Knowledge-Based Systems 268 (2023) 110482

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A capsule-vectored neural network for hyperspectral image classification

Xue Wang [a,b], Kun Tan [a,b,*], Pejun Du [c], Bo Han [d], Jianwei Ding [e]

[a] Key Laboratory of Geographic Information Science (Ministry of Education), East China Normal University, Shanghai 200241, China
[b] Key Laboratory of Spatial-Temporal Big Data Analysis and Application of Natural Resources in Megacities (Ministry of Natural Resources), East China Normal University, Shanghai 200241, China
[c] Key Laboratory for Satellite Mapping Technology and Applications of NASG, Nanjing University, Nanjing 210023, China
[d] The Institute of Remote Sensing Satellite, China Academy of Space Technology, Beijing, 100094, China
[e] The Second Surveying and Mapping Institute of Hebei, Shijiazhuang 050037, China

ARTICLE INFO

Article history:
Received 18 January 2023
Received in revised form 11 March 2023
Accepted 13 March 2023
Available online 15 March 2023

Keywords:
Hyperspectral image classification
Capsule
Fully convolutional network

ABSTRACT

The issues of feature redundancy and insufficient labeled samples impede the widespread application of hyperspectral images. Many researchers have designed complex networks in pursuit of hyperspectral image classification, which has brought unstable factors to the networks and resulted in poor performance in the case of insufficient labeled samples. The motivation of this paper is to promote the classification accuracy by enhancing the classification approach, instead of the feature extraction phase. Differing from the existing works which focus on elaborate and complicated network designs and complex training strategies, this work is aimed at integrating a vector-neuron capsule representation with a vanilla fully convolutional network (FCN) to obtain an improved hyperspectral image classification performance in the case of insufficient labeled samples. The proposed network, which is named the capsule-vectored neural network (CVNN), consists of an encoder–decoder feature learning part and a vector-neuron capsule transformation part. In the encoder–decoder part, a down-sampling step is employed twice in the fully convolutional framework. After feature extraction by the vanilla FCN, the output is transformed into a vectorial representation with a learnable transformation matrix. All of the vector neurons have a unified dimension, and the norm of the vector neurons is utilized as the logit that is fed into the softmax function to obtain the posterior probability matrix. The experimental results confirm that CVNN can tackle deep learning modeling well in the case of limited labeled samples. The classification accuracy of CVNN is also higher than that of the other advanced classification approaches.

1. Introduction

Hyperspectral remote sensing imagery has entered the era of mass production, which significantly improves the situation of image information lagging behind the spatio-temporal changes of ground objects. However, the widespread application and rapid development of remote sensing monitoring technology is hindered by the large amount of manual data annotation. Hyperspectral remote sensing imagery contains rich optical characteristics and spectral information, but the lack of labeled samples means that the model training and updating for classification applications are confronted with great challenges. How to tackle the modeling of the data in the case of insufficient labeled samples is an interesting research topic. Moreover, there is an urgent need to realize inter-class discrimination of ground objects and to promote the development of hyperspectral remote sensing image classification applications to be both data-driven and knowledge-driven.

Hyperspectral remote sensing imagery is made up of high-dimensional data cubes containing rich spatial and spectral information, which provides us with the ability to gain an insight into recognizing surface characteristics. Classification is one of the primary research topics, which involves assigning a unique label to each pixel in the image. Accurate classification and interpretation are of great significance for land-cover monitoring [1], vegetation assessment [2–4], pollution detection [5], atmospheric environmental monitoring [6], and mapping [7]. Many hyperspectral image classification approaches have been proposed over the past years, such as support vector machine (SVM) [8,9] and multinomial logistic regression (MLR) [8,9], which are aimed at

* Corresponding author at: Key Laboratory of Geographic Information Science (Ministry of Education), East China Normal University, Shanghai 200241, China.
E-mail address: tankuncu@gmail.com (K. Tan).

https://doi.org/10.1016/j.knosys.2023.110482
0950-7051/

exploring the boundaries in the native spectral feature space. Unfortunately, the correlation among the spectral bands is high and redundancy is one of the obstructing factors for classification. Furthermore, the high-dimensional features are dependent on sufficient information to avoid the Hughes phenomenon, which is a big limitation in the case of insufficient labeled samples. To solve this issue, researchers have proposed many feature reduction methods, such as band selection [10,11]. On the other hand, well-designed augmented spatial features can be employed to promote the classification accuracy, such as texture features [12], the gray-level co-occurrence matrix (GLCM) [13] and the topological properties [14,15]. However, the performance improvement brought by these hand-crafted features is limited. The hyperspectral image classification task can be split into two processes – feature extraction and classification – and when the two processes cannot be combined as an entirety, the optimal solution will not be globally optimal. Over the past years, deep learning algorithms have made many breakthroughs in the field of remote sensing interpretation. Deep learning has high feature mining and feature representation capabilities, and can extract the effective characteristics of remote sensing images. The end-to-end pattern of deep learning can also train the feature extraction and classification hyperplane optimization effectively.

Deep learning based hyperspectral image classification started with the deep representation of one-dimensional (1D) features. For example, deep restricted Boltzmann machines [16] and autoencoders [17] have been used to explore advanced features. Other works have constructed a spatio-spectral 1D feature vector to learn the deep representation [18].

Convolutional neural networks (CNNs) can be utilized in the hyperspectral data cube through the convolution operation, which conforms with the data organization mode. To deliver the spatio-spectral information, Yang et al. [19] utilized a CNN to extract the features from both the spectral and spatial domains. The extracted features were then input into fully connected layers for label prediction. To enable the CNN to capture deeper and wider features of hyperspectral imagery, Lee and Kwon [20] proposed a novel network which is able to integrate the local spatial and spectral relationships in neighborhood pixels, where the contextual interactions can be optimally utilized. Zhang et al. [21,22] investigated a diverse region-based CNN, where the semantic context information can be enhanced by merging a diverse set of spatial characteristics, which has been demonstrated to be important for classification. Hong et al. [23] proposed a transformer-based CNN, which can capture the spectrally local sequence information from neighboring bands and yield group-wise spectral embeddings. Hao et al. [24] proposed a two-branch structure for hyperspectral image classification, which consists of a stacked denoising autoencoder and a CNN for the spectral and spatial feature learning, respectively. The three-dimensional (3D) CNNs not only extract the convolution characteristics in both the spatial and spectral dimensions, but can also be employed for the feature extraction of hyperspectral imagery [25]. The above-mentioned CNN-based works are mainly carried out in patch-based models, in which the label of the pixel is inferred from the patch consisting of the surrounding pixels. However, the flaw of the patch-based inference approach is the many repeated steps, which cause a waste of computing resources and result in the low efficiency of the algorithm. Some works have introduced fully convolutional network (FCN)-based models into hyperspectral classification [1–3,26–28]. In an FCN, the last layer with full connection is replaced with an up-sampling operation, such as deconvolution or interpolation, which endows the FCN with high robustness and flexibility. Following the many breakthroughs in computer vision tasks, the FCN has now been modified for the application of hyperspectral image classification. For example, Jiang et al. [28] proposed the fully convolutional spatial propagation network (FCSPN) for hyperspectral image classification, which consists of a 3D FCN and a convolutional spatial propagation network. A residual unit and an attention mechanism are also used to achieve a high level of performance. In addition, Li et al. [26] proposed a deep learning framework based on an FCN, which includes convolution, deconvolution, and pooling operations. The experimental results showed that the enhanced deep features can obtain a superior performance. Zheng et al. [27] investigated an FCN-based classification method, in which the designed sampling strategy boosts the optimization process. In the proposed sampling strategy, the training samples in each category are transformed into a balance sequence, where the balance sequence can guarantee the convergence of the proposed model. Sun et al. [2,3] proposed a segmentation network based on an end-to-end FCN, which includes a fine label generation process to improve the distribution characteristics of the land objects. Wang et al. [1] explored the FCN-based network named the unified multiscale learning (UML) framework for hyperspectral image classification, which has a faster speed than the patch-based models. However, from hand-crafted features to the well-designed networks and elaborate attention mechanisms, the performance of the classification method is mainly dependent on the feature extraction. Meanwhile, these networks need sufficient labeled samples, and this precondition can rarely be met in hyperspectral image classification. To handle the case of classification with limited labeled samples, other approaches have been investigated, such as meta-learning [29] and semi-supervised learning [30]. Xue et al. [31] proposed S3Net – a spectral–spatial Siamese network – which can augment the training set by feeding sample pairs into each branch, thus enhancing the model separability. Chu et al. [32] introduced the discriminative information and local manifold structure of samples into a broad learning system to enhance the discriminative ability of the output weights and improve the performance in the case of limited labeled samples.

The above-mentioned works are mainly based on scalar neurons, which have been shown to be defective for classification [18]. Since the concept of the capsule network was first proposed [33], a number of methods have been proposed to improve the performance of the capsule network [34,35]. On this basis, capsule networks have been widely investigated in hyperspectral image classification [36,37]. A capsule network is composed of convolutional layers and capsule layers, which can address the intrinsic limitation of the convolution operation, such as the exponential inefficiency and the low robustness of affine transformations. Recently, several researchers have suggested that capsule-based networks have the potential to surpass the conventional convolutional networks in many aspects, such as the good performance with insufficient labeled samples. For example, Wang et al. [18] proposed Caps-TripleGAN, which combines a capsule network and a generative adversarial network to tackle the insufficient training samples. In addition, Paoletti et al. [38] developed a kind of refined capsule unit, which can represent the spatio-spectral features. Li et al. [36] proposed a 3D capsule network to enhance the robustness and achieve better generalization. In the conventional capsule network, the affine transformation process with a non-shared matrix and dynamic routing brings a high modeling capacity. However, the optimization of the transformation matrix and the weighted voting matrix are time-consuming, which hinders the development of capsule networks in hyperspectral image classification. Some methods related to computational burden and latency optimization have been introduced to address this issue [39–41]. Furthermore, the capsule-based models generally use patches for the image inference, and are thus affected by the intrinsic flaw of patch-based inference.

The motivation for this work can be described as follows:

(1) As previously mentioned, the capsule vector neuron has been shown to be more effective than the scalar neuron in learning separable features with limited samples, but its complex learning process (such as dynamic routing) hinders its application in hyperspectral image classification. In order to resolve this problem, the proposed model employs a learnable capsule transformation matrix to replace the dynamic routing, which can reduce the complexity of the optimization processing.

(2) The existing capsule-related hyperspectral image classification algorithms are mainly patch-based methods. However, the patch-based inference approach includes many repeated steps, which cause a waste of computing resources and result in the low efficiency of the algorithm. Therefore, the capsule vector neuron is embedded into an encoder–decoder FCN, and the embedded capsule network can reduce the full image inference time, compared with the patch-based methods.

Most recently, we investigated the capsule network and proposed Caps-TripleGAN to solve the problem of hyperspectral image classification in the case of limited labeled samples [18]. Caps-Triple GAN is carried out using patch-based inference and consists of a capsule network and a generative adversarial network (GAN). The former component is responsible for the main classification task, and the latter component is utilized for the training data augmentation. However, Caps-Triple GAN has the disadvantage of high time consumption, which can be ascribed to the dynamic routing process and the patch-based inference. Moreover, the complex unfolding optimization of the dynamic routing has a huge requirement for computational memory, which also limits the flexibility and the expandability of the capsule network, although it has been proved to surpass the convolutional networks in many aspects. In this regard, we replace the dynamic routing process to allow the vector-neuron capsule representation to be embedded into the FCN-based network, which can improve the classification accuracy by enhancing the classification approach, instead of relying on complex feature extraction with an unstable performance. To trade-off the computational overhead and the modeling accuracy, the capsule network is refined and embedded into the hyperspectral image classification based on an end-to-end FCN. The time-consuming dynamic routing is modified to form a trainable weight optimization process, which can accelerate the inference process significantly. With the high representation capacity and the fast inference, the capsule-vectored neural network (CVNN) can perform well in the case of limited labeled samples. The main contributions of this work are given as follows.

(1) We propose the innovative CVNN method for hyperspectral image classification in an encoder–decoder architecture. The network integrates a vector-neuron capsule representation with a vanilla FCN to obtain an improved hyperspectral image classification performance in the case of insufficient labeled samples

(2) We utilize affine transformation to replace the time-consuming dynamic routing process. The affine transformation enables the vector neurons to be optimized effectively. The time consumption can thus be reduced by the full image inference, compared with patch-based inference.

(3) Comprehensive comparisons between the recent deep learning based classification methods and the proposed CVNN method confirmed that the proposed method can obtain a state-of-the-art performance under the case of very few labeled samples.

The rest of this paper is organized as follows. Section 2 describes the previous works in this field. Section 3 introduces the proposed CVNN method. Section 4 details the real hyperspectral datasets utilized in the experiments, the experimental results, and the comparisons with other approaches. Finally, we draw our conclusions in Section 5.

2. Related works

2.1. The conventional CNNs and FCNs

The convolution operation is based on local connection, and the weight sharing and local feature extraction make CNNs more practicable than FCNs. The perception of the convolutional layer is determined by its receptive field. When the receptive field covers the whole data range, the convolutional layer is equivalent to a fully connected layer. However, the stacked receptive field means that CNNs still have to make a large number of parameter calculations. Therefore, the general CNN networks use pooling operations to reduce the scale of the network parameters. There are two main approaches used in the convolution-based inference models in hyperspectral image classification. One approach is to use patches of the whole image to feed into the CNN, and the other is to use the whole image based on an FCN. In the former approach, the label of each patch is matched with its center pixel; that is, the other pixels in this patch are employed as the neighborhood information for the center pixel. However, the modeling capability is heavily reliant on the size of the patch. The classification map for a certain hyperspectral image can be obtained by repetitive iterations though all the pixels, one after another. The output of a single layer in a CNN can be obtained by:

$$O = \mathcal{F}\left(z^{l}\right) = \mathcal{F}\left(\sum_{ic=1}^{IC_{l-1}} a^{l-1,t} * W^{s,c} + b^{s}\right) \tag{1}$$

$$\mathcal{F}(z) = \begin{cases} z, & if\ z > 0 \\ 0, & if\ z < 0 \end{cases} \tag{2}$$

where O represents the output in layer l, $a^{l-1,t}$ is the output of the prior layer in the t_{th} channel, $W^{s,c}$ is the affine weight matrix, and b^{s} is the bias. $\mathcal{F}$ denotes the nonlinear function. After the forward propagation through all the layers, a fully connected layer follows to calculate the logit into softmax.

Compared with the traditional CNN, the FCN abandons the following fully connected layer and replaces it with a convolutional layer. FCNs have been shown to perform well in semantic image segmentation, in which it is necessary to identify each pixel. Because the modeling object is the data in a patch, the size of the patch is the crucial factor determining the modeling capacity. Meanwhile, in an FCN, the dependence on the patch size is eliminated. Moreover, each pixel has a large enough receptive field and a tiny computational overhead. To avoid the loss of details when down-sampling, some tricks can be applied in FCNs, such as a skip connection between the down-sampling and up-sampling. Moreover, many elaborate attention mechanisms and multiscale feature fusion strategies [42] can be employed to boost the final performance.

2.2. Capsule network

The design of a neural network is based on the activity of the human visual cortex [43], while cortical cells also have specific impulses in response to a series of relative features, such as relative posture and position, which are discarded in a neural network. Therefore, it is necessary to introduce a feature descriptor which considers the relationship between the relative feature pose and feature attributes. Differing from [33,43] advocated that the visual cortex of the brain is organized into modules called "capsules". These capsules are designed to represent objects and their properties, such as speed, tone, light, shadow, deformation, or direction. Compared with the conventional neuron, the capsule can capture a more comprehensive representation of the features.

The former only cares about the existence or not, while the latter learns the combined thinking of the features. Moreover, the propagation mechanism in a capsule network should be the same as in a conventional neural network. The learned information from the prior layer should be assembled and transmitted into the next layer. In the capsule network propagation, it is necessary to consider both the existence of a certain feature and the high-level information, such as the form of the feature itself, and the relationship between the feature and other features, which is called dynamic routing. Moreover, the output of the capsule network is formed to act in the same way as that of a conventional neural network, and the nonlinear activation function is named the squash function.

In a capsule network, the object recognition process for the visual nerves is based on the relative characteristics. The relative position information of the objects that need to be recognized is constructed hierarchically. The hierarchical attributes can be regarded as inversion of the data, which can rebuild the original data. The correlation of the determination for the fed instance is dependent on whether each feature maintains the relative hierarchical attributes integrally.

3. Methodology

3.1. Capsule representation

In a CNN, the capacity of all the neurons in the same layer is identical and there is no internal hierarchical organization, which leads to uncertain classification of an instance from a different view point. The shared weight adopted in a CNN is aimed at extracting stable features by the neural network when there are slight changes in the image, which is exactly what the capsule network can do. In a capsule network, a capsule is defined as a carrier container loading many neurons that represent specific features appearing in the data. Each loaded neuron describes a corresponding attribute of a feature entity. The connection between capsules represents the position relationship between each feature entity. Sabour et al. [33] advocated employing vector neurons to represent capsules, where the two basic vector concepts of "length" and "direction" are utilized to describe the attributes. Length represents the probability of the existence of the feature entity, and direction describes the position attributes of the feature entity.

Each capsule exists in the form of vector neurons, and the calculation process for a vector neuron includes weight multiplication, summation, and nonlinear transformation of the activation function, which is similar to the neuron calculation method in the traditional neural network. Meanwhile, before multiplying the weights, an additional affine transformation is carried out on the relative position matrix related to the current layer. Table 1 lists the main differences between a capsule vector neuron and a conventional scalar neuron.

The calculation process for each capsule is as follows:

(1) Step 1: the affine transform on the output of the prior layer is used to obtain the relative posture of this layer.

(2) Step 2: the features are summed based on the weight matrix, which determines the degree of the features' influence.

(3) Step 3: the node is activated with a nonlinear activation function, while limiting the output length to 0–1.

Dynamic routing is carried out to obtain the weight matrix in Step 2. The weight matrix is also known as the "coupling coefficient", and it determines which low-level feature entities are in charge of the high-level feature entities. With dynamic routing, the connection weight of a lower-layer capsule close to the capsule center is increased, and the weight for a capsule far away from the capsule center is decreased. The weight matrix is defined as $I \times J$, where I and J respectively denote the number of capsules in the prior layer and the current layer. The update of the weight matrix is different from back-propagation, and is called dynamic routing.

Since the output of this layer is input into the following layer, both the value and direction should be considered. In Step 3, a squash function is set in the capsule network, i.e., the output s_j is numerically squashed while maintaining the direction of s_j.

$$\mathrm{sq}(s_j) = \frac{\|s_j\|^2}{1+\|s_j\|^2}\frac{s_j}{\|s_j\|} \tag{3}$$

where the front part is to squash the vector into 0 to 1. The norm of the output will tend to be 1 when the norm of s_j is very large; conversely, if the norm of s_j is very large, the norm of the output vector will approach 0. The latter is utilized to keep the direction vector.

The norm of the vector is utilized to represent the probability of the existence of a special feature for the classification. For the correct ground object class, the layer should have a long output vector in the corresponding capsule. The interval losses for each separate category are defined as follows:

$$\begin{aligned} L_C = {} & T_C max\left(0, m^+ - \|cap_C\|\right)^2 \\ & + \lambda\left(1 - T_C\right) max\left(0, \|cap_C\| - m^-\right)^2 \end{aligned} \tag{4}$$

where L_C denotes the distance between the positive and negative samples to the hyperplane, C represents the category, and T_C is the indicator function for the classification based on one-hot encoding. The corresponding position is 1 to represent class C.

3.2. Transformation to capsule vector

The capsule vector is obtained by stacking the multiple convolutional features in the capsule network, and the propagation between capsules should employ an affine transformation in advance. The capsule vector should be fully connected between two layers. The affine transformation matrix is $W \in \mathbb{R}^{I\times J}$. The vector after multiplying the matrix is:

$$U_{i,j} = W_{i,j} cap_i \tag{5}$$

where $U_{i,j}$ denotes the vector having the same length as the capsules in the following layer. After the affine transformation, $U_{i,j}$ should be summed with a different weight:

$$s_i = \sum_i c_{i,j} U_{i,j} \tag{6}$$

where $c_{i,j}$ is the weight matrix (which is also known as the coupling coefficient), which is optimized by dynamic routing. After this, the output cap_j is obtained by Eq. (1). In dynamic routing, an additional b_{ij} is employed to indicate the consistency between $U_{i,j}$ and cap_j, and the coupling coefficient $c_{i,j}$ is updated by:

$$c_{i,j} = \frac{exp\left(b_{ij}\right)}{\sum_k exp\left(b_{ik}\right)} \tag{7}$$

where k denotes the number of capsules in the following layer. b_{ij} is calculated by the combination of the input of the low-layer capsule and the output of the high-layer capsule, which is implemented by:

$$b_{i,j} = b_{i,j} + U_{i,j} \cdot cap_j \tag{8}$$

Differing from the conventional updating mechanism based on loss calculation and back-propagation, the updating of $c_{i,j}$ and $b_{i,j}$ is implemented in a feedforward process. Therefore, the dynamic

Table 1
Pseudocode of the proposed algorithm.

Algorithm CVNN framework for application to a hyperspectral dataset

Input: $HSI \in \mathbb{R}^{h\times w\times b}$, training label $T_{train} \in \mathbb{R}^{h\times w\times 1}$, testing label $T_{test} \in \mathbb{R}^{h\times w\times 1}$

Initialize: Network CVNN (feature extraction part FE and capsule transformation part CT);

Batch size bs (a multiple of the number of categories).

Initialize the weight matrix W in FE

Initialize the transformation matrix $\breve{W}$ in CT

for number of training iterations **do**:

Fix the location of the training label for each batch from the T_{train}.

Generate the batch training label image $\mathrm{T}' \in \mathbb{R}^{h\times w\times 1}$ by the fixed location.

Generate the training pair: $[HSI,\ \mathrm{T}']$

Input the HSI into CVNN and obtained the extracted feature F

for c **do**:

Obtain $\mathbf{\Gamma} = \left(\breve{W}^T\breve{W}\right)^{-1}$

Choose F' from F according to the location of T'

Calculate $V^TV = F'^T\breve{W}\mathbf{\Gamma}\breve{W}^TF'$

Obtain $VN = \|V\|_2$= sqrt (V^TV)

$L \leftarrow -\sum_c y_c log(VN_c)$

Update transformation matrix $\breve{W}$ with the gradient:

$$\nabla_{\theta_{\breve{W}}}\left[\frac{1}{\mathrm{bs}}L\right] = \frac{\delta_{\theta_{\breve{W}}}L}{\delta\|V\|_2}\frac{\delta\|V\|_2}{\delta\breve{W}}$$

Update weight matrix W with the gradient:

$$\nabla_{\theta_W}\left[\frac{1}{\mathrm{bs}}L\right]$$

Endfor

Endfor

Output: the index of the max value of VN for all pixels

routing can be regarded as gradually adjusting the probability of the low-level capsule to that of the high-level capsule.

The capsule and dynamic routing are an effective way address the limitations such as the lack of robustness to transformations

in the traditional deep learning networks. However, despite the good performance, the dynamic routing has a huge overhead and is time-consuming [44], which is not beneficial to hyperspectral image classification. Moreover, the capsule network is difficult to modify with an FCN framework and is also time-consuming. As such, many methods have been proposed to solve the intrinsic flaws of capsule networks.

To tackle the cumbersome dynamic routing algorithm, we introduce learnable transformation [21,22] to simplify the propagation of the capsule network. Because of its ability to include the features and the relative information, the vector neuron is employed following the FCN feature extraction. The projection is carried out on the last layer, to play the role of classification. For brevity, the m-dimensional features extracted by the FCN are denoted as $F \in \mathbb{R}^{h\times w\times m}$, where h and w represent the height and width of the image, respectively. The goal of the transformation is to obtain the category capsule for F using a $(h\times w)\times c\times d$ matrix, where c is the number of categories and d is the dimension of the capsule. For a certain pixel, the label indicator is determined by a vector $I \in \mathbb{R}^{c\times d}$. Differing from the design of interval losses, the norm of the indicator vector is regarded as the logit and fed into the softmax function for the posterior probability.

In a traditional FCN, the transformation is carried out using a $1\times 1\times c$ convolution operation. The learnable transformation matrix is utilized to yield the capsule vector, which is different from the conventional capsule network. Firstly, the extracted feature F is obtained:

$$F = FCN_{bb}(HSI) \tag{9}$$

where $HSI \in \mathbb{R}^{h\times w\times b}$ is the hyperspectral image matrix, and b is the spectral dimension. FCN_{bb} denotes the fully convolutional backbone network, which can be detailed as follows:

$$IF_3 = DL_2\left(DL_1\left(PL(HSI)\right)\right) \tag{10}$$

where $PL(\cdot)$ is the initial convolution layer. $DL_1(\cdot)$ and $DL_2(\cdot)$ are the two down-sampling layers. For convenience, the output intermediate features of $PL(\cdot)$ and $DL_1(\cdot)$ are denoted as IF_1 and IF_2, respectively. The down-sampling is followed with the up-sampling layer:

$$UF_1 = \left[UL_1\left(IF_3\right), CL_1\left(IF_2\right)\right] \tag{11}$$

$$F = CL_3\left(\left[UL_2\left(UF_1\right), CL_2\left(IF_1\right)\right]\right) \tag{12}$$

where $UL_1(\cdot)$ and $UL_2(\cdot)$ represent the up-sampling layer. UF_1 represents the intermediate features after $UL_1(\cdot)$. $CL_1(\cdot)$, $CL_2(\cdot)$ and $CL_3(\cdot)$ are the convolutional layers. F is the extracted feature which is fed into the capsule transformation matrix.

After the feature extraction, the advanced features are organized into a vector representation by the transformation matrix. In the capsule network, the organization form is as follows:

$$cap = group_d(f_{conv}) \tag{13}$$

where f_{conv} denotes the output of the primary convolutional layer. $group_d$ represents the merge operation, which assembles the d channels as a d-dimensional vector. In CVNN, the d-dimensional vector is obtained by projection transformation $\breve{W}$ with a learnable parameter. The advantage of the learnable transformation matrix is that the parameter of the matrix can be optimized by conventional back-propagation. In this work, each pixel in f_{conv} represents the corresponding spatio-spectral features obtained by the encoder–decoder feature extraction process, which differs from the approach used by Zhang et al. [21,22]. The trainable transformation matrix can thus be implemented in the FCN. Because the transformation is carried out on the 3D feature cube, the dimension of the transformation matrix should stay the same as that of the feature cube.

Firstly, we initialize the learnable transformation matrix $\breve{W} \in \mathbb{R}^{m\times c\times d}$, where each column in $\breve{W}$ denotes the projection subspace of the capsule vector, which consists of the basis of the subspace $span(\breve{W})$. The projection V on all the subspace can be obtained as follows:

$$V = argmin_{V\in span(\breve{W})}\|F - V\| \tag{14}$$

We utilize $\{\breve{w}_1, \breve{w}_2, \ldots, \breve{w}_c\}$ as the basis, forming the transformation matrix $\breve{W}_x$ of f_x, which can be written as follows:

$$\begin{aligned}[v_1, v_2, \ldots, v_c] &= \frac{\langle \breve{w}_1, f_x\rangle}{\langle \breve{w}_1, \breve{w}_1\rangle}\breve{w}_1 + \frac{\langle \breve{w}_2, f_x\rangle}{\langle \breve{w}_2, \breve{w}_2\rangle}\breve{w}_2 + \cdots + \frac{\langle \breve{w}_c, f_x\rangle}{\langle \breve{w}_c, \breve{w}_c\rangle}\breve{w}_c \\ &= [\breve{w}_1, \breve{w}_2, \ldots, \breve{w}_c]\begin{bmatrix} \frac{\breve{w}_1{}^{\mathrm{H}}}{\langle \breve{w}_1, \breve{w}_1\rangle} \\ \frac{\breve{w}_2{}^{\mathrm{H}}}{\langle \breve{w}_2, \breve{w}_2\rangle} \\ \cdots \\ \frac{\breve{w}_c{}^{\mathrm{H}}}{\langle \breve{w}_c, \breve{w}_c\rangle} \end{bmatrix} f_x \\ &= \breve{W}_x\left(\breve{W}_x{}^{\mathrm{H}}\breve{W}_x\right)^{-1}\breve{W}_x{}^{\mathrm{H}} f_x \end{aligned} \tag{15}$$

where $\breve{W}_x\left(\breve{W}_x{}^{\mathrm{H}}\breve{W}_x\right)^{-1}\breve{W}_x{}^{\mathrm{H}}$ can be written as $\breve{W}\left(\breve{W}^{\mathrm{H}}\breve{W}\right)^{-1}\breve{W}^{\mathrm{H}}$ for all the input features F, and has a closed-form solution:

$$V = \breve{W}\breve{W}^{+}F \tag{16}$$

where $\breve{W}^{+}$ is the Moore–Penrose pseudoinverse [45]. The capsule vector of each pixel consists of $V \in \mathbb{R}^{h\times w\times c\times d}$ and is the representation of the advanced features. Each pixel corresponds to c vectors, and each vector is denoted as $v \in \mathbb{R}^{d}$. In CVNN, the capsule vector can be utilized by the decoder part to reconstruct the original features. The reconstruction loss can be added to optimize the model.

The capsule vector is in charge of different characteristics, and is regarded as mutually independent. To this end, the transformation is orthogonal projection. That is to say, the columns in $\breve{W}$ are independent and $\breve{W}^{+}$ can be regarded as $(\breve{W}^{T}\breve{W})^{-1}\breve{W}^{T}$. The norm of V in the third dimension is used to indicate the class of each pixel in the $h\times w$ feature map.

$$\|V\|_2 = \sqrt{V^{T}V} = \sqrt{F^{T}\breve{W}\left(\breve{W}^{T}\breve{W}\right)^{-1}\breve{W}^{T}F} \tag{17}$$

where $\|V\|_2 \in \mathbb{R}^{h\times w\times c}$. The norm of V is regarded as the logit, and the loss is obtained by the softmax function.

After the loss calculation, the gradient is back-propagated, and all the components of the proposed CVNN method can be trained in the end-to-end framework.

3.3. The learning process

The final classification map is obtained by the feature extraction phase and the classification phase in CVNN. Firstly, the whole hyperspectral image $HSI \in \mathbb{R}^{h\times w\times b}$ is fed into the feature extraction part, which is a kind of encoder–decoder structure for high-level abstract feature learning. The extracted features have the same width and length as the original hyperspectral image, and are processed by the capsule transformation. After the transformation, the features are projected to the capsule feature space using the projection matrix. The spanning set is determined by the number of capsules, which is consistent with the categories. The basis of each span in the capsule transformation matrix gives rise to each capsule. The extracted features then turn into a set of characteristic vectors, and the vectors are regarded as the capsule vectors. The norm of a capsule vector indicates the probability of

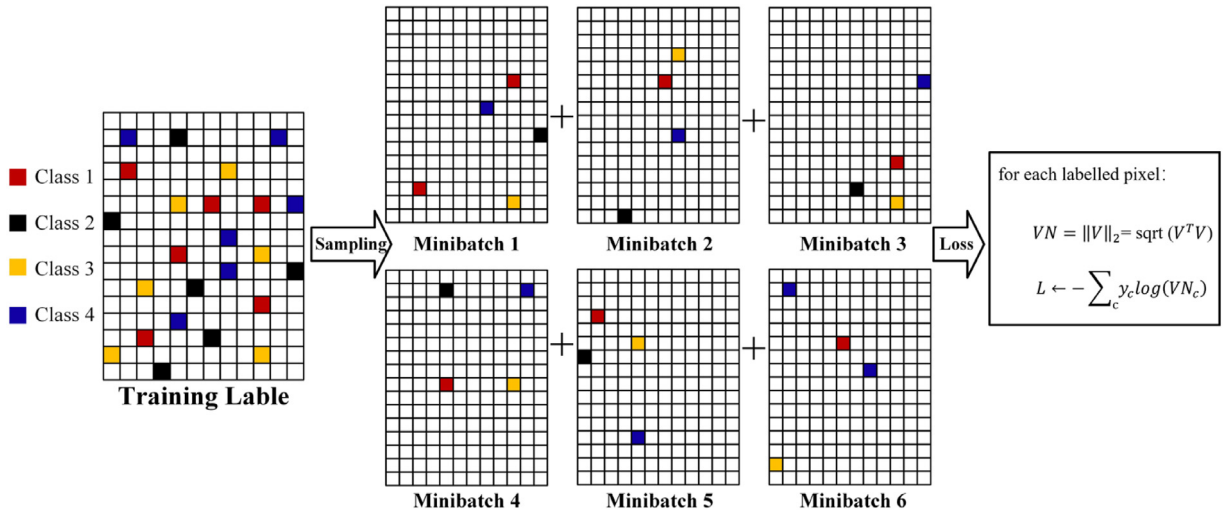


Fig. 1. Labeled pixel sampling in the learning process.

belonging to a class. The category corresponding to the maximum norm is the final label for the pixel. After this, all the pixels in the hyperspectral image can be labeled by CVNN.

In order to control the experimental conditions, the number of labeled training samples is 10 samples per class, and the other labeled pixels are employed in the accuracy assessment in each dataset. Ten samples per class in the training process are divided into model learning and model validation sets [27]. Because CVNN is based on the encoder–decoder structure, the whole hyperspectral image is fed into the CVNN framework, and the labeled image keeps the same length and width as the original input. In this way, the pixels of the labeled image are partly labeled based on the training dataset. Only some of the labeled samples in each category are selected to generate the labeled image $T' \in \mathbb{R}^{h\times w}$ in one training iteration, which is illustrated in Fig. 1. For example, the number of categories in the Kansas Advanced Hyperspectral Imager (AHSI) dataset is seven and T' contains seven labeled pixels which cover all categories. The loss calculation is executed based on the labeled pixels rather than the whole image. Because the labeled data pixels are few in number, only one labeled sample per class is employed for the model validation, and the other nine labeled samples are used for the model learning.

3.4. Capsule-vectored neural network

The proposed CVNN consists of a fully convolutional part and a capsule vector learning part. Because of the good performance of the capsule vector in the discrimination process, CVNN can obtain a high accuracy using a vanilla FCN, which is performed as shown in Fig. 2.

CVNN employs encoding and decoding processes. In the encoding process, three layers are used for the feature representation, in which two down-sampling operations are adopted for the down-scaling. Firstly, the initial part is utilized to extract the primary features of $HSI \in \mathbb{R}^{h\times w\times b}$, without any scale changes. The output of the initial part is $f_1 \in \mathbb{R}^{h\times w\times m_1}$. After the initial part, the following convolution is of stride 2, which generates the down-sampling feature $f_2 \in \mathbb{R}^{h/2\times w/2\times m_2}$. f_2 is then fed into another down-sampling layer which has the same stride, and the output is $f_3 \in \mathbb{R}^{h/4\times w/4\times m_3}$. The output is conserved for the lateral connection of the decoding process during the encoding process. Before the lateral connection, additional convolution is carried out for the fine adjustment. In the decoding process, CVNN adopts interpolation for the up-sampling, on account of the shortcut connection. The channels of the output feature are doubled, which should be adjusted by an additional convolutional layer. After the two up-sampling layers, the extracted feature F is obtained. The transformation matrix is utilized to obtain the capsule vector. To this end, the posterior probability can be achieved by the norm of the capsule vector using the softmax function.

The supervised training is carried out by sampling. Inspired by Zheng et al. [27], the whole hyperspectral image is fed into the CVNN framework, and only the labeled pixels are utilized in the transformation to the capsule vector and in the calculation of the loss in the training process. In each training iteration, random sampling is carried out on the labeled pixels, and the chosen samples are employed for the loss calculation. Table 1 details the proposed learning algorithm.

4. Experiments and discussion

In this paper, the overall accuracy (OA) and kappa coefficient are employed to report the performance of the proposed method. The OA is the ratio of the number of correctly classified pixels to total pixels. The kappa coefficient refers to the consistency between the model labels and the actual labels of all the pixels. The insufficient sample scenario for the training dataset was achieved by running the model with 10 labeled pixels in each category. The counterpart methods consisted of SVM as a representative conventional classifier, a vanilla CNN as a representative of the traditional patch-based methods, a vanilla CapsNet as a representative of the traditional capsule network methods, the multiscale residual network (MSRN) [46], A^2S^2 ResNet [47], the FCN-based method of fast patch-free global learning (FPGA) [27], UML [1], and the few-shot method of S3Net [31]. The details of each counterpart methods are given as follows:

(1) CNN-4 uses a 2D CNN with two layers and the patch size is 4.
(2) CapsNet-4 is the original capsule network, which consists of patch-based inference and dynamic routing optimization.
(3) MSRN features a multiscale residual module on a mixed depthwise CNN, and has been shown to have a good performance for hyperspectral image classification.
(4) A^2S^2 ResNet is an attention-based deep learning method using 3D convolution and an adaptive spectral–spatial kernel.

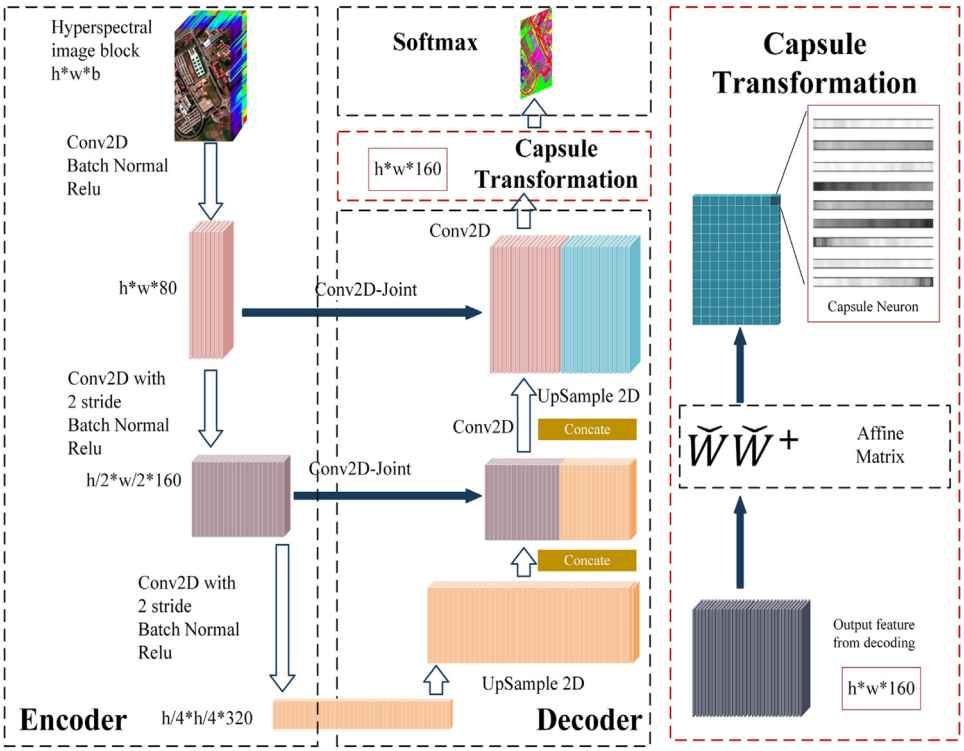


Fig. 2. The structure of the CVNN framework for hyperspectral image classification.

(5) FPGA is a widely utilized method based on an FCN, which has the same training strategy as CVNN.
(6) UML is an FCN-based model that employs multiscale feature fusion and channel shuffle.
(7) S3Net is a lightweight spectral–spatial network for few-shot hyperspectral image classification, which is based on patch-based inference.

The hyperparameter C and gamma of SVM with restricted Boltzmann machine (RBM) kernel were optimized. MSRN and the vanilla CNN are 2D CNNs and A^2S^2 ResNet is based on a 3D CNN. To test the effect of the capsule vector, a CNN with capsule representation was also used, which is named CapsNet [18]. All the deep learning based methods were deployed on an NVIDIA GeForce RTX 3090 graphics processing unit (GPU). For a fair comparison, the patch size was fixed as 4 in the vanilla CNN and CapsNet, and the corresponding counterpart methods were named CNN-4 and CapsNet-4, respectively. The parameter settings of MSRN and A^2S^2 ResNet were consistent with the optimal settings in the original works. In CapsNet-4, the capsule layer followed the conventional CNN with two layers. After the feature extraction of the patches, the features were squeezed into a 1D vector to indicate the posterior probability. In CVNN, the output of the feature extraction had 160 channels, which was transformed into the capsule vector. The length of each capsule vector was 4, and the number of capsules was consistent with the number of categories. The kernel of each convolution operation in the FCN-based and patch-based methods was uniformly 3×3, and the number of training iterations was 1500. The sampling approach was based on Zheng et al. [27]. The optimizer was the Adam optimizer [48]. The total parameter size of CVNN was 2.399M, 2.47M, 2.57M, and 2.43M on the Pavia University, Indian Pines, Kansas AHSI, and Houston CASI datasets, respectively.

4.1. Dataset description

4.1.1. Indian Pines AVIRIS dataset

The Indian Pines dataset was collected by the Airborne Visible Infrared Imaging Spectrometer (AVIRIS). The width and height of this image is 145×145 pixels and the spatial resolution is 17 m. There are 16 categories of objects in the Indian Pines dataset, including different kinds of mixed objects. The many kinds of vegetation bring a challenge to the classification task. The given spectral features include 220 spectral bands with a 5-nm spectral resolution after noisy band removal. Fig. 3 depicts the image and the corresponding labeled categories.

4.1.2. Pavia University ROSIS dataset

The Pavia University dataset was acquired by the Reflective Optics System Imaging Spectrometer (ROSIS). The width and height of this image is 610×340 pixels and the spatial resolution is 1.3 m. There are nine categories of objects in the Pavia University dataset, including different kinds of vegetation and artificial ground objects. The scattered objects represent a classification challenge. The given spectral features include 103 spectral bands with a 4.0-nm band width after noisy band removal. Fig. 4 depicts the image and the corresponding labeled categories.

4.1.3. Kansas AHSI dataset

The Kansas AHSI dataset was acquired by the visible-shortwave infrared Advanced Hyperspectral Imager (AHSI), and is the only satellite-based hyperspectral imagery dataset among these four datasets. The AHSI instrument is carried onboard the GF-5 satellite of China. The width and height of this image is 650×340 pixels. The spatial resolution is 30 m, which brings a challenge to the classifier performance. There are seven labeled categories in the Kansas AHSI dataset. Fig. 5 depicts the image and the corresponding labeled categories.

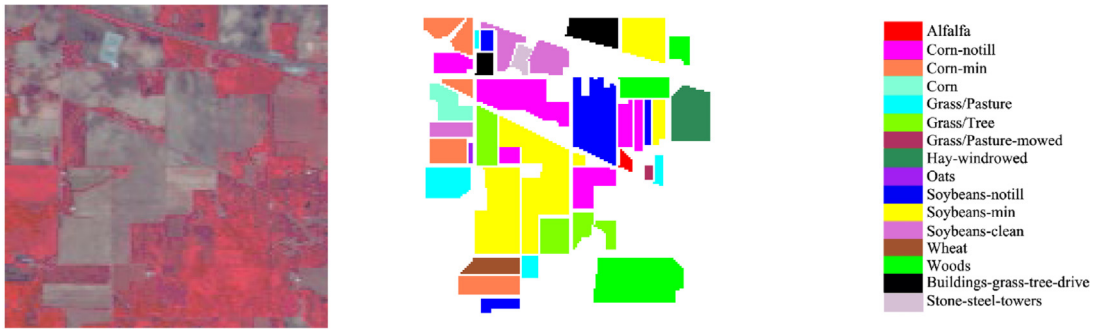


Fig. 3. The image and the corresponding labeled categories for the Indian Pines AVIRIS dataset.

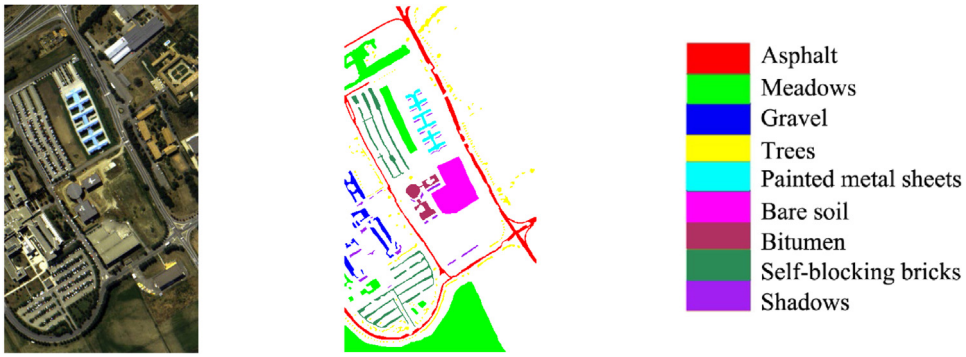


Fig. 4. The image and the corresponding labeled categories for the Pavia University ROSIS dataset.

Table 2
Number of labeled pixels in the Indian Pines dataset.

Class name	Training set number	Test set number	Labeled sample number
Alfalfa	10	36	46
Corn-Notill	10	1418	1428
Corn-Mintill	10	820	830
Corn	10	227	237
Grass-Pasture	10	473	483
Grass-Trees	10	720	730
Grass-Pasture-Mowed	10	18	28
Hay-Windrowed	10	468	478
Oats	10	10	20
Soybean-Notill	10	962	972
Soybean-Mintill	10	2445	2455
Soybean-Clean	10	583	593
Wheat	10	195	205
Woods	10	1255	1265
Bs-Grss-Trs-Drs	10	376	386
Stone-Steel-Towers	10	83	93

4.1.4. Houston CASI dataset

The Houston CASI dataset was acquired by the ITRES Research Ltd. CASI-1500 sensor over the University of Houston and its surroundings in Texas, USA. There are 349 × 1905 pixels and 144 bands in the spatial and spectral dimensions of the Houston CASI image, respectively. The spatial resolution is 2.5 m. There are 15 labeled categories in the Houston CASI dataset. Fig. 6 depicts the image and the corresponding labeled categories.

4.2. Experiments with the Indian Pines dataset

In order to control the experimental conditions, the number of labeled training samples was 10 samples per class, and the other labeled pixels were employed in the accuracy assessment. The training and test sets in the Indian Pines dataset are listed in Table 3 (see Table 2).

The accuracy of the different approaches is reported in Table 4. Because of the sparse labeled data and the large number of categories, the OAs for all the methods are lower than 90%. The proposed CVNN method obtains the best result with an OA of 82.07%, which is 3.22% higher than the OA of the second-best approach. The OA of S3Net is higher than that of the other comparison methods, with an OA of 80.33% (and a kappa of 0.7764), which demonstrates the validity of weighted contrastive learning in the case of limited labeled samples. The third-best performing method is FPGA, with an OA of 78.85% and a kappa of 0.7613, and FPGA outperforms the other methods by over 10% in OA, at least. The results of the deep learning based methods are better than the result of the traditional SVM method, because of the more efficient feature extraction and the utilization of spatio-spectral features. The original CapsNet-4 is superior to CNN-4, which reveals the advancement of the capsule vector when compared with the scalar neuron. The proposed CVNN obtains the highest accuracy on nine categories, which is over half of the categories in the Indian Pines dataset. For the classes of Alfalfa, Grass-Pasture-Mowed, Hay-Windrowed, and Oats, the classification OA is 100%.

Fig. 7 shows the classification maps generated by each method. The result of SVM contains the most obvious salt-and-pepper noise, followed by the result of CNN-4. The obvious salt-and-pepper noise is improved in the results of MSRN and A^2S^2 ResNet, which use a larger patch size than CNN-4 and CapsNet-4. Significant noise appears on the boundary of the classification map obtained by MSRN. The Indian Pines dataset contains many kinds of vegetation, and there is less differentiation in the spectral features. The dataset was collected over an agricultural region, and there are several roads between the parcels of cultivated fields. The pixels of the roads can be classified well by SVM, CNN-4, and CapsNet-4. Meanwhile, in the results of MSRN, A^2S^2 ResNet, and FPGA, the profiles of the roads are obscured by the over-smooth classification results. Moreover, the boundaries of the cultivated fields are not sharp in the results of MSRN, A^2S^2

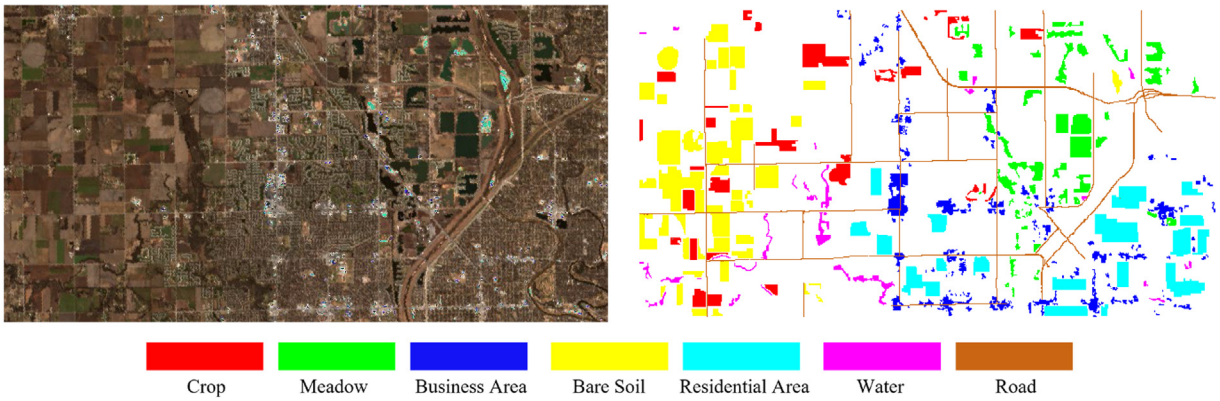


Fig. 5. The image and the corresponding labeled categories for the Kansas AHSI dataset.

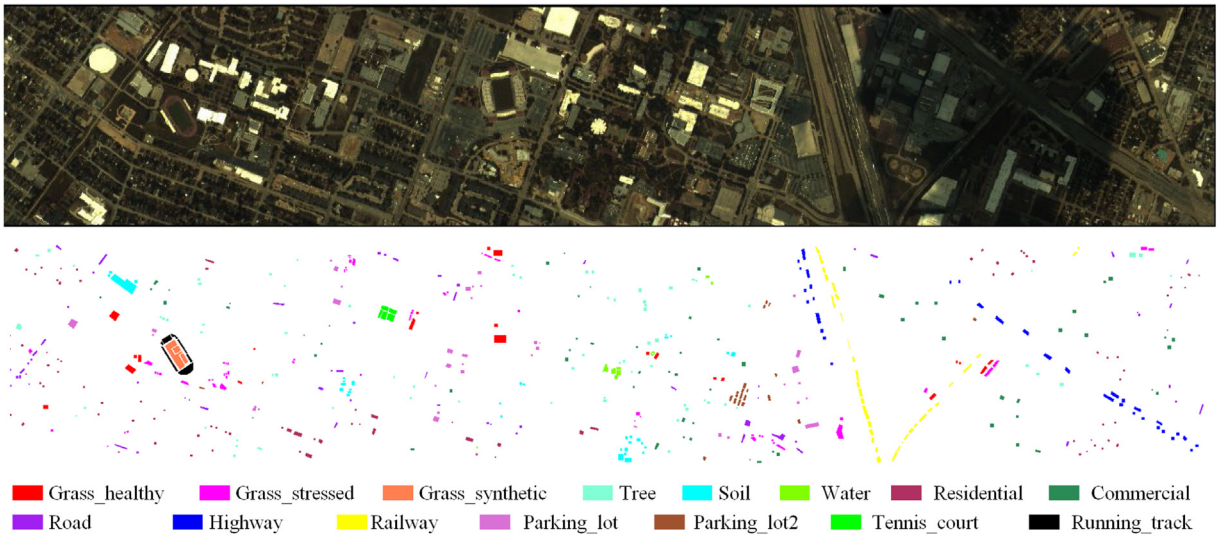


Fig. 6. The image and the corresponding labeled categories for the Houston CASI dataset.

Table 3
The accuracy of the different methods for the Indian Pines dataset.

Category	SVM	CNN-4	CapsNet-4	MSRN	A^2S^2ResNet	FPGA	UML	S3Net	CVNN
Alfalfa	86.11	97.22	81.25	100	100	100	97.22	100	100
Corn-Notill	35.97	38.93	48	46.21	37.31	55.57	61	66.74	69.82
Corn-Mintill	58.54	38.79	44.96	75.93	45.22	85.73	78.17	58.04	91.71
Corn	57.71	39.65	87.92	100	86.3	74.89	69.13	81.53	97.8
Grass-Pasture	83.93	84.71	26.49	52.1	92.52	82.03	86.47	85.47	78.86
Grass-Trees	76.81	79.72	73.86	95.97	96.81	93.19	99.17	88.67	90
Grass-Pasture-Mowed	88.89	83.33	100	100	88.89	100	100	100	100
Hay-Windrowed	60.26	74.36	77.46	54.69	99.78	98.93	95.73	90.5	100
Oats	100	50	100	80	100	100	100	100	100
Soybean-Notill	64.66	66.81	75.64	65.49	54.08	86.28	85.65	83.28	86.9
Soybean-Mintill	28.26	54.5	44.09	47.35	59.99	70.76	62.94	77.21	82.54
Soybean-Clean	47.34	25.56	38.54	52.36	64.55	79.93	70.15	80.97	86.79
Wheat	91.79	84.1	90.29	98.46	97.95	93.33	96.92	100	96.92
Woods	59.44	47.65	74.49	89.52	85.02	83.03	87.97	93.12	88.84
Bs-Grss-Trs-Drs	28.46	43.63	29.73	90	80.43	100	100	96.77	99.2
Stone-Steel-Towers	87.95	90.36	84.13	97.59	87.95	98.8	98.8	100	96.39
OA	50.59	53.44	55.77	64.82	67.02	78.85	77.51	80.33	**85.58**
KAPPA	0.4546	0.4768	0.504	0.6098	0.6283	0.7613	0.7464	0.7764	**0.8366**

ResNet, and FPGA, which is caused by the large patch size and the loss of detailed features in the down-sampling. Fig. 8 logs the training loss and testing loss of CVNN through the training process, where the training loss is stable when the iterations near 100 and the testing loss is stable when the iterations are about 800.

Floating-point operations (FLOPs) and the running time are utilized for the complexity analysis of each method, which are

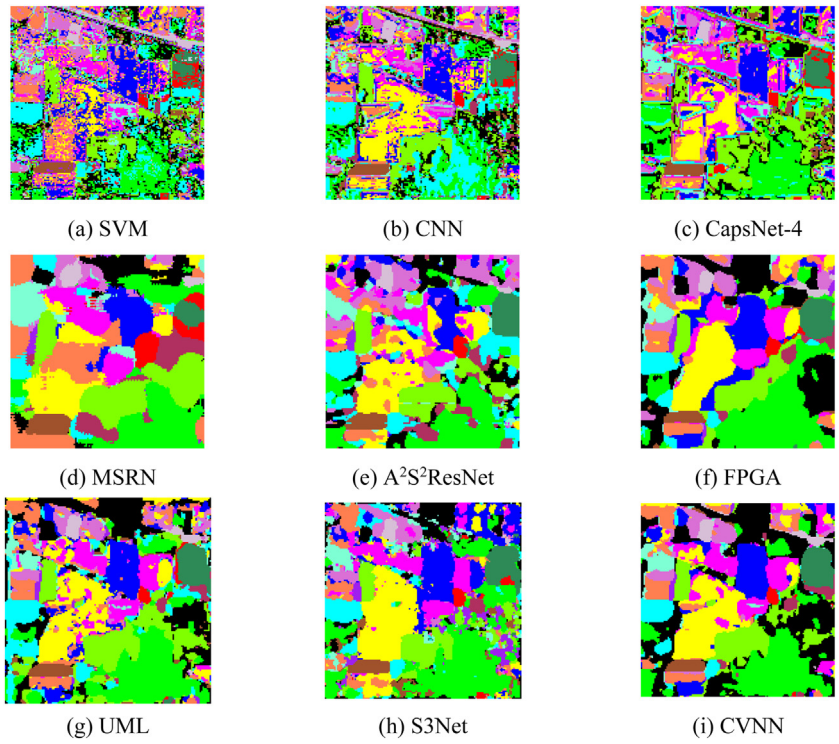


Fig. 7. The results obtained by the different methods with the Indian Pines dataset.

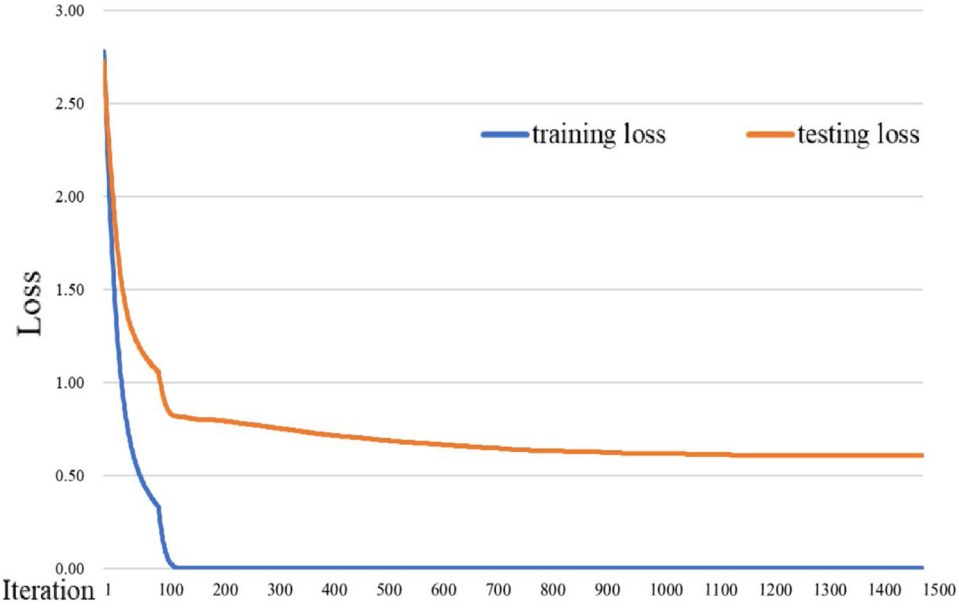


Fig. 8. The loss curves of CVNN on the Indian Pines dataset.

listed in Table 4. The calculation of FLOPs was conducted on the inference data of each deep learning method. For the patch-based methods, such as CNN-4 Capsule-4, MSRN, A^2S^2 ResNet, and S3Net, the FLOPs were calculated on the patch. For the FCN-based methods, such as FPGA, UML and CVNN, the FLOPs were calculated on the whole image. Therefore, the total FLOPs of the patch-based methods are a summation of all the layers in the network for the input hyperspectral patch, multiplied by the pixel number. The inference speed of the FCN-based methods is faster than that of the patch-based methods because the data patch is obtained by sliding a window in a one-step stride, which involves many repetitive calculations.

4.3. Experiments with the Pavia University dataset

The training and test sets in the Pavia University dataset are listed in Table 5. The test set number is tens of thousands of times more than the training set number in some categories.

Table 5 lists the accuracies of all the approaches for the Pavia University dataset. Among the seven methods, only the accuracy of CVNN is more than 90%, at 92.48%, which is higher than the second-best performing method by 4.73%. The second-best performing method is UML, with an OA of 89.91% and a kappa of 0.8578. As shown in Fig. 9, the three FCN-based approaches are superior to the other methods, not just in terms of the OA

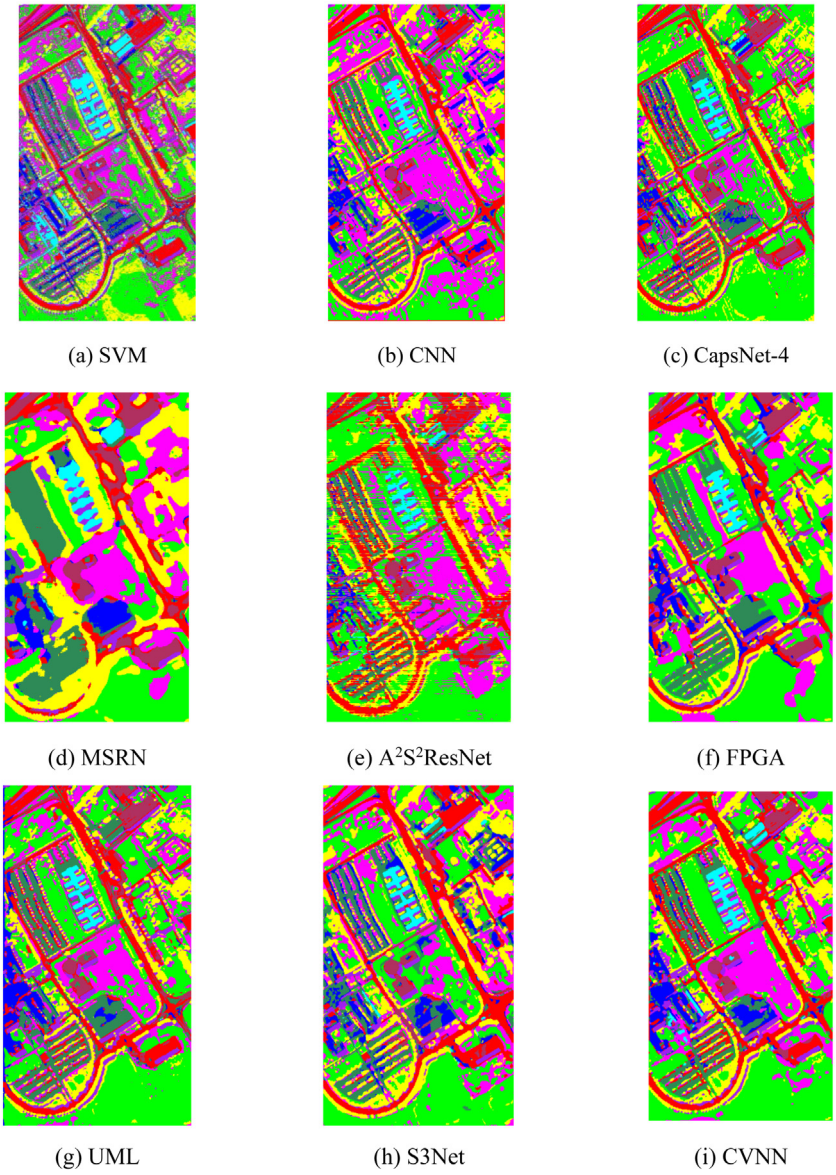


Fig. 9. The results obtained by the different methods for the Pavia University dataset.

Table 4
Complexity analysis of the different approaches on the Indian Pines dataset.

Indicator	CNN-4	CapsNet-4	MSRN	A^2S^2ResNet	FPGA	UML	S3Net	CVNN
Running time (s)	5.58	5.74	5.82	12.96	1.23	1.98	16.06	0.52
GFLOPs	11.63	120.68	31.52	3838.14	17.13	21.46	1341.61	34.12

and kappa, but also the classification map. Compared with FPGA, CVNN has a more concise and efficient network structure for feature extraction, and the capsule transformation endows CVNN with a remarkable feature representation ability. The classification map of CNN-4 is better in terms of salt-and-pepper noise than that of SVM. The performance of CapsNet exceeds that of the other patch-based methods, with an OA of 84.22% and a kappa of 0.7905, which demonstrates that the vector neuron is a more efficient approach in feature learning. Moreover, the noise in the classification result is improved, compared with the vanilla CNN (see Table 6).

The three FCN-based approaches obtain a higher OA than the other methods, which indicates that the FCN framework is favorable in hyperspectral image classification. The OA and kappa values of SVM are the lowest, at 67.19% and 0.5907, respectively,

Table 5
Number of labeled pixels in the Pavia University dataset.

Class name	Training set number	Test set number	Labeled sample number
Asphalt	10	6621	6631
Meadows	10	18 639	18 649
Gravel	10	2089	2099
Trees	10	3054	3064
Painted metal sheets	10	1335	1345
Bare soil	10	5019	5029
Bitumen	10	1320	1330
Self-blocking bricks	10	3672	3682
Shadows	10	937	947

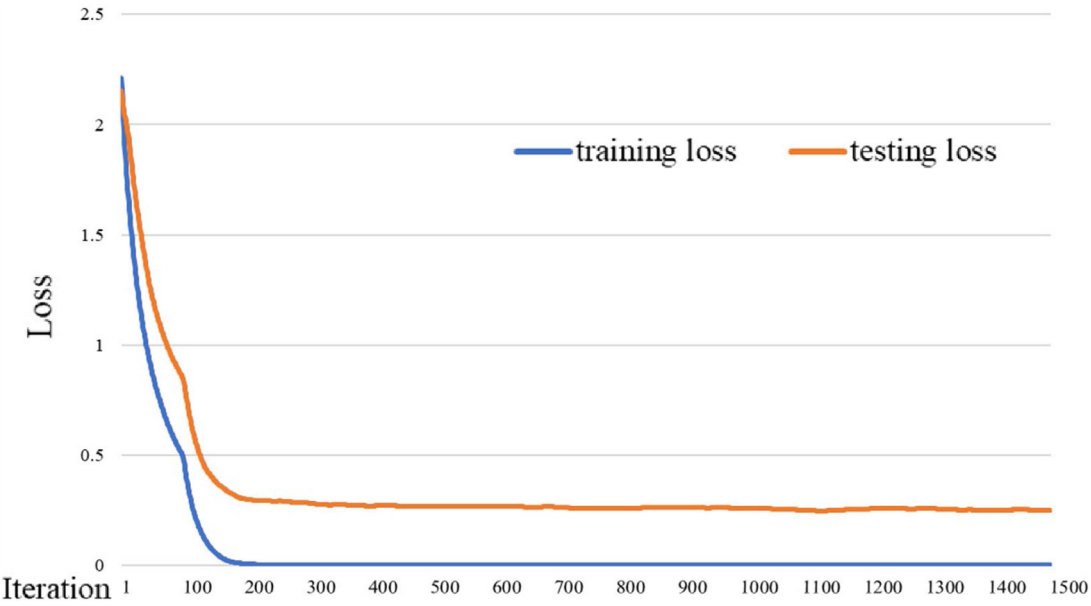


Fig. 10. The loss curve of CVNN on the Pavia University dataset.

Table 6
The accuracy of the different methods for the Pavia University dataset.

Category	SVM	CNN-4	CapsNet-4	MSRN	A^2S^2ResNet	FPGA	UML	S3Net	CVNN
Asphalt	68.95	92.51	79.8	57.78	84.36	90.15	78.9	93.47	95.67
Meadows	60.94	57.29	94.79	87.52	89.75	80.86	89.85	89.14	88.31
Gravel	63.14	77.84	61.83	33.42	26.94	72.33	85.3	47.79	87.94
Trees	87.3	87.69	90.76	93.23	74.78	95.81	94.79	96.82	96.99
Painted metal sheets	99.63	100	99.55	85.84	86.52	100	99.25	99.7	99.93
Bare soil	62.56	91.59	50.91	91.45	80.39	97.47	85.36	49.42	98.85
Bitumen	85.98	85.83	96.74	82.73	81.51	92.58	97.312	99.16	95.08
Self-blocking bricks	60.84	97.52	79.87	89.19	81.24	97.98	97.14	80.31	93.36
Shadows	99.68	98.83	99.68	74.49	92.32	99.47	99.36	95.49	99.89
OA	67.19	76.56	84.22	80.76	82.44	87.75	89.11	83.68	**92.48**
KAPPA	0.5907	0.7098	0.7905	0.758	0.7705	84.28	85.78	0.784	**0.9023**

Table 7
Complexity analysis of the different approaches on the Pavia University dataset.

Indicator	CNN-4	CapsNet-4	MSRN	A^2S^2ResNet	FPGA	UML	S3Net	CVNN
Running time (s)	22.02	23.85	16.72	38.41	1.41	2.03	109.4	0.83
GFLOPs	113.24	4946.49	311.11	20 775.63	128.34	168.51	6864.11	277.41

which means that 10 labeled samples per class is not enough for the optimization of SVM. The performance of S3Net is better than that of MSRN and A^2S^2 ResNet, which means that the Siamese structure is effective in handling few-shot learning. The salt-and-pepper noise and stripe noise are relatively significant in the results of SVM and A^2S^2 ResNet. The result of MSRN is over-smooth and cannot represent the real shapes of the ground objects, which is caused by the large patch size. The patch-based models of CNN-4, A^2S^2 ResNet, and MSRN utilize patches step by step for the whole image classification inference, and the labels of two adjacent pixels are obtained based on two almost identical image patches, which results in a high probability of the same labels. This inference pattern brings unfavorable effects in the boundary pixels of the different objects. With the larger patch size, the mislabeling phenomenon is more obvious.

The results of FPGA and CVNN are closer to the true distribution of the land cover, and the result of CVNN shows clearer profiles for the scattered objects, such as the solitary trees and the objects on the roof. In SVM, the inference is pixel to pixel, i.e., the spectral vector of the pixel generates the label of the pixel, which guarantees retention of the details. Unfortunately, keeping the detailed information introduces salt-and-pepper noise in the classification map. In both the patch-based methods and FCN-based methods, the convolution operation takes the neighborhood context into account and labels each pixel by all the features in the receptive field, which leads to the loss of detailed information. The result of the proposed CVNN method shows that this method is able to keep the details while avoiding the salt-and-pepper noise, which can be attributed to the capsule transformation and simple network design.

The FLOPs and running times of all methods are reported in Table 7. The inference time of CVNN is shorter than that of the other methods, at 0.83 s. Although the vector-neuron representation brings more FLOPs in CVNN, the simple layer propagation with capsule transformation consumes a minimal amount of time. Fig. 10 logs the training loss and testing loss of CVNN through the training process, where the training loss is stable when the iterations are about 200, and the testing loss is stable when the iterations are about 300.

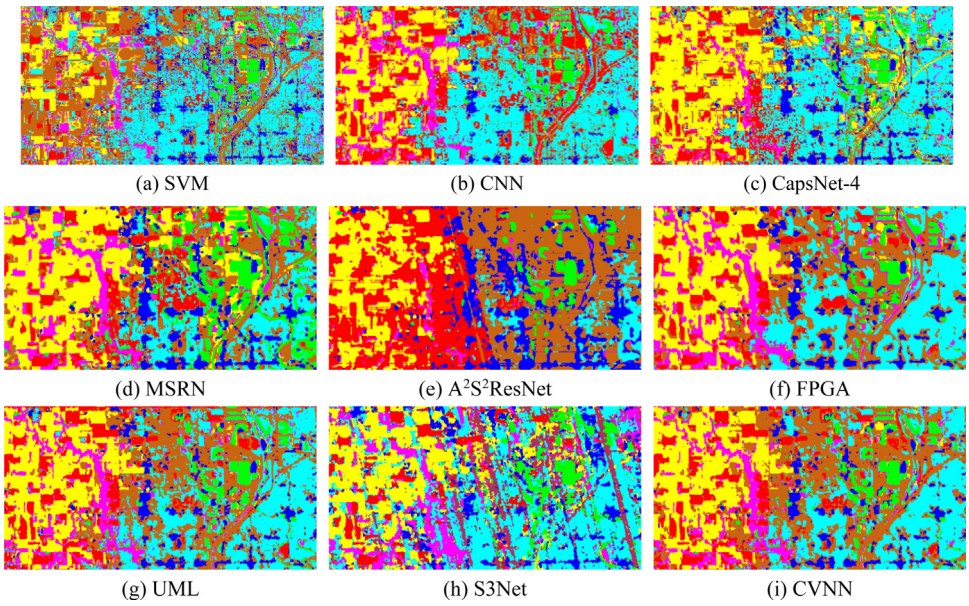


Fig. 11. The results obtained by the different methods for the Kansas AHSI dataset.

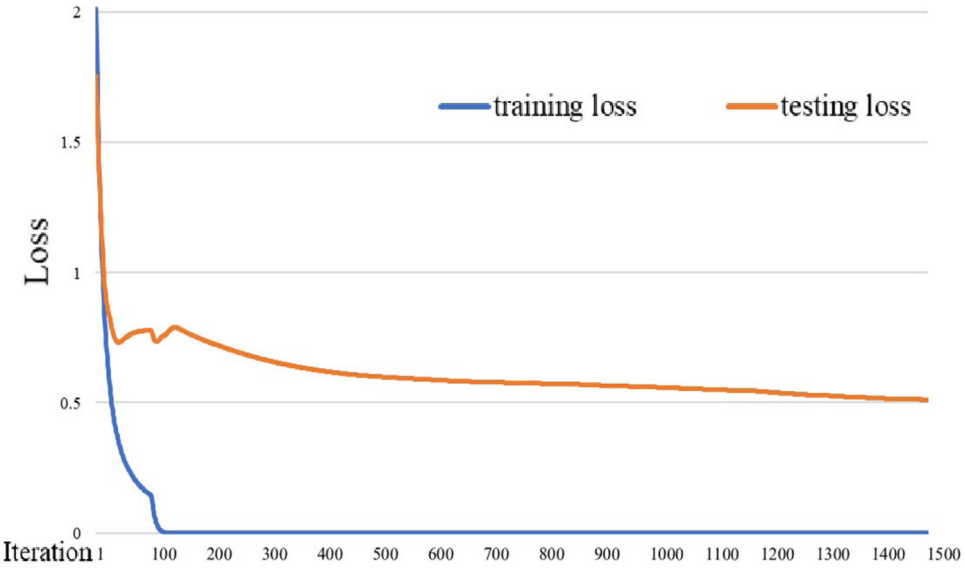


Fig. 12. The loss curve of CVNN on the Kansas AHSI dataset.

4.4. Experiments with the Kansas AHSI dataset

The training and test sets in the Kansas AHSI dataset are listed in Table 8. The test set number is tens of thousands of times more than the training set number in some categories. Ten labeled pixels were randomly selected for the training of each method. The Kansas AHSI dataset has the lowest spatial resolution among the four datasets, which means that there are some seriously mixed pixels. Moreover, the number of spectral bands in the Kansas AHSI dataset is the highest, with 330 bands from 400 nm to 2500 nm. Because of the scene being of an urban community, the land covers have complex boundaries.

Table 9 lists the OA and kappa values of all methods on the Kansas AHSI dataset. The OAs of CapsNet-4, FPGA, UML, and CVNN are higher than 80%. The superiority of the vector neuron is again proved by the result of CapsNet-4. The three best performances are obtained by the three FCN-based methods, i.e., FPGA, UML and CVNN, with OAs of 85.87%, 87.1%, and 88.14%, respectively, which are over 5% higher than the OAs of the other methods. A^2S^2 ResNet achieves the worst performance, with an OA of 71.26% and a kappa of 0.6566, which are lower than the OA and kappa of SVM.

Fig. 11 illustrates the classification results of all the approaches. The salt-and-pepper noise in the result of SVM is serious, but

Table 8
Number of labeled pixels in the Kansas AHSI dataset.

Class name	Training set number	Test set number	Labeled sample number
Crop	10	5193	5203
Meadow	10	5458	5468
Business area	10	4685	4695
Bare soil	10	12 078	12 088
Residential area	10	8225	8235
Water	10	1673	1683
Road	10	4571	4581

Table 9
Accuracy of the different approaches for the Kansas AHSI dataset.

Category	SVM	CNN-4	CapsNet-4	MSRN	A^2S^2ResNet	FPGA	UML	S3Net	CVNN
Crop	76.04	97.78	93.35	93.74	88.35	95.32	98.07	94.22	98.22
Meadow	96.94	92.58	88.09	99.51	94.78	93.29	84.69	95.36	97.82
Business area	88.88	67.76	78.08	95.55	90.66	90.22	91.14	74.38	87.62
Bare soil	44.19	78.65	88.36	81.57	76.71	83.07	84.44	83.65	86.93
Residential area	84.67	93.68	78.07	66.17	44.98	91.59	88.64	78.8	90
Water	91.21	82.8	86.76	82.28	58.12	94.74	93.9	86.99	93.78
Road	57.73	23.54	37.67	37.09	41.15	55.72	63.25	32.92	63.49
OA	71.32	78.69	80.17	79.14	71.26	85.87	87.1	79.09	**88.14**
Kappa	0.6613	0.742	0.7583	0.7487	0.6566	0.8298	0.8444	0.7459	**0.8566**

Table 10
Complexity analysis of the different approaches on the Kansas AHSI dataset.

Indicator	CNN-4	CapsNet-4	MSRN	A^2S^2ResNet	FPGA	UML	S3Net	CVNN
Running time (s)	26.53	28.49	41.62	105.35	2.47	2.49	9.83	1.17
GFLOPs	120.34	6296.29	331.54	68 867.05	175.76	211.16	4406.96	325.71

Table 11
Number of labeled pixels in the Houston CASI dataset.

Class name	Training set number	Test set number	Labeled sample number
Grass_healthy	10	1241	1251
Grass_stressed	10	1244	1254
Grass_synthetic	10	689	697
Tree	10	1234	1244
Soil	10	1232	1242
Water	10	315	325
Residential	10	1258	1268
Commercial	10	1234	1244
Road	10	1242	1252
Highway	10	1217	1227
Railway	10	1225	1235
Parking_lot	10	1223	1233
Parking_lot2	10	459	469
Tennis_court	10	418	428
Running_track	10	650	660

is much better in the results of CNN-4 and CapsNet-4. The salt-and-pepper noise does not appear in the result of MSRN. There is horizontal stripe noise in the result of A^2S^2 ResNet, which may have been caused by the direction of the inference and the instability of the model. There is a road in the right part of the image, which is misclassified in the result of CNN-4. A narrow road is represented by just one pixel in hyperspectral imagery with a 30-m spatial resolution, and the models with a large patch size cannot successfully identify the "Road" category in the Kansas AHSI dataset. The "Water" category is also not recognized in the results obtained by SVM and A^2S^2 ResNet.

The FLOPs and running times of all the methods are reported in Table 10. The inference time of CVNN is shorter than that of the other methods, at 1.17 s. The inference times of the three FCN-based methods are shorter than those of the patch-based methods. Fig. 12 logs the training loss and testing loss of CVNN through the training process, where the training loss is stable when the iterations are about 100, and the testing loss is stable when the iterations are about 1400.

4.5. Experiments with the Houston CASI dataset

The training and test sets of the Houston CASI dataset are listed in Table 11. The test set number is hundreds of times more than the training set number in most of the categories.

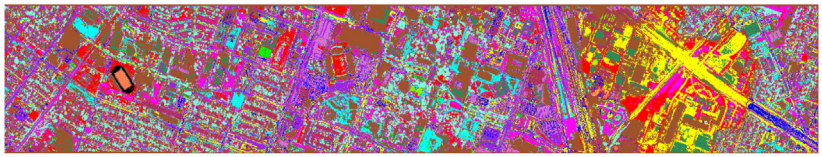
Table 12 lists the accuracies of all the approaches for the Houston CASI dataset. The proposed CVNN method obtains the best performance in over half the categories. Only the accuracies of A^2S^2 ResNet, FPGA, UML, and CVNN are over 80%. The highest accuracy of 89.16% is obtained by CVNN, which is higher than the accuracy of the second-best performing method by 1.9%. Because of the scene being the biggest among the four datasets, the local features from the patch-based models cannot represent the complex land cover well in the Houston CASI dataset. The accuracies of the three FCN-based models are higher than those of the other methods. The OA of S3Net is higher than that of SVM, CNN-4, and CapsNet-4, which proves that the features extracted by the Siamese structure have good separability. The three best performances are obtained by the three FCN-based methods, i.e., FPGA, UML, and CVNN, with OAs of 86.29%, 87.26%, and 89.16%, respectively, which are over 4% higher than the OAs of the other approaches.

Fig. 13 shows the classification maps of all the methods. The integrity of the FCN-based methods of FPGA, UML, and CVNN is better than that of the other methods. The results of SVM, CNN-4, and CapsNet-4 contain lots of salt-and-pepper noise. Because of the cloud cover in the right of the Houston CASI dataset, the results of the comparison methods are impacted. Many pixels are misclassified into "Highway" or "Railway" under the cloud-covered area in the results of SVM, CNN-4, CapsNet-4, MSRN, A^2S^2 ResNet, FPGA, and UML. The results of S3Net and CVNN show a better performance in the cloud-covered area. The FLOPs and running times are reported in Table 13. The inference time of CVNN is 1.9 s, which is faster than that of the other methods. Because of the simple network structure, the FLOPs of CVNN are less than those of UML. Fig. 8 logs the training loss and testing loss of CVNN through the training process, where the training loss and testing loss are stable when the iterations are about 300, which indicates that CVNN has been trained effectively with the limited labeled samples. When the curves converge, the error on the test dataset is also stable (see Fig. 14).

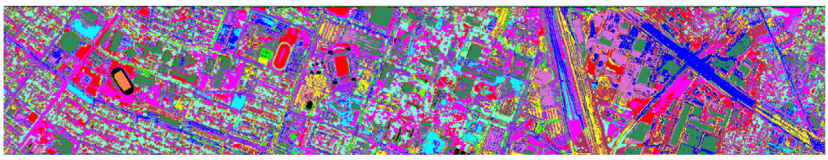
4.6. Discussion of the experimental settings

In this section, we describe how the performance of CVNN was evaluated based on different numbers of labeled samples, to explore the sensitivity of the proposed method toward different sizes of training sets. For the different numbers of labeled samples (5, 10, and 15 samples per class), the OAs obtained on the four datasets are shown in Fig. 15. The OA and kappa obtained on the four datasets are reported in Table 14.

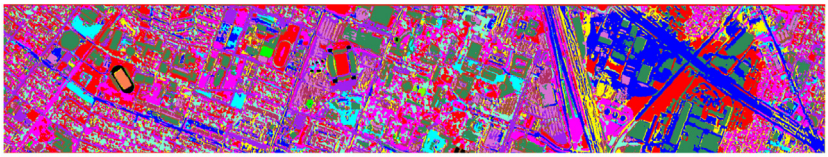
On the Indian Pines dataset, the OA when using 5 labeled samples per class is less than 80%, and the training process is unstable. The increasing trend in accuracy when using 10 and 15 labeled samples per class is more stable than with 5 labeled samples. Although the initial OA for 15 labeled samples is lower than that for 10 labeled samples, the OA becomes the highest after



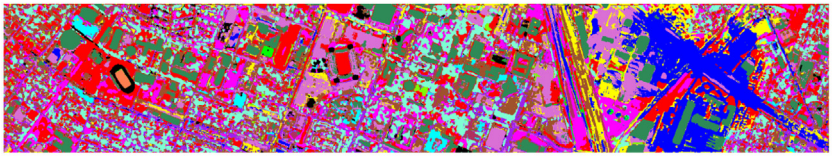
(a) SVM



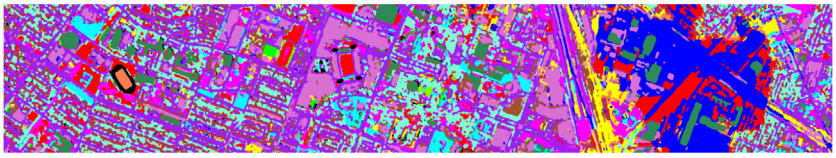
(b) CNN-4



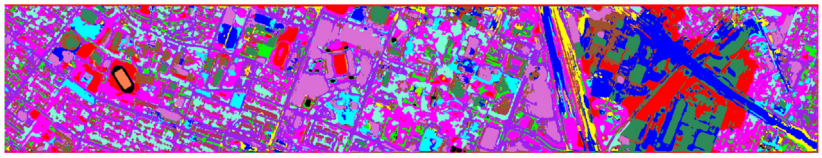
(c) CapsNet-4



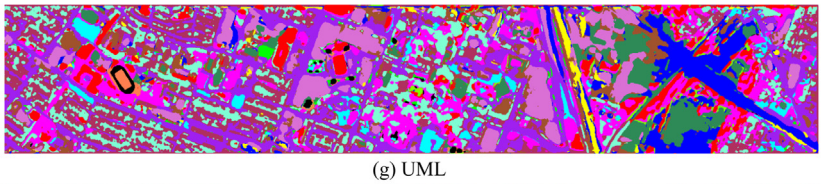
(d) MSRN



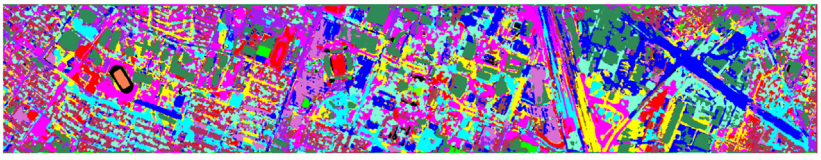
(e) A^2S^2ResNet



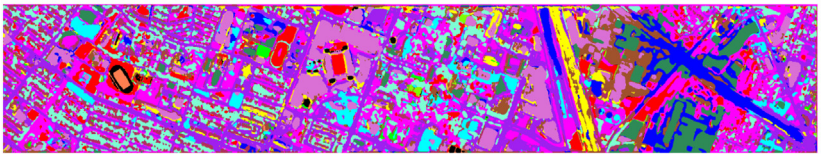
(f) FPGA



(g) UML



(h) S3Net



(i) CVNN

Fig. 13. The results obtained by the different methods for the Houston CASI dataset.

Table 12
The accuracy of the different methods with the Houston CASI dataset.

Category	SVM	CNN-4	CapsNet-4	MSRN	A^2S^2ResNet	FPGA	UML	S3Net	CVNN
Grass_healthy	83	94.43	100	99.27	99.27	97.4	97.18	93.12	97.9
Grass_stressed	71.3	98.39	86.58	94.86	86.98	84.97	97.19	76.92	98.95
Grass_synthetic	98.11	94.47	99.13	98.25	100	100	100	100	100
Tree	70.83	91.98	71.39	91.73	92.46	93.84	93.44	92.6	94.25
Soil	89.45	97.65	98.7	97.48	99.59	100	99.84	92.01	100
Water	84.44	63.49	76.19	87.94	81.59	81.27	87.94	86.13	90.79
Residential	64.79	61.53	72.5	32.19	70.37	75.68	84.89	63.93	81.96
Commercial	76.26	66.37	78.69	62.92	40.56	59.48	43.27	69.57	50.16
Road	75.04	61.43	52.17	41.79	76.94	86.23	90.66	53.44	92.75
Highway	57.75	62.2	87.67	86.2	49.38	92.85	85.37	75.66	92.11
Railway	44.33	67.18	58.86	66.09	78.18	66.04	68.24	64.34	71.35
Parking_lot	48.81	43.58	44.89	91.01	97.71	89.04	90.68	72.33	95.09
Parking_lot2	23.97	36.6	83	95.64	90.2	84.53	96.3	97.14	92.37
Tennis_court	88.76	90.67	100	96.17	100	100	100	100	100
Running_track	90.15	92.92	94.01	99.85	100	100	100	100	100
OA	70.11	75.29	78.06	79.715	82.06	86.29	87.26	79.16	**89.16**
KAPPA	0.6767	0.7326	0.7627	0.781	0.806	0.8519	0.8624	0.7748	**0.8828**

Table 13
Complexity analysis of the different approaches on the Houston CASI dataset.

Indicator	CNN-4	CapsNet-4	MSRN	A^2S^2ResNet	FPGA	UML	S3Net	CVNN
Running time (s)	43.55	49.66	84.68	197.79	2.15	2.85	301.37	1.93
GFLOPs	365.66	5099.36	1003.92	100 205.4	420.83	853.17	61 573.95	873.55

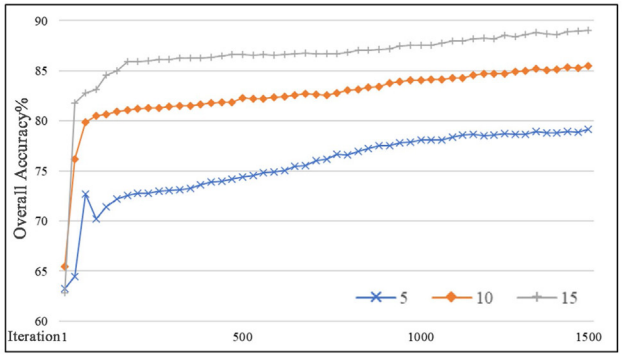
Table 14
The accuracy of CVNN with different numbers of training samples on the different datasets.

Dataset	5 samples per class		10 samples per class		15 samples per class	
	OA	Kappa	OA	Kappa	OA	Kappa
Indian Pines	79.07	0.7628	85.58	0.8366	88.94	0.8536
Kansas AHSI	76.13	0.7334	88.14	0.8566	90.21	0.8809
Pavia University	80.4	0.7865	92.48	0.9023	97.08	0.9525
Houston CASI	80.57	0.7792	89.16	0.8828	91.6	0.8843

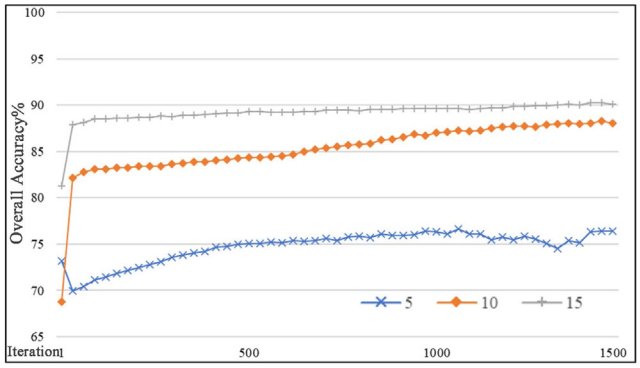
Fig. 14. The loss curve of CVNN on the Houston CASI dataset.

the training process is completed. On the Kansas AHSI dataset, the OA for 5 labeled samples presents a downward trend after 1000 iterations. The OA for 10 and 15 labeled samples improves significantly after the first few iterations. The OA is improved by over 10% from 5 labeled samples to 15 labeled samples per class. The OA presents an unstable upward tendency on the Pavia University dataset with 5 labeled samples, which is the same as on the Indian Pines and Kansas AHSI datasets. The improvement effect of 15 labeled samples becomes more significant after 1000 iterations, compared with 10 labeled samples. On the Houston CASI dataset, the increasing trend in accuracy is stable with 5, 10, and 15 labeled samples. Overall, the experiments on the four datasets reveal the stable classification ability of the proposed CVNN method.

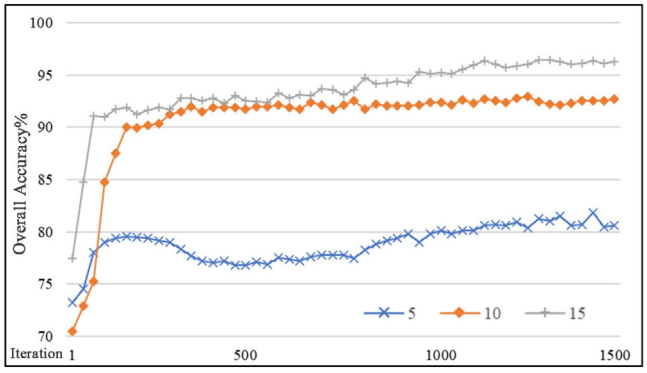
The two hyperparameters of CVNN are c and d, where c is the number of vector neurons, which is consistent with the number of categories, and d is the dimension of each vector. The dimension of the capsule neuron of the category capsules is one of the crucial factors for the final output, so we chose 2, 4, 6, 8, and 10 as the value range to perform additional experiments. The five different dimensions were performed on the four datasets, as listed in Tables 15–18, where the best result is highlighted with bold and the second-best is underlined.



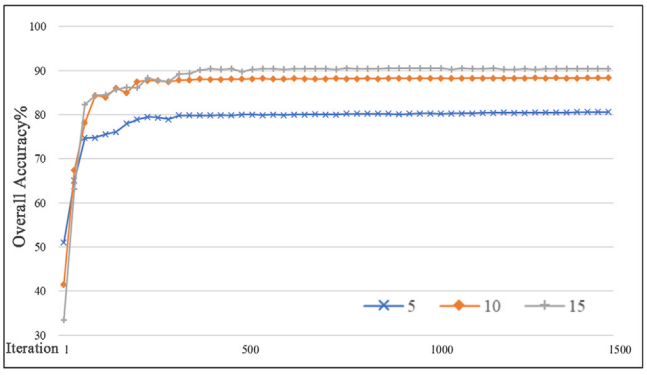
(a) Indian Pines dataset



(b) Kansas AHSI dataset



(c) Pavia University dataset



(d) Houston CASI dataset

Fig. 15. Overall classification accuracies obtained for the four hyperspectral image datasets using CVNN with 5, 10, and 15 labeled samples per class.

Table 15
The accuracy of CVNN with different vector dimensions for the Indian Pines dataset.

Category	$d = 2$	$d = 4$	$d = 6$	$d = 8$	$d = 10$
Alfalfa	100	100	100	100	100
Corn-Notill	65.23	69.82	60.01	61.78	64.53
Corn-Mintill	90.85	91.71	94.02	92.68	90.24
Corn	99.11	97.8	100	97.36	98.24
Grass-Pasture	81.4	78.86	83.72	84.14	81.61
Grass-Trees	96.53	90	84.58	92.5	93.47
Grass-Pasture-Mowed	100	100	100	100	100
Hay-Windrowed	99.36	100	99.57	100	98.29
Oats	100	100	100	100	100
Soybean-Notill	86.49	86.9	88.36	84.62	87.01
Soybean-Mintill	80.65	82.54	87.2	85.97	85.19
Soybean-Clean	80.27	86.79	86.96	77.02	82.16
Wheat	86.87	96.92	97.44	98.97	95.38
Woods	93.71	88.84	92.43	90.04	85.34
Bs-Grss-Trs-Drs	99.2	99.2	98.94	99.47	99.73
Stone-Steel-Towers	95.18	96.39	93.78	96.39	96.39
OA	84.97	85.58	**86.67**	85.19	84.96
KAPPA	0.8292	0.8366	**0.8484**	0.8316	0.8294

Table 16
The accuracy of CVNN with different vector dimensions for the Pavia University dataset.

Category	$d = 2$	$d = 4$	$d = 6$	$d = 8$	$d = 10$
Asphalt	68.95	95.67	93.84	91.19	89.28
Meadows	60.94	88.31	86.99	88.13	89.08
Gravel	63.14	87.94	92.3	88.13	89.95
Trees	87.3	96.99	96.4	95.28	95.81
Painted metal sheets	99.63	99.93	99.7	100	100
Bare soil	62.56	98.85	97.33	96.17	98.55
Bitumen	85.98	95.08	89.55	93.26	96.89
Self-blocking bricks	60.84	93.36	94.25	95.64	94.39
Shadows	99.68	99.89	99.79	99.89	99.47
OA	90.9	**92.48**	91.58	91.5	92.02
KAPPA	0.8825	**90.23**	0.8908	0.8895	0.8963

Table 17
The accuracy of CVNN with different vector dimensions for the Kansas AHSI dataset.

Category	$d = 2$	$d = 4$	$d = 6$	$d = 8$	$d = 10$
Crop	90.14	98.22	93.82	95.94	94.03
Meadow	89.5	97.82	98.2	97.23	94.52
Business area	93.11	87.62	71.21	89.18	78.99
Bare soil	89.52	86.93	91.6	85.2	84.9
Residential area	86.35	90	73.4	87.89	87.84
Water	93.07	93.78	93.66	94.92	95.34
Road	35.33	63.49	51.26	57.47	60.07
OA	84.77	**88.14**	82.56	86.44	84.91
Kappa	0.8153	**85.66**	0.7888	0.8363	0.8179

Table 18
The accuracy of CVNN with different vector dimensions for the Houston CASI dataset.

Category	$d = 2$	$d = 4$	$d = 6$	$d = 8$	$d = 10$
Grass_healthy	99.52	97.9	96.86	95.73	98.71
Grass_stressed	95.9	98.95	98.07	97.27	95.1
Grass_synthetic	100	100	100	100	100
Tree	92.46	94.25	92.22	93.27	93.6
Soil	100	100	99.92	100	100
Water	92.06	90.79	90.79	81.59	84.76
Residential	76.79	81.96	86.96	81.72	79.73
Commercial	39.95	50.16	42.14	40.6	40.44
Road	90.42	92.75	84.62	89.69	91.38
Highway	79.38	92.11	79.62	89.81	82.74
Railway	65.59	71.35	66.12	63.27	72.33
Parking_lot	94.19	95.09	87.41	90.11	89.82
Parking_lot2	92.81	92.37	92.81	95.64	94.12
Tennis_court	97.13	100	99.76	99.76	100
Running_track	100	100	100	100	100
OA	85.77	**89.16**	85.83	86.32	86.38
Kappa	84.63	**88.28**	84.7	85.22	85.29

Table 19
The accuracy of the CVNN framework with different data noise.

Dataset	AHSI		AVIRIS		ROSIS		HST	
Indicator	OA	Kappa	OA	Kappa	OA	Kappa	OA	Kappa
10%stripe_noise	86.3071	0.8347	62.76	0.5874	91.49	0.8895	87.86	0.8688
30%stripe_noise	86.2569	0.8341	60.66	0.5497	90.86	0.8825	86.97	0.8592
Gaussian	79.91	0.7594	31.74	0.1951	70.86	0.6219	74.8	0.7281

The different vector dimensions have a big impact on the final performance, of up to 6% in OA. Most of the experiments show that CVNN can represent all the features adequately when the capsule dimension is 4. The capsule vector with a dimension of 4 obtains the second-best OA and kappa on the Indian Pines dataset. Comparing the results of all the experiments, the best vector dimension setting was determined as 4 for all the datasets. With these settings, the OAs for the Indian Pines, Pavia University, Kansas AHSI, and Houston CASI datasets are 85.58%, 92.48%, 88.14%, and 89.16%, respectively.

4.7. The robustness of the CVNN method to noise

The noise of remote sensing imagery impacts the performance of remote sensing image processing [49]. In this section, we describe how the robustness to noise of the proposed CVNN method was analyzed. Firstly, Gaussian noise and stripe noise, which are two kinds of common noise in hyperspectral imagery, were utilized as simulated noise for the original datasets. Stripe noise can be one pixel wide or multiple pixels wide. The gray-level value of the bright stripe is higher than that of the surrounding normal pixels, while the gray-level value of the dark stripe is lower than that of the surrounding normal pixels. Gaussian noise is a kind of spot-like noise. In this work, the stripe noise ratio was set to 10% and 30% (see Table 19).

Table 18 lists the classification accuracies for the different datasets with different data noise. It can be seen that the Gaussian noise and stripe noise cause a loss of the detection accuracy. The CVNN method with 10% stripe noise obtains a higher accuracy than with 30% stripe noise. The Gaussian noise impacts the OA even more than the stripe noise in hyperspectral image classification using CVNN.

5. Conclusion

In this paper, an FCN-based method with modified capsule transformation has been proposed to handle the inadequate feature representation of the traditional models and the poor classification performance under the case of insufficient labeled samples. The motivation of this work was to improve the classification accuracy through enhancing the classification step, instead of the feature extraction phase. Differing from the existing works which focus on elaborate and complicated network designs and complex training strategies, this work was aimed at integrating vector neurons with a vanilla FCN network to obtain a superior classification performance. The proposed network, which is named the capsule-vectored neural network (CVNN), consists of an encoder–decoder part and a vector-neuron transformation part. In the encoder–decoder part, a fully convolutional framework with a two times down-sampling step is utilized. To verify the boosting effect of the vectorial FCN, the widely used multiscale learning strategies and attention mechanisms are omitted. After the feature extraction by the vanilla FCN, the output is transformed into a vectorial representation with a learnable transformation matrix. All the vector neurons have a unified dimension, and the norm of the vector neurons is utilized as the logit that is input into the softmax function to obtain the posterior probability matrix. After supervised learning with the limited labeled samples, the optimized transformation matrix is able to transform the features learned by the FCN to the more advanced vectorial feature space. Four real hyperspectral datasets, including both airborne and spaceborne images, were utilized to evaluate the performance of CVNN. The experimental results confirmed that CVNN can tackle deep learning modeling well in the case of limited labeled samples, while achieving a good trade-off between keeping the minor characteristics and restraining the salt-and-pepper noise. The classification accuracy of CVNN was also higher than that of the other state-of-the-art classification methods.

In the aspect of future studies, two key points could be further pursued. Firstly, the more advanced attention mechanisms in the computer vision field could be investigated to learn the fine-grained image features. Moreover, the proposed method is based on supervised learning. How to combine the deep learning methods with either semi-supervised or unsupervised learning to solve the hyperspectral image classification task is another possible future research direction.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was jointly supported by the Natural Science Foundation of China (grant nos. 42171335 and 42001350), the Shanghai Municipal Science and Technology Major Project (grant no. 22511102800), the Postdoctoral Science Foundation of China (grant no. 2021M691016), and the National Civil Aerospace Project of China (grant no. D040102).

References

[1] X. Wang, K. Tan, P. Du, C. Pan, J. Ding, A unified multiscale learning framework for hyperspectral image classification, IEEE Trans. Geosci. Remote Sens. 60 (2022) 1–19.

[2] H. Sun, X. Zheng, X. Lu, A supervised segmentation network for hyperspectral image classification, IEEE Trans. Image Process. 30 (2021) 2810–2825.

[3] W. Sun, K. Liu, G. Ren, W. Liu, G. Yang, X. Meng, J. Peng, A simple and effective spectral-spatial method for mapping large-scale coastal wetlands using China ZY1-02D satellite hyperspectral images, Int. J. Appl. Earth Obs. Geoinf. 104 (2021) 102572.

[4] T. Hou, W. Sun, C. Chen, G. Yang, X. Meng, J. Peng, Marine floating raft aquaculture extraction of hyperspectral remote sensing images based decision tree algorithm, Int. J. Appl. Earth Obs. Geoinf. 111 (2022) 102846.

[5] C. Niu, K. Tan, X. Jia, X. Wang, Deep learning based regression for optically inactive inland water quality parameter estimation using airborne hyperspectral imagery, Environ. Pollut. 286 (2021) 117534.

[6] C. Liu, C. Xing, Q. Hu, S. Wang, S. Zhao, M. Gao, Stereoscopic hyperspectral remote sensing of the atmospheric environment: Innovation and prospects, Earth-Sci. Rev. 226 (2022) 103958.
[7] G. Yang, K. Huang, W. Sun, X. Meng, D. Mao, Y. Ge, Enhanced mangrove vegetation index based on hyperspectral images for mapping mangrove, ISPRS J. Photogramm. Remote Sens. 189 (2022) 236–254.
[8] K. Tan, J. Hu, J. Li, P. Du, A novel semi-supervised hyperspectral image classification approach based on spatial neighborhood information and classifier combination, ISPRS J. Photogramm. Remote Sens. 105 (2015) 19–29.
[9] K. Tan, J. Zhang, Q. Du, X. Wang, GPU parallel implementation of support vector machines for hyperspectral image classification, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 8 (10) (2015) 4647–4656.
[10] K. Tan, E. Li, Q. Du, P. Du, Hyperspectral image classification using band selection and morphological profiles, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 7 (1) (2013) 40–48.
[11] K. He, W. Sun, G. Yang, X. Meng, K. Ren, J. Peng, Q. Du, A dual global–local attention network for hyperspectral band selection, IEEE Trans. Geosci. Remote Sens. 60 (2022) 1–13.
[12] R.M. Haralick, K. Shanmugam, I.H. Dinstein, Textural features for image classification, IEEE Trans. Syst. Man Cybern. (6) (1973) 610–621.
[13] X. Huang, X. Liu, L. Zhang, A multichannel gray level co-occurrence matrix for multi/hyperspectral image texture representation, Remote Sens. 6 (9) (2014) 8424–8445.
[14] D. Hong, L. Gao, J. Yao, B. Zhang, A. Plaza, J. Chanussot, Graph convolutional networks for hyperspectral image classification, IEEE Trans. Geosci. Remote Sens. 59 (7) (2020) 5966–5978.
[15] B. Lyu, M. Hamdi, Y. Yang, Y. Cao, Z. Yan, K. Li, S. Wen, T. Huang, Efficient spectral graph convolutional network deployment on memristive crossbars, IEEE Trans. Emerg. Top. Comput. Intell. (2022).
[16] M.H. Rafiei, W.H. Khushefati, R. Demirboga, H. Adeli, Supervised deep restricted Boltzmann machine for estimation of concrete, ACI Mater. J. 114 (2) (2017) 237.
[17] H. Luo, Y.Y. Tang, X. Yang, L. Yang, H. Li, Autoencoder with extended morphological profile for hyperspectral image classification, in: 2017 3rd IEEE International Conference on Cybernetics, CYBCONF, IEEE, 2017.
[18] X. Wang, K. Tan, Q. Du, Y. Chen, P. Du, Caps-TripleGAN: GAN-assisted CapsNet for hyperspectral image classification, IEEE Trans. Geosci. Remote Sens. 57 (9) (2019) 7232–7245.
[19] J. Yang, Y.-Q. Zhao, J.C.-W. Chan, Learning and transferring deep joint spectral–spatial features for hyperspectral classification, IEEE Trans. Geosci. Remote Sens. 55 (8) (2017) 4729–4742.
[20] H. Lee, H. Kwon, Going deeper with contextual CNN for hyperspectral image classification, IEEE Trans. Image Process. 26 (10) (2017) 4843–4855.
[21] L. Zhang, M. Edraki, G.-J. Qi, Cappronet: Deep feature learning via orthogonal projections onto capsule subspaces, Adv. Neural Inf. Process. Syst. (2018) 31.
[22] M. Zhang, W. Li, Q. Du, Diverse region-based CNN for hyperspectral image classification, IEEE Trans. Image Process. 27 (6) (2018) 2623–2634.
[23] D. Hong, Z. Han, J. Yao, L. Gao, B. Zhang, A. Plaza, J. Chanussot, SpectralFormer: Rethinking hyperspectral image classification with transformers, IEEE Trans. Geosci. Remote Sens. 60 (2021) 1–15.
[24] S. Hao, W. Wang, Y. Ye, T. Nie, L. Bruzzone, Two-stream deep architecture for hyperspectral image classification, IEEE Trans. Geosci. Remote Sens. 56 (4) (2017) 2349–2361.
[25] Z. Ge, G. Cao, X. Li, P. Fu, Hyperspectral image classification method based on 2D–3D CNN and multibranch feature fusion, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 13 (2020) 5776–5788.
[26] J. Li, X. Zhao, Y. Li, Q. Du, B. Xi, J. Hu, Classification of hyperspectral imagery using a new fully convolutional neural network, IEEE Geosci. Remote Sens. Lett. 15 (2) (2018) 292–296.
[27] Z. Zheng, Y. Zhong, A. Ma, L. Zhang, FPGA: Fast patch-free global learning framework for fully end-to-end hyperspectral image classification, IEEE Trans. Geosci. Remote Sens. 58 (8) (2020) 5612–5626.
[28] Y. Jiang, Y. Li, S. Zou, H. Zhang, Y. Bai, Hyperspectral image classification with spatial consistence using fully convolutional spatial propagation network, IEEE Trans. Geosci. Remote Sens. 59 (12) (2021) 10425–10437.
[29] K. Gao, B. Liu, X. Yu, P. Zhang, X. Tan, Y. Sun, Small sample classification of hyperspectral image using model-agnostic meta-learning algorithm and convolutional neural network, Int. J. Remote Sens. 42 (8) (2021) 3090–3122.
[30] J. Shi, X. Zhang, X. Liu, Y. Lei, G. Jeon, Multicriteria semi-supervised hyperspectral band selection based on evolutionary multitask optimization, Knowl.-Based Syst. 240 (2022) 107934.
[31] Z. Xue, Y. Zhou, P. Du, S3Net: Spectral–spatial siamese network for few-shot hyperspectral image classification, IEEE Trans. Geosci. Remote Sens. 60 (2022) 1–19.
[32] Y. Chu, H. Lin, L. Yang, D. Zhang, Y. Diao, X. Fan, C. Shen, Hyperspectral image classification based on discriminative locality preserving broad learning system, Knowl.-Based Syst. 206 (2020) 106319.
[33] S. Sabour, Frosst N., G.E. Hinton, Dynamic routing between capsules, Adv. Neural Inf. Process. Syst. (2017) 30.
[34] R. Zeng, Y. Song, A fast routing capsule network with improved dense blocks, IEEE Trans. Ind. Inform. 18 (7) (2021) 4383–4392.
[35] R. Renzulli, M. Grangetto, Towards efficient capsule networks, in: 2022 IEEE International Conference on Image Processing, ICIP, IEEE, 2022.
[36] H.-C. Li, W.-Y. Wang, L. Pan, W. Li, Q. Du, R. Tao, Robust capsule network based on maximum correntropy criterion for hyperspectral image classification, Ieee J. Sel. Top. Appl. Earth Obs. Remote Sens. 13 (2020) 738–751.
[37] W.-Y. Wang, H.-C. Li, Y.-J. Deng, L.-Y. Shao, X.-Q. Lu, Q. Du, Generative adversarial capsule network with ConvLSTM for hyperspectral image classification, IEEE Geosci. Remote Sens. Lett. 18 (3) (2020) 523–527.
[38] M.E. Paoletti, J.M. Haut, R. Fernandez-Beltran, J. Plaza, A. Plaza, J. Li, F. Pla, Capsule networks for hyperspectral image classification, IEEE Trans. Geosci. Remote Sens. 57 (4) (2018) 2145–2160.
[39] B. Lyu, S. Wen, K. Shi, T. Huang, Multiobjective reinforcement learning-based neural architecture search for efficient portrait parsing, IEEE Trans. Cybern. (2021).
[40] W. Li, S. Wen, K. Shi, Y. Yang, T. Huang, Neural architecture search with a lightweight transformer for text-to-image synthesis, IEEE Trans. Netw. Sci. Eng. 9 (3) (2022) 1567–1576.
[41] B. Lyu, Y. Yang, S. Wen, T. Huang, K. Li, Neural architecture search for portrait parsing, IEEE Trans. Neural Netw. Learn. Syst. 34 (2023) 10.
[42] W. Sun, K. Ren, X. Meng, G. Yang, C. Xiao, J. Peng, J. Huang, MLR-DBPFN: A multi-scale low rank deep back projection fusion network for anti-noise hyperspectral and multispectral image fusion, IEEE Trans. Geosci. Remote Sens. 60 (2022) 1–14.
[43] D.H. Hubel, T.N. Wiesel, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, J. Physiol. 160 (1) (1962) 106.
[44] J. Gu, V. Tresp, H. Hu, Capsule network is not more robust than convolutional network, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021.
[45] A. Ben-Israel, T.N. Greville, Generalized Inverses: Theory and Applications, Springer Science & Business Media, 2003.
[46] H. Gao, Y. Yang, C. Li, L. Gao, B. Zhang, Multiscale residual network with mixed depthwise convolution for hyperspectral image classification, IEEE Trans. Geosci. Remote Sens. 59 (4) (2020) 3396–3408.
[47] S.K. Roy, S. Manna, T. Song, L. Bruzzone, Attention-based adaptive spectral–spatial kernel ResNet for hyperspectral image classification, IEEE Trans. Geosci. Remote Sens. 59 (9) (2020) 7831–7843.
[48] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[49] D. Hong, N. Yokoya, J. Chanussot, X.X. Zhu, An augmented linear mixing model to address spectral variability for hyperspectral unmixing, IEEE Trans. Image Process. 28 (4) (2018) 1923–1938.