with embedded fpga platform for convolutional neural network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2016, pp. 26–35.

Jiangbo Xi is currently an assistant professor with the School of Geology Engineering and Geomatics at Chang'an University, China. He received his B.S. degree in electronic and information science and technology from Jilin University, China, in 2008, and the Ph.D. degree in signal and information processing from the University of Chinese Academy of Sciences (UCAS), China, in 2017. He was a joint Ph.D. student in electrical and computer engineering at Purdue University, USA, from April 2015 to April 2017. His current research focuses on deep learning, machine learning, and objects detection in optical sequential images.

Okan K. Ersoy (F'86) is a professor of Electrical and Computer Engineering at Purdue University. His research interests include neural networks, machine learning and pattern recognition, decision analytics, digital signal/image processing and recognition, transform and time-frequency methods, diffractive optics, and their applications. He has published approximately 250 papers, two books and several book chapters in his areas of research. He also holds 5 patents. He is a fellow of IEEE, and a fellow of Optical Society of America.

Jianwu Fang (M'15) received the Ph.D. degree in SIP (signal and information processing) from the Univerisity of Chinese Academy of Sciences, China in 2015. He is currently an Associate Professor with Laboratory of Traffic Vision Safety (LOTVS) in the School of Electronic and Control Engineering, Chang'an University, Xi'an, China. He has published many papers on IEEE-TIE, IEEE-TCYB, IEEE-TCSVT, and AAAI, etc. His research interests include computer vision and pattern recognition.

Tianjun Wu received the B.S. degree in information science and the M.S. degree in applied mathematics from Chang'an University, China, in 2009 and 2012 respectively, and the Ph.D. degree in cartography and geographical information system from the University of Chinese Academy of Sciences (UCAS), China, in 2015. He is currently an Associate Professor in the Department of Mathematics and Information Science, Chang'an University. His research interests include the intelligent information extraction from remote sensing images etc..

Xin Wei received the Bachelor degree in Microelectronics from Jilin University, China, in 2014. He is currently pursuing the Ph.D. degree in SIP (signal and information processing) at the University of Chinese Academy of Sciences (UCAS). His research interests include the embedded system, star identification and machine learning.

Chaoying Zhao (SM'18) received his M.S. and Ph.D degrees in geodesy and surveying engineering from Chang'an University, China, in 2002 and 2009, respectively. His research interests include SAR and interferometric SAR techniques development and their applications. He is currently the professor with the School of Geology Engineering and Geomatics, Chang'an University, China. He is the PI or Co-PI of five NSFC projects and National Key R&D Program of China.