Accelerating Convolutional Neural Network-Based Hyperspectral Image Classification by Step Activation Quantization

Shaohui Mei, *Senior Member, IEEE*, Xiaofeng Chen, Yifan Zhang, *Member, IEEE*, Jun Li, *Fellow, IEEE*, and Antonio Plaza, *Fellow, IEEE*

***Abstract*— Convolutional neural networks (CNNs) have achieved excellent feature extraction capabilities in remotely sensed hyperspectral image (HSI) classification. This is due to their ability to learn representative spatial and spectral features. However, it is difficult for conventional computers to classify HSIs quickly enough for practical use in many applications, mainly because of the large number of calculations and parameters needed by deep learning-based methods. Although several weight quantization methods achieved remarkable results in network compression, the network acceleration effect is still not significant because a full exploration of the potential of network acceleration brought by network weight quantization is still absent from the literature. In this article, a new step activation quantization method is proposed to constrain the input of the network layer of the CNN so that the data can be represented by low-bit integers. As a result, floating-point operations can be replaced with integer operations to greatly accelerate the forward (inference) step of the network. Specifically, nonlinear uniform quantization is adopted in this work to restrain the input of the CNN in the forward inference of the step activation quantization layer, and two functions (constant and tanh-like) are used in the backpropagation step to avoid gradient vanishing and noise. Our newly proposed step activation quantization acceleration method is applied to a CNN for HSI with two well-known benchmark data sets and the experimental results demonstrate that the proposed method is very effective in terms of both memory savings and computation acceleration, with only a slight decrease in classification accuracy. Specifically, our method reduces memory requirements in 13.6× and obtains around 10× speedup with regard to the original real-valued network version.**

***Index Terms*— Activation quantization, convolutional neural networks (CNNs), hyperspectral image (HSI) classification, model acceleration and compression, weight quantification.**

Manuscript received February 13, 2020; revised July 28, 2020, December 3, 2020, and January 20, 2021; accepted February 6, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61671383, in part by the Natural Science Foundation of Shaanxi Province under Grant 2018JM6005, and in part by the Fundamental Research Funds for the Central Universities under Grant 3102018AX001. *(Corresponding author: Shaohui Mei.)*

Shaohui Mei, Xiaofeng Chen, and Yifan Zhang are with the School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710129, China (e-mail: meish@nwpu.edu.cn).

Jun Li is with the School of Geography and Planning, Sun Yat-sen University, Guangzhou 510275, China.

Antonio Plaza is with Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura, 10003 Caceres, Spain.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TGRS.2021.3058321.

Digital Object Identifier 10.1109/TGRS.2021.3058321

I. Introduction

HYPERSPECTRAL images (HSIs) contain a wealth of spectral reflectance data collected from tens of hundreds of contiguous spectral bands [1]. With such abundant spectral information, pixels in an HSI can be classified into a number of predefined categories of ground materials. As a result, HSIs have been widely utilized to deal with a variety of earth remote-sensing tasks, such as environmental mapping, climate change, geological research, monitoring of environmental hazards, and so on [2]–[4].

A large number of studies have been conducted in the literature for effective HSI classification. Generally, HSI classification can be implemented in two steps: 1) feature extraction and 2) feature classification. Feature extraction aims to enhance the separability among various classes in HSIs. For example, principal component analysis (PCA) [5] transforms a set of linearly related variables to linearly irrelevant variables through orthogonal transformations, and has been widely applied to HSIs [6], [7]. Manifold learning-based methods have been proposed to explore the manifold embedded in the high-dimensional feature space, such as locally linear embedding [8], Laplacian eigenmap [9], local tangent space alignment [10], and so on. Graph-based discriminant analysis methods extract effective features by preserving the graph-based structure of HSIs while enhancing the separability of different classes, including the graph-based discriminant analysis with spectral similarity (denoted as GDA-SS) [11], the sparse and low-rank graph-based discriminant analysis (SLGDA) [12], the Laplacian regularized collaborative graph-based discriminant analysis (LapCGDA) [13], and so on. Regarding feature classification, the support vector machine (SVM) is one of the most widely used methods for this purpose [14]. It was originally proposed to find the best separation hyperplane in feature space by making the interval between the positive and negative samples maximal on the training set, and has been widely applied to HSI classification [15], [16]. The spatial-contextual information in HSIs has also been considered in SVM-based classification using composite kernels [17] or morphological profiles [18].

Recently, deep learning-based works have attracted a lot of attention in machine learning [19], [20]. In these studies, both feature extraction and feature classification are conducted

by one single framework, and the performance of these two steps can be jointly enhanced in a learning process that often involves the analysis of a large amount of data. As a result, deep learning-based techniques have been widely used for HSI classification [21]–[25]. For example, Hu *et al.* [26] designed a convolutional neural network (CNN) with five layers to classify HSIs in the spectral domain. Mei *et al.* [27] integrated spatial information into CNNs (using the mean and standard variation of pixels in a spatial neighborhood) to improve classification accuracy. Lee and Kwon [28] proposed a spectral-spatial feature-based classification (SSFC) framework that uses dimensionality reduction and deep learning for spectral and spatial feature extraction. Chen *et al.* [29] employed several convolutional and pooling layers to extract nonlinear, discriminative, and invariant features from HSIs, which can be used for image classification and target detection. Ma *et al.* [30] designed an end-to-end deconvolution network (with skip architecture) to learn spectral-spatial features, which can effectively address the problem of information loss introduced by the pooling operation at lower layers.

Although deep learning-based methods have achieved excellent performance on high-performance computing platforms, it is difficult for conventional processors to cope with the expensive computations needed by deep architectures [31], especially for real-time onboard processing on hardware platforms. Therefore, new techniques for deep model compression and acceleration are highly required [32], [33]. In [34], Forrest used small convolution kernels and decreased the number of channels to reduce the parameters of a deep model, and the downsampling operation was also postponed to reduce the network depth. Experiments on AlexNet [35] demonstrated that this technique can reduce network weights around 50 × without significantly compromising classification accuracy. This method reduced the weight parameters by increasing the depth of the network, but did not preserve the parallel nature of the network. As a result, the training time increased significantly (the compression efficiency of 50× was mainly due to the huge weight parameter structure of Alexnet). Li *et al.* [36] pruned some meaningless convolution filters to accelerate the model, and achieved 34% and 38% reductions in inference costs for the VGG-16 [37] and ResNet-110 [38], respectively, using the CIFAR10 data set. However, this method consumes significant time in the network training. Since the redundancy of each CNN is different, it is difficult to determine the pruning scale of the network. As a result, the operability of this method is poor in practice. Courbariaux *et al.* [39] introduced the BinaryNet method to train a deep neural network in which the weights and activations were constrained to 1 or −1, and multiplication was neither used in the convolution operation nor in the fully connected layers. Consequently, the weights required by the network were reduced by 32 times. Rastegari *et al.* [40] proposed two efficient approximations to a standard CNN: binary weight networks (whose filters are approximated with binary values), and XNOR networks (in which both the filter and input to the convolution layers are binary). The corresponding versions of AlexNet [35] achieved excellent results in ImageNet classification. The two aforementioned methods are based on weight quantization, which has strong operability. However, these methods are mainly focused on weight compression and quantization in network design (without considering the loss in network precision caused by such compression of weights and input quantization). In addition, such strategy of using binary inputs and binary weights does not adjust the number of quantization bits, which leads to inevitable information loss.

In this article, we address the aforementioned issues and introduce a new step activation quantization method that constrains the input of the network layer, so that the data can be represented by low-bit integers. This can significantly simplify hardware design for on-board implementations since we can replace floating-point operations with integer operations to greatly accelerate the forward inference of the network. In addition, our method allows balancing the speedup ratio and the classification accuracy of the network (by adjusting the number of quantization bits). Specifically, our step activation quantization mechanism adopts nonlinear uniform quantization to restrain the input of the CNN in the forward inference of the step activation quantization layer, and two functions are utilized in the backpropagation (including a constant function and a tanh-like function) to avoid gradient vanishing and noise.

To illustrate the advantages of our newly developed method, we design a typical CNN for HSIs classification and further replace the convolutional and activation layers of the CNN with binary convolutional layers and step activation quantization layers, respectively. Experiments with two benchmark data sets are carried out to evaluate the performance of our method, evaluating not only the classification accuracy but also the memory usage and the speed of the forward inference step.

The remainder of this article is organized as follows. Section II introduces the proposed method. Section III presents our experimental settings and results. Section IV concludes this article with some remarks and hints at plausible future research lines.

II. Proposed Method

A. Rationale

In the conventional CNN for HSIs classification, most of the computational burden and weight parameters are concentrated in the convolution layers. If the computation operations can be optimized effectively in the convolutional layer, the memory required by weight parameters and calculations will be greatly reduced, so that the CNN can be easily deployed to a computing platform with limited storage and computing resources, such as a digital signal processor (DSP) or a field programmable gate array (FPGA), which are currently the standards for on-board satellite data processing. The operations performed at the convolutional and fully connected layers of the CNN are shown in the following equation:

$$I \otimes W = \sum_{i=1}^{w} \sum_{j=1}^{d} \sum_{k=1}^{h} (I_{i,j,k} \times W_{i,j,k}) \tag{1}$$

where $I \subset \mathbf{R}^{w \times d \times h}$ represents the input to the layer, $\otimes$ denotes the convolution operation, and $W \subset \mathbf{R}^{w \times d \times h}$ represents the kernel in the convolutional layer or the weight in the fully

connected layers (the biases in the fully connected layer are not considered), $\{w, d, h\}$ are the scales of the data block of input I and weight W of network. The parameters in I and W are single-precision floating-point numbers represented by 32 bits.

After analyzing (1), we can infer that there are $w{\times}d{\times}h$ multiplications and $w{\times}d{\times}h$-1 additions in a convolution operation. It is noteworthy that multiplications and additions of 32-bit single-precision floating-point numbers are very time consuming in hardware and resource-intensive. Here, we develop a new method to compress and accelerate CNNs by replacing 32-bit single-precision floating-point numbers with low-bit numbers. Our goal is to use (2) and (3) to replace (1)

$$I = M\alpha, \quad W = N\beta, \tag{2}$$

$$I \otimes W = \left(\sum_{i=1}^{w} \sum_{j=1}^{d} \sum_{k=1}^{h} (M_{i,j,k} \times N_{i,j,k}) \right) \times \alpha\beta \tag{3}$$

where α and β are 32-bit single-precision floating-point numbers, and M and N are low-bit integers with size $w \times d \times h$. In this context, one convolution operation includes two 32-bit single-precision floating-point multiplication operations, while the others are all low-bit integer operations. In this article, a novel step activation quantization method is proposed to constrain the input to convolutional layers and then the weight quantization strategy in [40] is used to pruning convolutional layers. These modifications based on compressing the weights and adopting activation quantization can greatly reduce the memory required by each weight parameter, which can significantly compress the CNN effectively and improve its operational speed.

B. Weight Quantization

In [40], a new method was proposed to restrain the weight parameters of a CNN to binary form. The formula adopted for the convolutional layer is the same as (1). Equation (4) shows the convolution operation process with binary weight parameters

$$I \otimes W \approx (I \oplus W)\alpha \tag{4}$$

where $\oplus$ indicates a convolution without any multiplication. By defining $B \in \{+1, -1\}^{w\times d\times h}$, and a scaling factor $\alpha \subset \mathbf{R}^{+}$ to approximate the real-value weight W, it can be derived that $W \approx \alpha B$. Since the weights are binary, the convolution operation can be replaced by additions and subtractions. Therefore, the convolution operation in (1) can be approximated by (4).

The optimal estimation for $W \approx \alpha B$ can be obtained by solving the following problem:

$$\alpha^*, B^* = \underset{\alpha, B}{\operatorname{argmin}} J(B, \alpha) \tag{5}$$

where the objective function $J(B, \alpha)$ is defined as

$$\begin{aligned} J(B, \alpha) &= \| W - \alpha B \|^2 \\ &= \alpha^2 B^{\mathrm{T}} B - 2\alpha W^{\mathrm{T}} B + W^{\mathrm{T}} W. \end{aligned} \tag{6}$$

Such problem can be simply solved by assigning $B_{i,j,k} = +1$ when $W_{i,j,k} \geq 0$, and $B_i, j, k = -1$, when $W_{i,j,k} < 0$. The optimal solution for B is then

$$B^* = \operatorname{sign}(W). \tag{7}$$

To find the optimal value for the scaling factor α^*, the derivative of J with respect to α is calculated and set to zero value

$$\frac{\partial J}{\partial \alpha} = 2n\alpha - 2W^{\mathrm{T}} B = 0 \tag{8}$$

to achieve

$$\alpha^* = \frac{W^{\mathrm{T}} B^*}{n}. \tag{9}$$

By replacing B^* with sign(W), the following expression is derived:

$$\alpha^* = \frac{W^{\mathrm{T}} \operatorname{sign}(W)}{n} = \frac{\sum |\, W_i \,|}{n} = \frac{1}{n} \| W \|_{l1} \tag{10}$$

where $\| \cdot \|_{l1}$ represents ℓ_1 norm of a matrix.

C. Activation Quantification

In this section, a new activation quantization method, called step activation quantification (SAQ), is proposed to restrain the data input to the convolutional layers and fully connected layers, so that low-bit numbers are employed for representation rather than 32-bit single-precision floating-point numbers. Furthermore, a new activation layer (powered by the proposed SAQ layer) is designed. Our newly proposed method will be elaborated from the following two viewpoints: forward inference and backpropagation.

1) Forward Inference: The schematic of SAQ is given in Fig. 1. Nonlinear uniform quantization is adopted to restrain the input of the CNN, as shown in the following equation:

$$Q(x) = \begin{cases} ns, & ns \leq x < (n+1)s \\ \ldots, & \ldots\ldots \\ s, & s \leq x < 2s \\ 0, & 0 \leq x < s \\ 0, & -s \leq x < 0 \\ -s, & -2s \leq x < -s \\ \ldots, & \ldots\ldots \\ -ns, & -(n+1)s \leq x < -ns \end{cases} \tag{11}$$

where x is an element of I, and s is the quantization step, which can be used to adjust the quantization error. I is converted to $Q(I)$ by SAQ in (12) with $M \in \mathbf{Z}^{w\times d\times h}$, so that floating point numbers are replaced with low-bit integers in the convolutional operation

$$Q(I) = M \times s. \tag{12}$$

By doing this, the cost (in terms of processing time and resources) in the convolution operation can be greatly decreased. Sign function is a special form of step-activated quantization function; when the quantization order is 2, Q(x) is transformed into the form of Sign(x). Recently, full-wave activation functions such as Sigmoid and Tanh are less widely used in deep neural networks. On the contrary, half-wave

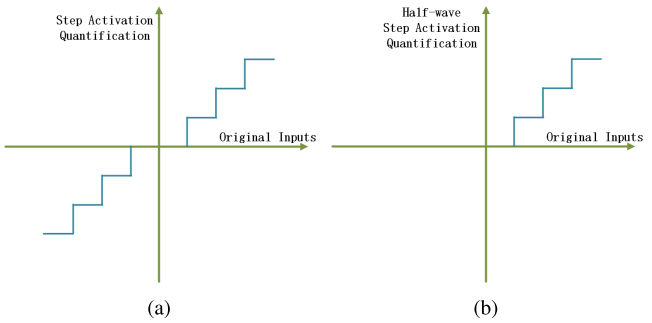


Fig. 1. (a) Step activation quantization. (b) Half-wave step activation quantization.

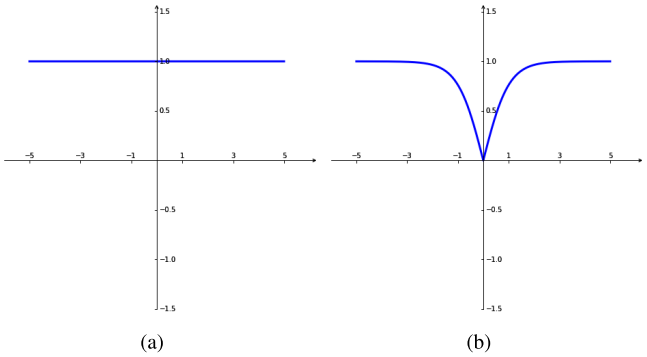


Fig. 2. Functions employed as the matching gradient functions for step quantization. (a) Constant function. (b) Tanh-like function.

activation functions (such as ReLU [41] and PReLU [41]) are more frequently used.

The quantization error is shown in the following equation:

$$E(x)^2 = (Q(x) - x)^2 \leq s^2 \tag{13}$$

$$E(I) = \sum_{i=1}^{h}\sum_{j=1}^{d}\sum_{k=1}^{w}(Q(I_{i,j,k}) - I_{i,j,k})^2 \leq wdh \times s^2 \tag{14}$$

where x and $I_{i,j,k}$ denote an element of the input vector. It can be clearly seen that, with the increase of quantization order, both *E(x)* and *E(I)* approach to zero. In the training phase, we can achieve a balance between the computation cost and the prediction accuracy of the network by simply adjusting the quantization order of the network input.

2) Backpropagation Approximation: One important observation resulting from the previous step is that the gradient of the step quantization function is almost zero everywhere. As a result, in the backpropagation process, the gradient will disappear. To ensure that the parameters of all network layers are updated in each iteration, the constant and tanh-like functions (shown in Fig. 2) are employed as the matching gradient functions for step quantization.

With the proposed constant function, as shown in (15), its gradient is always 1 for all inputs, and its corresponding forward inference function is shown in (16). The error between $Q_(x)$ and x is shown in (17). It is obvious that there is a slight gradient mismatch between the forward inference and backward propagation steps, which can be decreased by adjusting the quantization order of the activation layer

$$\widetilde{Q}_1'(x) = 1 \tag{15}$$

$$\widetilde{Q}_1(x) = x \tag{16}$$

$$|Q(x) - x| \leq s. \tag{17}$$

The tanh-like gradient function is shown in (18). As it can be seen, the gradient is almost zero when the input is small, and the gradient gets close to 1 with increasing input values. Compared with (15), the tanh-like function can effectively reduce the interference of input noise. The higher the $|x|$, the closer the tanh-like function gets to a constant function

$$\widetilde{Q}_2'(x) = \begin{cases} \dfrac{e^x - e^{-x}}{e^x + e^{-x}}, & 0 \leq x \\ \dfrac{e^{-x} - e^x}{e^x + e^{-x}}, & x < 0. \end{cases} \tag{18}$$

D. Calculation and Algorithmic Implementation

Our newly proposed SAQ leads to the design of new convolution and activation layers. When it is used for hyperspectral classification, a typical CNN is first required to be constructed and trained to achieve optimal classification accuracy. Then, according to the weight and input-scale distribution of the network, the quantization order and quantization bit number of the network are determined and the binary convolution layers and SAQ layers are used to replace the corresponding convolution layers and activation layers. Finally, the quantization order and quantization bit number of the network in the training phase is further adjusted to achieve best performance. In summary, a pseudocode of our new algorithm is given in Algorithm 1.

Algorithm 1

Require:

CNN used for HSI classification.

Binary convolutional layer, powered by a weight binarization method.

Step activation quantization layer.

Do:

1: Train the CNN to achieve the best classification performance.
2: Replace the related convolutional layers and activation layers in the CNN with binary convolutional layers and step activation quantification layers, respectively.
3: Adjust the parameters of the step activation quantification layer, which include quantization orders and quantization range.
4: Train the new CNN to achieve the best classification results.
5: Analyze and compare the experimental data and the results obtained in steps 1 and 4.
6: Repeat steps 3-5 to achieve the best classification performance.

III. Experimental Results

In this section, a CNN is adopted for HSI classification to illustrate the reliability of the proposed method. Two

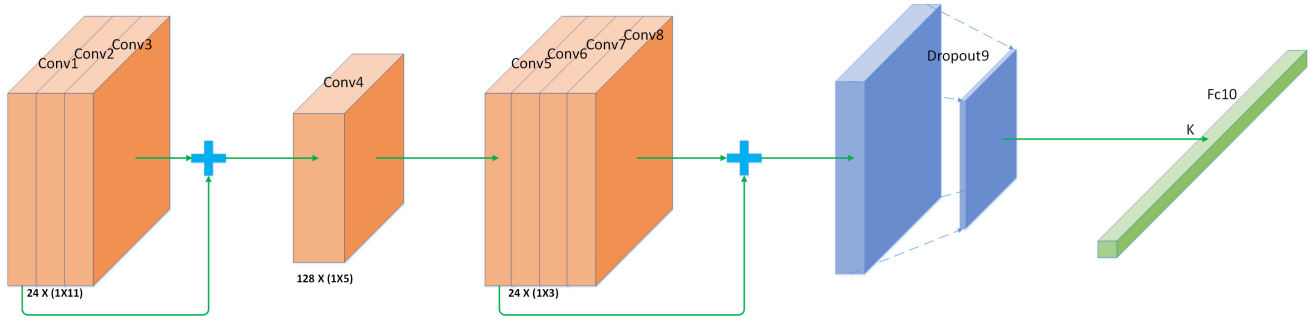


Fig. 3. Structure of the CNN adopted for HSI classification.

data sets collected by different hyperspectral sensors with different resolutions and land-cover types are used to evaluate the effectiveness of the proposed quantification strategy for HSI classification. The network structure and data sets are introduced in Section III-A. Then, the experimental settings for our network and the compared networks are presented and discussed. Section III-E reports various experimental results in terms of classification accuracy, memory usage, computational time, and visual interpretation.

A. CNN for HSI Classification

The CNN structure proposed for HSI classification is shown in Fig. 3, where it can be seen that eight convolution layers and a fully connection layer are utilized for spectral feature extraction and classification, respectively. In the convolution layers, skip connections are mainly used to avoid overfitting and accuracy degradation caused by the increase of layers [38]. Each convolutional structure consists of a convolution layer, a batch normalization layer, and an activation layer. The parameters of the network layer are shown in detail in Table I. Each layer has 24 filters with a shape of 1×11 for the first three convolutional layers, while the filter shape of the convolutional layers 5–8 is 1×3. Under the premise of ensuring classification accuracy, we increase the depth of the network to illustrate the applicability of the method to the greatest possible extent, and it will be used as the benchmark-CNN model in the experiment.

For network compression, the convolutional layer and ReLu activation layer of the benchmark-CNN are first replaced with binary convolutional layer and step activation layer, respectively, resulting in a new network that step activation network with binary weight (denoted as SAWB-CNN). When the proposed SAQ algorithm is applied to SAWB-CNN, SAWB-C and SAWB-T represent SAWB-CNN with constant and tanh-like functions used as gradient functions in the step activation quantization layer, respectively.

B. Data Sets Description

In this section, we describe two benchmark HSIs that have been adopted for the experiments in this work.

1) The Pavia Centre data set was acquired by the Reflective Optics System Imaging Spectrometer (ROSIS) over Pavia city, Northern Italy, in 2002. It consists of 1096×1096 pixels with 1.3 m spatial resolution and 115 spectral bands covering the 400-1000nm wavelength range. A black region of 381×1096 pixels and 13 channels of noise is removed, resulting in a two-part image. A region with 1096×715 pixels with 102 channels is employed in our experiments. There are nine landcover types in the reference data set, including vegetables, soil, and buildings. The pseudo color and ground-truth maps are shown in Fig. 4. Particularly, for this data set, the spectral resolution is low and the spectral range is narrow, while the spatial resolution is high.
2) The Salinas Valley data set was captured by the Airborne Visible Infrared Imaging Spectrometer (AVIRIS) over Salinas Valley, CA, USA in 1998. This data set contains 512×217 pixels with 224 spectral bands, covering the spectral range from 400 to 2500 nm, with spatial resolution of 3.7 m. Except unknown samples, 16 classes are labeled in the available ground truth, most of them in agricultural fields (such as vegetable fields, vineyards, and bare soil). The pseudocolor and ground-truth maps are shown in Fig. 5. The ground covers are quite homogeneous, covering large areas.

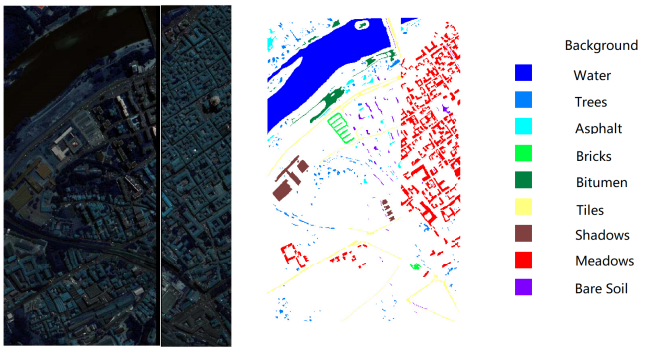


Fig. 4. Pseudo color image and the ground-truth map of the ROSIS Pavia Centre data set.

C. Experimental Settings

In our experiments, the two networks shown in Table I are used for comparison, including the benchmark-CNN and our SAWB-CNN. Several measurements are utilized to provide an objective and quantitative evaluation, including two widely used metrics for HSI classification: overall accuracy

TABLE I
CONFIGURATION AND MEMORY USAGE OF THE CNN ADOPTED FOR HSI CLASSIFICATION

Benchmark-CNN		**SAWB-CNN (SAWB-T and SAWB-C)**		**Kernel Size**
Convolutional layer1	1.03 KB	Convolutional layer1	1.03 KB	24×(1×11)
Batch normalization layer1	0.375 KB	Batch normalization layer1	0.375 KB	24
Relu1	\	SAQ1	\	\
Convolutional layer2	24.75 KB	Binary Convolutional layer2	0.87 KB	24×(1×11)
Batch normalization layer2	0.375 KB	Batch normalization layer2	0.375 KB	24
Relu2	\	SAQ2	\	\
Convolutional layer3	24.75 KB	Binary Convolutional layer3	0.87 KB	24×(1×11)
Batch normalization layer3	0.375 KB	Batch normalization layer3	0.375 KB	24
Relu3	\	SAQ3	\	\
Res_add1	\	Res_add1	\	\
Batch normalization layer r1	0.375 KB	Batch normalization layer r1	0.375 KB	24
Relu r1	\	SAQ r1	\	\
Convolutional layer4	60 KB	Binary Convolutional layer4	2.375 KB	128×(1×5)
Batch normalization layer4	2 KB	Batch normalization layer4	2 KB	128
Relu4	\	SAQ4	\	\
Convolutional layer5	36 KB	Binary Convolutional layer5	1.219 KB	24×(×3)
Batch normalization layer5	0.375 KB	Batch normalization layer5	0.375 KB	24
Relu5	\	SAQ5	\	\
Convolutional layer6	6.75 KB	Binary Convolutional layer6	0.305 KB	24×(1×3)
Batch normalization layer6	0.375 KB	Batch normalization layer6	0.375 KB	24
Relu6	\	SAQ6	\	\
Convolutional layer7	6.75 KB	Binary Convolutional layer7	0.305 KB	24×(1×3)
Batch normalization layer7	0.375 KB	Batch normalization layer7	0.375 KB	24
Relu7	\	SAQ7	\	\
Convolutional layer8	6.75 KB	Binary Convolutional layer8	0.305 KB	24×(1×3)
Batch normalization layer8	0.375 KB	Batch normalization layer8	0.375 KB	24
Relu8	\	SAQ8	\	\
Res_add2	\	Res_add2	\	\
Batch normalization layer r2	0.375 KB	Batch normalization layer r2	0.375 KB	24
Relu r2	\	SAQ r2	\	\
Dropout layer9	\	Dropout layer9	\	0.5
Fully connected layer10	\	Fully connected layer10	\	9\16
Overall	**172.16 KB**	\	**12.65 KB(13.6×)**	\

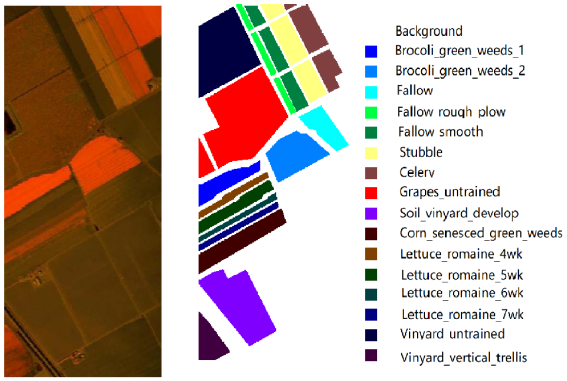


Fig. 5. Pseudo color image and the ground-truth map of the AVIRIS Salinas Valley data set.

(OA, representing the percentage of samples which are classified correctly) and average accuracy (AA, representing the averaged OA of each class). For each class in a certain scene, training samples are randomly selected, and the remaining samples are used for testing. To avoid the bias caused by different training samples, all results are averaged over ten Monte Carlo runs, with randomly selected training samples.

The configuration of the hyperparameters of the two networks considered in experiments are the same (displayed in Table II). ADAM algorithm is selected as the optimizer, which is capable of adaptively adjusting the learning rate. The learning rate is initialized as 0.01, beta1 as 0.99, beta2 as 0.999, and epsilon as 10^{-8}. The batch size is set to 64, and the loss function is categorical cross entropy.

In the sample CNN used for classification, the input to the network is selected to be the mean of the 9 × 9-pixel neighborhood around each pixel to consider spatial-contextual information, as in [27]. As a result, the dimensionality is set to 102 × 1 for the Pavia Centre data set, and to 204 × 1 for the Salinas Valley data set, respectively.

TABLE II
HYPERPARAMETER CONFIGURATION OF THE CONSIDERED NETWORKS

Hyperparameter	Configure
Batch Size	64
Loss function	Categorical crossentropy
Optimizer	ADAM

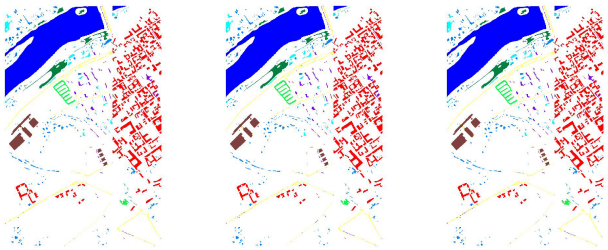
In our experiments, the quantization order of the forward inference of step activation quantization layer is set to 4, and 2 bits are needed to represent a parameter.

D. Experimental Results and Analysis

1) Classification Accuracy: The classification results obtained for the two considered data sets are listed in Tables III and IV, respectively. For both data sets, 200 randomly selected labeled samples per class are used for training, and the remaining ones are used for testing.

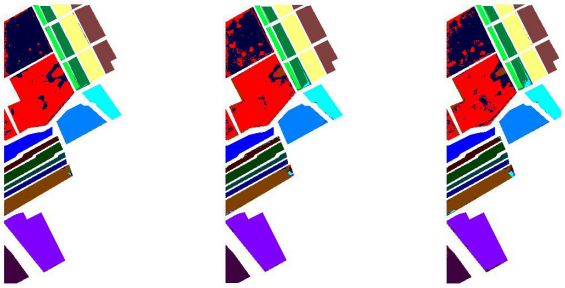
In the Pavia Centre data set, it can be clearly observed that, the AA and OA of the benchmark-CNN and our SAWB-C are almost the same, indicating the strong classification ability of our SAWB-C. Additionally, SAWB-C exhibits better classification accuracy on several classes, such as trees, bitumen, and meadows (99.22%, 98.01%, and 99.15%, respectively). The classification abilities of SAWB-C and SAWB-T are quite different, mainly because of the different gradient functions of the step activation quantization. The classification accuracies achieved by SAWB-T in the classes: water, trees, and meadows are significantly higher than those obtained by the benchmark-CNN and SAWB-C, with accuracies of 99.99%, 99.58%, and 99.36%, respectively. However, due to the poor classification accuracy of SAWB-T in the asphalt class, the AA of SAWB-T is greatly reduced (the classification accuracy in this class is only 83.73%).

In the Salinas Valley data set, the benchmark-CNN exhibits the best accuracy in almost every class (with OA and AA of 99.35% and 98.03%, respectively). Compared with the benchmark-CNN, SAWB-C has a decrease of 2.7% in OA and 2.01% in AA. This is mainly because of the poor classification accuracy in the class: vinyard_untrained, which contains a large number of samples. This is also a remarkable result when compared with bit quantization of deep neural network in other research areas [39], [42]. The classification ability achieved by SAWB-C and SAWB-T in the Salinas Valley data set is also quite different, as already observed in the Pavia Centre data set. The classification accuracies obtained by SAWB-T in the classes: fallow, fallow rough allow, lettree romaine 4 wk, ettree romaine 6 wk, lettuce romaine, and vinyard untrained are significantly higher than those achieved by SAWB-C (with classification accuracies of 99.04%, 96.48%, 97.24%, 98.90%, 99.99%, and 83.26%, respectively), and the classification accuracies obtained in the other classes are slightly lower than those achieved by SAWB-C. Generally, many parameters with small scale are involved in a well-trained CNN. Since SAWB-T cannot updated these small-scale parameters effectively, its performance is generally poor than the SAWB-C in both Pavia and Salinas data sets.



(a) (b) (c)

Fig. 6. Classification maps generated by different networks for the Pavia Centre Data Set. (a) Benchmark-CNN. (b) SAWB-C. (c) SAWB-T.



(a) (b) (c)

Fig. 7. Classification maps generated by different networks for the Salinas Valley Data Set. (a) Benchmark-CNN. (b) SAWB-C. (c) SAWB-T.

The corresponding classification maps obtained by the two networks on the two considered data sets are shown in Figs. 6 and 7, respectively. It is also confirmed that SAWB-CNN (including SAWB-C and SAWB-T) achieves a better approximation of the sample CNN for HSI classification purposes. In general, we can conclude that the SAQ method achieves excellent results in the HSI classification task, making the network performance very robust in terms of classification.

2) Memory Usage: For the Pavia Centre data, the input to the benchmark-CNN is selected to be the mean of a 9 $\times$ 9-pixel neighborhood around each pixel to consider spatial-contextual information as in [27]. This results in dimensionality of 102 $\times$ 1, and the number of neurons of the fully connected layer for classification is equal to the number of land-cover types. For Salinas Valley data set, the input to the benchmark-CNN is set to 204 $\times$ 1, which is also stems from averaging pixels in a 9 $\times$ 9 neighborhood. For the output, the number of neurons in the fully connected layer is set to 16 to classify the 16 types of ground objects in this scene. The other parameters of the network are the same for the two data sets.

The memory usage of the sample CNN for the benchmark-CNN (as well as SAWB-CNN) are listed in Table I. It can be clearly observed that the memory usage of the benchmark-CNN is 13.6 $\times$ more than that of SAWB-CNN. As shown in Table I, the network parameters are mainly concentrated in the convolutional layers and batch normalization layers, and the convolutional layers take up more than 80% of the weight parameters of the whole of network.

Taking convolutional layer 2 as an example to illustrate the process of weight compression, there are 24 filters with

TABLE III
CLASS-SPECIFIC ACCURACIES, OA AND AA ON THE PAVIA CENTRE DATA SET

Class	Label	Training	Testing	Benchmark-CNN	SAWB-C	SAWB-T
Water	1	200	65771	99.99	99.99	99.99
Trees	2	200	7398	98.00	99.22	99.58
Asphalt	3	200	2890	98.38	93.27	83.73
Bricks	4	200	2485	99.99	99.03	97.42
Bitumen	5	200	6384	97.59	98.01	97.73
Tiles	6	200	9048	98.53	96.99	98.15
Shadows	7	200	7087	99.44	97.50	95.67
Meadows	8	200	42626	99.13	99.15	99.36
Bare Soil	9	200	2663	97.82	97.41	97.30
AA(%)	/	/	/	**98.76**	**97.84 (↓0.92)**	**96.55 (↓2.21)**
OA(%)	/	/	/	**99.35**	**99.12 (↓0.23)**	**98.95 (↓0.4)**

TABLE IV
CLASS-SPECIFIC ACCURACIES, OA, AND AA OVER THE SALINAS VALLEY DATA SET

Class	Label	Training	Testing	Benchmark-CNN	SAWB-C	SAWB-T
Brocoli_green_weeds_1	1	200	1809	99.99	99.94	99.83
Brocoli_green_weeds_2	2	200	3526	99.89	99.57	99.09
Fallow	3	200	1776	99.66	98.48	99.04
Fallow_rough_plow	4	200	1194	94.14	94.56	96.48
Fallow_smooth	5	200	2478	99.72	93.99	89.31
Stubble	6	200	3759	99.20	99.99	99.79
Celery	7	200	3379	99.85	99.99	99.79
Grapes_untrained	8	200	11071	88.60	88.87	82.11
Soil_vinyard_develop	9	200	6003	99.52	99.07	98.18
Corn_senesced_green_weeds	10	200	3078	97.01	93.34	92.76
Lettuce_romaine_4wk	11	200	868	98.50	94.35	97.24
Lettuce_romaine_5wk	12	200	1727	99.65	98.78	98.90
Lettuce_romaine_6wk	13	200	716	99.86	99.86	99.99
Lettuce_romaine_7wk	14	200	870	99.99	98.16	97.70
Vinyard_untrained	15	200	7068	92.87	77.89	83.26
Vinyard_vertical_trellis	16	200	1607	99.99	99.94	98.20
AA(%)	/	/	/	**98.03**	**96.02 (↓2.01)**	**95.73 (↓2.3)**
OA(%)	/	/	/	**96.02**	**93.32 (↓2.7)**	**92.22 (↓3.8)**

a shape of 1×11, and the number of channels of each filter is 24. We can utilize a 32-bit single-precision floating-point number and 264 binary numbers to represent all the weight parameters in a filter by adopting our weight binarization method. Compared with the benchmark-CNN, SAWB-CNN can achieve $28.5\times$ reduction in terms of memory usage of the convolutional layer 2.

Generally, SAWB-CNN can achieve $28.5\times$ and $13.6\times$ compression compared to the benchmark-CNN in the memory usage for the convolutional layer and the whole network, respectively. This is an excellent compression efficiency, particularly if we take into account that the classification ability of network remains practically the same.

3) Network Forward Inference Speed in the CPU: The processing times of each network layer for the benchmark-CNN and SAWB-CNN over the two considered data sets are given in Table V, in which the time unit is the number of clock cycles that the CPU needs to perform a 32-bit single-precision floating-point operation once. It is noted that the network forward inference speed of SAWB-T and SAWB-C is the same since they only differ in backward step. The difference of processing times between the two data sets is caused by the difference of input data sizes ($1 \times 1 \times 102$ and $1 \times 1 \times 204$ for the Pavia and Salinas data sets, respectively).

As shown in Table V, the running time of benchmark-CNN is nearly ten times more than that of SAWB-CNN. SAQ accelerates the entire network by specifically accelerating the convolution layers with the largest number of parameters, and also, the mathematical operations in the whole network.

Let us take a complete convolutional structure (including convolutional layer 2, batch normalization layer 2, and ReLu

TABLE V

PROCESSING TIME OF THE BENCHMARK-CNN AND SAWB-CNN OVER THE PAVIA CENTRE AND SALINAS VALLEY DATA SETS. THE TIME UNIT IS THE NUMBER OF CLOCK CYCLES THAT THE CPU NEEDS TO PERFORM A 32-BIT SINGLE-PRECISION FLOATING-POINT OPERATION ONCE

Benchmark-CNN			**SAWB-CNN (SAWB-T and SAWB-C)**		
Layer name	**Pavia**	**Salinas**	**Layer name**	**Pavia**	**Salinas**
Convolutional layer1	46368	97776	Convolutional layer1	46368	97776
Batch normalization layer1	8832	18624	Batch normalization layer1	8832	18624
Relu1	2208	4656	SAQ1	4416	9312
Convolutional layer2	1163616	2453712	Binary Convolutional layer2	77280	162960
Batch normalization layer2	8832	18624	Batch normalization layer2	8832	18624
Relu2	2208	4656	SAQ2	4416	9312
Convolutional layer3	1163616	2453712	Binary Convolutional layer3	77280	162960
Batch normalization layer3	8832	18624	Batch normalization layer3	8832	18624
Relu3	2208	4656	SAQ3	4416	9312
Res_add1	2208	4656	Res_add1	2208	4656
Batch normalization layer r1	8832	18624	Batch normalization layer r1	8832	18624
Relu r1	2208	4656	SAQ r1	4416	9312
Convolutional layer4	2753280	5873664	Binary Convolutional layer4	195840	417792
Batch normalization layer4	46080	98304	Batch normalization layer4	46080	98304
Relu4	11520	24576	SAQ4	23040	49152
Convolutional layer5	1656720	3534336	Binary Convolutional layer5	105840	225792
Batch normalization layer5	8640	18432	Batch normalization layer5	8640	18432
Relu5	2160	4608	SAQ5	4320	9216
Convolutional layer6	308880	658944	Binary Convolutional layer6	23760	50688
Batch normalization layer6	8640	18432	Batch normalization layer6	8640	18432
Relu6	2160	4608	SAQ6	4320	9216
Convolutional layer7	308880	658944	Binary Convolutional layer7	23760	50688
Batch normalization layer7	8640	18432	Batch normalization layer7	8640	18432
Relu7	2160	4608	SAQ7	4320	9216
Convolutional layer8	308880	658944	Binary Convolutional layer8	23760	50688
Batch normalization layer8	8640	18432	Batch normalization layer8	8640	18432
Relu8	2160	4608	SAQ8	4320	9216
Res_add2	2160	4608	Res_add2	2160	4608
Batch normalization layer r2	8640	18432	Batch normalization layer r2	8640	18432
Relu r2	2160	4608	SAQ r2	4320	9216
Dropout layer9	\	\	Dropout layer9	\	\
Fully connected layer10	19431	73712	Fully connected layer10	19431	73712
Overall	**7889799**	**16804208**	\	**784599(10×)**	**1699760(9.9×)**

layer 2) as an example to illustrate how we can accelerate the processing of Salinas Valley data set. For the convolutional layer 2, the shape of the input and output data is $24 \times 1 \times 194$, and the shape of the filter is $24 \times 1 \times 11$, which means there are 1 229 184 multiplications and 1 224 528 additions. The benchmark-CNN requires 2 453 712 clock cycles while, for our SAWB-CNN, the CPU requires $16\times$ 2-bits operations in one clock cycle, which means 162 960 clock cycles. For the batch normalization layer 2, both the benchmark-CNN and SAWB-CNN require 18 624 clock cycles. For the ReLu layer 2, the benchmark-CNN needs 4656 clock cycles and SAWB-CNN needs 9312 clock cycles, because of the 2-bit quantization strategy. It can be seen that SAWB-CNN runs nearly $12.97\times$ faster than the benchmark-CNN in the considered convolutional structure.

Generally, SAWB-CNN can run around nearly ten times faster than the benchmark-CNN in the whole process of the network and the convolutional structure, respectively, because of weight binarization and the introduction of a step activation quantization layer. Once again, these are substantial acceleration factors considering that the classification accuracy remains almost the same and that the whole process described in this work has been conducted on a CPU, without the need to use a graphics processing unit (GPU) accelerator.

E. Discussion

1) Quantification Bit: Generally, more bits used for quantization, better performance will be achieved. We vary the quantization bit in the proposed algorithm from 2 to 4, and thus the quantization order of the forward inference of step activation quantization layer varies from 4 to 16. The constant function is used as the gradient function of activation quantization function. The Salinas Valley data set is used for validation, in which other parameters are consistent with the previous experiments. As shown in Table VI, when more bits are used for quantization, the prediction accuracy slightly increases since the quantization error of the network becomes smaller. However, when only 2 bits are used for quantization, in which the quantization order of the forward inference of step activation quantization layer is set as 4, the performance only

TABLE VI

EXPERIMENTAL RESULTS OF THE PROPOSED SAQ ALGORITHM USING DIFFERENT QUANTIFICATION BITS OVER SALINAS VALLEY DATA SET

Quantification Bit (Quantization Order)	**2 (4)**	**3 (8)**	**4 (16)**
Brocoli_green_weeds_1	99.94	99.94	99.67
Brocoli_green_weeds_2	99.57	99.82	99.94
Fallow	98.48	99.04	98.76
Fallow_rough_plow	94.56	95.89	90.28
Fallow_smooth	93.99	96.41	96.97
Stubble	99.99	100	100
Celery	99.99	99.7	99.85
Grapes_untrained	88.87	86.96	87.75
Soil_vinyard_develop	99.07	99.23	99.47
Corn_senesced_green_weeds	93.34	94.31	95.61
Lettuce_romaine_4wk	94.35	98.85	97.93
Lettuce_romaine_5wk	98.78	99.19	99.94
Lettuce_romaine_6wk	99.86	100	100
Lettuce_romaine_7wk	98.16	97.93	98.74
Vinyard_untrained	77.89	83.05	86.39
Vinyard_vertical_trellis	99.94	100	100
AA(%)	**96.02**	**96.90**	**96.96**
OA(%)	**93.32**	**93.96**	**94.62**
Memory Usage(KB)	12.6617	12.6734	12.6969
Forward Inference Time (Number of Clock Cycles)	1,699,760	1,766,000	1,832,240

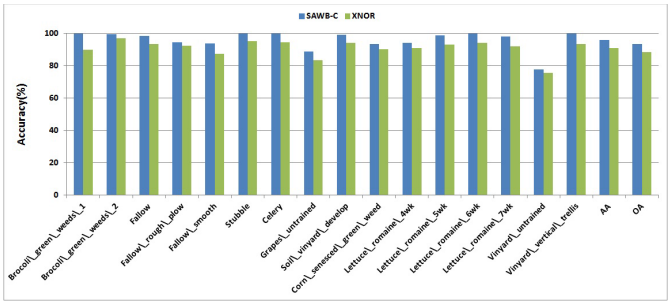


Fig. 8. Comparison with the XNOR-network in [40] over Salinas Valley Data Set.

drops a little. As a result, in practical application, the quantization bit in the proposed SAQ algorithm can be adjusted according to the computational resources of targeted platform to balance the quantization error and prediction accuracy.

The memory usage and forward inference time when different quantization bits are used are also considered as shown in Table VI. It is observed that the memory usage slightly increases when more quantization bits are used. This is because all the SAQ layers share the same parameters and these parameters only take a small part of whole parameters involved in CNNs. It is also observed that the forward inference time slightly increases when more quantization bits are used since more calculation are required for more quantization bits. However, the increase is not large since activation layers do not involve too much calculation in the whole network calculation.

2) Comparative Experiment With XNOR *Network:* In this section, the XNOR strategy is also applied to the sample CNN in Fig. 3 for hyperspectral classification. The comparison results of the proposed SAQ algorithm over the XNOR-network on Salinas Valley data set are shown in Fig. 8. Obviously, the classification accuracy of SAQ algorithm, in terms of class-specific performance, OA and AA, is significantly higher than that of XNOR net proposed in [40]. The inference time of XNOR-net and the proposed SAQ algorithm is also considered. For the Salinas Valley data set, XOR-net consumes about 1 567 280 Clock Cycles while the proposed SAQ algorithm spends slightly more calculations, which is about 1 699 760 Clock Cycles. Actually, the XNOR network quantified the half-wave activation function such as ReLU [41]. Its quantization error is large, resulting in poor prediction performance. However, the SAQ algorithm significantly improves the prediction accuracy of the network by increasing the quantization bit width by 1 bit. When more bits are used, better performance will also be achieved as previously discussed.

IV. CONCLUSION AND FUTURE WORK

In this article we have developed a new method that allows, for the first time in the HSI classification literature, to significantly reduce the memory requirements and processing speed of deep learning architectures. This has significant implications in terms of hardware implementation of established classifiers, such as the CNN, for on-board HSI data exploitation. Specifically, a new quantization method based on an activation layer is proposed, which can be combined with a weight binarization method to constrain the network layers with binary weights and low-bit inputs. Such quantization strategy is applied in this work to a traditional CNN for HSI classification, obtaining around 13.6× reduction in terms of memory saving, and 10× reduction in terms of computing time on a CPU (without using a GPU) platform, with the classification accuracy almost unchanged.

Since the proposed the quantization bit in the proposed SAQ algorithm can be adjusted according to the computational resources of targeted platform to balance the quantization error and prediction accuracy, in future, we will extended to different on-chip processors by considering their onboard computational resources, such as FPGA, DSP, and so on. In addition, combining the proposed work with different weight quantification strategies will also be considered for network compression and acceleration.

REFERENCES

[1] X. Xu, J. Li, C. Wu, and A. Plaza, "Regional clustering-based spatial preprocessing for hyperspectral unmixing," *Remote Sens. Environ.*, vol. 204, pp. 333–346, Jan. 2018.

[2] L. He, J. Li, C. Liu, and S. Li, "Recent advances on spectral–spatial hyperspectral image classification: An overview and new guidelines," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 3, pp. 1579–1597, Mar. 2018.

[3] X. Xu, J. Li, and S. Li, "Multiview intensity-based active learning for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 2, pp. 669–680, Feb. 2018.

[4] W. Wei, L. Zhang, Y. Jiao, C. Tian, C. Wang, and Y. Zhang, "Intracluster structured low-rank matrix analysis method for hyperspectral denoising," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 866–880, Feb. 2019.

[5] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, Jul. 1997.

[6] X. Jia and J. A. Richards, "Segmented principal components transformation for efficient hyperspectral remote-sensing image display and classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 37, no. 1, pp. 538–542, Jan. 1999.

[7] Y. Ren, L. Liao, S. J. Maybank, Y. Zhang, and X. Liu, "Hyperspectral image spectral-spatial feature extraction via tensor principal component analysis," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 9, pp. 1431–1435, Sep. 2017.

[8] G. Chen, "Dimensionality reduction of hyperspectral imagery using improved locally linear embedding," *J. Appl. Remote Sens.*, vol. 1, no. 1, Mar. 2007, Art. no. 013509.

[9] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. Int. Conf. Neural Inf. Process. Syst., Natural Synth.*, 2001, pp. 585–591.

[10] Z. Zhang and H. Zha, "Principal manifolds and nonlinear dimensionality reduction via tangent space alignment," *SIAM J. Sci. Comput.*, vol. 26, no. 1, pp. 313–338, Jan. 2004.

[11] F. Feng, W. Li, Q. Du, and B. Zhang, "Dimensionality reduction of hyperspectral image with graph-based discriminant analysis considering spectral similarity," *Remote Sens.*, vol. 9, no. 4, p. 323, Mar. 2017.

[12] W. Li, J. Liu, and Q. Du, "Sparse and low-rank graph for discriminant analysis of hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 7, pp. 4094–4105, Jul. 2016.

[13] W. Li and Q. Du, "Laplacian regularized collaborative graph for discriminant analysis of hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 12, pp. 1–11, 2016.

[14] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002.

[15] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 8, pp. 1778–1790, Aug. 2004.

[16] Y. Bazi and F. Melgani, "Toward an optimal SVM classification system for hyperspectral remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 11, pp. 3374–3385, Nov. 2006.

[17] G. Camps-Valls, L. Gomez-Chova, J. Munoz-Mari, J. Vila-Frances, and J. Calpe-Maravilla, "Composite kernels for hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 3, no. 1, pp. 93–97, Jan. 2006.

[18] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson, "Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 11, pp. 3804–3814, Nov. 2008.

[19] B. Li, Y. Dai, and M. He, "Monocular depth estimation with hierarchical fusion of dilated CNNs and soft-weighted-sum inference," *Pattern Recognit.*, vol. 83, pp. 328–339, Nov. 2018.

[20] X. Song, Y. Dai, and X. Qin, "Deeply supervised depth map super-resolution as novel view synthesis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 8, pp. 2323–2336, Aug. 2019.

[21] W. Li, G. Wu, F. Zhang, and Q. Du, "Hyperspectral image classification using deep pixel-pair features," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 2, pp. 844–853, Feb. 2017.

[22] S. Mei, X. Yuan, J. Ji, Y. Zhang, S. Wan, and Q. Du, "Hyperspectral image spatial super-resolution via 3D full convolutional neural network," *Remote Sens.*, vol. 9, no. 11, p. 1139, Nov. 2017.

[23] M. Zhang, W. Li, and Q. Du, "Diverse region-based CNN for hyperspectral image classification," *IEEE Trans. Image Process.*, vol. 27, no. 6, p. 2623, Jun. 2018.

[24] J. Li *et al.*, "Hyperspectral image super-resolution by band attention through adversarial learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 6, pp. 4304–4318, Jun. 2020.

[25] S. Mei, R. Jiang, X. Li, and Q. Du, "Spatial and spectral joint super-resolution using convolutional neural network," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 7, pp. 4590–4603, Jul. 2020.

[26] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *J. Sensors*, vol. 2015, pp. 1–12, Jan. 2015.

[27] S. Mei, J. Ji, J. Hou, X. Li, and Q. Du, "Learning sensor-specific spatial-spectral features of hyperspectral images via convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 8, pp. 4520–4533, Aug. 2017.

[28] H. Lee and H. Kwon, "Going deeper with contextual CNN for hyperspectral image classification," *IEEE Trans. Image Process.*, vol. 26, no. 10, pp. 4843–4855, Jul. 2016.

[29] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 6232–6251, Oct. 2016.

[30] X. Ma, A. Fu, J. Wang, H. Wang, and B. Yin, "Hyperspectral image classification based on deep deconvolution network with skip architecture," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 8, pp. 4781–4791, Jun. 2018.

[31] T. Liu, S. Xie, Y. Zhang, J. Yu, L. Niu, and W. Sun, "Feature selection and thyroid nodule classification using transfer learning," in *Proc. IEEE Int. Symp. Biomed. Imag.*, Apr. 2017, pp. 1096–1099.

[32] H. Chen, F. Zhang, B. Tang, Q. Yin, and X. Sun, "Slim and efficient neural network design for resource-constrained SAR target recognition," *Remote Sens.*, vol. 10, no. 10, p. 1618, Oct. 2018.

[33] F. Zhang, Y. Liu, Y. Zhou, Q. Yin, and H.-C. Li, "A lossless lightweight CNN design for SAR target recognition," *Remote Sens. Lett.*, vol. 11, no. 5, pp. 485–494, May 2020.

[34] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5 mb model size," 2016, *arXiv:1602.07360*. [Online]. Available: https://arxiv.org/abs/1602.07360

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[36] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," 2016, *arXiv:1608.08710*. [Online]. Available: http://arxiv.org/abs/1608.08710

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[39] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to $+1$ or -1," 2016, *arXiv:1602.02830*. [Online]. Available: http://arxiv.org/abs/1602.02830

[40] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.

[41] B. Xu *et al.*, "Empirical evaluation of rectified activations in convolutional network," 2015, *arXiv:1505.00853*. [Online]. Available: https://arxiv.org/abs/1505.00853

[42] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave Gaussian quantization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5406–5414.

Shaohui Mei (Senior Member, IEEE) received the B.S. degree in electronics and information engineering and the Ph.D. degree in signal and information processing from Northwestern Polytechnical University, Xi'an, China, in 2005 and 2011, respectively.

He is an Associate Professor with the School of Electronics and Information, Northwestern Polytechnical University. He was a Visiting Student at The University of Sydney, Sydney, NSW, Australia, from October 2007 to October 2008. His research interests include hyperspectral remote sensing image processing and applications, intelligent signal and information acquisition and processing, video processing, and pattern recognition.

Dr. Mei serves as an Associate Editor for the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING (JSTARS) and a reviewer for more than 20 international famous academic journals. He was a recipient of the Excellent Doctoral Dissertation Award of Shaanxi Province in 2014, the Best Paper Award of the IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS) 2017, and a Best Reviewer of the IEEE JSTARS in 2019. He also served as the Registration Chair for the IEEE China Summit and International Conference on Signal and Information Processing (ChinaSIP) 2014.

Xiaofeng Chen received the B.S. degree in electronics and information engineering and M.S. degree in signal and information processing from Northwestern Polytechnical University, Xi'an, China, in 2017 and 2020, respectively.

His research interests include neural networks' acceleration and hardware implementation.

Yifan Zhang (Member, IEEE) received the B.S. degree in electron and information technology and the M.S. and Ph.D. degrees in signal and information processing from Northwestern Polytechnical University, Xi'an, China, in 2001, 2004, and 2007, respectively.

From 2007 to 2010, she worked as a Post-Doctoral Researcher with Vision Laboratory, Department of Physics, University of Antwerp, Antwerp, Belgium. She is an Associate Professor with the School of Electronics and Information, Northwestern Polytechnical University. Her research interest includes hyperspectral image analysis, image fusion, and image restoration.

Jun Li (Fellow, IEEE) received the B.S. degree from Hunan Normal University, Changsha, China, in 2004, the M.Sc. degree in remote sensing and photogrammetry from Peking University, Beijing, China, in 2007, and the Ph.D. degree in electrical and computer engineering from the Instituto Superior Tecnico, Technical University of Lisbon, Lisbon, Portugal, in 2011.

From 2011 to 2012, she was a Post-Doctoral Researcher with the Department of Technology of Computers and Communications, University of Extremadura, Badajoz, Spain. She is a Professor with the School of Geography and Planning, Sun Yat-sen University, Guangzhou, China. Since 2013, she has obtained several prestigious funding grants at the national and international level. Her research interests include remotely sensed hyperspectral image analysis, signal processing, supervised/semisupervised learning, and active learning.

Dr. Li is an Editor of the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING and the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING.

Antonio Plaza (Fellow, IEEE) received the M.Sc. and Ph.D. degrees in computer engineering from the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura, Caceres, Spain, in 1999 and 2002, respectively.

He is the Head of the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura. He is one of the top-cited authors in Spain and in the University of Extremadura. His main research interests comprise remotely sensed hyperspectral image analysis, signal processing, and efficient implementations of large-scale scientific problems on high-performance computing architectures, including commodity Beowulf clusters, heterogeneous networks of computers and clouds, and specialized computer architectures, such as field-programmable gate arrays or graphical processing units.

Dr. Plaza was a member of the Editorial Board of the IEEE GEOSCIENCE AND REMOTE SENSING NEWSLETTER from 2011 to 2012 and the *IEEE Geoscience and Remote Sensing Magazine* in 2013. He was also a member of the Steering Committee of the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING. He served as the Director of Education Activities for the IEEE Geoscience and Remote Sensing Society (GRSS) from 2011 to 2012 and the President of the Spanish Chapter for the IEEE GRSS from 2012 to 2016. He served as the Editor-in-Chief for the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING from 2013 to 2017.