测绘出版社
SURVEYING AND MAPPING PRESS

A Deep Double-Channel Dense Network for Hyperspectral Image Classification

Kexian WANG[1], Shunyi ZHENG[1], Rui LI[1], Li GUI[2]

1. *School of Remote Sensing and Information Engineering*, *Wuhan University*, *Wuhan* 430079, *China*; 2. *School of Electronic Information*, *Wuhan University*, *Wuhan* 430072, *China*

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, the deep learning method usually requires numerous computational resources and high-quality labelled datasets, while the expenditures of high-performance computing and data annotation are expensive. In this paper, to reduce the dependence on massive calculation and labelled samples, we propose a deep Double-Channel dense network (DDCD) for Hyperspectral Image Classification. Specifically, we design a 3D Double-Channel dense layer to capture the local and global features of the input. And we propose a Linear Attention Mechanism that is approximate to dot-product attention with much less memory and computational costs. The number of parameters and the consumptions of calculation are observably less than contrapositive deep learning methods, which means DDCD owns simpler architecture and higher efficiency. A series of quantitative experiences on 6 widely used hyperspectral datasets show that the proposed DDCD obtains state-of-the-art performance, even though when the absence of labelled samples is severe.

Key words: 3D Double-Channel dense layer; Linear Attention Mechanism; Deep Learning (DL); hyperspectral classification

Citation: Kexian WANG, Shunyi ZHENG, Rui LI, et al. A Deep Double-Channel Dense Network for Hyperspectral Image Classification[J]. Journal of Geodesy and Geoinformation Science, 2021, 4(4): 46-62. DOI: 10.11947/j.JGGS.2021.0404.

1 Introduction

Researches about hyperspectral image (HSI) have become a heated research area in recent years. HSI is a kind of optical remote sensing image with high spectral resolution[1]. And a variety of applications have been developed in many areas such as topsoil organic carbon estimation[2], plant traits prediction[3], and salient object detection[4], among others. 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.

Early attempts, including support vector machines (SVM)[5], distance measure (DM) calculation[6], and maximum likelihood (MLH) criterion[7], focus on spectral features of HSI. Multiple classifier methods and ensemble learning are also introduced such as AdaBoost[8] and Random Forests (RF)[9]. However, the adjacent pixels possibly belong to the same category, which is neglected by the above-mentioned spectral-based methods. Meanwhile, HSI datasets are normally organized as 3D cubes format. Thus, it is feasible to integrate the spatial and spectral features in complementary form. Inspired by this insight, 3D Gabor filter[10-11], 3D scattering wavelet transform[12], and 3D discrete wavelet transform[13] are proposed. However, the high dependency on hand-crafted descriptors restricts the flexibility and adaptability of these methods.

Deep Learning (DL) is powerful to capture

Received date: 2021-02-28; accepted date: 2021-08-30
Foundation support: National Natural Science Foundations of China (No. 41671452); China Postdoctoral Science Foundation Funded Project (No. 2017M612510)
First author: Kexian WANG (1998—), male, majors in hyperspectral image classification and deep learning.
E-mail: kxwang@whu.edu.cn
Corresponding author: Rui LI
E-mail: lironui@whu.edu.cn

nonlinear and hierarchical features automatically, and has influenced many domains such as computer vision (CV)[14-16], and has shown its impact on photogrammetry[17-19]. In Literature [20], He et al. proposed an Encoder-Decoder network for road extraction. Zuo et al.[21] introduced a deformable convolution to enhance the adaptability of convolutional networks to spatial transformation. Zheng et al.[22] designed a fast patch-free global learning (FPGA) framework. Jia et al.[23] proposed an end-to-end deep U-net-based model.

Deep Learning is also applied to HSI classification. In Literature [24], Sun et al. proposed an end-to-end fully convolutional segmentation network (FCSN) to simultaneously identify labels of all pixels in an HSI cube. Pan et al.[25] designed DSSNet which can extract spatial-spectral information via an end-to-end manner. 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 Literature [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 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 mechanisms 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).

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 costs of computing and annotation are rather expensive. Thousands of samples should be labelled for training to obtain a passable performance, and the experiments are often conducted on high-powered GPU.

DBDA[37] uses the dot-product attention mechanism, the memory and calculation cost of the dot-product attention mechanism increases quadratically with the size of the input time and space. The complexity of the dot-product attention mechanism is 9 $O(N^2)$, so we introduce linear attention to reduce the complexity to $O(N)$.

In this paper, to reduce the dependence on massive calculation and labelled samples, we design a 3D Double-Channel network architecture (DDCD) for HSI Classification. Firstly, the proposed framework contains a 3D Double-Channel dense layer to obtain receptive fields and extract the visual features in different scales. Subsequently, the extracted features are fed into a 3D convolution layer to reduce the dimension. Whereafter, the spectral and spatial features are refined by the proposed Linear Attention Mechanism separately. Finally, we flatten the dimension reduction of extracted features and determine the category by the softmax function. The existence of the 3D Double-Channel dense layer significantly enhances the capacity of feature extraction. Thus, a small number of labelled samples are sufficient to train an excellent model. The number of parameters and the consumptions of calculation are observably less than contrapositive deep learning methods. The four significant contributions of this paper could be listed as follows:

① Based on DenseNet, We propose a 3D Double-Channel dense layer that respectively captures the local and global features of the input. The 3D Double-Channel dense layer can obtain receptive fields at different levels and efficiently extract visual features.

② We proposed a linear attention mechanism which reduces the complexity of the attention mechanism from $O(N^2)$ to $O(N)$.

③ Based on 3D Double-Channel dense layer, we propose an end-to-end framework. The number of parameters and the consumptions of calculation are observably less than contrapositive deep learning methods.

④ Substantial experiments are conducted on 6 widely used HSI benchmarks. A sequence of quantitative experiences on 6 widely used datasets and the results show that the proposed DDCD obtains state-of-the-art performance.

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

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

2 Methodology

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

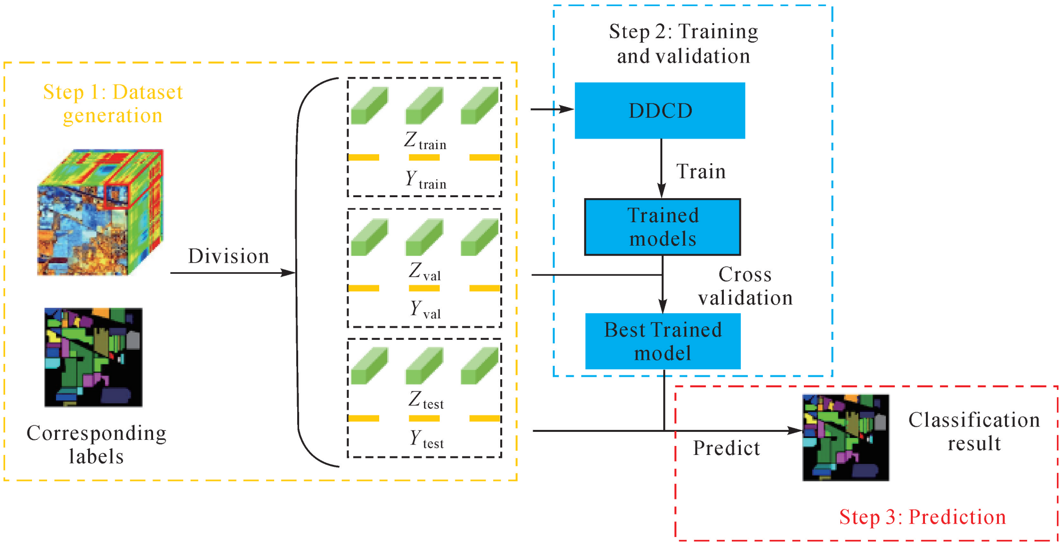

Fig.1 Three steps of the DDCD

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 $\{z_1, z_2, \cdots, z_n\} \mathbb{R}^{p\times p\times b}$. Next, the 3D-cube Z is 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 whole framework of DDCD can be seen in Fig.2, and the flowchart of DDCD is shown in Fig.3. The following part provides a detailed procedure of DDCD illustrated with the Indian Pines (IP) dataset example. The IP dataset comprises 145×145 pixels with 200 bands, which means the size of IN is 145×145×200. Nevertheless, only 10, 249 pixels own corresponding labels, and the others are background. Tab.2 gives the details of IP. The patch size is set as 9×9. For the matrixes mentioned below such as (9×9×97, 24), the 9×9×97 denotes the numerical value of height, width and depth of the 3D-cube, and 24 means the count of 3D-cubes generated by the convolution layer. And details about the implements are provided in Tab.1. The sensitivity analysis of the parameters in Tab.1 is provided in Section.

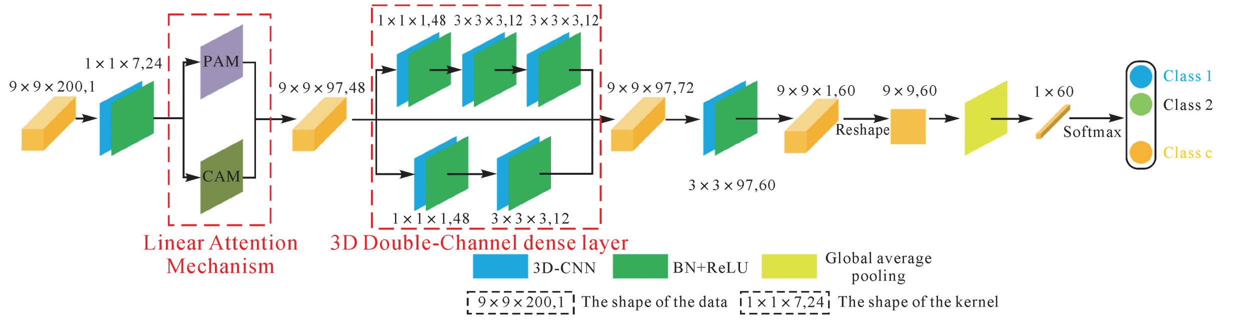

Fig.2 The structure of the DDCD

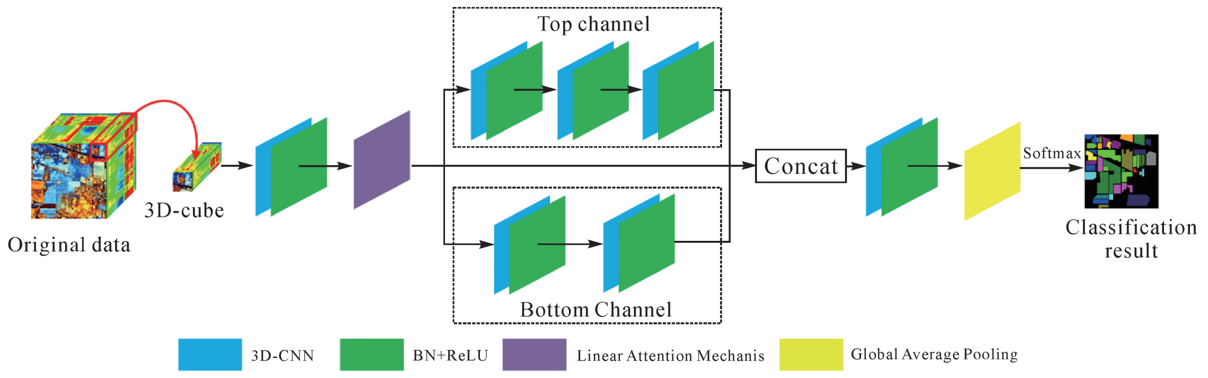

Fig.3 The flowchart of the DDCD methodology

Tab.1 The implements details about DDCD

Layer name		Kernel size	Group	Output size
Input		-	-	(9×9×200,1)
3D-CNN+BN+ReLU		(1×1×7)	1	(9×9×97,24)
Linear Attention		-	-	(9×9×97,24)
Mechanism		-	-	(9×9×97,24)
Concatenate		-	-	(9×9×97,48)
3D two-way dense layer	3D-CNN+ BN+ReLU	(1×1×1)	3	(9×9×97,48)
	3D-CNN+ BN+ReLU	(3×3×3)	3	(9×9×97,12)
	3D-CNN+ BN+ReLU	(3×3×3)	3	(9×9×97,12)
	3D-CNN+ BN+ReLU	(1×1×1)	3	(9×9×97,48)
	3D-CNN+ BN+ReLU	(3×3×3)	3	(9×9×97,12)
Concatenate		-	-	(9×9×97,72)
3D-CNN+BN+ReLU		(3×3×97)	3	(9×9×1,60)
Global Average Pooling		-	-	(1×60)

2.1 Spectrum extraction using 3D-CNN

3D-CNN is capable of obtaining spatial and spectral features simultaneously, and Batch Normalization (BN) layers can improve numerical stability. Thus, we consider 3D-CNN with a BN layer as an example to elaborate the mechanism of 3D-CNN. Supposing that the size of input 3D feature maps is expressed as (height×width×depth, channels), and the shape of the convolution kernel is (k_h×k_w×k_d, channels). The operation of 3D-CNN is similar to the operation of 2D-CNN. The convolution operations are implemented between the convolution kernel and sliding windows in the shape of ($k_h \times k_w \times k_d$), and obtained values constitute the output 3D feature maps. Another important parameter, stride, determines the distance of width and height traversed per slide of the sliding windows. A diagrammatic sketch can be seen in Fig.4. Concretely, the operation of 3D-CNN can be formalized as

$$\boldsymbol{X}_{i,j}^{x,y,z} = \sum_m \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} \boldsymbol{W}_{i,j,m}^{p,q,r} \boldsymbol{X}_{i-1,m}^{(x+p),(y+q),(z+r)} + b_{i,j} \qquad (1)$$

Where $\boldsymbol{X}_{i,j}^{x,y,z}$ denotes the jth feature cube at position (x,y,z) in the ith layer; m means the feature maps generated by the $(i-1)$ layer; $\boldsymbol{W}_{i,j,m}^{p,q,r}$ represent the column weight of the mth feature cube at position (p,q,r); $b_{i,j}$ is the jth feature cube in the ith layer's bias items of the filter; and P_i and Q_i respectively express the height and width of the kernel, while R_i

means spectral bands of HSI.

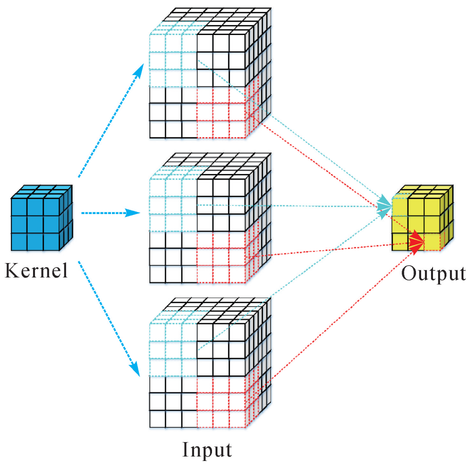

Fig.4 The diagrammatic sketch of the 3D-CNN with a BN

Then, the output of the BN layer can be calculated as

$$\hat{\boldsymbol{x}}_i=\frac{\boldsymbol{x}_i-E(\boldsymbol{x}_i)}{\sqrt{\mathrm{Var}(\boldsymbol{x}_i)+\varepsilon}} \tag{2}$$

$$\boldsymbol{y}_i=\sigma(\gamma_i\hat{\boldsymbol{x}}_i+\beta_i) \tag{3}$$

Where $\boldsymbol{x}_i$ is the output of the BN layer. Var(·) and E(·) represent the variance function and expectation of the input. γ and β are two trainable parameters, and the normalized result $\hat{\boldsymbol{x}}_i$ can be scaled by γ and shifted by β. σ(·) expresses the activation function, which is set as ReLU in our model.

As the quality of extracted features restricts the performance of the model and the convolution kernel size determines the receiving field, how to design the size of the convolution kernel is the crux of the network[38]. If the kernel size is set as 1×1, then the layer simply realizes linear combinations or integrates spatial pixels. Hence, the kernel in the shape of $1\times1\times d$ ($d>1$) could exploit the spectral features while reserving the spatial features, as kernel sizes of width and height are set to 1.

The encoded spectral features within bands lead to the high dimensionality of hyperspectral image. Nevertheless, a kernel in the shape of ($1\times1\times d$) and with a stride in the ($1,1,s_d$) can refine the high-dimensional bands of the hyperspectral image, where $s_d>1$.

Hence, in DDCD, a 3D-CNN layer with the size of (1×1×7) is firstly adopted to refine the spectral features and reduce the spectral dimensions. And the stride is set as (1,1,2).

2.2 Feature abstraction by Double-Channel dense block

Normally, the deeper convolutional network would obtain better performance. Nevertheless, substantial layers and excess parameters cause 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.5(a) 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

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

Where H_m represents a hidden block comprised of convolution layers, BN layers, and activation layers. $x_0,x_1,\cdots,x_{m-1}$ denote the output generated by the 1st, 2nd, ⋯, $m-1$th dense blocks. As shown in Fig.5(b), sufficient connections ensure adequate information flow through the network. An m-layer DenseNet owns $m(m+1)/2$ connections, while general convolutional network with m layers only has m connections.

Supposing the channels' number of x_0 in Fig.5(b) is c_0, and each block contains k kernels. Since DenseNet directly concatenates the output of each block at the channel dimension, so the channels' number of c _ m in Fig. 5 (b) can be formulated as

$$c_m=c_0+(m-1)\times k \tag{5}$$

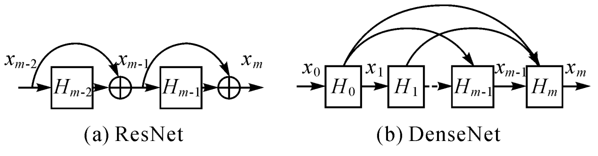

Fig.5 The structure of ResNet and DenseNet

When applied to HSI classification, dense blocks which usually stack three dense layers of the same size are designed to extract spectral and spatial features from HSI. The dense spectral block with the size of $(1\times1\times k_d)$ is capable of exploiting spectral features, and the dense spatial block with the size of $(k_h\times k_w\times1)$ is able to capture spatial features. The above operations depict the general utilization of dense blocks, which can be seen in FDSSC, and DBMA. And the framework of FDSSC is shown in Fig.6. However, there are two points that need to discuss the usage of the dense block.

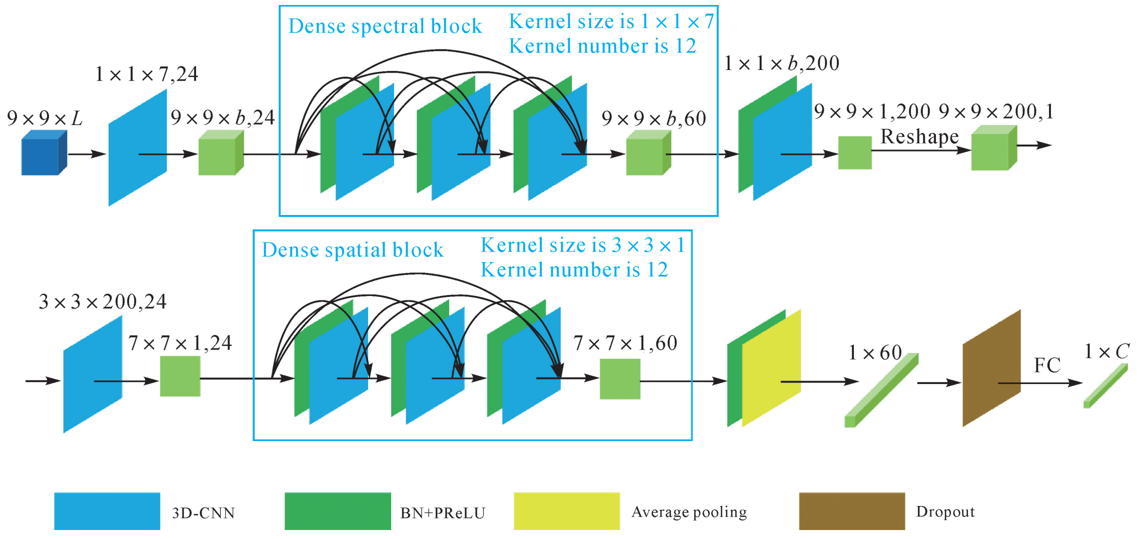

Fig.6 The framework of FDSSC

FDSSC firstly captures spectral features and subsequently exploits spatial features, and the output of the dense spectral block serves as the input of dense spatial block, while the later procedure of spatial feature extraction might undermine the former extracted spectral features. DBMA copes with this problem by designing double branches to capture spectral and spatial features respectively. As for DDCD, we extract spectral and spatial features simultaneously.

Another issue is that the dense blocks used in FDSSC and DBMA are in the same size, which means the receiving field of a dense block is constant. Generally, the larger the convolution kernel size, the larger the receiving field and the more global vision, which augments the scope of areas observed in the image and enhances feature extraction. Conversely, a decrease in the size of the convolution kernel would induce the receiving field and obtain the local vision. However, the global visual patterns and the local visual patterns both contain available features. Thus, in DDCD, we design a Double-Channel dense block to exploit the global and local features simultaneously.

The structure of the Double-Channel dense block can be seen in Fig. 7, with the input 3D feature maps in the shape of $(p\times p\times b, c_0)$. The top channel of the block contains a 1×1×1 convolution layer and two stacked 3×3×3 convolution layers. The 1×1×1 convolution layer transforms the channels of the input, while the receiving field of two stacked 3×3×3 convolution layers is equivalent to a 7×7×7 convolution layers. Thus, the top channel is capable of capturing global visual patterns and generates the output in the shape of $(p\times p\times b, 12)$. Similarly, the bottom Channel of the block harnesses a 1×1×1 convolution layer to transform the channels of the input. And the attached single 3×3×3 convolution layer exploits local visual patterns, which also generate the output in the shape of $(p\times p\times b, 12)$. Subsequently, the concat operation is implemented between the input and the outputs of the top channel and the bottom channel, and obtains the feature maps with the size of $(p\times p\times b, c_0+24)$.

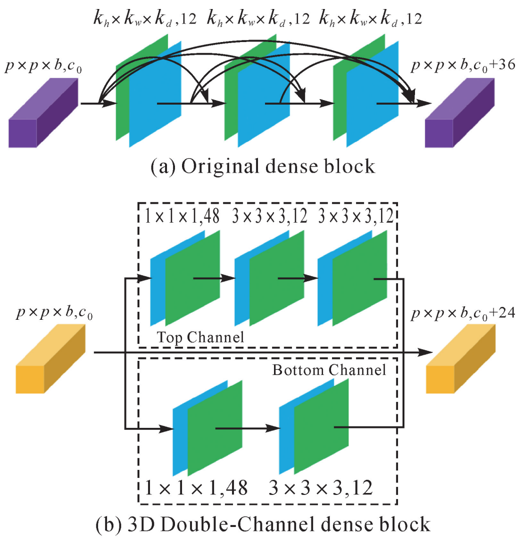
$k_h\times k_w\times k_d$,12
$k_h\times k_w\times k_d$,12
$k_h\times k_w\times k_d$,12
$p\times p\times b, c_0$
$p\times p\times b, c_0+36$
(a) Original dense block
1×1×1,48
3×3×3,12
3×3×3,12
$p\times p\times b, c_0$
$p\times p\times b, c_0+24$
Top Channel
Bottom Channel
1×1×1,48
3×3×3,12
(b) 3D Double-Channel dense block
🟦 3D-CNN
🟩 BN+ReLU

Fig.7 The structure of the Double-Channel dense block

2.3 Dimension reduction and information aggregation

In this section, we firstly harness a $(3\times3\times97, 60)$ 3D-CNN layer to reduce the dimension of bands and reshape the output into $(9\times9, 60)$. Then, a global average pooling operation is implemented to obtain the feature maps and aggregate information. Finally, the classification result is generated via a fully connected layer and softmax function.

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

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

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

2.4 Linear attention mechanism

In this section, we refer to the linear attention mechanism[39-40] and extend it from 2D to 3D. We illustrate that substituting the attention from the conventional softmax attention to the first-order approximation of Taylor expansion leads to linear time and memory complexity.

Supposing N and D_x denote the length of input sequences and the number of input dimensions, given a feature $\boldsymbol{X}=[\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N] \in \mathbb{R}^{N\times D_x}$, dot-product attention utilize three projected matrices $\boldsymbol{W}_q \in \mathbb{R}^{D_x\times D_k}$, $\boldsymbol{W}_k \in \mathbb{R}^{D_x\times D_k}$, and $\boldsymbol{W}_v \in \mathbb{R}^{D_x\times D_v}$ to generate corresponding query matrix $\boldsymbol{Q}$, the key matrix $\boldsymbol{K}$, and the value matrix $\boldsymbol{V}$

$$\left.\begin{aligned} \boldsymbol{Q} &= \boldsymbol{X}\boldsymbol{W}_q \in \mathbb{R}^{N\times D_k}, \\ \boldsymbol{K} &= \boldsymbol{X}\boldsymbol{W}_k \in \mathbb{R}^{N\times D_k}, \\ \boldsymbol{V} &= \boldsymbol{X}\boldsymbol{W}_v \in \mathbb{R}^{N\times D_v} \end{aligned}\right\} \quad (7)$$

A normalization function ρ evaluates the similarity between the ith query feature $\boldsymbol{q}_i^{\mathrm{T}} \in \mathbb{R}^{D_k}$ and the jth key feature $\boldsymbol{k}_j \in \mathbb{R}^{D_k}$ by $\rho(\boldsymbol{q}_i^{\mathrm{T}}\boldsymbol{k}_j) \in \mathbb{R}^1$. The dot-product attention module computes the value at position i via aggregating the value features from all positions based on weighted summation

$$D(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \rho(\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}})\boldsymbol{V} \quad (8)$$

The softmax is the common normalization function

$$\rho(\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}) = \mathrm{softmax}_{\mathrm{row}}(\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}) \quad (9)$$

Where $\mathrm{softmax}_{\mathrm{row}}$ indicates applying the softmax function along each row of matrix $\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}$.

The $\rho(\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}})$ models the similarities between all pairs of positions. However, as $\boldsymbol{Q} \in \mathbb{R}^{N\times D_k}$ and $\boldsymbol{K}^{\mathrm{T}} \in \mathbb{R}^{D_k\times N}$, the product between $\boldsymbol{Q}$ and $\boldsymbol{K}^{\mathrm{T}}$ belongs to $\mathbb{R}^{N\times N}$, which leads to $O(N^2)$ memory complexity and $O(N^2)$ computational complexity. Therefore, the high resource-demanding of dot-product greatly limits the application of large inputs. An improvement method is introducing the kernel method.

Under the condition of softmax normalization function, the ith row of result matrix generated by dot-product attention module according to Eq. (8) can be written as

$$D(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})_i = \frac{\sum_{j=1}^{N} e^{q_i^{\mathrm{T}} k_j} \boldsymbol{v}_j}{\sum_{j=1}^{N} e^{q_i^{\mathrm{T}} k_j}} \quad (10)$$

Then, Eq. (10) can be generalized for any normalization function and rewritten as

$$\left.\begin{aligned} D(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})_i &= \frac{\sum_{j=1}^{N} \operatorname{sim}(\boldsymbol{q}_i,\boldsymbol{k}_j)\boldsymbol{v}_j}{\sum_{j=1}^{N} \operatorname{sim}(\boldsymbol{q}_i,\boldsymbol{k}_j)} \\ \operatorname{sim}(\boldsymbol{q}_i,\boldsymbol{k}_j) &\geqslant 0 \end{aligned}\right\} \tag{11}$$

If $\operatorname{sim}(\boldsymbol{q}_i,\boldsymbol{k}_j)=e^{q_i^{\mathrm{T}}k_j}$, Eq. (11) is equivalent to Eq. (4). And $\operatorname{sim}(\boldsymbol{q}_i,\boldsymbol{k}_j)$ can be further expanded as $\operatorname{sim}(\boldsymbol{q}_i,\boldsymbol{k}_j)=\phi(\boldsymbol{q}_i)^{\mathrm{T}}\varphi(\boldsymbol{k}_j)$, where $\phi(\cdot)$ and $\varphi(\cdot)$ can be seen as kernel smoothers[40]. Then Eq. (10) can be rewritten as

$$D(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})_i = \frac{\sum_{j=1}^{N} \phi(\boldsymbol{q}_i)^{\mathrm{T}}\varphi(\boldsymbol{k}_j)\boldsymbol{v}_j}{\sum_{j=1}^{N} \phi(\boldsymbol{q}_i)^{\mathrm{T}}\varphi(\boldsymbol{k}_j)} \tag{12}$$

and then can simplify as

$$D(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})_i = \frac{\phi(\boldsymbol{q}_i)^{\mathrm{T}}\sum_{j=1}^{N} \varphi(\boldsymbol{k}_j)\ \boldsymbol{v}_j^{\mathrm{T}}}{\phi(\boldsymbol{q}_i)^{\mathrm{T}}\sum_{j=1}^{N} \varphi(\boldsymbol{k}_j)} \tag{13}$$

As $\boldsymbol{K}\in\mathbb{R}^{D_k\times N}$ and $\boldsymbol{V}^{\mathrm{T}}\in\mathbb{R}^{N\times D_v}$, the product between $\boldsymbol{K}$ and $\boldsymbol{V}^{\mathrm{T}}$ belongs to $\mathbb{R}^{D_k\times D_v}$, which considerably reduces the complexity of the dot-product attention mechanism. For example, Literature [41] adopts $\phi(\boldsymbol{x})=\varphi(\boldsymbol{x})=\operatorname{softplus}(\boldsymbol{x})$.

Different previous researches, we conceive linear attention mechanism based on first-order approximation of Taylor expansion on Eq. (10)

$$e^{\boldsymbol{q}_i^{\mathrm{T}}\boldsymbol{k}_j}\approx 1+\boldsymbol{q}_i^{\mathrm{T}}\boldsymbol{k}_j \tag{14}$$

However, the above approximation cannot guarantee the non-negativity. To ensure $\boldsymbol{q}_i^{\mathrm{T}}\boldsymbol{k}_j\geqslant -1$, we can normalize $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$ by l_2 norm

$$\operatorname{sim}(\boldsymbol{q}_i,\boldsymbol{k}_j) = 1 + \left(\frac{\boldsymbol{q}_i}{\|\boldsymbol{q}_i\|_2}\right)^{\mathrm{T}}\left(\frac{\boldsymbol{k}_j}{\|\boldsymbol{k}_j\|_2}\right) \tag{15}$$

Then, Eq. (12) can be rewritten as

$$D(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})_i = \frac{\sum_{j=1}^{N}\left(1 + \left(\frac{\boldsymbol{q}_i}{\|\boldsymbol{q}_i\|_2}\right)^{\mathrm{T}}\left(\frac{\boldsymbol{k}_j}{\|\boldsymbol{k}_j\|_2}\right)\right)v_j}{\sum_{j=1}^{N}\left(1 + \left(\frac{\boldsymbol{q}_i}{\|\boldsymbol{q}_i\|_2}\right)^{\mathrm{T}}\left(\frac{\boldsymbol{k}_j}{\|\boldsymbol{k}_j\|_2}\right)\right)} \tag{16}$$

and simplified as

$$D(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})_i = \frac{\sum_{j=1}^{N}\boldsymbol{v}_j + \left(\frac{\boldsymbol{q}_i}{\|\boldsymbol{q}_i\|_2}\right)^{\mathrm{T}}\sum_{j=1}^{N}\left(\frac{\boldsymbol{k}_j}{\|\boldsymbol{k}_j\|_2}\right)\boldsymbol{v}_j^{\mathrm{T}}}{N + \left(\frac{\boldsymbol{q}_i}{\|\boldsymbol{q}_i\|_2}\right)^{\mathrm{T}}\sum_{j=1}^{N}\left(\frac{\boldsymbol{k}_j}{\|\boldsymbol{k}_j\|_2}\right)} \tag{17}$$

The above equation can be written ina vectorized form as

$$D(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \frac{\sum_j \boldsymbol{V}_{i,j} + \left(\frac{\boldsymbol{Q}}{\|\boldsymbol{Q}\|_2}\right)\left(\left(\frac{\boldsymbol{K}}{\|\boldsymbol{K}\|_2}\right)^{\mathrm{T}}\boldsymbol{V}\right)}{N + \left(\frac{\boldsymbol{Q}}{\|\boldsymbol{Q}\|_2}\right)\sum_j\left(\frac{\boldsymbol{K}}{\|\boldsymbol{K}\|_2}\right)^{\mathrm{T}}_{i,j}} \tag{18}$$

As $\sum_{j=1}^{N}\left(\frac{k_j}{\|k_j\|_2}\right)\boldsymbol{v}_j^{\mathrm{T}}$ and $\sum_{j=1}^{N}\left(\frac{k_j}{\|k_j\|_2}\right)$ can be calculated and reused for every query, time and memory complexity of the proposed linear attention mechanism based on Eq. (18) is $O(N)$.

3 Experimental Results

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

3.1 Data description

In this paper, we use six HSI datasets, i.e., the Indian Pines (IN) dataset, the Pavia University (UP) dataset, the Pavia Centre (PC) dataset, the-HyRANK (HV) dataset, the Kennedy Space Center (KSC) dataset and the Botswana dataset (BS), to design the experiments. The details of six datasets are provided in Tab.2—Tab.7, and false-color composites and corresponding category labels are rendered in Fig.8.

Indian Pines (IP): 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 Paviacentre, Italy. UP has 1096×715 pixels with 103 bands and 9 land cover categories.

HyRANK (HV): The HV dataset is obtained via the Hyperion sensor (EO-1, USGS) over Dioni.

HV has 250×1376 pixels with 176 bands and 12 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.

3.2 Experimental setting

As we mentioned above, we select very limited training and validation samples to save calculation costs. The proportions of training and validation samples are sent as 3% for IN, HV and KSC, 1.2% for BS, 0.5% for UP, and 0.1% for PC.

To evaluate the performance, we compare DDCD with SVM[5], CDCDD[27], SSRN[31], FDSSC[32] and DBMA[34]. All experiments are implemented on a standard and mediocre laptop, which is configured with 8 GB of physical memory and 20 GB virtual memory, an i7-7700HQ CPU, and an NVIDIA GTX1060 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 a 0.000 5 learning rate, and the optimizer is Adam.

Tab.2 The samples for each class for training, validation, and testing of the Indian Pines (IP) 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	10 249	307	307	9635

Tab.3 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	18 463
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	42 356

Tab.4 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	65 841
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	42 742
9	Shadows	2863	3	3	2857
	Total	148 152	145	145	147 862

Tab.5 The samples for each class for training, validation, and testing of the HyRANK (HV) dataset

No.	Class	Total number	Train	Val	Test
1	Dense urban fabric	1262	37	37	1188
2	Mineral extraction sites	204	6	6	192
3	Non-irrigated arable land	614	18	18	578
4	Fruit trees	150	4	4	142
5	Olive groves	1768	53	53	1662
6	Coniferous forest	361	10	10	341
7	Densesclerophyllous vegetation	5035	151	151	4733
8	Sparce sclerophyllous vegetation	6374	191	191	5992
9	Sparcely vegetated areas	1754	52	52	1650
10	Rocks and sand	492	14	14	464
11	Water	1612	48	48	1516
12	Coastal water	398	11	11	376
	Total	20 024	595	595	19 429

Tab.6 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

Tab.7 The samples for each class for training, validation, and testing of the Botswana (BS) dataset

No.	Class	Total number	Train	Val	Test
1	Scrub	761	22	22	717
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

3.3 Experimental results

The experimental results with different methods for 6 datasets are demonstrated in Tab.8—Tab.13.

For IN dataset, our proposed DDCD achieves the best results with 95.44% ± 1.59% OA, 94.31% ± 1.23% AA, and 0.947 9±0.013 1 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. Nevertheless, since training samples are extremely limited, the overfitting phenomenon occurs in DBMA. As DDCD adopts a more adaptive and flexible attention mechanism, 5.55% improvement in OA is promoted. With the proposed DDCD, it can remain reliable and stable even though the training samples are finite. Specifically, the proposed method improves the OA by 5.55%, the AA by 6.86%, and the Kappa by 0.063 1 compared to DBMA.

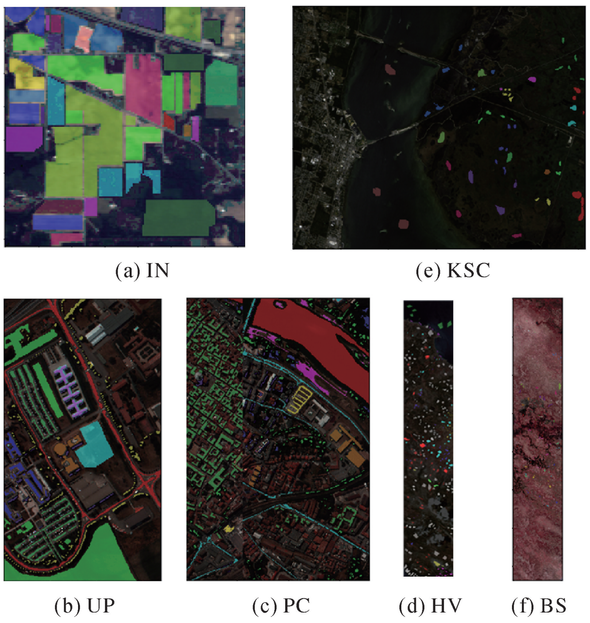

Fig. 8 False color composites and corresponding category labels

For the UP dataset, our proposed DDCD achieves the best results with 96.98%±1.23% OA, 96.21%±1.30% AA, and 0.958 6±0.016 4 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% training samples for training. Thus, the performance of CDCNN surpasses the SVM. Even though our model cannot make each category's 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 3.19%, the AA by 2.47%, and the Kappa by 0.041 7 compared to DBMA.

For the PC dataset, our proposed DDCD achieves the best results with 97.54%±0.33% OA, 93.06%±1.28% AA, and 0.965 1±0.004 7 Kappa with 0.1% training samples. The imbalanced category distribution is a significant feature of the PC dataset. For example, there are 65 971 pixels in Class 1, while there are only 2685 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 the proposed DDCD merely obtains 76.95%±9.00% accuracy, the precision is 12.43% higher than DBMA.

For the HV dataset, our proposed DDCD achieves the best results with 96.44%±0.57% OA, 96.63%±1.18 % AA, and 0.955 8±0.007 Kappa with 3% training samples. Similarly, 3% of training samples are enough, as the HV dataset owns sufficient samples. But unlike the UP dataset which only has 9 classes, the HV dataset has 12 classes. Thus, CDCNN obtains the worst performance. And the proposed method improves the OA by 1.38%, the AA by 1.2%, and the Kappa by 0.017 2 compared to DBMA.

For the KSC dataset, our proposed DDCD achieves the best results with 96.79%±1.26% OA, 95.21%±1.87% AA, and 0.964 9±0.015 6 Kappa with 3% training samples. The accuracies of Class 3—7 with no more than 7 training samples are unsatisfactory of other methods. Our DDCD obtains 77.36%, 85.34%, 93.52%, 97.68% and 92.17% of Class 3—7, which are 3.02%, 23.83%, 20.26%, 9.80%, 7.08% higher than DBMA. And the proposed method improves the OA by 4.90%, the AA by 6.72%, and the Kappa by 0.055 2 compared to DBMA.

Tab.8 The categorized results for the IN dataset using 3% training samples

Class	SVM/(%)	CDCNN/(%)	SSRN/(%)	FDSSC/(%)	DBMA/(%)	Proposed/(%)
1	29.34 ±3.60	50.17 ±6.79	79.16 ±8.97	97.08 ±2.71	94.80 ±3.69	98.37 ±1.63
2	55.51 ±0.32	56.59 ±4.46	86.15 ±2.21	96.24 ±1.91	91.08 ±0.72	95.64 ±1.37
3	62.66 ±1.07	53.17 ±3.07	91.67 ±2.98	93.14 ±2.65	85.40 ±5.48	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	95.37 ±1.58
5	85.30 ±1.28	84.05 ±5.66	97.69 ±1.89	98.42 ±0.64	97.43 ±0.51	98.24 ±0.73
6	82.11 ±1.52	89.03 ±2.82	95.85 ±1.56	97.02 ±0.87	96.76 ±1.18	98.12 ±0.79
7	64.17 ±6.13	46.28 ±8.09	90.93 ±6.09	72.21 ±12.14	52.66 ±9.25	73.47 ±22.19
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
9	42.40 ±10.06	52.17 ±13.04	74.64 ±10.86	71.29 ±18.60	62.66 ±6.34	90.21 ±4.17
10	63.01 ±2.72	52.87 ±7.61	85.75 ±3.81	86.01 ±4.24	82.43 ±3.32	91.31 ±2.35
11	64.09 ±1.25	67.79 ±3.09	88.65 ±1.80	91.56 ±4.05	90.54 ±1.83	95.67 ±1.66
12	48.50 ±1.15	44.67 ±3.28	86.34 ±2.73	90.63 ±2.75	80.10 ±5.37	91.02 ±3.59
13	87.37 ±2.35	87.12 ±2.60	99.00 ±1.00	99.79 ±0.20	98.55 ±0.79	99.05 ±0.69
14	89.71 ±0.41	91.17 ±1.25	95.52 ±0.55	97.01 ±1.69	97.14 ±0.75	96.51 ±1.47
15	61.51 ±2.73	73.97 ±1.00	94.28 ±1.54	93.24 ±2.18	86.18 ±2.21	95.69 ±2.31
16	97.64 ±1.29	94.36 ±1.33	94.15 ±2.15	96.99 ±1.69	94.55 ±3.93	95.57 ±1.87
OA	68.69 ±0.50	66.90 ±7.38	90.24 ±1.18	93.16 ±1.96	89.89 ±1.33	95.44 ±1.59
AA	66.62 ±1.37	68.05 ±2.06	90.12 ±1.54	92.36 ±3.08	87.45 ±2.34	94.31 ±1.23
K×100	63.93 ±0.49	62.36 ±7.78	88.84 ±1.36	92.18 ±2.28	88.48 ±1.51	94.79 ±1.31

For the BS dataset, our proposed DDCD achieves the best results with 95.66%±1.30% OA, 96.17%±1.05% AA, and 0.954 0±0.014 1 Kappa with 1% training samples. BS is a small dataset with only 3248 labelled samples, and we merely select 40 samples for training and 40 samples for validation. DDCD not only obtains state-of-the-art performance but also generates the lowest standard deviation.

Specifically, our method's standard deviation of OA is 1.30, while it is 2.30 and 2.80 for DBMA and FDSSC. And the proposed method improves the OA by 2.15%, the AA by 1. 70%, and the Kappa by 0.054 3 compared to DBMA.

Tab.9 The categorized results for the UP dataset using 0.5% training samples

Class	SVM/(%)	CDCNN/(%)	SSRN/(%)	FDSSC/(%)	DBMA/(%)	Proposed/(%)
1	83.61 ±2.58	87.30 ±2.83	98.89 ±0.47	97.42 ±1.06	92.91 ±0.94	97.31 ±2.43
2	84.96 ±2.07	92.65 ±1.12	97.96 ±0.37	98.69 ±0.34	96.03 ±2.1	99.13 ±0.41
3	58.75 ±5.38	45.81 ±12.22	74.34 ±10.03	91.34 ±6.61	89.41 ±4.36	94.13 ±5.81
4	96.37 ±0.86	95.02 ±2.65	98.98 ±0.47	97.75 ±1.59	96.86 ±1.48	98.00 ±1.66
5	94.99 ±1.16	96.96 ±1.27	99.93 ±0.06	99.67 ±0.11	99.49 ±0.16	99.71 ±0.17
6	81.90 ±4.16	82.71 ±4.28	91.07 ±4.24	98.72 ±0.27	96.86 ±0.92	97.64 ±2.46
7	53.26 ±13.41	69.82 ±8.51	78.69 ±4.42	96.53 ±1.30	95.18 ±4.49	98.67 ±1.33
8	71.36 ±1.96	65.38 ±2.04	77.71 ±4.16	74.33 ±2.14	81.67 ±2.05	86.21 ±2.17
9	99.89 ±0.03	93.89 ±1.76	98.60 ±0.78	97.17 ±0.95	92.78 ±3.73	98.11 ±0.95
OA	82.63 ±2.95	85.82 ±1.62	92.92 ±1.26	95.32 ±1.24	93.79 ±1.53	96.98 ±1.23
AA	80.57 ±4.68	81.06 ±2.93	90.68 ±1.93	94.62 ±2.14	93.47 ±1.96	96.21 ±1.30
K×100	76.23 ±4.56	81.08 ±2.11	90.66 ±1.63	93.78 ±1.66	91.69 ±2.15	95.86 ±1.64

Tab.10 The categorized results for the PC dataset using 0.1% training samples

Class	SVM/(%)	CDCNN/(%)	SSRN/(%)	FDSSC/(%)	DBMA/(%)	Proposed/(%)
1	99.75 ±0.09	97.08 ±0.75	99.98 ±0.02	99.85 ±0.13	99.84 ±0.05	99.74 ±0.24
2	83.36 ±2.23	82.01 ±4.28	97.05 ±0.85	87.73 ±4.72	93.32 ±2.77	93.27 ±5.32
3	62.47 ±5.34	85.20 ±5.92	77.15 ±5.88	84.17 ±7.69	76.82 ±5.15	86.67 ±9.59
4	63.15 ±5.80	49.91 ±12.72	65.43 ±6.89	63.46 ±17.03	64.52 ±2.83	76.95 ±12.00
5	82.76 ±4.16	73.20 ±4.87	89.23 ±2.16	90.01 ±3.43	87.02 ±3.23	93.22 ±6.24
6	83.52 ±2.27	85.77 ±2.24	87.29 ±4.79	88.96 ±3.80	87.87 ±3.94	90.76 ±2.20
7	91.88 ±0.97	86.91 ±7.96	99.64 ±0.30	94.53 ±5.19	99.14 ±0.35	97.92 ±1.69
8	95.26 ±2.57	97.28 ±0.69	98.19 ±0.86	99.66 ±0.10	99.55 ±0.18	99.58 ±0.09
9	99.77 ±0.10	92.57 ±3.53	97.99 ±1.77	92.43 ±7.23	95.61 ±2.34	99.46 ±0.83
OA	93.87 ±1.64	92.92 ±2.08	96.36 ±1.39	96.54 ±0.78	96.50 ±0.86	97.54 ±0.33
AA	84.66 ±0.85	83.33 ±5.99	90.22 ±2.24	88.98 ±4.36	89.30 ±2.18	93.06 ±1.28
K×100	91.27 ±2.38	89.86 ±3.00	94.83 ±1.97	95.10 ±1.11	95.04 ±1.22	96.51 ±0.47

Tab.11 The categorized results for the HV dataset using 3% training samples

Class	SVM/(%)	CDCNN/(%)	SSRN/(%)	FDSSC/(%)	DBMA/(%)	Proposed/(%)
1	74.27 ±0.37	85.98 ±4.20	93.80 ±5.34	89.81 ±7.88	89.92 ±7.10	93.38 ±4.40
2	88.07 ±1.51	89.08 ±11.56	99.43 ±1.14	99.79 ±0.42	100.0 ±0.00	98.81 ±2.38
3	85.93 ±2.39	73.49 ±16.74	93.25 ±3.74	79.05 ±15.69	93.31 ±6.61	93.68 ±4.17
4	89.01 ±3.21	67.04 ±20.48	83.92 ±21.15	59.09 ±34.54	91.79 ±8.37	91.63 ±16.18
5	86.80 ±0.36	85.82 ±4.81	88.97 ±1.88	93.37 ±1.81	88.18 ±2.22	93.17 ±2.20
6	97.09 ±1.04	88.71 ±9.59	99.23 ±0.95	99.59 ±0.68	98.52 ±0.81	99.58 ±0.83
7	95.83 ±0.95	94.86 ±1.86	97.07 ±0.70	97.26 ±1.22	96.87 ±2.53	97.92 ±1.92
8	86.28 ±3.11	89.26 ±4.77	93.38 ±1.15	95.34 ±1.55	96.17 ±1.42	95.86 ±0.86
9	83.29 ±0.57	84.98 ±7.15	95.80 ±2.03	91.25 ±8.38	91.97 ±6.47	96.23 ±2.03
10	91.40 ±0.05	89.61 ±9.03	96.13 ±5.02	97.61 ±4.78	98.42 ±2.12	99.33 ±1.35
11	94.56 ±0.68	81.73 ±2.41	99.91 ±0.12	99.96 ±0.05	100.0 ±0.00	100.0 ±0.00
12	100.0 ±0.00	39.64 ±48.57	99.84 ±0.31	100.0 ±0.00	100.0 ±0.00	100.0 ±0.00
OA	88.74 ±1.31	87.85 ±2.85	94.80 ±0.52	94.52 ±1.48	95.06 ±1.41	96.44 ±0.57
AA	89.38 ±0.58	80.85 ±7.66	95.06 ±1.51	91.84 ±3.84	95.43 ±1.13	96.63 ±1.18
K×100	85.99 ±067	84.90 ±3.58	93.53 ±0.66	93.19 ±1.85	93.86 ±1.77	95.58 ±0.70

Tab.12 The categorized results for the KSC dataset using 3% training samples

Class	SVM/(%)	CDCNN/(%)	SSRN/(%)	FDSSC/(%)	DBMA/(%)	Proposed/(%)
1	89.75 ±1.54	94.67 ±1.74	95.17 ±1.58	98.97 ±0.71	98.96 ±1.00	99.17 ±0.90
2	86.65 ±2.65	62.59 ±3.97	93.72 ±2.01	94.97 ±2.57	91.96 ±3.20	98.32 ±2.23
3	66.28 ±5.25	47.27 ±10.10	82.05 ±11.36	78.79 ±8.20	74.34 ±6.66	77.36 ±12.37
4	41.40 ±3.67	34.20 ±4.45	57.43 ±9.23	62.66 ±6.56	61.51 ±4.46	85.34 ±4.53
5	52.04 ±4.55	5.50 ±5.50	62.15 ±19.78	71.91 ±19.34	73.26 ±9.47	93.52 ±2.98
6	54.60 ±3.45	61.68 ±6.00	80.83 ±11.73	84.76 ±9.03	87.88 ±6.42	97.68 ±3.09
7	72.43 ±2.88	17.88 ±13.45	81.49 ±9.25	85.36 ±7.13	85.09 ±3.05	92.17 ±2.52
8	84.08 ±2.82	62.07 ±6.42	92.59 ±3.23	99.00 ±0.62	93.71 ±2.89	97.10 ±1.07
9	82.88 ±2.70	76.51 ±2.27	93.38 ±1.50	99.63 ±0.23	93.30 ±2.00	99.21 ±0.80
10	96.48 ±1.84	73.94 ±7.53	99.48 ±0.45	100.0 ±0.00	96.63 ±1.96	99.53 ±0.94
11	92.93 ±0.96	94.69 ±2.55	97.84 ±0.87	99.13 ±0.87	99.95 ±0.05	98.79 ±1.54
12	90.61 ±2.56	83.50 ±4.64	97.01 ±1.22	98.71 ±0.49	93.95 ±1.18	98.34 ±0.95
13	99.78 ±0.17	98.21 ±0.33	99.95 ±0.05	100.0 ±0.00	99.77 ±0.23	99.70 ±0.48
OA	84.10 ±2.27	75.88 ±3.13	89.72 ±1.87	94.22 ±2.64	91.89 ±1.35	96.79 ±1.26
AA	77.68 ±1.86	62.51 ±6.27	87.16 ±2.78	90.30 ±6.12	88.49 ±2.34	95.21 ±1.87
K×100	82.28 ±2.54	73.11 ±3.49	88.54 ±2.09	93.56 ±2.93	90.97 ±1.51	96.49 ±1.56

Tab.13 The categorized results for the BS dataset using 1% training samples

Class	SVM/(%)	CDCNN/(%)	SSRN/(%)	FDSSC/(%)	DBMA/(%)	Proposed/(%)
1	99.85 ±0.15	71.31 ±18.08	99.33 ±0.41	96.96 ±1.65	98.16 ±0.68	98.21 ±0.96
2	77.30 ±5.42	44.29 ±16.83	92.30 ±3.35	82.61 ±8.66	95.55 ±3.73	92.00 ±8.88
3	74.28 ±5.94	67.96 ±17.77	99.41 ±0.50	100.0 ±0.00	98.76 ±0.84	99.59 ±0.64
4	57.35 ±3.13	43.36 ±18.85	81.01 ±5.13	82.32 ±4.02	81.99 ±4.47	91.39 ±4.69
5	82.65 ±2.10	57.80 ±15.31	86.27 ±5.19	89.41 ±3.42	86.74 ±4.43	87.98 ±5.17
6	53.09 ±4.22	52.47 ±13.71	90.93 ±3.97	95.46 ±2.27	89.21 ±3.86	94.83 ±2.52
7	95.29 ±4.14	87.84 ±4.85	100.0 ±0.00	97.11 ±2.60	94.58 ±2.36	99.76 ±0.47
8	75.02 ±6.92	59.67 ±16.04	95.21 ±1.90	96.98 ±1.29	98.93 ±0.84	97.90 ±2.64
9	70.95 ±4.50	71.29 ±18.55	90.24 ±2.63	87.71 ±6.26	95.47 ±4.05	99.27 ±0.90
10	68.53 ±3.80	65.13 ±17.94	86.39 ±4.84	94.16 ±3.70	93.08 ±4.11	92.39 ±8.13
11	93.00 ±1.57	61.53 ±16.87	99.05 ±0.62	99.11 ±0.89	95.32 ±3.60	99.39 ±0.60
12	84.85 ±3.46	52.88 ±14.96	97.56 ±1.66	97.26 ±2.34	97.50 ±1.53	98.30 ±0.63
13	79.47 ±4.56	71.31 ±18.07	97.69 ±1.38	91.44 ±4.20	98.90 ±1.00	99.13 ±0.59
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
OA	74.73 ±2.45	61.01 ±26.42	92.90 ±1.37	92.69 ±2.80	93.51 ±2.30	95.66 ±1.30
AA	77.25 ±1.58	61.73 ±27.86	93.96 ±1.18	93.59 ±2.63	94.47 ±1.90	96.17 ±1.45
K×100	72.70 ±2.63	58.50 ±27.32	92.31 ±1.48	92.08 ±3.03	92.97 ±2.49	95.40 ±1.41

3.4 Investigation of the number of kernel size of 3D Double-Channel dense layer

In this part, we investigate the impact of the kernel size for classification accuracy. We set the kernel size of the 3D two-way dense layer as 3× 3, 5× 5, 7× 7, 9× 9, 11× 11, and 13× 13 to evaluate the precision and time cost with the BS dataset example.

As shown in Fig.9, when the kernel size continues to increase, the accuracy gradually decreases, but the time consumption multiplies, so we choose 3×3 as the kernel size of the 3D Double-Channel dense layer.

3.5 Effectiveness of 3D Double-Channel dense layer and linear attention mechanism

The linear attention mechanism is added in front of the 3D Double-Channel dense layer. To verify the effectiveness of the linear attention mechanism, we conduct an ablation experiment of the attention mechanism. As can be seen in Fig.10, the linear attention mechanism indeed promotes the precision on 6 datasets. Averagely, the linear attention mechanism improves 0.92% OA on 6 datasets compared to the original network.

There are two channels in 3D Double-Channel dense layer. The top Channel contains two stacked 3×3×3 convolution layers, which is equivalent to a 5×5×5 kernel size and obtains global information. The bottom Channel uses a 3×3×3 kernel to exploit local visual patterns. On the basis of the linear attention mechanism ablation experiment, to verify the effectiveness of the Double-Channel dense layer, we further remove a 3×3×3 convolutional layer of the top Channel. And we also take the original dense layer into comparison.

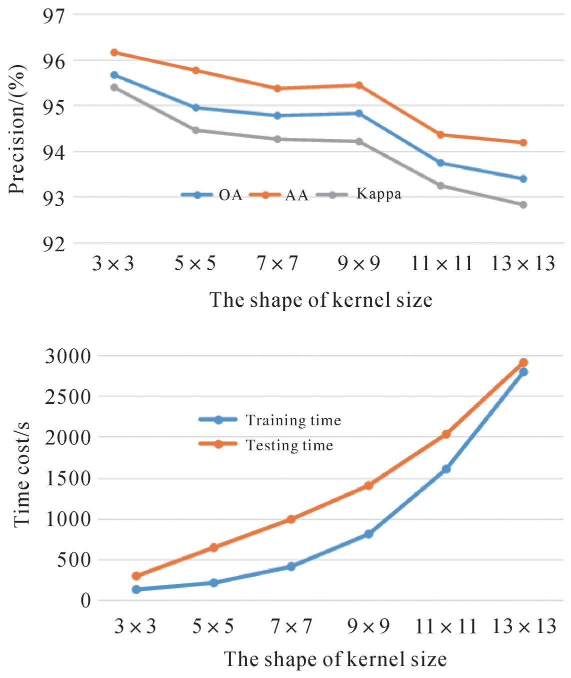

Fig.9 The precision and time cost between different kernel sizes of the dense layer

As can be seen in Fig. 10, the 3D Double-Channel dense layer indeed promotes the precision on 6 datasets. Averagely, the Double-Channel dense layer improves 1.71% OA on 6 datasets compared to the original dense layer. And two stacked 3×3×3 convolution layers improve 1.49% OA on 6 datasets compared to a single convolution layer.

3.6 Investigation of the number of patch size

The patch size is a crucial factor for 3D-cube-based methods. The larger patch size means more adjacent spatial information is taken into consideration. However, the larger patch size also leads to more computation complexity. We set the 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.

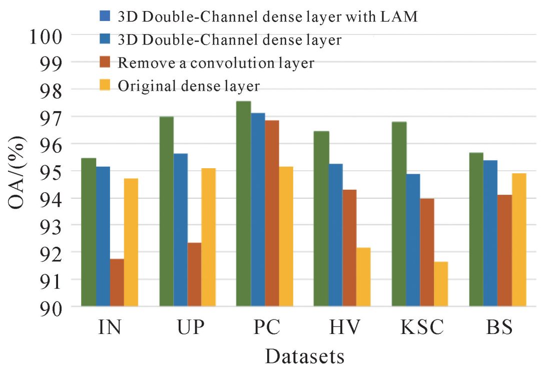

Fig. 10 The OA comparison between 3D Double-Channel dense layer with linear attention mechanism, 3D Double-Channel dense layer (3D Double-Channel dense layer without a 3×3×3 convolution layer, and original dense layer)

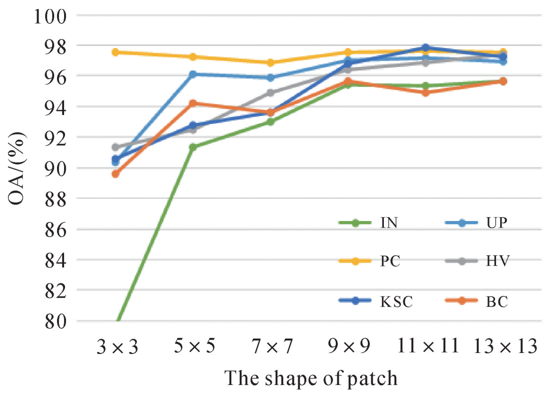

Fig.11 The OA comparison between different patch sizes of DDCD

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 the 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 DDCD.

3.7 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 the increasing number of training samples. As long as sufficient samples are provided, all methods obtain almost perfect performances. Meanwhile, as shown in Fig.12, the accuracy gaps between different methods are narrowing with increasing training samples. It is worth noting that our DDCD surpasses other methods on most occasions. Even though the samples are finite, DDCD still delivers robust performance.

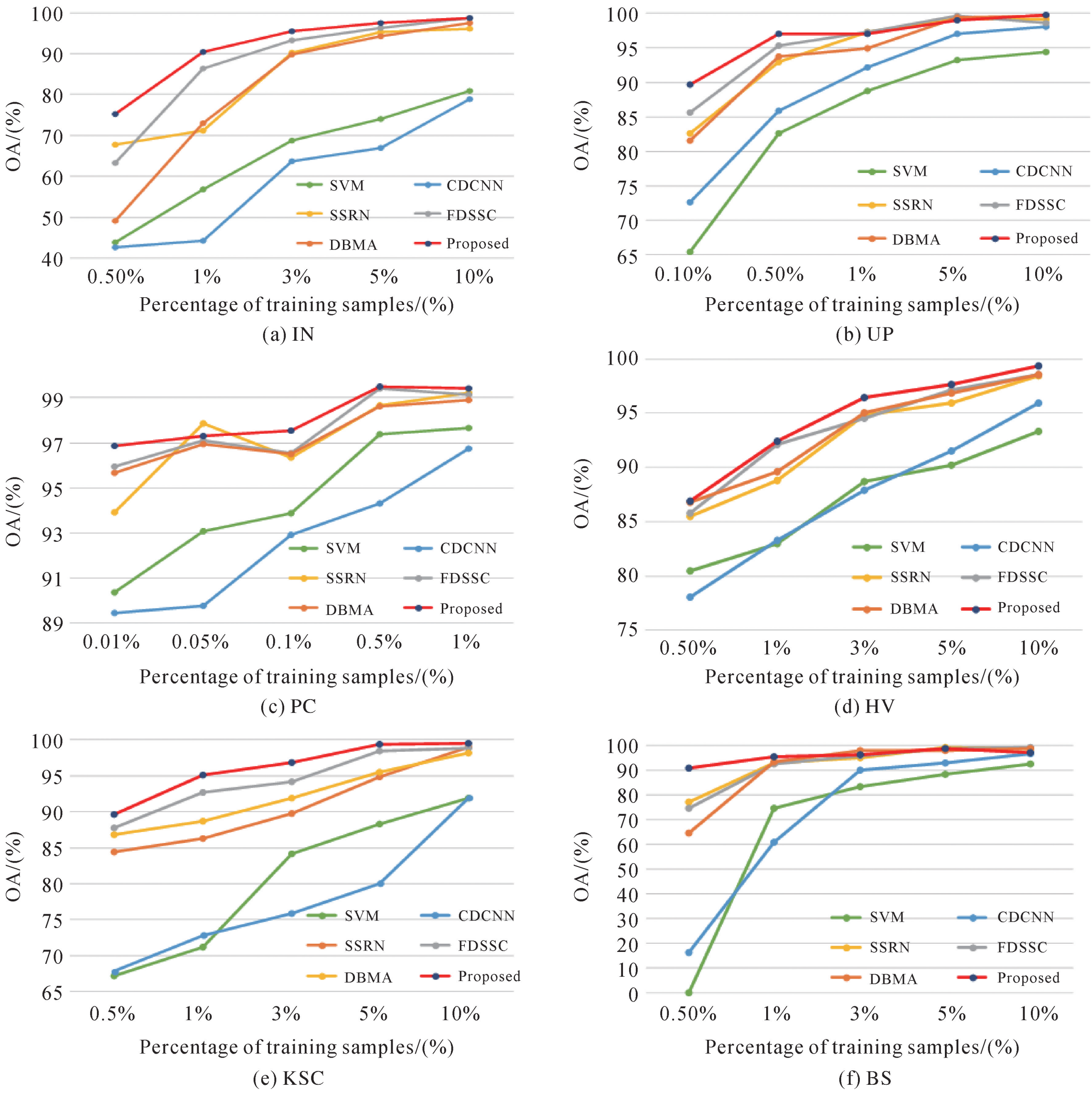

Fig.12 The OA of SVM, CDCNN, CDCNN, SSRN, FDSSC, DBMA and proposed DDCD with different ratios of training samples

4 Conclusion

In this paper, for HSI Classification, we design a 3D Double-Channel network architecture (DDCD) based on DenseNet. For our DDCD, we design a 3D Double-Channel dense layer to obtain local and global receptive fields simultaneously. Besides, we introduce a linear attention mechanism which reduces the complexity of the attention mechanism from $O(N^2)$ to $O(N)$. The number of parameters and the consumptions of calculation is observably less than comparative deep learning methods. A series of quantitative experiments on 6 widely-used datasets demonstrate that the proposed method obtains state-of-the-art performance, even though when the labelled samples are inadequate.

References

[1] LI Zhaokui, HUANG Lin, HE Jinrong. A multiscale deep middle-level feature fusion network for hyperspectral classification [J]. Remote Sensing, 2019, 11(6): 695.

[2] HONG Yongsheng, GUO Long, CHEN Songchao, et al. Exploring the potential of airborne hyperspectral image for estimating topsoil organic carbon: effects of fractional-order derivative and optimal band combination algorithm[J]. Geoderma, 2020, 365: 114228.

[3] ROCHA A D, GROEN T A, SKIDMORE A K. Spatially-explicit modelling with support of hyperspectral data can improve prediction of plant traits[J]. Remote Sensing of Environment, 2019, 231: 111200.

[4] DAI Yuchao, ZHANG Jing, HE Mingyi, et al. Salient object detection from multi-spectral remote sensing images with deep residual network[J]. Journal of Geodesy and Geoinformation Science, 2019, 2(2): 101-110. DOI: 10.11947/j.JGGS.2019.0211.

[5] MELGANI F, BRUZZONE L. Classification of hyperspectral remote sensing images with support vector machines[J]. IEEE Transactions on Geoscience and Remote Sensing, 2004, 42(8): 1778-1790.

[6] DU Qian, CHANG C I. A linear constrained distance-based discriminant analysis for hyperspectral image classification [J]. Pattern Recognition, 2001, 34(2): 361-373.

[7] EDIRIWICKREMA J, KHORRAM S. Hierarchical maximum-likelihood classification for improved accuracies [J]. IEEE Transactions on Geoscience and Remote Sensing, 1997, 35(4): 810-816.

[8] CHAN J C W, PAELINCKX D. Evaluation of Random Forest and Adaboost tree-based ensemble classification and spectral band selection for ecotope mapping using airborne hyperspectral imagery[J]. Remote Sensing of Environment, 2008, 112(6): 2999-3011.

[9] HAM J, CHEN Y C, CRAWFORD M M, et al. Investigation of the random forest framework for classification of hyperspectral data[J]. IEEE Transactions on Geoscience and Remote Sensing, 2005, 43(3): 492-501.

[10] ZHU Zexuan, JIA Sen, HE Shan, et al. Three-dimensional Gabor feature extraction for hyperspectral imagery classification using a memetic framework[J]. Information Sciences, 2015, 298: 274-287.

[11] BAU T C, SARKAR S, HEALEY G. Hyperspectral region classification using a three-dimensional Gabor filterbank[J]. IEEE Transactions on Geoscience and Remote Sensing, 2010, 48(9): 3457-3464.

[12] TANG Yuanyan, LU Yang, YUAN Haoliang. Hyperspectral image classification based on three-dimensional scattering wavelet transform[J]. IEEE Transactions on Geoscience and Remote Sensing, 2015, 53(5): 2467-2480.

[13] CAO Xiangyong, XU Lin, MENG Deyu, et al. Integration of 3-dimensional discrete wavelet transform and Markov random field for hyperspectral image classification[J]. Neurocomputing, 2017, 226: 90-100.

[14] LIU Ziwei, LUO Ping, WANG Xiaogang, et al. Deep learning face attributes in the wild[C]//Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). Santiago: IEEE, 2015: 3730-3738.

[15] WANG L, ZHANG C, LI R, et al. Scale- aware neural network for semantic segmentation of multi-resolution remote sensing images[J]. Remote Sensing, 2021, 13(24): 5015.

[16] LI R, ZHENG S, ZHANG C, et al. ABCNet: Attentive bilateral contextual network for efficient semantic segmentation of Fine-Resolution remotely sensed imagery[J]. ISPRS Journal of Photogrammetry and Remote Sensing, 2021, 181: 84-98.

[17] GONG Jianya, JI Shunping. Photogrammetry and deep learning [J]. Acta Geodaetica et Cartographica Sinica, 2018, 47(6): 693-704. DOI: 10.11947/j.AGCS.2018.20170640 .

[18] FAN Dazhao, ZHANG Yongsheng. Satellite image matching method based on deep convolutional neural network [J]. Journal of Geodesy and Geoinformation Science, 2019, 2(2): 90-100. DOI: 10.11947/j.JGGS.2019.0210 .

[19] SUN Long, WU Tao, SUN Guangcai, et al. Object detection research of SAR image using improved faster region-based convolutional neural network [J]. Journal of Geodesy and Geoinformation Science, 2020, 3(3): 18-28. DOI: 10.11947/j.JGGS.2020.0302.

[20] HE Hao, WANG Shuyang, WANG Shicheng, et al. A road extraction method for remote sensing image based on encoder-decoder network[J]. Journal of Geodesy and Geoinformation Science, 2020, 3(2): 16-25. DOI: 10.11947/j.JGGS.2020.0202.

[21] ZUO Zongcheng, WANG Wen, ZHANG Dongying. A remote sensing image semantic segmentation method by combining deformable convolution with conditional random fields [J]. Journal of Geodesy and Geoinformation Science, 2020, 3(3): 39-49. DOI: 10.11947/j.JGGS.2020.0304.

[22] ZHENG Zhuo, ZHONG Yanfei, MA Ailong, et al. FPGA: fast patch-free global learning framework for fully end-to-end hyperspectral image classification[J]. IEEE Transactions on Geoscience and Remote Sensing, 2020, 58(8): 5612-5626.

[23] JIA Zhuang, LU Wenkai. An end-to-end hyperspectral image classification method using deep convolutional neural network with spatial constraint [J]. IEEE Geoscience and Remote Sensing Letters, 2021, 18(10): 1786-1790.

[24] SUN Hao, ZHENG Xiangtao, LU Xiaoqiang. A supervised segmentation network for hyperspectral image classification[J]. IEEE Transactions on Image Processing, 2021, 30: 2810-2825.

[25] PAN Bin, XU Xia, SHI Zhenwei, et al. DSSNet: a simple dilated semantic segmentation network for hyperspectral imagery classification [J]. IEEE Geoscience and Remote Sensing Letters, 2020, 17(11): 1968-1972.

[26] ZHAO Wenzhi, DU Shihong. Spectral-spatial feature extraction for hyperspectral image classification: a dimension reduction and deep learning approach[J]. IEEE Transactions on Geoscience and Remote Sensing, 2016, 54(8): 4544-4554.

[27] LEE H, KWON H. Going deeper with contextual CNN for hyperspectral image classification[J]. IEEE Transactions on Image Processing, 2017, 26(10): 4843-4855.

[28] CHEN Yushi, JIANG Hanlu, LI Chunyang, et al. Ghamisi. Deep feature extraction and classification of hyperspectral images based on convolutional neural networks[J]. IEEE Transactions on Geoscience and Remote Sensing, 2016, 54(10): 6232-6251.

[29] HE Kaiming, ZHANG Xiangyu, REN Shaoqing, et al. Deep residual learning for image recognition[C]//Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas: IEEE, 2016: 770-778.

[30] HUANG Gao, LIU Zhuang, VAN DER MAATEN L, et al. Densely connected convolutional networks[C]//Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu: IEEE, 2017: 2261-2269.

[31] ZHONG Zilong, LI J, LUO Zhiming, et al. Spectral-spatial residual network for hyperspectral image classification: a 3-D deep learning framework [J]. IEEE Transactions on Geoscience and Remote Sensing, 2018, 56(2): 847-858.

[32] WANG Wenju, DOU Shuguang, JIANG Zhongmin, et al. A fast dense spectral-spatial convolution network framework for hyperspectral images classification [J]. Remote Sensing, 2018, 10(7): 1068.

[33] HAUT J M, PAOLETTI M E, PLAZA J, et al. Visual attention-driven hyperspectral image classification [J]. IEEE Transactions on Geoscience and Remote Sensing, 2019, 57 (10): 8065-8080.

[34] MA Wenping, YANG Qifan, WU Yue, et al. Double-branch multi-attention mechanism network for hyperspectral image classification[J]. Remote Sensing, 2019, 11(11): 1307.

[35] WOO S, PARK J, LEE J Y, et al. CBAM: convolutional block attention module [C]//Proceedings of the 15th European Conference on Computer Vision (ECCV). Munich: Springer, 2018: 3-19.

[36] FU Jun, LIU Jing, TIAN Haijie, et al. Dual attention network for scene segmentation[C]//Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach: IEEE, 2019: 3141-3149.

[37] LI Rui, ZHENG Shunyi, DUAN Chenxi, et al. Classification of hyperspectral image based on double-branch dual-attention mechanism network [J]. Remote Sensing, 2020, 12 (3): 582.

[38] LI R, DUAN C, ZHENG S, et al. MACU-Net for semantic segmentation of fine-resolution remotely sensed images[J]. IEEE Geoscience and Remote Sensing Letters, 2021.

[39] LI R, ZHENG S, DUAN C, et al. Multistage attention resunet for semantic segmentation of fine-resolution remote sensing images [J]. IEEE Geoscience and Remote Sensing Letters, 2021.

[40] WANG L, LI R, WANG D, et al. Transformer meets convolution: a bilateral awareness network for semantic segmentation of very fine resolution urban scene images[J]. Remote Sensing, 2021, 13(16): 3065.

[41] LI R, ZHENG S, ZHANG C, et al. Multiattention network for semantic segmentation of fine-resolution remote sensing images [J]. IEEE Transactions on Geoscience and Remote Sensing, 2021.