LiteDenseNet: A Lightweight Network for Hyperspectral Image Classification

Rui Li, Chenxi Duan, and Shunyi Zheng

***Abstract*—Hyperspectral Image (HSI) classification based on deep learning has been an attractive area in recent years. However, as a kind of data-driven algorithm, deep learning method usually requires numerous computational resources and high-quality labelled dataset, while the cost of high-performance computing and data annotation is expensive. In this paper, to reduce dependence on massive calculation and labelled samples, we propose a lightweight network architecture (LiteDenseNet) based on DenseNet for Hyperspectral Image Classification. Inspired by GoogLeNet and PeleeNet, we design a 3D two-way dense layer to capture the local and global features of the input. As convolution is a computationally intensive operation, we introduce group convolution to decrease calculation cost and parameter size further. Thus, the number of parameters and the consumptions of calculation are observably less than contrapositive deep learning methods, which means LiteDenseNet owns simpler architecture and higher efficiency. A series of quantitative experiences on 6 widely used hyperspectral datasets show that the proposed LiteDenseNet obtains the state-of-the-art performance, even though when the absence of labelled samples is severe.**

***Index Terms*—3D two-way dense layer, deep learning (DL), hyperspectral classification**

I. Introduction

As an significant Earth observation technology, remote sensing is able to capture remote sensing images via sensors on aircrafts or satellites without physical contact [1].

The optical remote sensing is a major branch of remote sensing and has been applied in many fields including super resolution land cover mapping [2], drinking water protection [3] and object detection [4]. In recent years, scholars have increasingly focus on hyperspectral image (HSI), a kind of optical remote sensing images with high spectral resolution [5]. And a variety of applications have been development in many areas such as topsoil organic carbon estimation [6], plant traits prediction [7], and anomaly detection [8], among others. A basic research of HSI is supervised classification, whose objective is classifying labelled pixels in the image correctly. However, it is a great challenge to handle the redundant spectral information under limited data.

Support vector machines (SVM) [9], distance measure (DM) calculation [10], and maximum likelihood (MLH) criterion [11], and other early attempts focus on the spectral features of HSI. Multiple classifier methods and ensemble learning are also introduced such as AdaBoost [12] and Random Forests (RF) [13]. However, the adjacent pixels possibly fit into the same category, which is neglected by above-mentioned spectral-based methods. Meanwhile, HSI dataset are normally organized as 3D cubes format. Thus, it is feasible to integrate the spatial and spectral features in complementary form. Typical methods include 3D Gabor filter [14], [15], 3D scattering wavelet transform [16], and 3D discrete wavelet transform [17]. However, the high dependency on hand-crafted descriptors restricts the flexibility and adaptability of these methods.

Deep Learning (DL) is powerful to capture nonlinear and hierarchical features automatically, and has influenced many domains such as computer vision (CV) [18], natural language processing (NLP) [19], and automatic speech recognition (ASR) [20]. As a typical and basic classification task, there are many DL methods which have been introduced to HSI classification.

Chen et al. employed the Stacked Autoencoders (SAE) to extract serviceable features in [21]. Zhang et al. [22] used a Recursive Autoencoder (RAE) to capture high-level features from the adjacent pixels. In [23], based on Restricted Boltzmann Machine (RBM) and Deep Belief Network (DBN), Chen et al. proposed a novel hyperspectral image classification method. Recently, Zhou et al. [24] designed a compact and discriminative SAE (CDSAE) to discriminatively exploit low-dimensional feature. Zhou et al. [25] proposed a semi-supervised stacked autoencoders (Semi-SAEs) to deal with limited availability of samples.

However, the input of the above-mentioned algorithms is one-dimensional. Though the spatial features is exploited, the original structure is devastated. Fortunately, the emergence of Convolutional Neural Networks (CNN) renders some novel ideas, as CNN could capture spatial features while maintaining the immanent space structure. Zhao et al. [26] used CNN as the feature extractor in their framework. Lee et al. [27] proposed a deeper and wider network, Contextual Deep CNN (CDCNN). In [28], Chen et al. designed the feature extractor based on 3D-CNN.

Commonly, deeper networks are tougher to train, whereas they could capture finer information. The advent of the Residual

This work was supported in part by the National Natural Science Foundations of China (No. 41671452). *(Corresponding author: Rui Li.)*

R. Li and S. Zheng are with School of Remote Sensing and Information Engineering, Wuhan University, Wuhan 430079, China (e-mail: lironui@whu.edu.cn; syzheng@whu.edu.cn).

C. Duan is with the State Key Laboratory of Information Engineering in Surveying, Mapping, and Remote Sensing, Wuhan University, Wuhan 430079, China; chenxiduan@whu.edu.cn (e-mail: chenxiduan@whu.edu.cn).

Network (ResNet) [29] and the Dense Convolutional Network (DenseNet) [30] solves the dilemma to a great extent. Inspired by the ResNet, Zhong et al. [31] designed a Spectral-Spatial Residual Network (SSRN). Wang et al. [32] proposed a Fast Dense Spectral-Spatial Convolution (FDSSC) algorithm motivated by DenseNet.

To obtain more discriminative features, the attention mechanism was introduced to refine and optimize the feature maps. Haut et al. [33] incorporated attention mechanism into ResNet. Ma et al. [34] designed a Double-Branch Multi-Attention mechanism network (DBMA) based on the Convolutional Block Attention Module (CBAM) [35]. Motivated by Dual Attention Network (DANet) [36], Li et al. [37] proposed a Double-Branch Dual-Attention mechanism network (DBDA), and obtained the state-of-the-art results.

Inspired by the progress of DL domains, some novel network structures could also be seen in the literatures. Mou et al. [38] proposed a Recurrent Neural Networks (RNN) framework where hyperspectral images were analyzed through sequential perspective. Active Learning (AL) [39], Generative Adversarial Network (GAN) [40] and Semi-Supervised Learning (SSL) [41] are introduced to alleviate the severe absence of labelled samples in HSI. In [42], Capsule Networks (CapsNets) were adopted to reduce the complexity of the network. Furthermore, superpixel-based methods [43], Self-taught Learning [44], and Self-pace Learning [45] are also noteworthy.

Although performances have been enhanced by leaps and bounds, the requirement of DL for computational resources and training samples are huge and striking, while the cost of computing and annotation are rather expensive. In this paper, motivated by the GoogLeNet [46] and PeleeNet [47], we design a lightweight network architecture (LiteDenseNet) for HSI Classification. The number of parameters and the consumptions of calculation are observably less than contrapositive deep learning methods. The three significant contributions of this paper could be listed as follows:

(1) Based on DenseNet, we propose a 3D two-way dense layer which respectively capture the local and global features of the input. Based on 3D two-way dense layer, we propose an end-to-end lightweight framework, LiteDenseNet.

(2) We introduce group convolution to decrease calculation cost and parameter size. Thus, the number of parameters and the consumptions of calculation are observably decreased compared with contrapositive deep learning methods.

(3) A sequence of quantitative experiences on 6 widely used datasets show that the proposed LiteDenseNet obtains the state-of-the-art performance.

The remainder of this paper is arranged as follows: In Section 2, we briefly introduced the related work. In Section 3, we illustrate the detailed structure of LiteDenseNet. The experimental results are provided and analyzed in Sections 4. Finally, in Section 6 we draw a conclusion of the entire paper.

All of our code will be publicly available at https://github.com/lironui/LiteDenseNet as soon as possible.

II. Related Work

In this section, we will briefly introduce the basic units used in LiteDenseNet, related lightweight network architectures, and comparative deep learning methods.

A. *HSI Classification Framework Based on 3D-Cube*

The pixel-based methods only harness spectral features, and use the pixel individually to train the network. In contrast, 3D-cube-based methods capture both spectral and spatial features, and take both the target pixel and adjacent pixels as input in 3D-cube format. In other words, the significant distinction between 3D-cube-based methods and pixel-based methods is that the shape of the former input is $p \times p \times b$, while the shape of the latter input is $1 \times 1 \times b$, where $p \times p$ denotes the number of adjacent pixels and b represents the count of spectral bands.

B. *3D-CNN with Batch Normalization*

3D-CNN with a Batch Normalization (BN) [48] layer is a frequently used component in 3D-cube-based methods. As there are sufficient information of HSI both in the spatial and spectral dimensions, it is 3D-CNN which should be adopted to simultaneously obtain spatial and spectral features, and BN layers can improve numerical stability.

As shown in Fig. 1, supposing the shape of input feature maps is $(p_m \times p_m \times b_m, n_m)$, a 3D-CNN layer in the size of $(\alpha_{m+1} \times \alpha_{m+1} \times c_{m+1}, k_{m+1})$ would generate the output feature maps of size $(p_{m+1} \times p_{m+1} \times b_{m+1}, n_{m+1})$. The ith output of the $(m+1)$th 3D-CNN with BN can be computed as:

$$\boldsymbol{X}_i^{m+1} = \mathrm{R}\left(\sum_{j=1}^{n_m} \widehat{\boldsymbol{X}}_j^m * \boldsymbol{H}_i^{m+1} + b_i^{m+1}\right) \tag{1}$$

$$\widehat{\boldsymbol{X}}^m = \frac{\boldsymbol{X}^m - E(\boldsymbol{X}^m)}{Var(\boldsymbol{X}^m)} \tag{2}$$

where $\boldsymbol{X}_j^m \in \mathbb{R}^{p \times p \times b}$ is the jth input, and $\widehat{\boldsymbol{X}}^m$ is the output of the BN layer. $Var(\cdot)$ and $E(\cdot)$ represent the variance function and expectation of the input. $\boldsymbol{H}_i^{m+1}$ and b_i^{m+1} denote the weights and biases of the convolution layer, $R(\cdot)$ means the activation function, and $*$ is the convolutional operation.

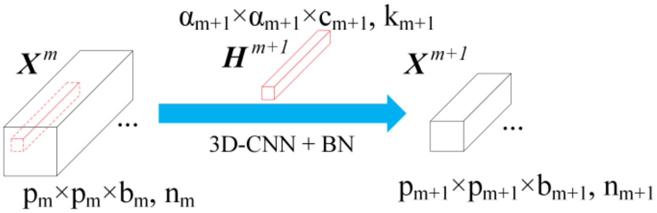

$\alpha_{m+1}\times\alpha_{m+1}\times c_{m+1}$, k_{m+1}
X^m
H^{m+1}
X^{m+1}
...
3D-CNN + BN
...
$p_m\times p_m\times b_m$, n_m
$p_{m+1}\times p_{m+1}\times b_{m+1}$, n_{m+1}

Fig. 1. The architecture of 3D-CNN with BN layer.

C. *ResNet and DenseNet*

Normally, the deeper convolutional network would obtain better performance. Nevertheless, substantial layers and excess parameters cause the troublesome vanishing and exploding gradients problems. ResNet [29] and DenseNet [30] are efficient and valid skills to solve the problem.

In ResNet, a skip connection is appended to the CNN. As shown in Fig. 2a, H represents a hidden block, which contains convolution layers, BN layers and activation layers. The skip connection, an identity mapping, empowers the input to directly get through the network. The basic element of ResNet is named residual block, and the output of the mth residual block can be computed as:

$$x_m = H_m(x_{m-1}) + x_{m-1} \tag{3}$$

Based on ResNet, DenseNet thoroughly connects all layers of the network. Instead of summating the output feature maps like ResNet, DenseNet concatenates the output of each layer at the channel dimension. The basic unit in DenseNet is named dense block, and the output of the mth dense block can be calculated as:

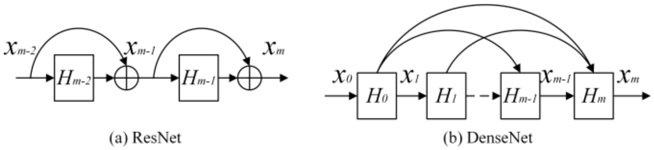
$$x_m = H_m[x_0, x_1, \dots, x_{m-1}] \tag{4}$$

where H_m represents a hidden block comprised of convolution layers, BN layers, and activation layers. $x_0, x_1, \dots, x_{m-1}$ denote the output generated by the 1st, 2nd, …, m-1th dense blocks. As shown in Fig. 2b, sufficient connections ensure adequate information flow through the network. A m-layer DenseNet owns $m(m+1)/2$ connections, while general convolutional network with m layers only has m connections.

Supposing the size of x_0 in Fig. 2b is $(p \times p \times b, n)$, and the shape of convolution lay is $(1 \times 1 \times 1, k)$, then each block outputs $(p \times p \times b, k)$ feature maps. Since DenseNet directly concatenates output of each block at the channel dimension, so the channels' number of x_m in Fig. 2b can be formulated as:

$$k_m = k_0 + (m-1) \times k \tag{5}$$

where k_0 is the channels′ number of x_0.



(a) ResNet (b) DenseNet

Fig. 2. The structure of ResNet and DenseNet.

D. Lightweight Network Architecture

The huge number of parameters and the high consumptions of calculation limit the application scenarios of deep learning methods. Even though deep learning frameworks could obtain brilliant performance, the overmuch time costs and monetary costs are the dominating weakness of them. In recent years, a fair number of lightweight network architectures have been proposed to escape this dilemma, including SqueezeNet [49], Xception [50], MobileNet V1 [51], MobileNet V2 [52], MobileNet V3 [53], ShuffleNet V1 [54], ShuffleNet V2 [55], PeleeNet [47], and so on.

SqueezeNet: Three strategies are adopted by SqueezeNet [49] to reduce the complexity of network, including replacing 3×3 kernels with 1×1 kernels, decreasing the channels' number of 3×3 kernels, and down sampling in the network late.

Xception: Xception means extreme Inception module [50], and is based on the hypothesis that the convolutional neural networks can be completely decoupled to spatial correlations and cross-channels correlations. And depth-wise convolution was introduced to improve the Inception.

MobileNet V1: In MobileNet V1 [51], conventional convolution layers were separated to depth-wise convolution and pointwise convolution which has been proved that can substantially enhance computational efficiency and passably retain accuracy rating.

MobileNet V2: Based on MobileNet V1, Sandler et, al. [52] proposed MobileNet V2 and introduced inverted residuals and linear bottlenecks to make a more efficient structure by utilizing the low-rank nature of the problem.

MobileNet V3: By combining novel architecture and complementary search technologies, Howard A et, al. [53] designed MobileNet V3. To improve the performance, Squeeze-and-Excitation Networks (SENet) and h-swish activation were also applied as components of the network.

ShuffleNet V1: The pointwise group convolution and channel shuffle [54] was introduced to decrease computation cost. The shuffling operations enables the information exchange between the groups of channels to alleviate marginal effects.

ShuffleNet V2: Ma et, al. [55] provided four guidances for lightweight network design including equal channel width, appropriate number of group convolutions, avoiding network fragmentation and reducing element-wise operations. Based on four guidances, authors updated the architecture of ShuffleNet.

PeleeNet: Wang et, al. [47] built two-way dense layer, stem block, bottleneck layer with dynamic number of channels, transition layer without compression and composite function to simplify network structure. What is more, authors proved that depth-wise convolution is not the only method to design an efficient model.

E. Hyperspectral Image Classification

We will briefly introduce the frameworks which are going to be compared with our method, including SVM [9], CDCDD [27], SSRN [31], FDSSC [32], DBMA [34] and DBDA [37].

SVM: We harness SVM with a radial basis function (RBF) kernel [9] as pixel-based method, and all pixels with their spectral bands are individually fed into.

CDCNN: CDCNN [27] is based on 2D-CNN and ResNet, and the deeper and wider structure enables CDCNN to fully capture local spatio-spectral contextual interactions. The size of input is 3D-cube in the shape of $5 \times 5 \times$ b, where b denotes the number of spectral bands.

SSRN: The spatial and spectral residual blocks of the SSRN [31] alleviate the decline in the precision accuracy. When the training samples are limited, SSRN could still deliver robust performance. The shape of the 3D-cube input is set to $7 \times 7 \times b$.

FDSSC: To exploit spectral and spatial features respectively, the sizes of convolutional kernels in FDSSC [32] are different. Since BN, dropout layers and other skills are adopted in FDSSC, the convergence speed of FDSSC is fast. The shape of the input is $9 \times 9 \times$ b.

DBMA: There are two branches in DBMA [34] to capture spectral and spatial features separately, and channel-wise attention and spatial-wise attention are introduced to refine the extracted features. $7 \times 7 \times$ b is the input patch size.

DBDA: Motivated by DBMA, a more adaptive and flexible attention mechanism and a novel activation function are introduced to DBDA [37], which bring the state-of-the-art performance and accelerate the convergence speed. The shape of the input is $9 \times 9 \times$ b.

III. Proposed Method

Fig. 3 illustrates the three steps of the LiteDenseNet: dataset generation, training and validation, and prediction. Supposing that an HSI dataset X contains N pixels $\{x_1, x_2, \dots, x_n\} \in \mathbb{R}^{1\times1\times b}$, where b denotes the bands, the corresponding label vector is $\{y_1, y_2, \dots, y_n\} \in \mathbb{R}^{1\times1\times c}$, where c represent the number of land cover categories.

In the dataset generation step, $p \times p$ adjacent pixels of the target pixel x_i is chosen from the original image to obtain the 3D-cube set $\{z_1, z_2, \dots, z_n\} \in \mathbb{R}^{p\times p\times b}$. Next, the 3D-cube Z is randomly separated into Z_{train} , Z_{val} and Z_{test} . The corresponding labels are accordingly divided into Y_{train}, Y_{val} and Y_{test}. In the training and validation steps, we update the parameters using training samples, and validate and select the trained model using the validation set. In the prediction step, we verify the accuracy of the selected model.

The cross-entropy loss function is the commonly used quantitative evaluation index to measure the disparity between the ground truth and predicted results, which is defined as

$$C(\hat{y}, y) = \sum_{m=1}^{L} y_m \left(log \sum_{n=1}^{L} e^{\hat{y}_n} - \hat{y}_m \right) \qquad (6)$$

where $y = [y_1, y_2, \dots y_L]$ represents the ground truth and $\hat{y} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_L]$ denotes predicted results.

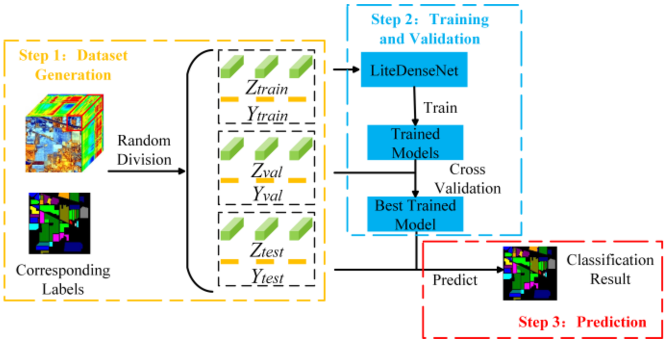


Fig. 3. The procedure of our proposed LiteDenseNet framework.

The whole framework of LiteDenseNet is shown in Fig. 6, and the following contents will introduce the details about the LiteDenseNet using the Indian Pines (IN) dataset as the example. The IN dataset comprises 145×145 pixels with 200 bands, which means the size of IN is $145 \times 145 \times 200$. Nevertheless, only 10, 249 pixels own corresponding labels, and the others are background. Table III gives details of IN. The patch size is set as 9×9. For the matrixes mentioned below such as $(9 \times 9 \times 97, 24)$, the $9 \times 9 \times 97$ denotes the numerical value of height, width and depth of the 3D-cube, and 24 means the count of 3D-cubes generated by convolution layer.

A. 3D Two-Way Dense Layer

Motivated by GoogLeNet [46] and PeleeNet [47], we design a 3D two-way dense layer to get receptive fields in different scales. The comparison between original dense layer and two-way dense layer is provided in Fig. 4.

Supposing the shape of the input is $(p \times p \times b, k_0)$, for the original dense layer, the 3D-cube is fed into a $1 \times 1 \times 1$ convolution layer and a $3 \times 3 \times 3$ convolution layer, and obtain the $(p \times p \times b, 24)$ output, and the concat operation is implemented between the input and the output and generate the 3D-cubes in the shape of $(p \times p \times b, k_0 + 24)$.

As for 3D two-way dense layer, the top way of the layer contains two stacked $3 \times 3 \times 3$ convolution to capture global visual patterns which generate the $(p \times p \times b, 12)$ output. And the bottom way of the layer harnesses a $3 \times 3 \times 3$ kernel size to exploit local visual patterns which also generate the output in the shape of $(p \times p \times b, 12)$. And the concat operation is implemented between the input and the outputs of two ways, which generates the 3D-cubes with the size of $(p \times p \times b, k_0 + 24)$. The $1 \times 1 \times 1$ convolution layers which exist in the top way and bottom way transform the channel of the input.

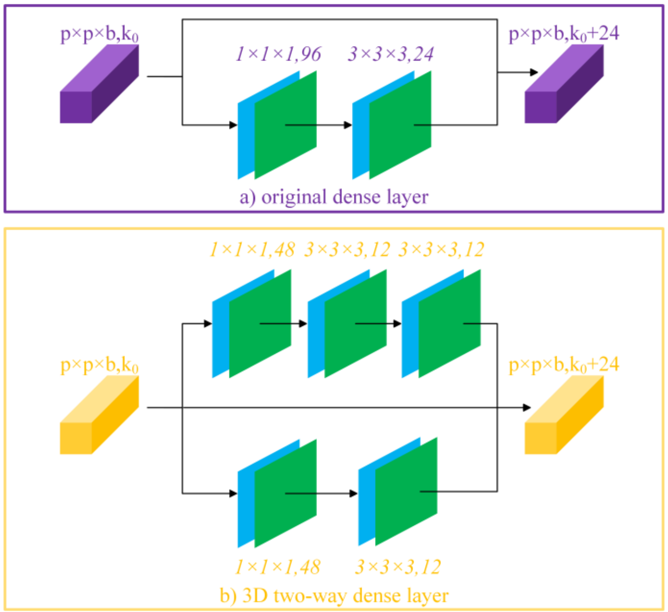


Fig. 4. The comparison between (a) original dense layer and (b) 3D two-way dense layer.

B. Group Convolution

Since convolutional operation is a computationally intensive operation, the goal to reduce the parameters and consumption of convolution has spawned a series of worthwhile works, and group convolution is one of the simple but effective skills of them. The comparison between a normal convolution layer and a convolution layer with 3 groups can be seen in Fig. 5.

The normal convolution layer in Fig. 5a typically own the same channels c_1 as the input. Nevertheless, for the normal convolution layer with 3 groups in Fig. 5b, 3 independent groups of $c_2/3$ channels operate on the fraction $c_1/3$ of the input, decreasing channel dimensions from $h \times w \times c_1$ to $h \times w \times (c_1/3)$. The groups prominently reduce the number of parameters and computational complexity, while maintain the dimensions of the input and output. Concretely, for the normal convolution layer in Fig. 5a, the number of parameters M_1 and the amount of computation N_1 can be calculated as:

$$M_1 = h \times w \times c_1 \times c_2 \tag{7}$$

$$N_1 = h \times w \times c_1 \times H_2 \times W_2 \times c_2 \tag{8}$$

As for the convolution layer with 3 groups in Fig. 5b, the number of parameters M_2 and the amount of computation N_2 can be formulated as:

$$M_2 = (h \times w \times (c_1/3) \times (c_2/3)) \times 3 \tag{9}$$

$$N_2 = (h \times w \times (c_1/3) \times H_2 \times W_2 \times (c_2/3)) \times 3 \tag{10}$$

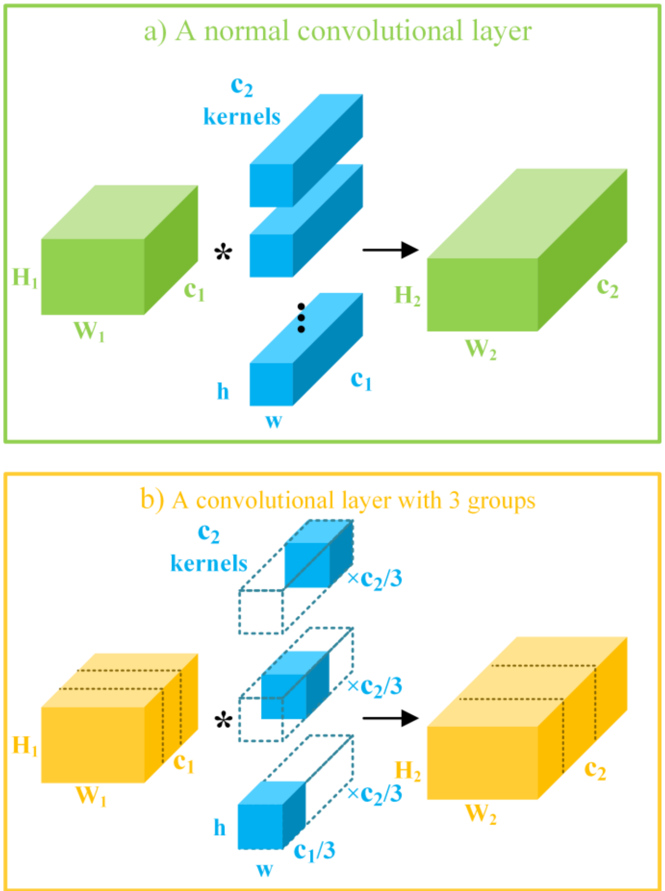

a) A normal convolutional layer
c2
kernels
H1
W1
c1
*
h
w
c1
H2
W2
c2
b) A convolutional layer with 3 groups
c2
kernels
×c2/3
×c2/3
×c2/3
H1
W1
c1
*
h
w
c1/3
H2
W2
c2

Fig. 5. The comparison between (a) a normal convolution layer and (b) a convolution layer with 3groups.

C. The Whole Framework of LiteDenseNet.

The whole framework of LiteDenseNet can be seen in Fig. 6, and the flowchart of LiteDenseNet is shown in Fig. 7. The following part provides a detailed procedure of LiteDenseNet illustrated with the IN dataset example.

Firstly, the $(9 \times 9 \times 200{,}1)$ input 3D-cube is fed into a 3D-CNN layer in the shape of $(1 \times 1 \times 7{,}24)$, while the down sampling stride is (1, 1, 2). Thus, we obtain a $(9 \times 9 \times 97{,}24)$ output. Then, a 3D two-way dense layer with group convolution is attached to exploit the information, and generate the feature maps with the size of $(9 \times 9 \times 97{,}48)$. In 3D two-way dense layer, the number of groups for convolutional operation are assigned as 3. Next, we harness a $(3 \times 3 \times 97{,}60)$ 3D-CNN layer to reduce the dimension of bands, and reshape the output into $(9 \times 9{,}60)$. Finally, we implement a global average pooling operation and obtain the feature maps in the shape of 1×60, and obtain the classification result via a fully connected layer and softmax function. And details about the implements are provided in Table I.

TABLE I
THE IMPLEMENTS DETAILS ABOUT LITEDENSENET

Layer name		Kernel Size	Group	Output Size
Input		-	-	$(9\times 9 \times 200, 1)$
3D-CNN+BN+ReLU		$(1\times 1 \times 7)$	1	$(9\times 9 \times 97{,}24)$
3D two-way dense layer	3D-CNN+ BN+ReLU	$(1\times 1 \times 1)$	3	$(9\times 9 \times 97{,}48)$
	3D-CNN+ BN+ReLU	$(3\times 3 \times 3)$	3	$(9\times 9 \times 97{,}12)$
	3D-CNN+ BN+ReLU	$(3\times 3 \times 3)$	3	$(9\times 9 \times 97{,}12)$
	3D-CNN+ BN+ReLU	$(1\times 1 \times 1)$	3	$(9\times 9 \times 97{,}48)$
	3D-CNN+ BN+ReLU	$(3\times 3 \times 3)$	3	$(9\times 9 \times 97{,}12)$
Concatenate		-	-	$(9\times 9 \times 97{,}60)$
3D-CNN+BN+ReLU		$(3\times 3 \times 97)$	3	$(9\times 9 \times 1{,}60)$
Global Average Pooling		-	-	(1×60)

D. Two Measures Taken to Prevent Overfitting.

Since LiteDenseNet is a lightweight architecture network, we select the number of training and validation samples at a minimal percentage to conserve calculating resources further. Nevertheless, limited training samples make the network be prone to overfitting. Thus, we adopt early stopping and dynamic learning rate to retard overfitting.

Early stopping strategy means if the accuracy on validation set is not reduce any more for certain epochs (we set the number as 20 in our model), thus we will terminate the training step to restrain overfitting and save the training consumption.

Since the learning rate is the vital hyper parameter of a network, a dynamic learning rate could support a network avoid trapping into local minima. We adopt the cosine annealing [56] method to dynamically adjust the learning rate as:

$$\eta_t = \eta^i_{min} + \frac{1}{2}\left(\eta^i_{max} - \eta^i_{min}\right)\left(+cos\left(\frac{T_{cur}}{T_i}\pi\right)\right) \tag{11}$$

where η_t denotes the learning rate and $[\eta^i_{min}, \eta^i_{max}]$ represents the scope of the learning rate. T_{cur} is the number of epochs which have been performed yet, and T_i regulates the number of epochs which will be performed in a cyclical period.

E. The Comparison of Parameters and Computational Complexity.

Table II demonstrates the comparison of parameters and computational complexity between 6 deep learning algorithms. The parameters and calculation cost of LiteDenseNet are significantly less than other 3D-CNN-based algorithms.

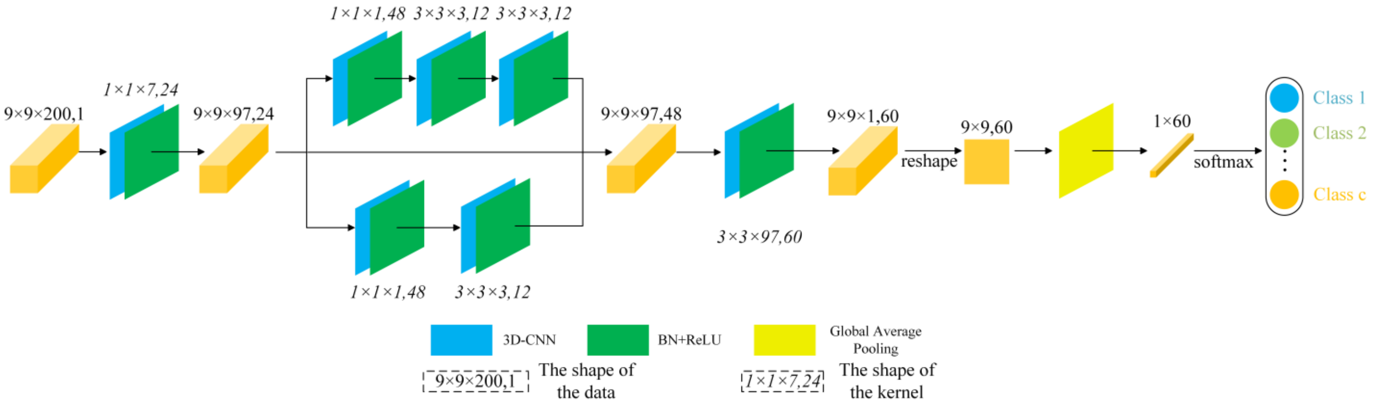


Fig. 6. The structure of the LiteDenseNet.

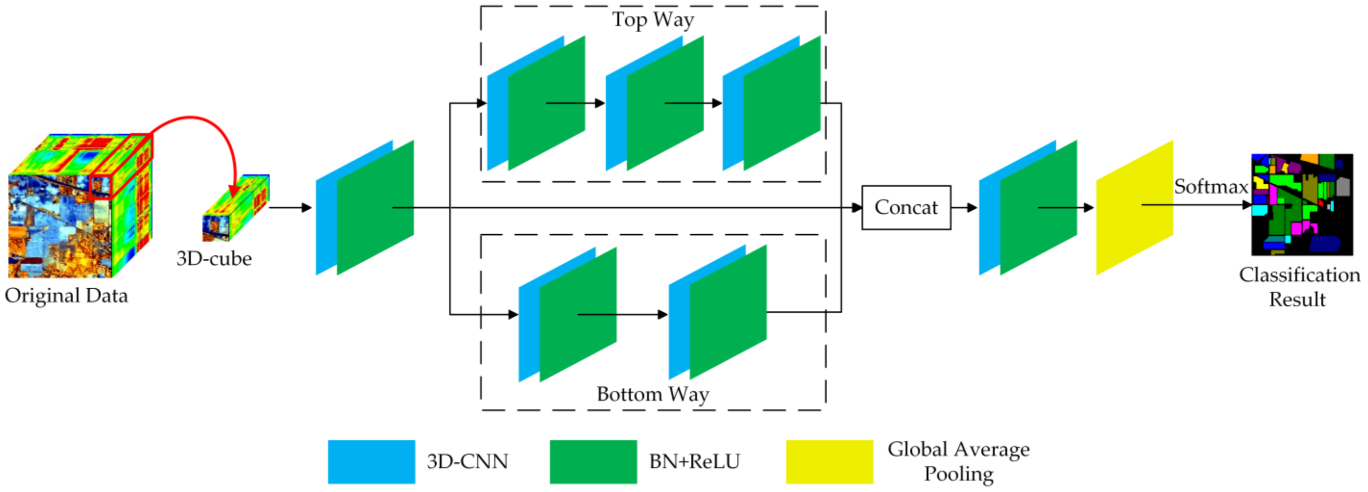


Fig. 7. The flowchart of the LiteDenseNet methodology.

TABLE II
THE COMPARISON OF PARAMETERS AND COMPUTATIONAL COMPLEXITY

Infrastructure		Network name	Input shape	Parameters
2D-CNN	CDCNN	5×5×200	1.06M	3.53M FLOPs
3D-CNN	SSRN	7 × 7 × 200	0.36M	95.36M FLOPs
	FDSSC	9 × 9 × 200	1.23M	175.66M FLOPs
	DBMA	7 × 7 × 200	0.61M	147.89M FLOPs
	DBDA	9 × 9 × 200	0.61M	244.81M FLOPs
	LiteDenseNet	9 × 9 × 200	0.11M	111.47M FLOPs

IV. EXPERIMENTAL RESULTS

To verify the accuracy the LiteDenseNet, we perform quantitative experiments on 6 widely used datasets and compare the performance between LiteDenseNet and other methods. We adopt overall accuracy (OA), average accuracy (AA), and Kappa coefficient (K) to evaluate the performance of each method.

A. Data Description.

In this paper, we use 6 HSI datasets, i.e. the Indian Pines (IN) dataset, the Pavia University (UP) dataset, the Pavia Centre (PC) dataset, the Salinas Valley (SV) dataset, the Kennedy Space Center (KSC) dataset and the Botswana (BS) dataset, to design the experiments. The details of six dataset are provided in Tables III-VIII, and false color images and corresponding category labels are rendered in Fig. 8.

Indian Pines (IN): The IN dataset is captured via Airborne Visible Infrared Imaging Spectrometer (AVIRIS) sensor over north-western Indiana. IN has 145 × 145 pixels with 200 bands and 16 land cover categories.

Pavia University (UP): The UP dataset is gathered through the Reflective Optics Imaging Spectrometer (ROSIS) sensor over the University of Pavia, Italy. UP has 610 × 340 pixels with 102 bands and 9 land cover categories.

Pavia Center (PC): The PC dataset is captured by the ROSIS sensor over the Pavia center, Italy. UP has 1096 × 715 pixels with 103 bands and 9 land cover categories.

Salinas Valley (SV): The SV dataset is obtained via the AVIRIS sensor over Salinas Valley, California. SV has 512 × 217 pixels with 204 bands and 16 land cover categories.

Kennedy Space Center (KSC): The KSC dataset is gathered through the AVIRIS sensor over the Kennedy Space Center, Florida. KSC has 512 × 614 pixels with 176 bands and 13 land cover categories.

Botswana (BS): The PC dataset is captured by the NASA EO-1 satellite over the Okavango Delta, Botswana. BS has 1476 × 256 pixels with 145 bands and 14 land cover categories.

B. Experimental Setting

As we mentioned above, we select very limited training and validation samples to save calculation cost. The proportions of training and validation samples are set as 3% for IN and KSC, 1% for BS, 0.5% for UP and SV, and 0.1% for PC.

To evaluate the performance, we compare LithDenseNet with SVM [9], CDCDD [27], SSRN [31], FDSSC [32], DBMA [34] and DBDA [37]. All experiments are conducted on a standard and mediocre laptop, which is configured with 8 GB of physical memory and 50 GB virtual memory, an i5-8250U CPU, and an NVIDIA GeForce MX150 GPU. All deep learning classifiers are implemented with PyTorch, and SVM is implemented with sklearn.

For all deep learning methods, the batch size is set as 16 with 0.0005 learning rate, and the optimizer is Adam.

TABLE III
THE SAMPLES FOR EACH CLASS FOR TRAINING, VALIDATION AND TESTING OF THE INDIAN PINES (IN) DATASET

No.	Class	Total number	Train	Val	Test
1	Alfalfa	46	3	3	40
2	Corn-notill	1428	42	42	1344
3	Corn-mintill	830	24	24	782
4	Corn	237	7	7	223
5	Grass-pasture	483	14	14	455
6	Grass-trees	730	21	21	688
7	Grass-pasture-mowed	28	3	3	22
8	Hay-windrowed	478	14	14	450
9	Oats	20	3	3	14
10	Soybean-notill	972	29	29	914
11	Soybean-mintill	2455	73	73	2309
12	Soybean-clean	593	17	17	559
13	Wheat	205	6	6	193
14	Woods	1265	37	37	1191
15	Buildings-Grass-Trees	386	11	11	364
16	Stone-Steel-Towers	93	3	3	87
	Total	10249	307	307	9635

TABLE IV
THE SAMPLES FOR EACH CLASS FOR TRAINING, VALIDATION AND TESTING OF THE PAVIA UNIVERSITY (UP) DATASET

No.	Class	Total number	Train	Val	Test
1	Asphalt	6631	33	33	6565
2	Meadows	18,649	93	93	18463
3	Gravel	2099	10	10	2079
4	Trees	3064	15	15	3034
5	Painted metal sheets	1345	6	6	1333
6	Bare Soil	5029	25	25	4979
7	Bitumen	1330	6	6	1318
8	Self-Blocking Bricks	3682	18	18	3646
9	Shadows	947	4	4	939
	Total	42,776	210	210	42356

TABLE V
THE SAMPLES FOR EACH CLASS FOR TRAINING, VALIDATION AND TESTING OF THE PAVIA CENTER (PC) DATASET

No.	Class	Total number	Train	Val	Test
1	Water	65 971	65	65	65841
2	Trees	7598	7	7	7584
3	Meadows	3090	3	3	3084
4	Bricks	2685	3	3	2679
5	Soil	6584	6	6	6572
6	Asphalt	9248	9	9	9230
7	Bitumen	7287	7	7	7273
8	Tiles	42 826	42	42	42742
9	Shadows	2863	3	3	2857
	Total	148152	145	145	147862

TABLE VI
THE SAMPLES FOR EACH CLASS FOR TRAINING, VALIDATION AND TESTING OF THE SALINAS VALLEY (SV) DATASET

No.	Class	Total number	Train	Val	Test
1	Brocoli-green-weeds-1	2009	10	10	1989
2	Brocoli-green-weeds-2	3726	18	18	3690
3	Fallow	1976	9	9	1958
4	Fallow-rough-plow	1394	6	6	1382
5	Fallow-smooth	2678	13	13	2652
6	Stubble	3959	19	19	3921
7	Celery	3579	17	17	3545
8	Grapes-untrained	11271	56	56	11159
9	Soil-vinyard-develop	6203	31	31	6141
10	Corn-senesced-green-weeds	3278	16	16	3246
11	Lettuce-romaine-4wk	1068	5	5	1058
12	Lettuce-romaine-5wk	1927	9	94	1824
13	Lettuce-romaine-6wk	916	4	4	908
14	Lettuce-romaine-7wk	1070	5	5	1060
15	Vinyard-untrained	7268	36	36	7196
16	Vinyard-vertical-trellis	1807	9	9	1789
	Total	54129	263	263	53603

TABLE VII
THE SAMPLES FOR EACH CLASS FOR TRAINING, VALIDATION AND TESTING OF THE KENNEDY SPACE CENTER (KSC) DATASET

No.	Class	Total number	Train	Val	Test
1	Scrub	761	22	22	717
2	CP hammock	243	7	7	229
3	CP/Oak	256	7	7	242
4	Slash pine	252	7	7	238
5	Oak/Broadleaf	161	4	4	153
6	Hardwood	229	6	6	217
7	Swamp	105	3	3	99
8	Graminoid marsh	431	12	12	407
9	Spartina marsh	520	15	15	490
10	Cattail marsh	404	12	12	380
11	Salt marsh	419	12	12	395
12	Mud flats	503	15	15	473
13	Water	927	27	27	873
	Total	5211	149	149	4913

C. Experimental Results

The experimental results with different methods for 6 datasets are demonstrated in Tables IX-XIV.

For IN dataset, our proposed LiteDenseNet achieves the best results with 95.52%±1.33% OA, 94.24%±0.83% AA, and 0.9490±0.0152 Kappa with 3% training samples. Since the training samples are severely limited and network structure is weak, CDCNN which is based on 2D-CNN obtains the worst accuracy with 66.90%±7.38% OA. Though SVM achieves better performance than CDCNN, the salt-and-pepper noise is obvious, which due to SVM individually uses target pixels and uses no adjacent spatial information. The 3D-CNN based methods far surpass CDCNN and SVM, on account of their incorporation of both spectral and spatial information for classification. FDSSC uses DenseNet instead of ResNet as its backbone, which leads to 2.92% enhancement in OA compared to SSRN. Motivated by FDSSC, DBMA captures the spatial and spectral features in two branches and brings spatial-wise attention and channel-wise attention in. Nevertheless, since training samples are extremely limited, overfitting phenomenon occurs in DBMA. As DBDA adopts a more adaptive and flexible attention mechanism, 3.14% improvement in OA is promoted. With proposed LiteDenseNet, it can remain reliable and stable even though the training samples are finite. Specifically, the proposed method improves the OA by 2.49%, the AA by 0.59%, and the Kappa by 0.0284 compared to DBDA.

TABLE VIII
THE SAMPLES FOR EACH CLASS FOR TRAINING, VALIDATION AND TESTING OF THE BOTSWANA (BS) DATASET

No.	Class	Total number	Train	Val	Test
1	Water	270	3	3	264
2	Hippo grass	101	2	2	97
3	Floodplain grasses1	251	3	3	245
4	Floodplain grasses2	215	3	3	209
5	Reeds1	269	3	3	263
6	Riparian	269	3	3	263
7	Fierscar2	259	3	3	253
8	Island interior	203	3	3	197
9	Acacia woodlands	314	4	4	306
10	Acacia shrublands	248	3	3	242
11	Acacia grasslands	305	4	4	297
12	Short mopane	181	2	2	177
13	Mixed mopane	268	3	3	262
14	Exposed soils	95	1	1	93
	Total	3248	40	40	3168

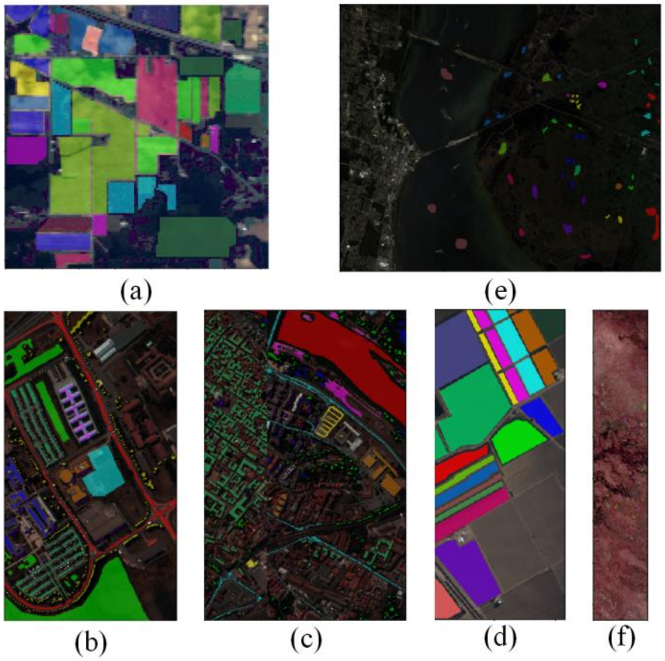


Fig. 8. False color images and corresponding category labels of (a) IN (b) UP (c) PC (d) SV (e) KSC and (f) BS.

For UP dataset, our proposed LiteDenseNet achieves the best results with 97.03%±0.59% OA, 96.52%±0.76% AA, and 0.9606±0.0079 Kappa with 0.5% training samples. As the samples in the UP dataset are abundant, there are enough samples for every category even if there are only 0.5% samples for training. Thus, the performance of CDCNN surpasses the SVM. Even though our model cannot make each category accuracy best, the precision of each class with our method is not lower than 86%, which means that our method can distinctively capture the features between different categories. And the proposed method improves the OA by 0.48%, the AA by 0.59%, and the Kappa by 0.0063 compared to DBDA.

For PC dataset, our proposed LiteDenseNet achieves the best results with 97.48%±0.39% OA, 92.98%±1.52% AA, and 0.9642±0.0055 Kappa with 0.1% training samples. The imbalanced category distribution is a significant feature of PC dataset. For example, there are 65, 971 pixels in class 1, while there are only 2, 685 pixels in class 4. As we just select 0.1% training samples, there are only 3 pixels of class 4 for training. For class 4, although proposed LiteDenseNet merely obtains 78.83%±6.80% accuracy, the precision is 5.82% higher than DBDA, and 14.31% than DBMA.

For SV dataset, our proposed LiteDenseNet achieves the best results with 97.24%±1.42% OA, 98.28%±0.60% AA, and 0.9693±0.0157 Kappa with 0.5% training samples. Similarly, 0.5% training samples are enough, as SV dataset owns sufficient samples. But unlike UP dataset which only has 9 classes, the SV dataset has 16 classes. Thus, CDCNN obtains a worst performance. And the proposed method improves the OA by 1.96%, the AA by 1.20%, and the Kappa by 0.0218 compared to DBDA.

For KSC dataset, our proposed LiteDenseNet achieves the best results with 96.68%±1.17% OA, 95.08%±1.83% AA, and 0.9630±0.0130 Kappa with 3% training samples. The accuracies of class 3-7 with no more than 7 training samples are unsatisfactory of other methods. Our LiteDenseNet obtains 79.19%, 83.24%, 92.42%, 98.10% and 91.87% of class 3-7, which are 2.77%, 16.50%, 20.20%, 12.25% and 14.28% higher than DBDA. And the proposed method improves the OA by 2.59%, the AA by 5.18%, and the Kappa by 0.0288 compared to DBDA.

For BS dataset, our proposed LiteDenseNet achieves the best results with 95.94%±0.93% OA, 96.20%±1.10% AA, and 0.9560±0.0100 Kappa with 1% training samples. BS is a small dataset with only 3, 248 labelled samples, and we merely select 40 samples for training and 40 samples for validation. LiteDenseNet not only obtains the state-of-the-art performance, but also generates lowest standard deviation. Specifically, our method's standard deviation of OA is 0.93, while it is 3.04, 2.30

TABLE IX
THE CATEGORIZED RESULTS FOR THE IN DATASET USING 3% TRAINING SAMPLES

Class	SVM	CDCNN	SSRN	FDSSC	DBMA	DBDA	Proposed
1	29.34±3.60	50.17±6.79	79.16±8.97	97.08±2.71	94.80±3.69	98.30±1.70	100.0±0.00
2	55.51±0.32	56.59±4.46	86.15±2.21	96.24±1.91	91.08±0.72	93.82±1.29	94.44±1.69
3	62.66±1.07	53.17±3.07	91.67±2.98	93.14±2.65	85.40±5.48	94.55±2.19	94.95±1.35
4	42.74±3.49	53.31±3.91	84.37±4.53	97.17±1.07	88.88±2.69	96.81±0.49	95.99±1.67
5	85.30±1.28	84.05±5.66	97.69±1.89	98.42±0.64	97.43±0.51	98.53±0.67	98.67±0.55
6	82.11±1.52	89.03±2.82	95.85±1.56	97.02±0.87	96.76±1.18	96.64±1.82	97.90±0.69
7	64.17±6.13	46.28±8.09	90.93±6.09	72.21±12.14	52.66±9.25	67.37±11.36	70.77±10.51
8	89.79±0.92	92.06±0.98	97.72±1.34	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00
9	42.40±10.06	52.17±13.04	74.64±10.86	71.29±18.60	62.66±6.34	78.46±7.65	89.28±3.41
10	63.01±2.72	52.87±7.61	85.75±3.81	86.01±4.24	82.43±3.32	85.55±7.24	92.11±1.91
11	64.09±1.25	67.79±3.09	88.65±1.80	91.56±4.05	90.54±1.83	93.98±2.96	96.37±0.76
12	48.50±1.15	44.67±3.28	86.34±2.73	90.63±2.75	80.10±5.37	88.15±2.66	90.82±3.34
13	87.37±2.35	87.12±2.60	99.00±1.00	99.79±0.20	98.55±0.79	98.78±0.98	100.0±0.00
14	89.71±0.41	91.17±1.25	95.52±0.55	97.01±1.69	97.14±0.75	96.19±1.08	97.45±0.38
15	61.51±2.73	73.97±1.00	94.28±1.54	93.24±2.18	86.18±2.21	94.15±0.90	94.09±2.46
16	97.64±1.29	94.36±1.33	94.15±2.15	96.99±1.69	94.55±3.93	93.59±4.00	95.04±2.06
OA	68.69±0.50	66.90±7.38	90.24±1.18	93.16±1.96	89.89±1.33	93.03±2.10	95.52±1.33
AA	66.62±1.37	68.05±2.06	90.12±1.54	92.36±3.08	87.45±2.34	92.18±1.00	94.24±0.83
K×100	63.93±0.49	62.36±7.78	88.84±1.36	92.18±2.28	88.48±1.51	92.06±2.38	94.90±1.52

TABLE X
THE CATEGORIZED RESULTS FOR THE UP DATASET USING 0.5% TRAINING SAMPLES

Class	SVM	CDCNN	SSRN	FDSSC	DBMA	DBDA	Proposed
1	83.61±2.58	87.30±2.83	98.89±0.47	97.42±1.06	92.91±0.94	96.23±0.67	97.62±1.23
2	84.96±2.07	92.65±1.12	97.96±0.37	98.69±0.34	96.03±2.1	99.02±0.22	98.94±0.33
3	58.75±5.38	45.81±12.22	74.34±10.03	91.34±6.61	89.41±4.36	93.95±2.64	90.93±4.17
4	96.37±0.86	95.02±2.65	98.98±0.47	97.75±1.59	96.86±1.48	97.91±0.41	98.30±0.64
5	94.99±1.16	96.96±1.27	99.93±0.06	99.67±0.11	99.49±0.16	99.52±0.20	99.60±0.14
6	81.90±4.16	82.71±4.28	91.07±4.24	98.72±0.27	96.86±0.92	97.31±0.66	98.62±0.52
7	53.26±13.41	69.82±8.51	78.69±4.42	96.53±1.30	95.18±4.49	96.59±1.29	99.46±0.23
8	71.36±1.96	65.38±2.04	77.71±4.16	74.33±2.14	81.67±2.05	85.55±3.97	86.33±1.92
9	99.89±0.03	93.89±1.76	98.60±0.78	97.17±0.95	92.78±3.73	97.30±0.87	98.91±0.45
OA	82.63±2.95	85.82±1.62	92.92±1.26	95.32±1.24	93.79±1.53	96.55±0.88	97.03±0.59
AA	80.57±4.68	81.06±2.93	90.68±1.93	94.62±2.14	93.47±1.96	95.93±1.08	96.52±0.76
K×100	76.23±4.56	81.08±2.11	90.66±1.63	93.78±1.66	91.69±2.15	95.43±1.16	96.06±0.79

TABLE XI
THE CATEGORIZED RESULTS FOR THE PC DATASET USING 0.1% TRAINING SAMPLES

Class	SVM	CDCNN	SSRN	FDSSC	DBMA	DBDA	Proposed
1	99.75±0.09	97.08±0.75	99.98±0.02	99.85±0.13	99.84±0.05	99.86±0.10	99.84±0.09
2	83.36±2.23	82.01±4.28	97.05±0.85	87.73±4.72	93.32±2.77	96.96±0.93	94.03±1.02
3	62.47±5.34	85.20±5.92	77.15±5.88	84.17±7.69	76.82±5.15	77.40±6.27	84.87±6.33
4	63.15±5.80	49.91±12.72	65.43±6.89	63.46±17.03	64.52±2.83	73.01±7.33	78.83±6.80
5	82.76±4.16	73.20±4.87	89.23±2.16	90.01±3.43	87.02±3.23	89.83±3.74	94.94±2.16
6	83.52±2.27	85.77±2.24	87.29±4.79	88.96±3.80	87.87±3.94	91.00±1.22	90.16±1.39
7	91.88±0.97	86.91±7.96	99.64±0.30	94.53±5.19	99.14±0.35	98.20±0.76	96.98±2.22
8	95.26±2.57	97.28±0.69	98.19±0.86	99.66±0.10	99.55±0.18	99.50±0.08	99.03±0.56
9	99.77±0.10	92.57±3.53	97.99±1.77	92.43±7.23	95.61±2.34	99.39±0.40	98.15±1.73
OA	93.87±1.64	92.92±2.08	96.36±1.39	96.54±0.78	96.50±0.86	97.24±0.23	97.48±0.39
AA	84.66±0.85	83.33±5.99	90.22±2.24	88.98±4.36	89.30±2.18	91.68±1.00	92.98±1.52
K×100	91.27±2.38	89.86±3.00	94.83±1.97	95.10±1.11	95.04±1.22	96.09±0.33	96.42±0.55

and 2.80 for DBDA, DBMA, and FDSSC. And the proposed method improves the OA by 1.70%, the AA by 1.27%, and the Kappa by 0.0184 compared to DBDA.

D. *Effectiveness of 3D Two-Way Dense Layer*

There are two ways in 3D two-way dense layer. The top way contains two stacked $3 \times 3 \times 3$ convolution layers, which is equivalent to a $5 \times 5 \times 5$ kernel size and obtain global information. The bottom way uses a $3 \times 3 \times 3$ kernel to exploit local visual patterns. To verify the effectiveness of two-way dense layer, we remove a $3 \times 3 \times 3$ convolution layer of the top way. And we also take original dense layer into comparison.

TABLE XII

The Categorized Results for the SV Dataset Using 0.5% Training Samples

Class	SVM	CDCNN	SSRN	FDSSC	DBMA	DBDA	Proposed
1	99.59±0.21	54.94±22.83	100.0±0.00	100.0±0.00	99.56±0.44	99.70±0.19	100.0±0.00
2	98.82±0.19	67.82±2.33	99.94±0.06	98.77±0.92	99.89±0.09	99.38±0.45	99.24±0.76
3	89.01±2.51	92.39±1.61	90.80±6.16	96.43±0.91	97.84±0.87	97.64±0.60	98.67±0.48
4	97.59±0.41	94.98±1.55	97.77±0.89	95.19±1.84	89.73±2.89	93.01±1.90	96.29±1.42
5	93.31±1.36	95.06±2.19	97.58±0.91	99.71±0.07	97.70±0.51	97.58±1.93	99.62±0.16
6	99.91±0.03	98.73±0.59	99.92±0.05	99.99±0.01	99.33±0.35	100.0±0.00	99.98±0.02
7	95.37±1.23	96.35±1.33	99.86±0.14	99.08±0.56	96.75±1.62	97.88±1.07	99.79±0.12
8	71.76±1.14	76.77±5.37	82.35±2.74	91.54±2.16	90.68±2.70	92.06±4.02	95.50±1.14
9	98.46±0.36	97.96±1.05	99.32±0.48	99.62±0.20	99.50±0.22	99.66±0.12	99.77±0.10
10	85.29±1.30	84.36±3.45	97.30±0.63	94.32±3.83	95.29±1.70	96.99±0.52	99.02±0.22
11	86.82±3.47	80.84±3.80	94.17±1.31	96.16±1.36	93.89±2.33	96.50±0.98	96.33±1.34
12	92.48±2.32	86.62±4.08	98.84±0.48	97.23±1.11	98.61±1.12	98.59±0.82	99.60±0.20
13	92.60±1.34	95.80±0.88	98.74±0.62	99.74±0.21	99.40±0.11	98.96±0.62	99.78±0.14
14	92.48±1.81	93.14±2.36	96.42±2.04	94.06±1.12	94.77±2.54	96.95±1.12	97.30±0.47
15	70.58±3.14	57.79±7.96	88.42±1.12	91.93±1.45	89.31±1.60	88.43±3.46	91.59±4.38
16	97.74±0.70	98.09±0.53	99.83±0.10	99.99±0.01	99.73±0.16	99.96±0.04	100.0±0.00
OA	86.88±0.80	82.09±2.57	93.27±1.56	95.88±1.27	95.04±1.09	95.28±1.34	97.24±1.42
AA	91.36±0.48	85.73±4.28	96.33±0.85	97.11±0.71	96.37±0.88	97.08±0.59	98.28±0.60
K×100	85.35±0.89	80.03±2.92	92.48±1.75	95.40±1.42	94.48±1.22	94.75±1.50	96.93±1.57

TABLE XIII

The Categorized Results for the KSC Dataset Using 3% Training Samples

Class	SVM	CDCNN	SSRN	FDSSC	DBMA	DBDA	Proposed
1	89.75±1.54	94.67±1.74	95.17±1.58	98.97±0.71	98.96±1.00	99.58±0.21	99.42±0.24
2	86.65±2.65	62.59±3.97	93.72±2.01	94.97±2.57	91.96±3.20	98.37±0.67	98.05±1.03
3	66.28±5.25	47.27±10.10	82.05±11.36	78.79±8.20	74.34±6.66	76.42±6.64	79.19±6.59
4	41.40±3.67	34.20±4.45	57.43±9.23	62.66±6.56	61.51±4.46	66.74±4.76	83.24±1.03
5	52.04±4.55	5.50±5.50	62.15±19.78	71.91±19.34	73.26±9.47	72.22±10.40	92.42±3.88
6	54.60±3.45	61.68±6.00	80.83±11.73	84.76±9.03	87.88±6.42	85.85±4.91	98.10±1.79
7	72.43±2.88	17.88±13.45	81.49±9.25	85.36±7.13	85.09±3.05	77.59±5.66	91.87±2.82
8	84.08±2.82	62.07±6.42	92.59±3.23	99.00±0.62	93.71±2.89	97.97±1.22	97.67±0.57
9	82.88±2.70	76.51±2.27	93.38±1.50	99.63±0.23	93.30±2.00	99.52±0.38	99.84±0.12
10	96.48±1.84	73.94±7.53	99.48±0.45	100.0±0.00	96.63±1.96	98.13±1.87	99.25±0.75
11	92.93±0.96	94.69±2.55	97.84±0.87	99.13±0.87	99.95±0.05	99.45±0.35	98.49±0.84
12	90.61±2.56	83.50±4.64	97.01±1.22	98.71±0.49	93.95±1.18	97.06±1.77	98.50±0.77
13	99.78±0.17	98.21±0.33	99.95±0.05	100.0±0.00	99.77±0.23	99.73±0.27	100.0±0.00
OA	84.10±2.27	75.88±3.13	89.72±1.87	94.22±2.64	91.89±1.35	94.09±0.86	96.68±1.17
AA	77.68±1.86	62.51±6.27	87.16±2.78	90.30±6.12	88.49±2.34	89.90±2.06	95.08±1.83
K×100	82.28±2.54	73.11±3.49	88.54±2.09	93.56±2.93	90.97±1.51	93.42±0.96	96.30±1.30

TABLE XIV

The Categorized Results for the BS Dataset Using 1% Training Samples

Class	SVM	CDCNN	SSRN	FDSSC	DBMA	DBDA	Proposed
1	99.85±0.15	71.31±18.08	99.33±0.41	96.96±1.65	98.16±0.68	96.31±1.46	98.15±0.20
2	77.30±5.42	44.29±16.83	92.30±3.35	82.61±8.66	95.55±3.73	91.09±6.82	89.77±6.68
3	74.28±5.94	67.96±17.77	99.41±0.50	100.0±0.00	98.76±0.84	99.84±0.10	99.58±0.26
4	57.35±3.13	43.36±18.85	81.01±5.13	82.32±4.02	81.99±4.47	83.52±4.11	92.12±1.79
5	82.65±2.10	57.80±15.31	86.27±5.19	89.41±3.42	86.74±4.43	87.43±4.90	85.29±4.44
6	53.09±4.22	52.47±13.71	90.93±3.97	95.46±2.27	89.21±3.86	87.97±3.02	94.84±2.47
7	95.29±4.14	87.84±4.85	100.0±0.00	97.11±2.60	94.58±2.36	98.30±1.42	97.86±1.57
8	75.02±6.92	59.67±16.04	95.21±1.90	96.98±1.29	98.93±0.84	100.0±0.00	96.98±1.27
9	70.95±4.50	71.29±18.55	90.24±2.63	87.71±6.26	95.47±4.05	99.55±0.45	95.61±2.05
10	68.53±3.80	65.13±17.94	86.39±4.84	94.16±3.70	93.08±4.11	95.68±3.61	99.13±0.87
11	93.00±1.57	61.53±16.87	99.05±0.62	99.11±0.89	95.32±3.60	98.94±0.43	99.08±0.36
12	84.85±3.46	52.88±14.96	97.56±1.66	97.26±2.34	97.50±1.53	91.03±8.42	99.00±0.62
13	79.47±4.56	71.31±18.07	97.69±1.38	91.44±4.20	98.90±1.00	99.34±0.66	99.77±0.23
14	69.90±12.95	57.42±13.93	100.0±0.00	99.78±0.22	98.43±0.98	100.0±0.00	99.56±0.44
OA	74.73±2.45	61.01±26.42	92.90±1.37	92.69±2.80	93.51±2.30	94.24±3.04	95.94±0.93
AA	77.25±1.58	61.73±27.86	93.96±1.18	93.59±2.63	94.47±1.90	94.93±2.68	96.20±1.10
K×100	72.70±2.63	58.50±27.32	92.31±1.48	92.08±3.03	92.97±2.49	93.76±3.29	95.60±1.00

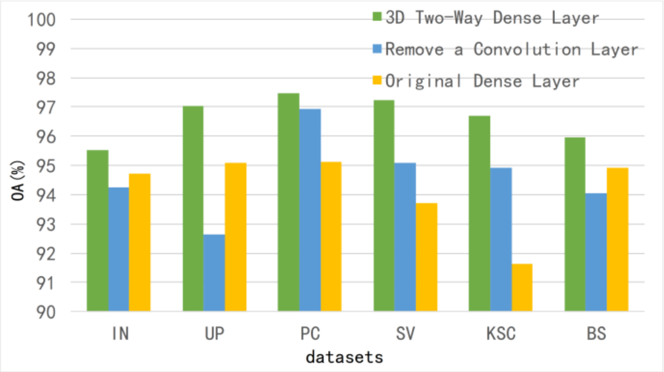


Fig. 9. The OA comparison between 3D two-way dense layer, 3D two-way dense layer without a 3 × 3 × 3 convolution layer, and original dense layer.

As can be seen in Fig. 9, the 3D two-way dense layer indeed promotes the precision on 6 datasets. Averagely, the two-way dense layer improves 2.45% OA on 6 datasets compared to original dense layer. And two stacked 3 × 3 × 3 convolution layers improve 2.01% OA on 6 datasets compared to a single convolution layer.

E. *Effectiveness of Group Convolution*

Since group convolution reduce consumption of convolution, we quantificationally analyzed the effectiveness of group convolution.

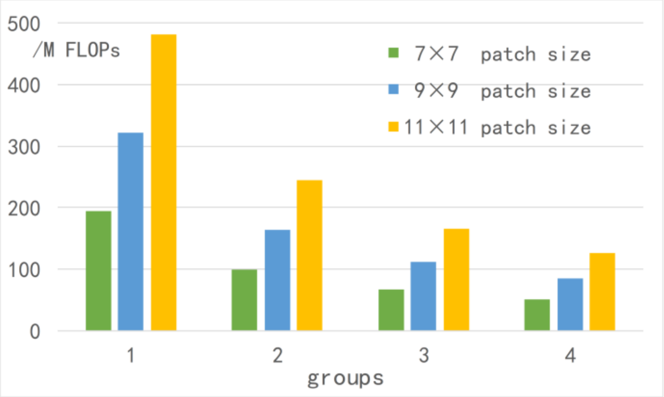


Fig. 10. The computational complexity comparison of LiteDenseNet between different convolution groups with 7 × 7, 9 × 9 and 11 × 11 patch size.

For LiteDenseNet with 1 group, a 9 × 9 × 200 input requires 321.91M FLOPs computing power, while for LiteDenseNet with 3 groups, the consumption is just 111.47M FLOPs. The comparison is provided in Fig. 10. As demonstrated in [55], superfluous groups may significantly reduce running speed, we set the number of groups as 3 for LiteDenseNet.

F. *Investigation of the Number of Patch Size*

The patch size is a crucial factor for 3D-cube-based methods. The larger patch size means the more adjacent spatial information are taken into consideration. However, the larger patch size also leads to the more computation complexity. We set patch size of input as 3 × 3, 5 × 5, 7 × 7, 9 × 9, 11 × 11 and 13 × 13 to evaluate the OA in 6 datasets.

As shown in Fig. 11, the accuracy is near the peak when the patch size is 9 × 9. And the 11 × 11 patch size just brings a negligible promotion in accuracy compared with 9 × 9. There is even a drop in accuracy on BS dataset, which may be due to the sparsity of the samples in BS. Thus, we chose 9 × 9 as the patch size of the input for LiteDenseNet.

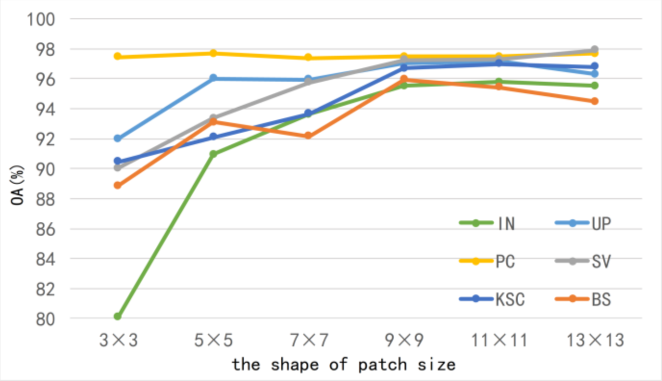


Fig. 11. The OA comparison between different patch size of LiteDenseNet.

G. *Investigation of the Number of Training Samples*

The performance of deep learning is highly dependent on a mass of labelled samples. In this part, we investigate scenarios with training samples in varying proportions.

Sure enough, the accuracy increases with increasing number of training samples. As long as sufficient samples are provided, all methods obtain almost perfect performances. Meanwhile, the accuracy gaps between different methods are narrowing with increasing training samples. It is worth noting that our LiteDenseNet surpasses other methods upon most occasions. Even though the samples are finite, LiteDenseNet still deliver robust performance.

V. Conclusion

In this paper, for HSI Classification, we design a lightweight network architecture (LiteDenseNet) based on DenseNet. The consumptions of calculation and the number of parameters is observably less than contrapositive deep learning methods.

First, we design a 3D two-way dense layer. The top way contains two stacked 3 × 3 × 3 convolution layer which exploits global visual patterns. And the bottom way captures local visual information using a single 3 × 3 × 3 convolution layer.

Second, as 3D-CNN is a computationally intensive operation, we introduce group convolution into LiteDenseNet. A great deal of quantitative comparisons between different methods illustrate that group convolution dramatically reduces the calculation cost and parameter size.

Third, a series of quantitative experiments on 6 widely-used datasets demonstrate that the proposed method obtains the state-of-the-art performance, even though when the absence of labelled samples is severe.

Even though the calculation cost and parameter size of LiteDenseNet are observably less than other methods, the actual time consumptions for training and testing of LiteDenseNet are not distinctly better than other methods. We think a significant factor is that the size of feature maps of LiteDenseNet are

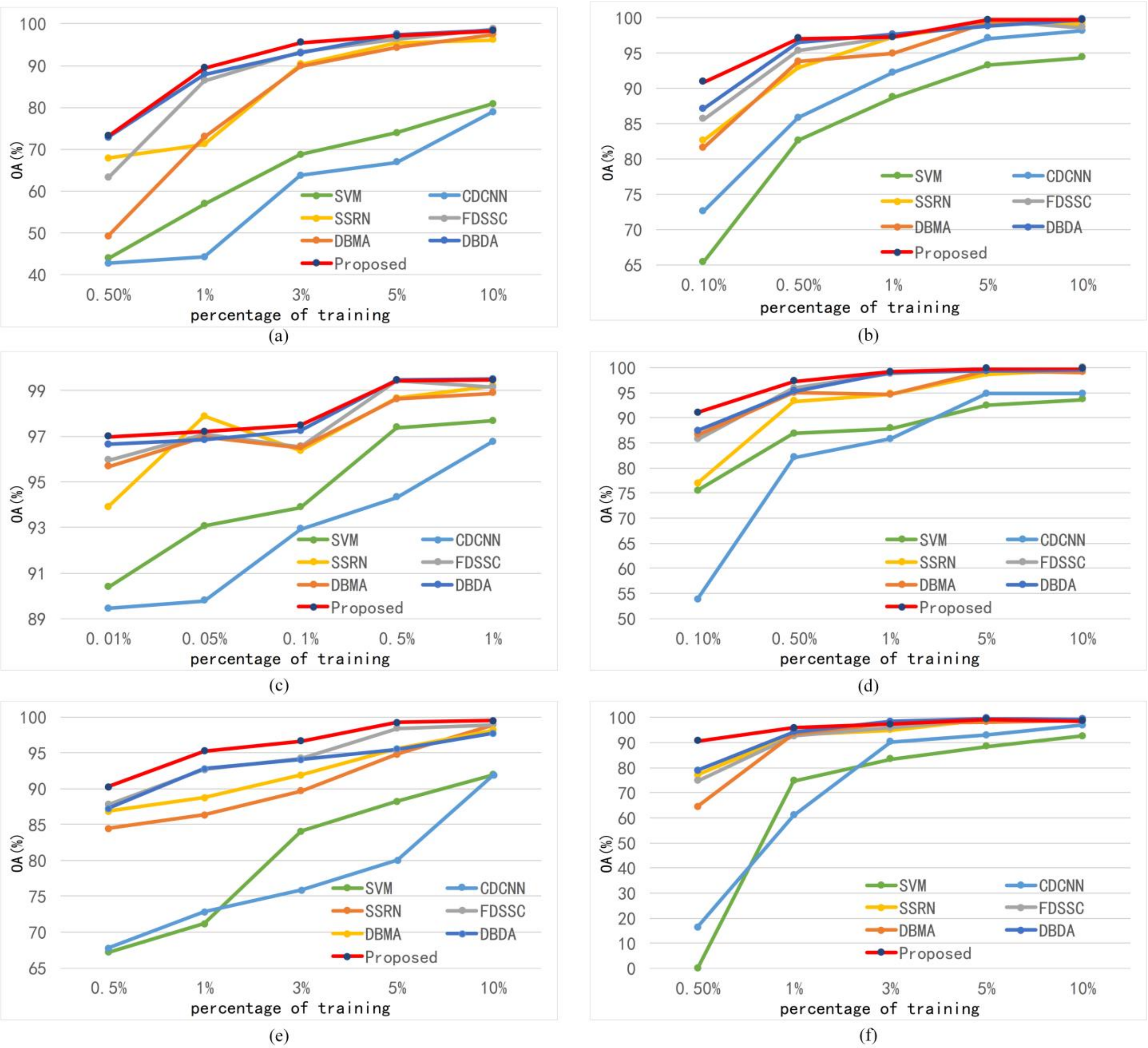


Fig. 12. The OA of SVM, CDCNN, CDCNN, SSRN, FDSSC, DBMA, DBDA and proposed LiteDenseNet with different ratios of training samples on the (a) IN, (b) UP, (c) PC, (d) SV, (e) KSC and (f) BS.

obviously larger than other methods. Thus, it is the memory access cost (MAC) which limits the speed of our method. An important future work is how to reduce the MAC and optimize LiteDenseNet further.

References

[1] Y. Zhong, A. Ma, Y. soon Ong, Z. Zhu, and L. Zhang, "Computational intelligence in optical remote sensing image processing," Appl. Soft Comput., vol. 64, pp. 75–93, 2018.

[2] P. Wang, L. Zhang, G. Zhang, H. Bi, M. Dalla Mura, and J. Chanussot, "Superresolution land cover mapping based on pixel-, subpixel-, and superpixel-scale spatial dependence with pansharpening technique," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 10, pp. 4082–4098, 2019.

[3] G. Wang, J. Li, W. Sun, B. Xue, A. Yinglan, and T. Liu, "Non-point source pollution risks in a drinking water protection zone based on remote sensing data embedded within a nutrient budget model," *Water Res.*, vol. 157, pp. 238–246, 2019.

[4] Z. Zhang, Y. Liu, T. Liu, Z. Lin, and S. Wang, "Dagn: A real-time uav remote sensing image vehicle detection framework," *IEEE Geosci. Remote Sens. Lett.*, 2019.

[5] Z. Li, L. Huang, and J. He, "A multiscale deep middle-level feature fusion network for hyperspectral classification," *Remote Sens.*, vol. 11, no. 6, p. 695, 2019.

[6] Y. Hong, L. Guo, S. Chen, M. Linderman, A. M. Mouazen, L. Yu, Y. Chen, Y. Liu, Y. Liu, H. Cheng et al., "Exploring the potential of airborne hyperspectral image for estimating topsoil organic carbon: Effects of fractional order derivative and optimal band combination algorithm," *Geoderma*, vol. 365, p. 114228, 2020.

[7] A. D. Rocha, T. A. Groen, and A. K. Skidmore, "Spatially-explicit modelling with support of hyperspectral data can improve prediction of plant traits," *Remote Sens. Environ.*, vol. 231, p. 111200, 2019.

[8] W. Xie, T. Jiang, Y. Li, X. Jia, and J. Lei, "Structure tensor and guided filtering-based algorithm for hyperspectral anomaly detection," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 7, pp. 4218–4230, 2019.

[9] 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, 2004.

[10] Q. Du and C.-I. Chang, "A linear constrained distance-based discriminant analysis for hyperspectral image classification," *Pattern Recogn.*, vol. 34, no. 2, pp. 361–373, 2001.

[11] J. Ediriwickrema and S. Khorram, "Hierarchical maximum-likelihood classification for improved accuracies," *IEEE Trans. Geosci. Remote Sens.*, vol. 35, no. 4, pp. 810–816, 1997.

[12] J. C.-W. Chan and D. Paelinckx, "Evaluation of random forest and adaboost tree-based ensemble classification and spectral band selection for ecotope mapping using airborne hyperspectral imagery," *Remote Sens. Environ.*, vol. 112, no. 6, pp. 2999–3011, 2008.

[13] J. Ham, Y. Chen, M. M. Crawford, and J. Ghosh, "Investigation of the random forest framework for classification of hyperspectral data," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 3, pp. 492– 501, 2005.

[14] Z. Zhu, S. Jia, S. He, Y. Sun, Z. Ji, and L. Shen, "Three-dimensional gabor feature extraction for hyperspectral imagery classification using a memetic framework," *Inform. Sciences*, vol. 298, pp. 274–287, 2015.

[15] T. C. Bau, S. Sarkar, and G. Healey, "Hyperspectral region classification using a three-dimensional gabor filterbank," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 9, pp. 3457–3464, 2010.

[16] Y. Y. Tang, Y. Lu, and H. Yuan, "Hyperspectral image classification based on three-dimensional scattering wavelet transform," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 5, pp. 2467–2480, 2014.

[17] X. Cao, L. Xu, D. Meng, Q. Zhao, and Z. Xu, "Integration of 3-dimensional discrete wavelet transform and markov random field for hyperspectral image classification," *Neurocomputing*, vol. 226, pp. 90–100, 2017.

[18] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 3730–3738.

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," [Online]. Available: https://arxiv.org/abs/1810.04805

[20] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," [Online]. Available: https://arxiv.org/abs/1904.08779

[21] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2094–2107, 2014.

[22] X. Zhang, Y. Liang, C. Li, N. Huyan, L. Jiao, and H. Zhou, "Recursive autoencoders-based unsupervised feature learning for hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 11, pp. 1928–1932, 2017.

[23] Y. Chen, X. Zhao, and X. Jia, "Spectral–spatial classification of hyperspectral data based on deep belief network," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 6, pp. 2381– 2392, 2015.

[24] P. Zhou, J. Han, G. Cheng, and B. Zhang, "Learning compact and discriminative stacked autoencoder for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 7, pp. 4823– 4833, 2019.

[25] X. Ma, H. Wang, and J. Geng, "Spectral–spatial classification of hyperspectral image based on deep auto-encoder," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 9, pp. 4073– 4085, 2016.

[26] W. Zhao and S. Du, "Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 8, pp. 4544–4554, 2016.

[27] 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, 2017.

[28] 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, 2016.

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

[30] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 4700–4708.

[31] 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, 2017.

[32] W. Wang, S. Dou, Z. Jiang, and L. Sun, "A fast dense spectral–spatial convolution network framework for hyperspectral images classification," *Remote Sens.*, vol. 10, no. 7, p. 1068, 2018.

[33] J. M. Haut, M. E. Paoletti, J. Plaza, A. Plaza, and J. Li, "Visual attention-driven hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 10, pp. 8065–8080, 2019.

[34] W. Ma, Q. Yang, Y. Wu, W. Zhao, and X. Zhang, "Double-branch multi-attention mechanism network for hyperspectral image classification," *Remote Sens.*, vol. 11, no. 11, p. 1307, 2019.

[35] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "Cbam: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19.

[36] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 3146–3154.

[37] R. Li, S. Zheng, C. Duan, Y. Yang, and X. Wang, "Classification of hyperspectral image based on double-branch dual-attention mechanism network," *Remote Sens.*, vol. 12, no. 3, p. 582, 2020.

[38] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep recurrent neural networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3639–3655, 2017.

[39] X. Cao, J. Yao, Z. Xu, and D. Meng, "Hyperspectral image classification with convolutional neural network and active learning," *IEEE Trans. Geosci. Remote Sens.*, 2020.

[40] L. Zhu, Y. Chen, P. Ghamisi, and J. A. Benediktsson, "Generative adversarial networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 9, pp. 5046–5063, 2018.

[41] H. Wu and S. Prasad, "Semi-supervised deep learning using pseudo labels for hyperspectral image classification," *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1259–1270, 2017.

[42] M. E. Paoletti, J. M. Haut, R. Fernandez-Beltran, J. Plaza, A. Plaza, J. Li, and F. Pla, "Capsule networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 4, pp. 2145–2160, 2018.

[43] S. Zhang, S. Li, W. Fu, and L. Fang, "Multiscale superpixel-based sparse representation for hyperspectral image classification," *Remote Sens.*, vol. 9, no. 2, p. 139, 2017.

[44] R. Kemker and C. Kanan, "Self-taught feature learning for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 5, pp. 2693–2705, 2017.

[45] J. Peng, W. Sun, and Q. Du, "Self-paced joint sparse representation for the classification of hyperspectral images," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 1183–1194, 2018.

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 1–9.

[47] R. J. Wang, X. Li, and C. X. Ling, "Pelee: A real-time object detection system on mobile devices," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 1963–1972.

[48] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," [Online]. Available: https://arxiv.org/abs/1502.03167

[49] 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," [Online]. Available: https://arxiv.org/abs/1602.07360

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

[51] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," [Online]. Available: https://arxiv.org/abs/1704.04861

[52] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 4510–4520.

[53] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan et al., “Searching for mobilenetv3,” *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 1314–1324.

[54] X. Zhang, X. Zhou, M. Lin, and J. Sun, “Shufflenet: An extremely efficient convolutional neural network for mobile devices,” *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 6848–6856.

[55] 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.

[56] I. Loshchilov and F. Hutter, “Sgdr: Stochastic gradient descent with warm restarts,” [Online]. Available: https://arxiv.org/abs/1608.0398

Rui Li received the bachelor’s degree in School of Automation Science and Engineering, South China University of Technology, Guangzhou, China in 2019. He is currently pursuing the master’s degree with the School of Remote Sensing and Information Engineering, Wuhan University, Wuhan, China.

His research interests include hyperspectral image classification, machine learning, and deep learning.

Chenxi Duan received the bachelor’s degree in College of Geology Engineering and Geomatics, Chang’an University, Xi’an, China in 2019. She is currently pursuing the master’s degree with the State Key Laboratory of Information Engineering in Surveying, Mapping, and Remote Sensing, Wuhan University, Wuhan, China.

Her research interests include cloud removal, machine learning, and deep learning.

Shunyi Zheng received the Post-Doctorate from the State Key Laboratory of Information Engineering in Surveying, Mapping, and Remote Sensing, Wuhan University, Wuhan, China, in 2002. He is currently a Professor with School of Remote Sensing and Information Engineering, Wuhan University, Wuhan, China. His research interests include remote sensing data processing, digital photogrammetry and three-dimensional reconstruction.

Mr. Zheng’s awards and honors include the First prize for scientific and technological progress in surveying and mapping 2012, the First prize for 2014 John I. Davidson President’s Award, and the First prize for scientific and technological progress in surveying and mapping 2019,.