IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 3, 2021, accepted March 22, 2021, date of publication March 24, 2021, date of current version April 7, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3068880

Scalable Wide Neural Network: A Parallel, Incremental Learning Model Using Splitting Iterative Least Squares

JIANGBO XI[1,2], **OKAN K. ERSOY**[3], **(Fellow, IEEE), JIANWU FANG**[4], **(Member, IEEE), MING CONG**[1,2], **XIN WEI**[5,6], **AND TIANJUN WU**[7]

[1]College of Geological Engineering and Geomatics, Chang'an University, Xi'an 710054, China
[2]Key Laboratory of Western China's Mineral Resources and Geological Engineering, Ministry of Education, Xi'an 710054, China
[3]School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA
[4]College of Transportation Engineering, Chang'an University, Xi'an 710064, China
[5]Xi'an Institute of Optics and Precision Mechanics, CAS, Xi'an 710119, China
[6]School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100039, China
[7]Department of Mathematics and Information Science, College of Science, Chang'an University, Xi'an 710064, China

Corresponding authors: Okan K. Ersoy (ersoy@purdue.edu) and Jianwu Fang (fangjianwu@chd.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFC1504805; in part by the National Natural Science Foundation of China under Grant 61806022 and Grant 41874005; in part by the State Key Laboratory of Geo-Information Engineering under Grant SKLGIE 2018-M-3-4; in part by the Fundamental Research Funds for the Central Universities under Grant 300102269103, Grant 300102269304, Grant 300102260301, and Grant 300102260404; and in part by the China Scholarship Council (CSC) through Scholarship under Grant 201404910404.

ABSTRACT With the rapid development of research on machine learning models, especially deep learning, more and more endeavors have been made on designing new learning models with properties such as fast training with good convergence, and incremental learning to overcome catastrophic forgetting. In this paper, we propose a scalable wide neural network (SWNN), composed of multiple multi-channel wide RBF neural networks (MWRBF). The MWRBF neural network focuses on different regions of data and nonlinear transformations can be performed with Gaussian kernels. The number of MWRBFs for proposed SWNN is decided by the scale and difficulty of learning tasks. The splitting and iterative least squares (SILS) training method is proposed to make the training process easy with large and high dimensional data. Because the least squares method can find pretty good weights during the first iteration, only a few succeeding iterations are needed to fine tune the SWNN. Experiments were performed on different datasets including gray and colored MNIST data, hyperspectral remote sensing data (KSC, Pavia Center, Pavia University, and Salinas), and compared with main stream learning models. The results show that the proposed SWNN is highly competitive with the other models.

INDEX TERMS Wide neural network, least squares, fast training, incremental learning.

I. INTRODUCTION

It is well known that one model cannot give answers to all kinds of tasks. Deep convolutional neural networks work excellently on image, video, and speech tasks, but except for these tasks, other learning models such as random forest [1], and XGBoost [2] still play an important role in applications such as booking tickets, hotel etc., and win in many Kaggle competition tasks. Therefore, it is helpful to study different kinds of learning models to push forward the development of machine learning, together with the development of deep learning architectures. For example, recently, Hinton proposed the capsules network [3], in which the neurons are organized as groups, also known as capsules, to represent an object or an object part by an activity vector. Kontschieder *et al.* [4] proposed deep neural decision forests, which use random, differentiable decision trees to help the representation learning in hidden layers of deep networks.

For complex learning tasks, learning models always become deep with many layers and a great number of parameters, often with good test performance and generalization. However, they are still poorly understood in a theoretical sense, and it is hard to characterize the training and generalization because of highly non-convex behaviour.

The associate editor coordinating the review of this manuscript and approving it for publication was Easter Selvan Suviseshamuthu.

Recently, researchers also did many works on wide learning [5], [6], or both wide and deep learning architectures [7], where the width refers to the number of hidden units in a fully connected layer or channels in a convolutional layer. A broad learning system (BLS) was also proposed by Chen and Liu [8] to learn incrementally and effectively without a deep learning structure. It was shown that the wide fully connected neural networks are equivalent to Gaussian Processes, which make the characterization of training and learning process simpler by evaluating the Gaussian process. The wide neural network can also be found to generalize better [5], [6].

One natural way to make learning much wider is using ensemble methods, for example, bagging [9] and boosting (including Adaboost) [10], in which multiple learners (or classifiers) are trained together to achieve better generalization. Ciregan *et al.* [11] proposed multi-column deep neural network (MCDNN) including 35 independent convolutional neural networks as an ensemble. Another benefit of ensemble learning is that the classifiers can be organized in a hierarchical approach, and therefore, the designed models can learn knowledge incrementally. Roy *et al.* [12] proposed Tree-CNN, which is a hierarchical deep convolutional neural network for incremental learning. An ensemble of RBF neural networks in decision tree structure with knowledge transfer (ERDK) [13] is composed of radial basis function (RBF) networks as tree nodes with a decision to sort features.

Incremental learning itself is an important desired property of learning models to overcome catastrophic forgetting [14]. The elastic weight consolidation (EWC) was proposed by Kirkpatrick *et al.* [15], which remembers learned knowledge by selectively reducing important weights on these knowledge. Lee *et al.* [16] proposed incremental moment matching (IMM), which uses the Bayesian neural network, and assumes that the posterior distribution of parameters of neural networks matches the moment of the posteriors incrementally. Stochastic configuration networks (SCNs) [17] was proposed by Wang *et al.*, which uses stochastic configuration to generate models incrementally. Dhar *et al.* [18] proposed learning without memorizing (LwM) to learn new classes and to preserve knowledge of old classes without sorting the used data. The forest of decision trees with RBF networks and knowledge transfer (FDRK) [19] was proposed to classify features using a tree of RBF nodes.

Two goals in developing learning models is to reduce both training and testing time so that they can be used in real-time or fast computing applications, such as with mobile terminal or edge computing. Recently, researchers did more and more work on developing lightweight deep learning models, including Xception [20], ShuffleNet v2 [21] and so on. These models are used more easily in embedded applications. Recently, lightweight learning models have also been proposed for hyperspectral image classification such as lightweight convolutional neural network [22] and spectral-spatial squeeze-and-excitation residual bag-of-features learning [23]. Another way to reduce the training time is using RBF network [24]–[27]. It has two layers including the kernal functions layer and the fully connected layer. The weights of the second layer are trained using least squares (LS). The recent works on RBF networks can be categorized into two types. (1) Using RBF networks as an ensemble: For example, multicolumn RBF network [25] uses k-d tree two select subsets of features, and learns these subsets with different RBF networks. It has excellent performance on classification of manually selected features, but it is difficult to use this kind of ensemble to do image classification. ERDK and FDRK can also be categorized into this type. (2) Using RBF kernels to design different kinds of activation functions for CNN: Such as deep RBF network [28], DeepLABNet [29], which are end-to-end deep Radial Basis networks with fully learnable basis functions, and deep RBF value functions for continuous control. These networks can be used for image classification or system control, but actually, they are still trained using gradient based methods, which is time consuming when the data is large and the architectures are deeper [30].

In general, the longstanding goals of a learning model include: (1) learning and testing efficiently and fast, which means the architecture can be optimized with less number of parameters, the training time can be reduced and accelerated; (2) robust generalization, which means the learning model has a good testing performance on previously unseen data; and (3) incremental learning to overcome catastrophic forgetting, which means it learns new knowledge without forgetting learned knowledge (can do life-long learning).

In this paper, we propose a scalable wide neural network (SWNN), considering the above goals, and the contributions are: (1) It can deal with both images and vectors as input samples. It is composed of multi-channel wide RBF (MWRBF) neural networks as learning stages, which can be generated incrementally to learn data without overfitting. The MWRBF extends the input instances in the wide direction as input to the RBF kernels, and the outputs are summed among channels to reduce the data, sorted, and subsampled to reduce the output number of Gaussian basis functions. (2) The SWNN can be trained fast because of using iterative splitting least squares along both feature and sample dimensions to save training time and to deal with large and high dimensional datasets compared with the discussed above other recent RBF related methods.

The rest of the paper is organized as follows: Section II provides an overview of related work on scalable learning. Section III presents the proposed scalable wide neural network. Section IV provides the experimental settings and the results with the proposed SWNN. In Sections V and VI, the discussion and conclusions are given.

II. RELATED WORK

A. MULTISTAGE SCALABLE LEARNING MODELS

Early works on multistage scalable learning models were proposed in 1990's. Ersoy *et al.* published works including the parallel, self-organizing, hierarchical neural networks (PSHNN) [31], and parallel consensual neural network

(PCNN) [32]. PSHNN is composed of multiple stages. Nonlinear transforms are performed on input dataset of all the stages. Except for the first stage, each stage only receives the rejected instances from the previous stage. Nonlinear transforms convert these samples into another space to make them easier to classify. They also designed competitive learning and safe rejection schemes [33] for PSHNN, and a network with continuous inputs and outputs [34]was developed. PCNN uses statistical consensus theory to organize stages of neural networks, and the final output is decided by the consensus result among all the stages of neural networks. Additionally, adaptive multistage image transform coding is proposed with an optimal method for bit allocation. The quantization errors of a previous stage are corrected by the current stage. Recently, scalable-effort classifiers [35], [36] were proposed, composed of multiple stages of classifiers with increasing architectural complexity as the stages increase. The number of stages is adjusted by the difficulty of the input instances.

B. TREE-LIKE AND DEEP SCALABLE LEARNING MODELS

Recently, a Tree-CNN was proposed by Roy *et al.* [37], which is a tree-like approach and the deep convolutional neural network is organized in a hierarchical way to overcome catastrophic forgetting and to realize lifelong learning. Jiang *et al.* [38] proposed another version of Tree-CNN, which includes a cluster algorithm and a Trunk-CNN as Tree CNN. The branch-CNNs are fine tuned by extracting a feature map and dividing it into fine classes. ERDK [13] uses decision tree to generate RBF networks as leaf nodes, and the class labels of a corresponding leaf node are determined by the RBF network of this node. The forest of decision trees with RBF networks and knowledge transfer (FDRK) [19] was proposed by Abpeykar, which classifies high-dimensional data using a tree composed of RBF tree nodes. Hybrid clustering is used to sort the features, which are used to generate nodes of RBF neural networks. A tree-based deep network [39] was proposed by Muhammad *et al.*, which has many branches composed of convolutional layers and can be used for edge devices. Zhou and Feng [40] proposed deep forest, also known as multi-Grained Cascade Forest (gcForest), which uses a decision tree to generate an ensemble and can be implemented in parallel. A conditional deep learning (CDL) was proposed by Panda *et al.* [41], in which the deep convolutional layer can be activated when the input is difficult to be classified. This method can adjust the architecture according to the difficulty of test instances, and thereby, the computing resources can be saved. Lin *et al.* [42] used focal loss to deal with imbalanced data in object detection. Kim *et al.* [43] proposed a regularization method to train the network and classify them correctly. Hou *et al.* [44] proposed a framework to learn a unified classifier incrementally. We see that researchers are paying more and more attention to develop new learning models or new methods to overcome the deficiencies of current deep learning models.

III. SCALABLE WIDE NEURAL NETWORK

A. MULTI-CHANNEL WIDE RBF NETWORK

The convolutional neural network has the property of using a number of convolutional kernels to extract features in the space domain on images with multiple channels. Another advantage of CNN is that although it has a great number of parameters and many layers, these parameters are highly reduced by using local connections and weights sharing. The local receptive field is used to define the area to perceive local features, while these features are extracted by a group of convolutional kernels with shared weights in different local areas of an image. Color images usually have three channels including red, green, and blue, while hyperspectral images have hundreds of bands, which can be considered as channels. CNNs can work very well for classification of such datasets. On the other hand, it is hard for the original RBF network to get a pretty good performance with such data. When there are many channels, and especially when the images are big, the input instances have large dimension after being flattened into a vector. It consumes large computing resources to train such a shallow RBF network with much greater number of hidden RBF kernels as hidden units. We proposed a multi-channel wide RBF neural network to use the advantages of RBF networks more efficiently on such datasets as shown in Fig. 1. There are mainly three characteristics with this network. (1) Local connections are used to reduce the number of RBF kernels, (2) RBF networks are generated for each local sliding window with multi channels, and they are combined in the wide direction as the MWRBF. (3) Subsampling is used to reduce the number of outputs from RBF kernels, and thereby, the number of linear weights are reduced [45].

Assume the input dataset is $\mathbf{X} \in \mathbf{R}^{d_1 \times d_2 \times d_3 \times N}$, where d_1, d_2, and d_3 are the sizes of instances in each dimension. Supposing the dataset is in the format of images, they denote the height, width, and channel of an input image. N denotes the number of data samples. Multi-channel sliding window with size $r_1 \times r_2 \times d_3$ is used from left to right, and top to bottom, to choose the local area. We also denote $r = r_1 \times r_2$. The default sliding stride is 1, therefore, the sliding index k satisfies $1 \leq k \leq K = (d_1 - r_1 + 1) \times (d_2 - r_2 + 1)$. The input instances of the local area for the kth sliding index is $\mathbf{x}_k \in \mathbf{R}^{r \times d_3 \times N}$ after being flattened in dimensions d_1 and d_2. $\mathbf{x}_k \in \mathbf{R}^{r \times d_3 \times N}$ can be rewritten as $x_k = \left[x_k^1, x_k^2, \cdots, x_k^{d_3} \right]$, in which each element represents an input data cube for lth channel $(1 \leq l \leq d_3)$, and the size of these elements is $r \times N$. These data cubes are fed into d_3 RBF networks with N_k^l Gaussian response functions $\{\phi_{k1}^l, \phi_{k2}^l, \ldots, \phi_{kN_k^l}^l\}$ as RBF kernels. The outputs of the Gaussian response functions are

$$\Phi_k^l = \left[\phi_{k1}^l \left(\mathbf{x}_k^{lT}\right), \quad \phi_{k2}^l \left(\mathbf{x}_k^{lT}\right), \quad \ldots, \quad \phi_{kN_k}^l \left(\mathbf{x}_k^{lT}\right) \right] \tag{1}$$

$$\phi_{ki}^l \left(\mathbf{x}_k^{lT}\right) = \exp\left(\frac{\left\| \mathbf{x}_k^{lT} - \mathbf{c}_i^l \right\|^2}{\sigma_i^{l2}} \right) \tag{2}$$

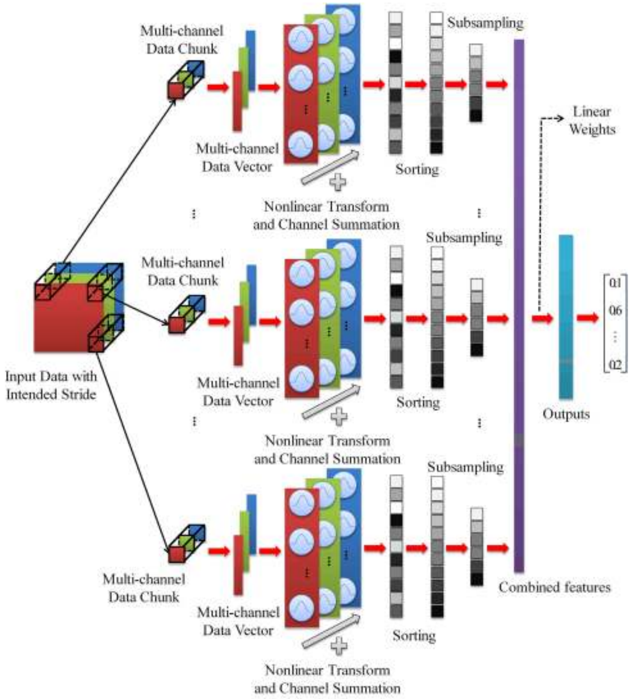


FIGURE 1. **The architecture of the MWRBF network. The MWRBF extends the input samples in the wide direction and sends them to the RBF kernels, then the response outputs are summed among channels to reduce the data, sorted, and subsampled to reduce the output features. The least squares is used to compute the weights fast. During testing, the sampled features can be computed in parallel in the wide direction.**

where $\mathbf{c}_i^l$, and σ_i^l are the parameters of the Gaussian function, respectively. $(\cdot)^T$ denotes the transpose of a vector or a matrix.

After the outputs of all the channels are obtained, the following three steps are processed:

(1) Channel summation: Outputs are summed together as

$$\Phi_k = \sum_{l=1}^{d_3} \Phi_k^l. \tag{3}$$

(2) Response sorting: Φ_k is summed along sample dimension and sorted in descending order:

$$\Phi_k' = sort\left(\sum_{col=1}^{N} \Phi_k \downarrow\right). \tag{4}$$

(3) Subsampling:

$$\Phi_{kS} = subsample\left(\Phi_k', N_{kS}\right) \tag{5}$$

where N_{kS} is the subsampling interval.

Channel summation reduces the number of computing channels, whereas response sorting and subsampling result in sorting different Gaussian responses and reserve abundant feature information with a reduced computational load.

Let $N_S = \sum_{k=1}^{K} N_k$ be the total number of outputs after subsampling, where N_k is the subsampling number for kth sliding. Then, the matrix $\Phi \in \mathbf{R}^{N \times N_S}$ with all the outputs is

$$\Phi = \left[\, \Phi_{1S}, \quad \Phi_{2S}, \quad \cdots, \quad \Phi_{KS} \,\right]. \tag{6}$$

The final outputs of the MWRBF network are given by

$$\mathbf{Y} = \Phi \mathbf{W}. \tag{7}$$

The least squares estimation of $\hat{\mathbf{W}}$ is obtained by minimizing the square error

$$\hat{\mathbf{W}} = \arg\min_{\mathbf{W}} \|\Phi\mathbf{W} - \mathbf{D}\|^2, \tag{8}$$

where $\mathbf{D}$ is the vector of desired outputs.

The pseudoinverse Φ^+ of Φ is used to calculate $\hat{\mathbf{W}}$ [46]:

$$\hat{\mathbf{W}} = \Phi^+ \mathbf{D} = \left(\Phi^T \Phi\right)^{-1} \Phi^T \mathbf{D}. \tag{9}$$

B. SCALABLE WIDE NEURAL NETWORK

It is known that CNNs usually need to be trained several hundreds of epochs together with validation process to reach a good performance, while MWRBF is only trained once using least squares. Therefore, we introduce more MWRBF networks incrementally to learn adequate features in a large dataset. Each MWRBF is just focused on reducing a proportion of training errors, which can be continuously reduced as the WRBF networks are added one after the other.

Another problem is that when the training dataset includes a large number of instances and features, it is hard to compute the weights with all the training samples at one time using the LS method. Therefore, the splitting iterative least squares (SILS) training method is proposed to train the proposed SWNN as described below.

1) THE ARCHITECTURE OF SWNN

The number of MWRBF networks depends on the size of the data and the complexity of the application. Assuming there are P MWRBF networks, the dataset $\mathbf{X}$ can be learned P times. The architecture is shown in Fig. 2. The first MWRBF is trained to learn the labels of the training data, then the second is trained with the same training set to learn the remaining error from the first MWRBF. The whole network is composed in this way. Each MWRBF has its own learning tasks, and the learning error can be reduced by one MWRBF after the other.

The learning process is organized in cascade using MWRBF networks as learning stages. Supposing the MWRBF networks at corresponding stage are $h_1, h_2, \ldots, h_P$, the outputs after subsampling fed to the linear layers of MWRBF networks are $\Phi_1, \Phi_2, \ldots, \Phi_P$ ($\Phi_i \in R^{N \times PN_S}$, $1 \leq i \leq P$), and the expected output of the dataset is $D \in R^{C \times N}$. Then, the output of the SWNN denoted as $\mathbf{Y}$ can be expressed as

$$\begin{aligned} Y &= \Phi W = [\Phi_1, \Phi_2, \ldots, \Phi_P]\, W \\ &= \Phi_1 W_1 + \Phi_2 W_2 + \cdots + \Phi_P W_P. \end{aligned} \tag{10}$$

2) TRAINING SWNN USING SPLITTING ITERATIVE LEAST SQUARES

The SWNN has many attractive properties. However, as the training data becomes much larger both in numbers of dimensions and in instances, the training process with the

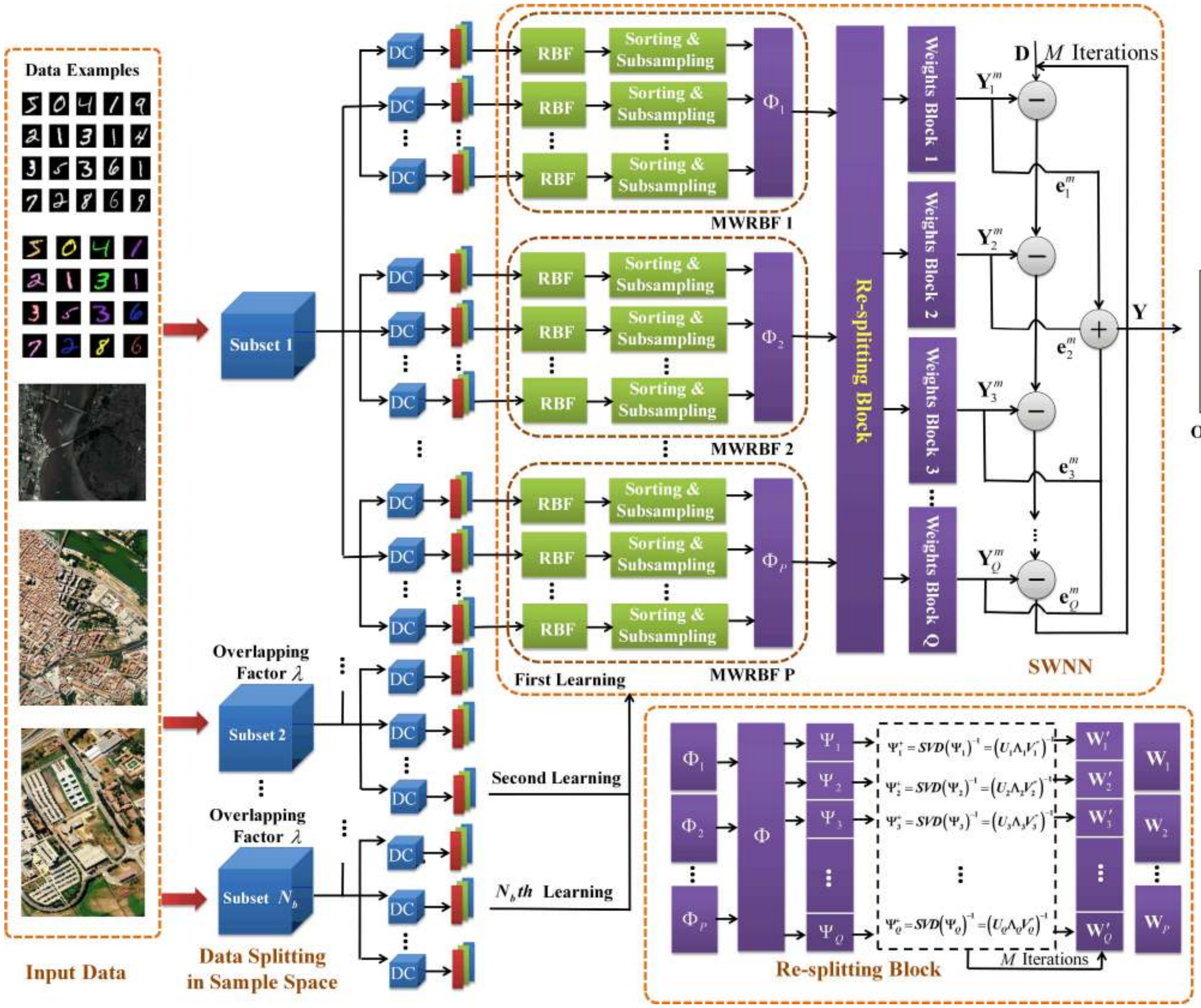


FIGURE 2. The architecture of the SWNN, and the re-splitting block (DC represents data chuck in the figure). The input data is split into subsets with overlapping factors λ to train the network continuously. The SWNN is composed of *P* MWRBF networks. Each MWRBF network learns to reduce a portion of training errors, and can be generated one by one incrementally. It is trained fast because of using least squares instead of gradient descent to compute the weights. For the large and high dimensional data, the splitting iterative least squares (SILS) along both feature (re-splitting block) and sample (training subsets) dimensions are used to reduce both computing load and time. The SWNN can be tested highly in parallel from different levels (from single MWRNF to SWNN).

LS method consumes both time and computing resources. In order to accelerate the training process and to reduce the concurrent computing resources, the splitting iterative training method is proposed, in which the training data is separated both in feature and sample space, and trained interactively.

First, in order to learn the features more accurately, the data separation is performed in the sample space. The proposed method trains the SWNN for N_c epochs using the shuffle operation. For each epoch, the training data X is separated into N_b parts with a proportion of overlap (use overlapping factor $\lambda, 0 \leq \lambda \leq 1$). For example, consider two MWRBF neural networks setup to describe the training process. Suppose, for the first epoch, we have

$$X = X_1^1, X_2^1, \ldots, X_{N_b}^1. \tag{11}$$

Then, for each training part input to the SWNN network, P Gaussian output matrices obtained for P MWRBF are

$$\Phi^1 = \left[\Phi_1^1, \Phi_2^1, \ldots, \Phi_P^1\right]. \tag{12}$$

The current output is

$$Y = \Phi W^1 = \Phi_1^1 W_1^1 + \Phi_2^1 W_2^1 + \cdots + \Phi_P^1 W_P^1. \tag{13}$$

Then, the MWRBF network h_1 is used to reduce the learning error by

$$\Phi_1^1 W_1^1\,(1) = D \tag{14}$$

The weights of h_1 is computed by

$$W_1^1\,(1) = \Phi_1^{1+} D. \tag{15}$$

The output of h_1 is computed by

$$Y_1^1(1) = \Phi_1^1 \Phi_1^{1+} D. \tag{16}$$

The remained error of h_1 is denoted by

$$e_1^1(1) = D - Y_1^1 = D - \Phi_1^1 \Phi_1^{1+} D. \tag{17}$$

Then, the second MWRBF h_2 is added to reduce the learning error left by

$$\Phi_2^1 W_2^1(1) = e_1^1(1). \tag{18}$$

The weights of h_2 are computed by

$$W_2^1(1) = \Phi_2^{1+} e_1^1(1). \tag{19}$$

The output of h_2 is computed by

$$Y_2^1(1) = \Phi_2^1 \Phi_2^{1+} e_1^1(1). \tag{20}$$

The remaining error of h_2 is denoted by

$$e_2^1(1) = e_1^1(1) - Y_2^1(1) = e_1^1(1) - \Phi_2^1 \Phi_2^{1+} e_1^1(1). \tag{21}$$

The second iteration of the two MWRBF networks is shown below. Other iterations are performed in the same way.

The weights of h_1 for the second iteration are computed by

$$\Phi_1^1 W_1^1(2) = Y_1^1(1) + e_P^1(1) \tag{22}$$

$$W_1^1(2) = \Phi_1^{1+}\left[Y_1^1(1) + e_P^1(1)\right]. \tag{23}$$

The output of h_1 for the second iteration is computed by

$$Y_1^1(2) = \Phi_1^1 \Phi_1^{1+}\left[Y_1^1(1) + e_P^1(1)\right]. \tag{24}$$

The remaining error of h_1 for the second iteration is

$$e_1^1(2) = Y_1^1(1) + e_P^1(1) - Y_1^1(2). \tag{25}$$

The weights of h_2 for the second iteration is computed by

$$\Phi_2^1 W_2^1(2) = Y_2^1 + e_1^1(2) \tag{26}$$

$$W_2^1(2) = \Phi_2^{1+}\left[Y_2^1(1) + e_1^1(2)\right]. \tag{27}$$

The output of h_2 for the second iteration is computed by

$$Y_2^1(2) = \Phi_2^1 \Phi_2^{1+}\left[Y_2^1(1) + e_1^1(2)\right]. \tag{28}$$

The remaining error of h_2 for the second iteration is

$$e_2^1(2) = Y_2^1(1) + e_1^1(2) - Y_2^1(2). \tag{29}$$

The remaining error reduction process can be performed in the same way. With P MWRBFs, the training process is similar. After all the MWRBFs processed in the first iteration, the process is iterated M times to continue reducing the learning error. The iteration stops when the validation performance stops improving for a given number of times.

After that, the second training part with an overlap proportion (to learn new knowledge with less forgetting of old knowledge) is used in the same way based on the remaining error in the first training part. For each iteration, the validation performance is evaluated and compared with the result of the previous iteration. If the performance decays for a given number of times (adjusted according to the task requirement), the training process will stop to avoid overfitting. After all the trained parts are used, the training data X is shuffled and split for the second epoch. This process continues until the maximum training epoch is reached, or validation performance stops improving. The optimizers like stochastic gradient descent (SGD), AdaGrad, AdaDelta, RMSProp, Adam, and especially the recently proposed diffGrad have excellent performance on training CNNs. These methods compute the gradients to find the trained parameters. For the proposed SWNN, the proposed iterative least square method is used to train the network, which uses pseudo inverse to compute the weights, and does not need to compute the gradients. This is completely different from the gradient based methods.

3) WEIGHTS RE-SPLITTING AND INCREMENTAL LEARNING

When the feature matrix Φ extracted by P MWRBFs is much greater than the number of training samples, the feature matrix Φ is split along the feature dimension to make the feature dimension close to the number of samples. Supposing the splitting number is Q, the output matrix is rewritten as

$$\begin{aligned} Y = \Phi W &= \left[\Psi_1, \Psi_2, \ldots, \Psi_Q\right] W \\ &= \Psi_1 W'_1 + \Psi_2 W'_2 + \cdots + \Psi_Q W'_Q. \end{aligned} \tag{30}$$

Then, $\left[\Psi_1, \Psi_2, \ldots, \Psi_Q\right]$ is used instead of $[\Phi_1, \Phi_2, \ldots, \Phi_P]$ to train the SWNN network. The training details are shown in Algorithm 1.

The number of MWRBF networks can be added incrementally depending on the complexity of the tasks. More specifically, the new MWRBF can be added one by one according to the remaining errors of current SWNN with a threshold T_s. Suppose the trained SWNN is composed of P MWRBF networks, and the new training subset is X_{new}. The remaining training error of the Pth MWRBF after training with X_{new} is e_P. The $(P+1)$th MWRBF is added by learning the weights using

$$W_{P+1} = \Phi_{P+1}^{+} e_P \tag{31}$$

where Φ_{P+1} is the subsampled outputs in the $(P+1)$th MWRBF.

4) SPEEDING UP THE TESTING OF THE SWNN

For deep neural networks, although the convolutional process can be performed in parallel, the testing process is still performed one layer after another. As the architecture goes deeper and deeper, the testing time becomes more and more important, especially in the applications of edge computing, and embedded computing. Researchers are making more and more endeavor on lightweight methods to compress and speed up the deep models [36]. The tree based deep network [39] is composed of many convolutional layers as parallel branches, and can be implemented more easily on edge devices.

The architecture of SWNN can be executed in parallel in three levels to reduce the testing time. (1) The testing

Algorithm 1 Training Procedure

```
Input: Original training dataset $X$; training epochs $N_c$; numbers of separated training parts, separated feature dimensions, and iterations: $N_b$, $Q$, and $M$; overlapping factor $\lambda$; initialized SWNN network composed of $P$ MWRBF networks: $h_1, h_2. \ldots, h_P$.
Output: Weights $W$ of Trained SWNN
Separate $X$ into $N_b$ training parts with overlapping factor $\lambda$: $X = X_1^1, X_2^1, \ldots, X_{N_b}^1$;
for epoch $i = 1 : N_c$ do
    for Data parts $j = 1 : N_b$ do
        Compute Caussian output matrix $\Phi$ of $P$ MWRBF networks using $X_i^j$: $\Phi^i = \left[\Phi_1^i, \Phi_2^i, \ldots, \Phi_P^i\right]$;
        Separate $\Phi^i$ into $Q$ parts:$\Phi^i = \left[\Psi_1, \Psi_2, \ldots, \Psi_Q\right]$;
        for Separated feature groups $k = 1 : Q$ do
            Compute the weight with $M$ maximum iterations and early stop with validation process:
            $\Phi_k^{i+} = SVD(\Phi_k^i)^{-1}$
            $W'^k_i = \Phi_k^{i+} e_k^i$;
        end for
    end for
end for
$W = \left[W'_1, W'_2, \ldots, W'_Q\right] = [W_1, W_2, \ldots, W_P]$
```

instances can be organized and stored in distributed way, (2) Each MWRBF can be implemented in parallel, (3) The multistages of MWRBF can be tested in parallel.

a: STORING THE TEST INSTANCES IN DISTRIBUTED WAY

Suppose the test data is X_t. Let the sliding window be $r_1 \times r_2$, and the sliding stride (Default value is 1) be determined. X_t is separated into $K = (d_1 - r_1 + 1) \times (d_2 - r_2 + 1)$ test data blocks, which are denoted as

$$X_t = X_{t1} \cup X_{t2} \cup \cdots \cup X_{tK}. \tag{32}$$

The test data blocks can be stored and used in parallel.

b: TESTING MWRBF NETWORKS IN PARALLEL

The trained MWRBF network is composed of K multi-channel RBF networks with data sorting and subsampling. Each multi-channel RBF is focused on a test data block, and performed independently from each other. The parallel testing outputs of all the test data blocks after sorting and subsampling are

$$\Phi_t S = [\Phi_{tS1}, \Phi_{tS2}, \ldots, \Phi_{tSK}]. \tag{33}$$

c: TESTING SWNN NETWORK IN PARALLEL

The SWNN network is composed of and trained with P MWRBF networks: $h_1, h_2, \ldots, h_P$. During training, each MWRBF is focused on reducing the total error of the current stage. Therefore, they are trained in cascade stage by stage. During testing, these networks can be implemented independently and run in parallel. The testing outputs of these P networks are

$$Y_t = [Y_{t1}, Y_{t2}, \ldots, Y_{tP}]. \tag{34}$$

The final test output of SWNN is

$$Y_f = \sum_{i=1}^{P} Y_{ti}. \tag{35}$$

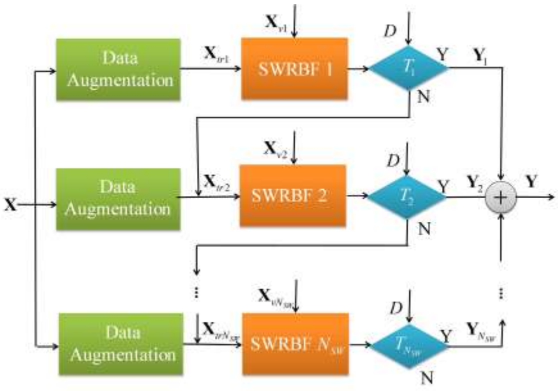


FIGURE 3. The architecture of the MSWNN.

C. MULTISTAGE SCALABLE WIDE NEURAL NETWORK

The SWNN can be trained by iterations with multiple MWRBF networks until the desired accuracy or maximum number of iterations is reached.

For datasets with large number of samples, multiple SWNNs can be organized as multistage scalable wide neural network (MSWNN) to make each SWNN learn a different subset of the dataset. Another reason for such design is that it can also learn misclassified samples from previous MWRBF. The architecture is shown in Fig. 3. Suppose the training dataset is D_{tr}, and it is split into M subsets, which are D_{s1}, D_{s2}, ..., and D_{sM}. The MSWNN $H_M SW$ is composed of M SWNNs, which are denoted as $H_1, H_2, \ldots,$ and H_M.

IV. EXPERIMENTS

A. DATASETS AND EXPERIMENTAL SETUP

The datasets in the experiments include the following:

(1) MNIST Data: The MNIST [47] data is handwritten digits, including a training set of 60000 instances, and a test set of 10000 instances. These digits are of size 28 × 28 pixels. The dataset samples are shown in Fig. 4 (a).

(2) Colored MNIST Data: The colored MNIST was generated to test the performance of the SWNN. The samples of the data are shown in Fig. 4 (b). We selected 10 different colors as the main colors. These colors were assigned to the instances randomly using uniform distribution. Then, the colored instances were transformed from RGB to HSI color model. We used the gray image as the intensity component of the HSI model, and finally transformed HST to RGB color model. The generating process is shown in Fig. 4 (c).

(3) KSC: The KSC dataset was acquired over the Kennedy Space Center in Florida on March 23, 1996, has 176 bands

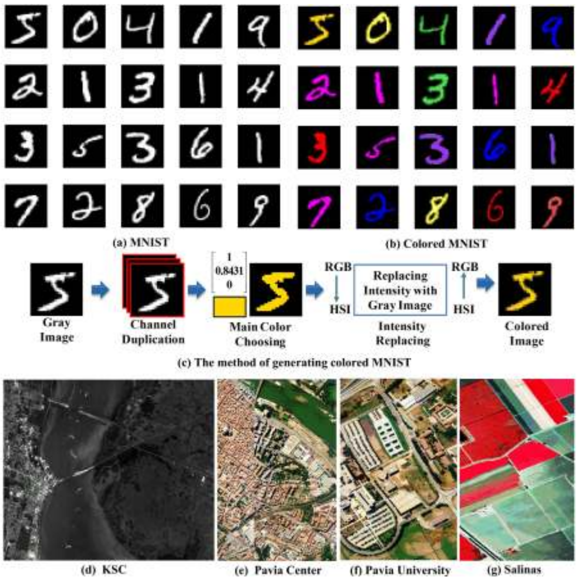


FIGURE 4. Experimental Datasets. (a) Original MNIST dataset. (b) Colored MNIST dataset. (c) Example of generating a colored handwriting digit sample. (d) KSC data. (e) Pavia Center data. (f) Pavia University data. (g) Salinas data.

after removing water absorption and 224 bands. The size of the image is 512 × 614, and there are 13 classes for classification. The data is shown in Fig. 4 (d).

(4) Pavia Center: The Pavia Center scenes were acquired by the ROSIS sensor over Pavia, in northern Italy. The number of bands is 102, and the size of the images is 1096 × 1096 (After discarding the pixels without information, the size is 1096 × 715). There are 9 classes in the ground truth. If all the spectral and spatial information are used, the computing load is excessive. The data is shown in Fig. 4 (e).

(5) Pavia University: The Pavia University scenes were acquired by the ROSIS sensor over Pavia. It has the size of 610 × 610 (After discarding the pixels without information, the size is 610 × 340). There are 9 classes in the ground truth. The data is shown in Fig. 4 (f).

(6) 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. The data is shown in Fig. 4 (g).

The experiments were implemented using a Work Station with Intel-i7-9700K CPU @ 3.6 GHz, and 64G memory. For all the experiments, the centers of RBF basis functions were randomly chosen from the training samples, and the width was chosen as a fixed number neither too big or too small.

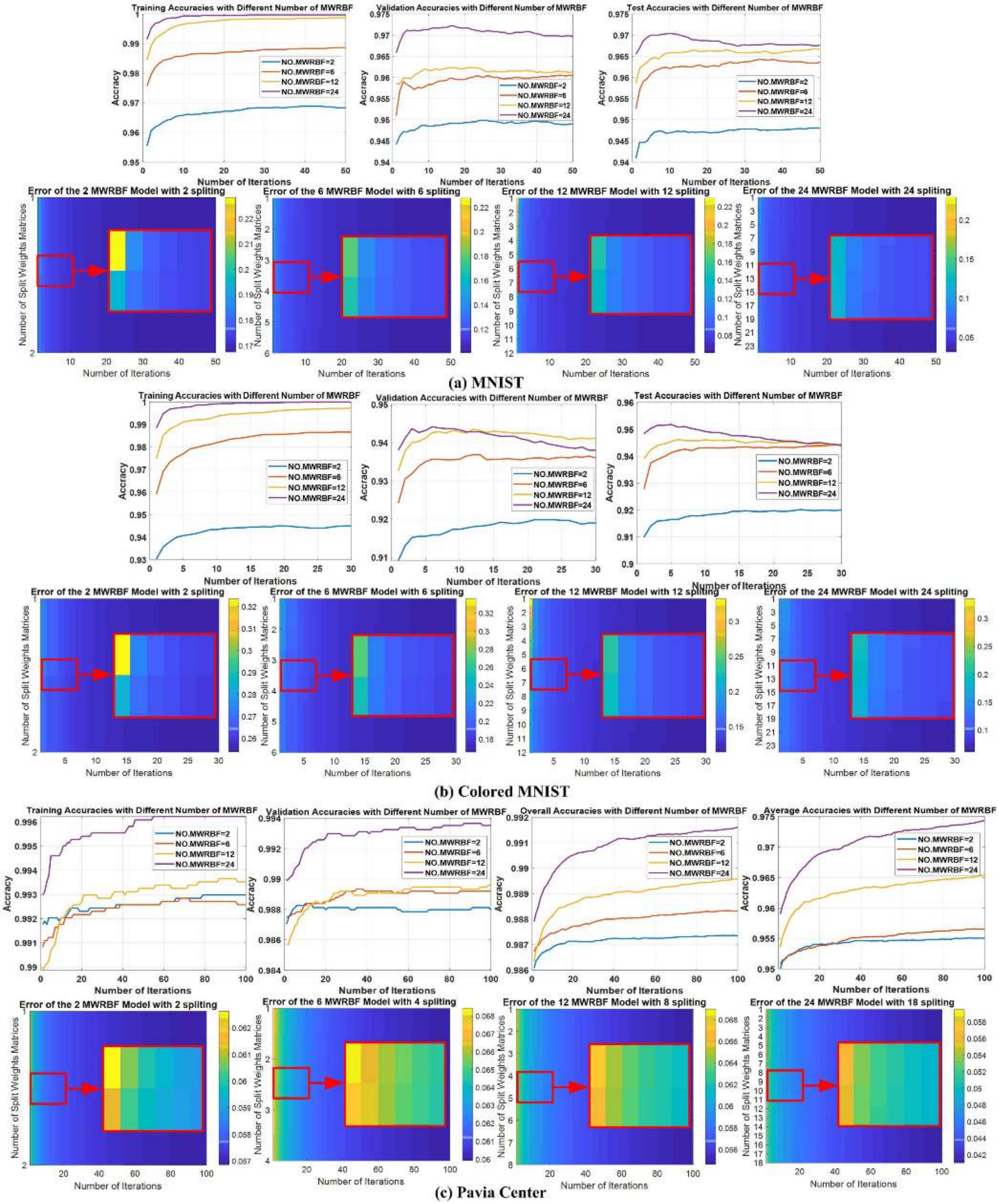
For datasets 3), 4), 5), and 6) in the following experiments, normalization was performed for each band. The spacial (pixels in a 3 × 3 window for each centered pixel) and spectral features were used for training and testing. Recently, different band selection methods were proposed such as using end-to-end framework [23], unsupervised network with dual-attention [48], which are very effective to reduce the bands. The effective organization and use of spatial and spectral features are very important. The hybrid spectral CNN (HybridSN) [49], and the 3D deep learning framework [50] were proposed to use these information effectively. These methods has excellent performance on hyperspectral classification. But, in the experiments, to compare the proposed method with different learning models fairly, the number of spectral features was reduced to the same number 15 by using the principal components analysis (PCA), and higher than 99% spectral information was preserved. No additional measures were used to obtain more spatial or spectral features to improve the classification performance. For KSC dataset, a small proportion (0.2) of the total pixels was chosen randomly as training and validation samples because there is a small number of samples for each class, and the remaining 0.6 portion of samples were used as testing samples. For Pavia Center, Pavia University, and Salinas, 0.05 proportion of the total pixels were chosen randomly as the training and validation sets, respectively. The remaining 0.9 portion was the testing set. The overall accuracy (OA), average accuracy (AA), and Kappa coefficient were used to evaluate the performance.

B. SPLITTING ITERATIVE LEARNING ALONG FEATURE DIMENSION

For hard tasks, a large number of MWRBF networks are needed to learn features accurately. Then, the total weight matrix is big, and an efficient method to compute with it is important. As we discussed in Section III, the matrix will be split into small matrices to compute the pseudo inverse quickly and to reduce the demanding computing resources.

Here we discuss splitting iterative least squares (SILS) training SWNN as a function of the number of MWRBFs. The numbers for training and validation samples are both 6000 for MNIST and colored MNIST. These samples were chosen randomly from the whole datasets, respectively. The test samples were the whole testing sets. The window size of MNIST and colored MNIST was 13, and the numbers of hidden units and RBF nodes after subsampling were 1000 and 50, respectively. For Pavia Center, the window size was 103, and the numbers of hidden units, and RBF nodes after subsampling were 800 and 50, respectively.

The results are shown in Fig. 5. The computing process is iterated to improve performance. Because the LS method is used to compute the weights, the errors are reduced a large amount during the first iteration, and then, the training process is completed with a small number of iterations. During the iterations, the validation and testing performance continues to improve at early iterations. Then, they improve less, or even get worse, while the training accuracy is still increasing. This means the model is overfitted. The changes of training errors with different number of splitting in feature space (weights matrices) are also shown in Fig. 5. It can be observed that as the number of iterations increases, the training error of each splitting part continuously decreases, which also demonstrates the effectiveness of splitting iterative learning in feature space.



(a) MNIST

(b) Colored MNIST

(c) Pavia Center

FIGURE 5. Accuracies of training, validation, testing, and training errors with different data. (a) MNIST. (b) Colored MNIST. (c) Pavia Center hyperspectral remote sensing data. The colorbar represents the amount of the training error. It is seen that as the numbers of MWRBFs (or splitting matrices) and iterations increase, the training error continuously decreases along both axes.

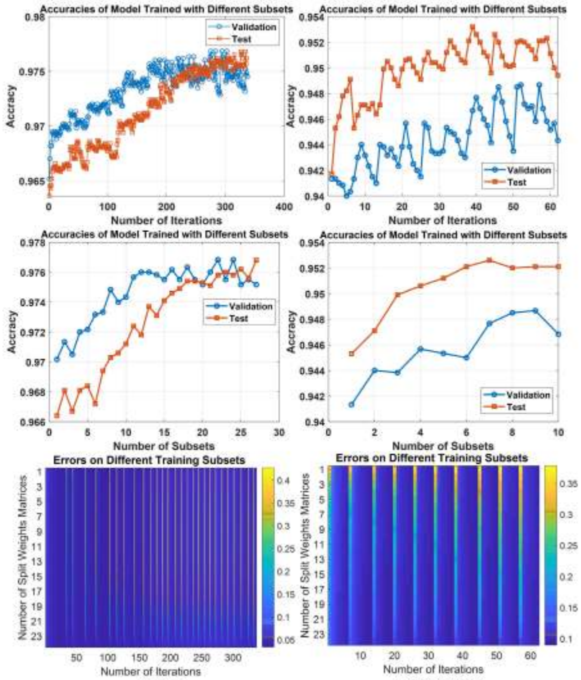
It is also seen that as the number of MWRBF networks increases, the performance of the model is improved, but the number of iterations is reduced. This means the model can reach its best performance with less number of iterations.

C. SPLITTING ITERATIVE LEARNING ALONG SAMPLE DIMENSION

Larger training sets can be split into smaller training sets to train the SWNN effectively. If there are not enough training examples, the augmentation method can be used to generate more training samples to continue training the SWNN. The main problem which may be encountered is catastrophic forgetting if large different training sets are used one by one to continue training the models.

Experiment was implemented to show the effectiveness of the proposed training method when splitting the training set into small subsets. The MNIST and colored MNIST were used in this experiment. The training sets were shuffled and separated into subsets with size 6000 including 600 new samples. The validation set was the last 6000 training samples which were not used during the training process in the experiment. The window size of the models was 13. The SWNN was composed of 24 MWRBF networks. The numbers of hidden units and subsampling were chosen as 3 and 1, respectively.

The experimental results are shown in Fig. 6. It is observed that the trends of validation and testing performance go up, and finally become flatten when the number of iterations increases. This is seen more clearly when showing the testing accuracies at maximums of validation accuracies during iterations for each training subset. Although both validation and testing accuracies drop at the beginning of iterations for each training subset, they become higher than the maximums in the previous training subset, which demonstrates that the training method of splitting along sample dimension is effective, and it can learn new knowledge without forgetting the learned knowledge. In the top sub-figure of Fig. 6 (b), the fluctuated curve pattern occurred when the new subset is used to train the network. That is because the network was learning new knowledge; and the weights trained by the previous subset were not suitable with new subset. Therefore, the validation and test accuracies dropped off at the beginning, and as the training was going on, they began to rise again, and finally, the best performance on the current subset was better than that of the previous subset. This demonstrates that the proposed SWNN can learn new knowledge incrementally without forgetting the learned knowledge. Another observed fluctuation is in the iterations of a subset. That is because during training, the performance of the SWNN improved continuously and then it became overfitting. The SWNN with best trained weights can be saved during the iterations in each subset during the validation process. The changes of training errors are also shown in Fig. 6. It is seen that the training errors decrease with the same subset for each splitting part in sample space (weights matrices) when iterations are performed. When the next training subset is input, the training errors grow up, and then begin decreasing again. This process provides a vision on the ability of incremental learning with iterative splitting in the sample space.



(a) MNIST

(b) Colored MNIST

FIGURE 6. Accuracies and training errors with different training subsets of different data. (a)MNIST. (b) Colored MNIST. The colorbar represents the amount of training error. It is seen that as the numbers of MWRBFs (or Splitting Matrices) and iterations increase, the training error continuously decreases within the training subset, and increases between the training subsets, but the total trend is still decreasing. This demonstrates that the SWNN can learn new knowledge without forgetting old knowledge continuously.

TABLE 1. Testing accuracies of SWNN with different number of MWRBF on gray and colored MNIST datasets.

Method	Test Acc. (%)	Test Acc. (%)
Num. of MWRBF	Gray MNIST data	Colored MNIST data
2	98.34	97.44
6	98.65	98.14
10	98.84	98.30
30	98.91	98.37
60	99.02	98.36
80	98.93	98.36

D. SWNN WITH DIFFERENT NUMBER OF MWRBF NETWORKS

Different number of MWRBF networks is generated for different tasks, so that the model with proper complexity can be found for a specific task.

Experiments were performed first on the MNIST and colored MNIST datasets. The SWNN was tested with 2 MWRBF networks as the minimum setup. The window size of each MWRBF was 13. The number of hidden units was 16, and the subsampling number was 8 for MNIST. The number of hidden units was 20, and the subsampling number was 10 for colored MNIST. The test accuracy is shown in Table 1.

TABLE 2. Testing accuracies of SWNN with different MWRBF neural networks on KSC, Pavia Center, and Pavia University datasets.

Num. of	KSC (%)			Pavia Center (%)			Pavia Univ. (%)		
MWRBF	OA	AA	Kappa	OA	AA	Kappa	OA	AA	Kappa
2	95.01	92.00	94.22	99.41	98.16	99.14	93.78	88.67	91.75
3	95.14	92.34	94.64	98.69	95.58	98.14	93.55	88.77	91.54
6	95.30	92.42	94.68	98.77	95.83	98.25	93.59	88.88	91.56
9	95.33	93.22	94.78	98.86	95.98	98.39	93.72	89.17	91.73
12	95.75	93.51	95.24	98.92	96.25	98.50	94.00	89.56	92.12
16	95.35	93.20	94.89	99.10	96.92	98.71	93.98	89.31	91.95
20	96.18	93.89	95.77	99.03	96.87	98.62	93.88	89.14	91.92
24	95.20	93.10	94.68	99.16	97.35	98.83	94.01	89.25	92.17
36	95.62	93.83	95.14	99.14	96.93	98.79	94.24	89.96	92.46
48	94.95	92.15	94.32	99.23	97.45	98.90	94.27	89.65	92.47

It is observed that with a small number of MWRBF networks, for example, 2, the SWNN has accuracies higher than 98.34% for MNIST and 97.44% for colored MNIST. That is because a single MWRBF already has enough complexity to learn features under current parameter settings. The total numbers of parameters were 40960 and 51200, respectively. As the number of MWRBF networks increases, the testing accuracies continue increasing and finally stops increasing, which means the model is becoming overfitting. The best test accuracy for current parameter settings were 99.02% for MNIST with 60 MWRBF networks. For colored MNIST, it is seen that the performance stops increasing when the accuracy reaches 98.37% with 30 MWRBF networks.

Experiments were also performed with KSC, Pavia Center, and Pavia University datasets. The window size for all the three datasets was 103. The number of hidden units for KSC was 200, and for Pavia Center and Pavia University, it was 800. The number of subsampling outputs was 50 for all three datasets.

The results are shown in Table 2. It is seen that, for KSC dataset, the model is overfitted when the number of MWRBF is 24, and the best OA and AA reach 96.18% and 93.89% for the current parameter settings, respectively. For Pavia Center, the OA and AA continue to increase to 99.23% and 97.45%, respectively. For Pavia University, the OA continues to increase to 94.27%, and the AA reaches 89.65%. When there were 2 MWRBF networks, the proposed method had already good performance on these three datasets, and the total numbers of parameters were 42900, 29700 and 29700, respectively.

According to these experiments, it is observed that, the more MWRBF networks are generated, the higher accuracy can be obtained as a SWNN. However, the accuracy decreases with too many MWRBF networks, which means that the model is overfitted. Therefore, the model can be scalable with an optimal number of MWRBF depending on the difficulty of each application (using the training errors and a given threshold).

TABLE 3. Parameter settings of SWNN on different datasets.

Datasets	Num. of MWRBF	Window size	Num. of hidden units	Num. of RBF nodes
MNIST	60	13	20	10
colored MNIST	96	13	20	10
KSC	48	103	10	5
Pavia Center	48	103	10	5
Pavia Univ.	36	83	10	5
Salinas	48	83	10	5

TABLE 4. Testing accuracies of different learning models on MNIST dataset.

Method	Accuracy (%)
RBF network (10 000 hidden units)	97.41
CNN (6c-2s-12c-2s)	98.63
RBF ensemble (10 000 hidden units, 5 RBF Nets)	97.50
CNN ensemble (6c-2s-12c-2s, 5 CNNs)	98.81
SVM [40]	98.60
MLP [51]	97.39
Random forest [40]	96.8
Deep belief nets [52]	98.87
RBM[53]	96.39
FRBM-STFN [53]	97.55
FRBM-GMF [53]	97.70
Elastic weight consolidation (EWC) [15]	98.2
Broad learning system [8]	98.74
Incremental moment matching (IMM) [16]	98.30
Conditional deep learning [41]	98.92
gc Forest [40]	99.26
SWNN	**99.04**

TABLE 5. Testing accuracies of different learning models on colored MNIST dataset.

Method	Accuracy (%)
MLP (1000 hidden units)	95.81
SAE (4000 hidden units)	95.01
RBF network (10 000 hidden units)	97.22
CNN (6c-2s-12c-2s)	97.71
RBF ensemble (10 000 hidden units, 5 RBF Nets)	97.28
CNN ensemble (6c-2s-12c-2s, 5 CNNs)	98.41
ResNet (8 layers,)	96.99
ResNet (20 layers)	98.32
SWNN	**98.52**

E. CLASSIFICATION PERFORMANCE OF SWNN

The performance of the proposed model was compared with different kinds of models based on the above datasets. The parameter settings of SWNN for these datasets are shown in Table 3. The results are shown in Tables 4, 5, and 6, and Fig. 7.

For MNIST, the SWNN was compared with RBF network with 10000 hidden units, CNN (6c-2s-12c-2s, c represents

TABLE 6. Testing accuracies of different learning models on KSC, Pavia Center, Pavia University, and Salinas datasets.

Method	KSC/ (%)			Pavia Center / (%)			Pavia Univ./ (%)			Salinas/ (%)		
	OA	AA	Kappa	OA	AA	Kappa	OA	AA	Kappa	OA	AA	Kappa
MLP	92.74	87.64	91.95	98.70	94.86	98.16	92.99	88.87	92.23	92.50	96.04	91.75
SAE	93.22	88.15	92.48	94.89	80.68	92.79	83.30	71.41	78.38	92.24	94.14	91.46
RBF	93.41	89.73	92.70	97.75	91.25	97.10	91.34	85.24	88.64	91.83	95.95	91.04
CNN	92.83	89.85	92.18	99.01	96.54	98.60	90.03	86.62	87.02	91.43	95.06	90.59
RBFE	93.44	89.81	92.73	97.95	91.24	97.10	91.39	85.34	88.71	92.06	96.02	91.28
CNNE	95.17	92.83	94.64	99.17	97.22	98.60	91.83	88.71	89.31	91.81	95.45	90.99
SWNN	**97.12**	**94.48**	**96.77**	**99.53**	**98.67**	**99.21**	**97.44**	**96.24**	**95.80**	**95.14**	**97.74**	**94.64**

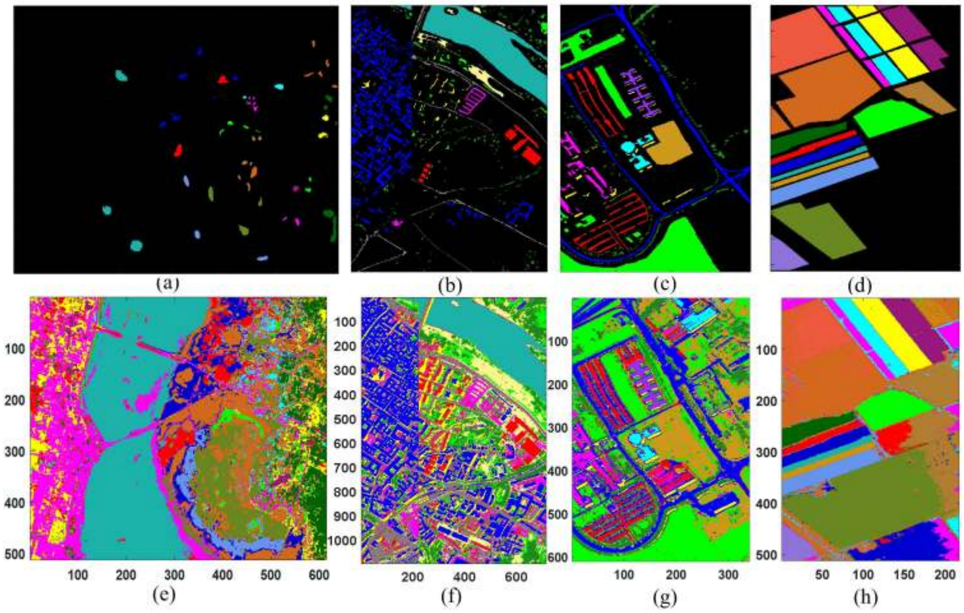


FIGURE 7. Ground truths of hyperspectral datasets: (a) KSC data, (b) Pavia Center data, (c) Pavia University data, and (d) Salinas data; Classification results of SWNN on hyperspectral datasets: (e) KSC data, (f)Pavia Center data, (g) Pavia University data, and (h) Salinas data.

convolutional layer, s represents subsampling or pooling layers, 5 × 5 kernels), RBF ensemble (1000 hidden units, 5 RBF networks), CNN ensemble (6c-2s-12c-2s, 5 CNNs), and also some other methods: SVM [40], multilayer perceptron [51], random forest [40], deep belief net [52], restricted Boltzmann machine (RBM) [53], fuzzy restricted Boltzmann machine (FRBM) [53]. It was also compared with recently proposed scalable and incremental learning methods such as conditional deep learning [41], elastic weight consolidation (EWC) [15], and incremental moment matching (IMM) [16]. The recently proposed different learning models including broad learning system (BLS) [8], and multi-grained cascade forest [40] were also compared. For colored MNIST, the ResNet with 8 layers (6 convolutional layers, 2 skip connection layers, and 1 fully connected layer, 3 × 3 kernels) and 26 layers (18 convolutional layers, 2 skip connection layers, and 1 fully connected layer, 3 × 3 kernels) were also compared under the same usage of data. Other parameter settings of models were the same as in MNIST. For KSC, Pavia Center and Pavia University, MLP with 500 hidden units, RBF with 2000 hidden units, and SAE with 200 and 50 hidden units for encoder and decoder were used to compare with SWNN, and parameter settings of other models were the same as in MNIST.

It is observed that, for MNIST dataset, the proposed SWNN reaches 99.04% accuracy, which is the second best result among the compared models, and a bit lower than the results of gc forest. For colored MNIST dataset, the proposed model has the best accuracy (98.52%) among the compared methods. For KSC, Pavia Center, Pavia University, and Salinas the OAs are 97.12%, 99.53%, 97.44%, and 95.14%, respectively, and the AAs are 94.48%, 98.67%, 96.24%, and 94.64%, respectively. These four groups of results including Kappa coefficients are the best among the compared learning models. It can be seen that the SWNN has stable and competitive performance on classification tasks, and actually, after extending hierarchically as MSWNN, the performance can be further improved.

F. EVALUATION OF PARALLEL TESTING

The parallel implementation of the testing process of the SWNN was evaluated in this experiment. The Pavia Center dataset was chosen considering it has a large number of

TABLE 7. Testing evaluation of parallel implementation with different number of MWRBF networks.

Test type	Number of MWRBF networks						
	2	3	4	5	6	7	8
Original testing time (s)	7.69	11.54	15.23	19.60	25.17	30.10	34.26
Parallel testing time (s)	5.24	5.56	5.73	5.72	5.94	5.94	5.98
Gain	**1.47**	**2.08**	**2.66**	**3.43**	**4.24**	**5.07**	**5.73**

TABLE 8. Testing evaluation of parallel implementation with CNN ensemble and SWNN.

Methods	Testing time original (s)	Testing time parallel (s)	Gain	OA (%)	AA (%)	Kappa (%)
CNN(6c-2s-12c-2s)	20.94	-	-	99.01	96.54	98.60
CNNE(5 CNN, 6c-2s-12c-2s)	75.61	23.17	3.26	99.17	97.22	98.60
SWNN (5 MWRBF)	**15.23**	**5.73**	**3.43**	**99.27**	**97.93**	**98.96**

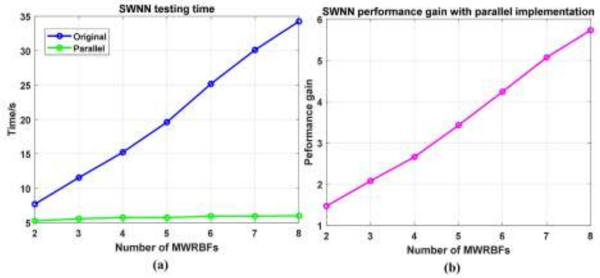


FIGURE 8. Evaluation of the testing process in parallel. (a) Original and parallel testing time of SWNN with different number of MWRBF networks. (b) The performance gain of SWNN with different number of MWRBF networks with parallel implementation of the testing process.

instances to compute the testing time. The testing process of the CNN ensemble with 5 CNNs (6c-2s-12c-2s) was implemented in parallel to compare with that of the SWNN. The hyperparameters of the two models were the same as in Section IV-E. The parallel testing process was run on the work station (CPU with 8 cores) with up to 8 parallel workers. The specific number of workers was the same as the numbers of MWRBFs for the SWNN and CNNs for the CNN ensemble, and each single learning model was run with an independent worker.

First, the testing process of the SWNN with different number of MWRBF networks were implemented in parallel to evaluate the performance gain of computing time. The results are show in Fig. 8, and Table 7. It is seen that as the number of MWRBF increases from 2 to 8, the performance gain of computing time is also increased to 5.73 times.

Secondly, the comparison results of parallel implementation of the CNN ensemble and SWNN are shown in Table 8. The SWNN is composed of 5 MWRBFs. It has a bit better classification performance than that of the CNN ensemble, and its original testing time is less than that of the CNN ensemble. It can be seen in the table that after parallel implementations, the SWNN can run much faster than CNN and CNNE, and it has larger performance gain of computing time than that of the CNN ensemble.

It can be seen from the above results that in spite of the hidden overhead, which includes allocating test data to different workers and gathering computing results from those works, the speeding up of the parallel implementation is close to the theoretical speedup number.

V. DISCUSSION

A. THE RELATIONSHIP OF NUMBERS OF HIDDEN UNITS, RBF NODES, AND MWRBF NETWORKS

There is a tradeoff between the complexity (numbers of hidden units and subsampled outputs) of a single MWRBF network and the number of MWRBFs. Actually, if the single MWRBF network has smaller number of hidden units and subsampled outputs, the training process can be trained more stably, and can get a better performance. Conversely, if the performance is decreased a bit, fewer MWRBFs are needed. This can be seen in Tables 1, 2, compared with the final classification performance in Section VI-E. The better performance is obtained when less number of hidden units and sampled outputs (RBF nodes), but more MWRBF networks are used.

B. ITERATIVE LEARNING ALONG BOTH FEATURE AND SAMPLE DIMENSIONS

In Fig. 5 (c), it is seen that the training process continues for more than 10 iterations, during which time the training, validation, and testing (OA and AA) performances are improved rapidly. When the number of iterations is larger than 20, all the performances improve much more slowly. A possible reason is that the numbers of features and samples of Pavia Center are big, and the model can learn more stably to reach its best performance. This result means that the learning process can be ceased after 20 iterations to save computing time and resources.

In Fig. 6, for each training subset, the validation performance reaches its maximum with even less than 5 iterations. After that, although the training performance is still improved (which can be seen from the changes of training errors in Fig. 6), the validation and testing performances do not improve or begin to drop. The reason is that although the training set is smaller (6000 training samples), the number of parameters of the SWNN is also small, and therefore, the model can learn stably, and reach its best performance quickly. These results mean that this training process can be stopped with only 5 iterations at a very early time point for each training subset.

The splitting along feature and sample dimensions can be stopped in time by using validation process, and because of using least squares, the training of SWNN starts with a very good performance, and can be stopped quickly with few iterations.

VI. CONCLUSION

Recently, researchers have done many works on developing different types of models compared with deep learning models, and they are also working improving the performances of previous learning models. For example, fast training, developing methods that can make models learn incrementally, which means learning new tasks without forgetting knowledge learned from old task. In this paper, a new model SWNN is proposed with scalable property, which is generated incrementally in the wide direction using the proposed MWRBF networks. The proposed model is trained using iterative splitting along feature and sample dimensions together with least squares to reduce the training error quickly. SWNN can deal with large number of instances and large number of features. Experiments were performed on MNIST, colored MNIST, KSC, Pavia Center, and Pavia University datasets, and compared with shallow and deep models. The results show that the proposed SWNN has comparatively excellent performance on classification tasks. The main limitation of the proposed work is that it cannot extract spatial features from images easily, because the image blocks in the sliding windows are flattened into vectors and learned by MWRBF directly. CNN can extract different levels of spatial features with a number of convolutional layers. We are working on solving this problem by (1) learning the spatial features directly for images with two dimensional Gaussian kernels; (2) extending the SWNN with multiple feature extracting layers to make it deeper, but still to be trained quickly.

REFERENCES

[1] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, Aug. 1998.

[2] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.

[3] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3856–3866.

[4] P. Kontschieder, M. Fiterau, A. Criminisi, and S. R. Bulo, "Deep neural decision forests," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1467–1475.

[5] B. Neyshabur, Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro, "The role of over-parametrization in generalization of neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–20.

[6] J. Lee, L. Xiao, S. S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington, "Wide neural networks of any depth evolve as linear models under gradient descent," 2019, *arXiv:1902.06720*. [Online]. Available: http://arxiv.org/abs/1902.06720

[7] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, 2016, pp. 7–10.

[8] C. L. P. Chen and Z. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 10–24, Jan. 2018.

[9] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.

[10] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[11] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3642–3649.

[12] D. Roy, P. Panda, and K. Roy, "Tree-CNN: A hierarchical deep convolutional neural network for incremental learning," *Neural Netw.*, vol. 121, pp. 148–160, Jan. 2020.

[13] S. Abpeykar and M. Ghatee, "An ensemble of RBF neural networks in decision tree structure with knowledge transferring to accelerate multi-classification," *Neural Comput. Appl.*, vol. 31, no. 11, pp. 7131–7151, May 2018.

[14] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Netw.*, vol. 113, pp. 54–71, May 2019.

[15] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, and D. Hassabis, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, 2017.

[16] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4652–4662.

[17] D. Wang and M. Li, "Stochastic configuration networks: Fundamentals and algorithms," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3466–3479, Oct. 2017.

[18] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa, "Learning without memorizing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 5138–5146.

[19] S. Abpeykar, M. Ghatee, and H. Zare, "Ensemble decision forest of RBF networks via hybrid feature clustering approach for high-dimensional data classification," *Comput. Statist. Data Anal.*, vol. 131, pp. 12–36, Mar. 2019.

[20] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1251–1258.

[21] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 116–131.

[22] S. Jia, Z. Lin, M. Xu, Q. Huang, J. Zhou, X. Jia, and Q. Li, "A lightweight convolutional neural network for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, early access, Aug. 19, 2020, doi: 10.1109/TGRS.2020.3014313.

[23] S. K. Roy, S. Chatterjee, S. Bhattacharyya, B. B. Chaudhuri, and J. Platoš, "Lightweight spectral–spatial squeeze-and-excitation residual bag-of-features learning for hyperspectral classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 8, pp. 5277–5290, Aug. 2020.

[24] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, Mar. 1991.

[25] A. O. Hoori and Y. Motai, "Multicolumn RBF network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 4, pp. 766–778, Apr. 2018.

[26] Y. W. Wong, K. P. Seng, and L.-M. Ang, "Radial basis function neural network with incremental learning for face recognition," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 4, pp. 940–949, Aug. 2011.

[27] S. Chen, B. Mulgrew, and P. M. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 570–590, Jul. 1993.

[28] P. H. Zadeh, R. Hosseini, and S. Sra, "Deep-RBF networks revisited: Robust classification with rejection," 2018, *arXiv:1812.03190*. [Online]. Available: http://arxiv.org/abs/1812.03190

[29] A. Hryniowski and A. Wong, "DeepLABNet: End-to-end learning of deep radial basis networks with fully learnable basis functions," 2019, *arXiv:1911.09257*. [Online]. Available: http://arxiv.org/abs/1911.09257

[30] K. Asadi, N. Parikh, R. E. Parr, G. D. Konidaris, and M. L. Littman, "Deep radial-basis value functions for continuous control," 2020, *arXiv:2002.01883*. [Online]. Available: http://arxiv.org/abs/2002.01883

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

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

[33] S. Cho, O. K. Ersoy, and M. R. Lehto, "Parallel, self-organizing, hierarchical neural networks with competitive learning and safe rejection schemes," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 40, no. 9, pp. 556–567, Sep. 1993.

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

[35] S. Venkataramani, A. Raghunathan, J. Liu, and M. Shoaib, ''Scalable-effort classifiers for energy-efficient machine learning,'' in *Proc. 52nd Annu. Design Automat. Conf.*, Jun. 2015, p. 67.

[36] P. Panda, S. Venkataramani, A. Sengupta, A. Raghunathan, and K. Roy, ''Energy-efficient object detection using semantic decomposition,'' *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 9, pp. 2673–2677, Sep. 2017.

[37] D. Roy, P. Panda, and K. Roy, ''Tree-CNN: A hierarchical deep convolutional neural network for incremental learning,'' *Neural Netw.*, vol. 121, pp. 148–160, 2020.

[38] S. Jiang, T. Xu, J. Guo, and J. Zhang, ''Tree-CNN: From generalization to specialization,'' *EURASIP J. Wireless Commun. Netw.*, vol. 2018, no. 1, p. 216, Dec. 2018.

[39] G. Muhammad, M. S. Hossain, and A. Yassine, ''Tree-based deep networks for edge devices,'' *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 2022–2028, Mar. 2020.

[40] Z. Zhou and J. Feng, ''Deep forest,'' 2017, *arXiv:1702.08835*. [Online]. Available: https://arxiv.org/abs/1702.08835

[41] P. Panda, A. Sengupta, and K. Roy, ''Conditional deep learning for energy-efficient and enhanced pattern recognition,'' in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, 2016, pp. 475–480.

[42] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, ''Focal loss for dense object detection,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, Feb. 2020.

[43] B. Kim, H. Kim, K. Kim, S. Kim, and J. Kim, ''Learning not to learn: Training deep neural networks with biased data,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 9012–9020.

[44] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, ''Learning a unified classifier incrementally via rebalancing,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 831–839.

[45] J. Xi, O. K. Ersoy, J. Fang, T. Wu, X. Wei, and C. Zhao, ''Parallel multistage wide neural network,'' Dept. Elect. Comput. Eng., Tech. Rep. 757, 2020.

[46] S. Aghagolzadeh and O. K. Ersoy, ''Optimal adaptive multistage image transform coding,'' *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, no. 4, pp. 308–317, Dec. 1991.

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

[48] Y. Cai, X. Liu, and Z. Cai, ''BS-nets: An end-to-end framework for band selection of hyperspectral image,'' *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 3, pp. 1969–1984, Mar. 2020.

[49] S. K. Roy, G. Krishna, S. R. Dubey, and B. B. Chaudhuri, ''HybridSN: Exploring 3-D–2-D CNN feature hierarchy for hyperspectral image classification,'' *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 2, pp. 277–281, Feb. 2020.

[50] Z. Zhong, J. Li, Z. Luo, and M. Chapman, ''Spectral–spatial residual network for hyperspectral image classification: A 3-D deep learning framework,'' *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 2, pp. 847–858, Feb. 2018.

[51] C. M. Bishop, *Pattern Recognition and Machine Learning* (Information Science and Statistics). New York, NY, USA: Springer-Verlag, 2006.

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

[53] C. L. P. Chen, C.-Y. Zhang, L. Chen, and M. Gan, ''Fuzzy restricted Boltzmann machine for the enhancement of deep learning,'' *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 6, pp. 2163–2173, Dec. 2015.

JIANGBO XI received the 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. From April 2015 to April 2017, he was a joint Ph.D. student in electrical and computer engineering at Purdue University, USA. He is currently an Assistant Professor with the School of Geology Engineering and Geomatics, Chang'an University, China. His current research interests include deep learning, machine learning, and objects detection in optical sequential images.

OKAN K. ERSOY (Fellow, IEEE) is currently a Professor of electrical and computer engineering with Purdue University. He has published approximately 250 articles, two books, and several book chapters in his areas of research. He also holds five patents. 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 is a Fellow of the Optical Society of America.

JIANWU FANG (Member, IEEE) received the Ph.D. degree in signal and information processing (SIP) from the University of Chinese Academy of Sciences, China, in 2015. He is currently an Associate Professor with Laboratory of Traffic Vision Safety (LOTVS), School of Electronic and Control Engineering, Chang'an University, Xi'an, China. He has published many articles on the IEEE Transactions on Industrial Electronics, the IEEE Transactions on Cybernetics, the IEEE Transactions on Circuits and Systems for Video Technology, AAAI, and so on. His research interests include computer vision and pattern recognition.

MING CONG received the Ph.D. degree from the State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing (LIESMARS), Wuhan University, China, in 2015. He is currently a Lecturer with the School of Geological Engineering and Surveying, Chang'an University, Xi'an, China. He has published many articles on remote sensing journals. His research interests include remote sensing image processing and photogrammetry.

XIN WEI received the bachelor's degree in microelectronics from Jilin University, China, in 2014. He is currently pursuing the Ph.D. degree in signal and information processing (SIP) with the University of Chinese Academy of Sciences (UCAS). His research interests include embedded systems, star identification, and machine learning.

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 with the Department of Mathematics and Information Science, Chang'an University. His research interest includes intelligent information extraction from remote sensing images.