Neurocomputing 486 (2022) 47–60

Contents lists available at ScienceDirect

Neurocomputing

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

ASPCNet: Deep adaptive spatial pattern capsule network for hyperspectral image classification ☆

Jinping Wang [a], Xiaojun Tan [a,b], Jianhuang Lai [c,d], Jun Li [a,*]

[a] School of Intelligent Systems Engineering, Sun Yat-sen University, Guangzhou 510006, China
[b] School of Intelligent Systems Engineering, Sun Yat-sen University & Southern Marine Science and Engineering Guangdong Laboratory (Zhuhai), Zhuhai 519082, China
[c] School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China
[d] Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, Sun Yat-sen University, Guangzhou 510006, China

ARTICLE INFO

Article history:
Received 10 May 2021
Revised 14 January 2022
Accepted 19 February 2022
Available online 24 February 2022
Communicated by Zidong Wang

Keywords:
Capsule networks
Adaptive spatial pattern neural network
Hyperspectral image classification

ABSTRACT

Previous studies have shown the great potential of capsule networks for the spatial contextual feature extraction from hyperspectral images (HSIs). However, the sampling locations of the convolutional kernels of capsules are fixed and cannot be adaptively changed according to the inconsistent semantic information of HSIs. Based on this observation, this paper proposes an adaptive spatial pattern capsule network (ASPCNet) architecture by developing an adaptive spatial pattern (ASP) unit, that can rotate the sampling location of convolutional kernels on the basis of an enlarged receptive field. Note that this unit can learn more discriminative representations of HSIs with fewer parameters. Specifically, two cascaded ASP-based convolution operations (ASPConvs) are applied to input images to learn relatively high-level semantic features, transmitting hierarchical structures among capsules more accurately than the use of the most fundamental features. Furthermore, the semantic features are fed into ASP-based conv-capsule operations (ASPCaps) to explore the shapes of objects among the capsules in an adaptive manner, further exploring the potential of capsule networks. Finally, the class labels of image patches centered on test samples can be determined according to the fully connected capsule layer. Experiments on three public datasets demonstrate that ASPCNet can yield competitive performance with higher accuracies than state-of-the-art methods. For the convenience of follow-up research and engineering applications, we packaged the algorithm into an arbitrary plug-in module and released it at https://github.com/Cimy-wang.

© 2022 Published by Elsevier B.V.

1. Introduction

Hyperspectral remote sensing technology has attracted much attention in recent years since it can include hundreds of contiguous spectral bands and capture more accurate and discriminative features for different objects, especially compared with panchromatic and multi-spectral images. Hyperspectral image classification (HSIC), which refers to automatically assigning a specific label for each pixel in a scene, has become an active topic in many research fields [1], such as defense and security [2,3], intelligent transportation, intelligent healthcare [4,5], and forest and unmanned aerial vehicle monitoring [6,7].

During the last few decades, a large number of pixelwise-based classifiers, mainly based on spectral signatures of hyperspectral image (HSI), have been proposed for classification tasks, such as support vector machines (SVMs) [8], logistic regression [9], *k*-nearest-neighbors [10,11], and random forests [12,13]. With the improvement in sensor spatial resolution, many spectral-spatial-based classification methods play roles [14–18], taking the spatial information of images into consideration and improving the classification accuracies. As one of the crucial steps in spectral-spatial classification, feature extraction-based methods have attracted much attention. Some state-of-the-art feature extraction methods, which are mainly based on general machine learning techniques, have achieved good classification performance, such as the image fusion and recursive filtering [19] method, the extended morphological profiles (EMP) [20] method, the edge-preserving filter (EPF) [21] method, and the image segmentation-based method [22–24]. However, the aforementioned feature extraction approaches mainly classify an image in a shallow manner, e.g.,

☆ This work was supported by the Key-Area Research and Development Program of Guangdong Province under Grant (2020B0909030005 and 2020B090921003) and Southern Marine Science and Engineering Guangdong Laboratory (Zhuhai) under Grant SML2020SP011.

* Corresponding author.
E-mail address: stslijun@mail.sysu.edu.cn (J. Li).

https://doi.org/10.1016/j.neucom.2022.02.058
0925-2312/© 2022 Published by Elsevier B.V.

the feature extractors and classifiers only focus on a single layer. Comparatively, deep learning-based techniques can allow computers to learn different image features through a series of hierarchical layers, and the learning process is totally automatic and efficient. Currently, deep learning models have shown their excellent performances in various applications and tasks, such as image-voice retrieval, cross-modal biometric matching, object detection, and image classification [25–29]. Typical deep learning networks for HSIC can be summarized into five categories [30]: stacked autoencoders [31,32], deep belief networks [33], recurrent neural networks [34], generative adversarial networks [35], and convolutional neural networks (CNNs) [36].

Unlike other deep learning methods, CNN models possess local receptive fields and shared weight architecture, and have shown strong abilities in the feature extraction process. Generally, CNNs can be grouped into two classes: (1) The first class extracts spectral-spatial characteristics by simultaneously using 3D filtering. As variants of CNNs, the dilated neural networks were introduced for HSIC [37–39], and the core idea of which is to avoid resolution reduction in the pooling layer while enlarging the receptive field through a dilated convolution strategy. Moreover, a multi-scale dilated residual CNN has been proposed to improve further classification performance [40]. (2) The second class extracts the spectral and spatial characteristics and subsequently fuses them. In the literature [41], a spectral-spatial fully convolutional network was proposed that introduces a dense conditional random field into two parallel flows to balance the obtained spectral and spatial features. In contrast, Cheng et al. developed an effective method for HSIC [42], where a metric learning regularization term is designed to learn a powerful spectral-spatial feature representation by fusing spectral feature and deep spatial feature. Li et al. used a fusion scheme with layer-specific regularization and smooth normalization to adaptively learn the fusion weights of local and global information [43]. Moreover, the fusion of multi-level and multiscale-based spatial-spectral CNNs also has potential for HSIC [44,45].

Although general CNNs are capable of image feature extraction, some shortcomings still exist that strongly threaten the performance of deep learning methods, such as problems related to the loss of local information caused by the pooling layer and an inability to understand spatial positional relationships among features. To overcome these issues, Sabour et al. proposed a capsule network (CapsuleNet) [46] that can achieve results superior to those of CNNs. Due to their advantages, CapsuleNets have been widely used in many image processing tasks, such as face recognition, optical character recognition, and scene classification [47,48]. Later, CapsuleNet was first introduced for HSIC in [49]. Motivated by the core ideas of CNNs, a modification of the traditional CapsuleNet named conv-capsule has been proposed for classification [50], which uses a conv-capsule unit to achieve a local connection and shared transform matrices. Experimental results indicate that the conv-capsule unit provides results competitive with those of traditional CapsuleNet.

However, there are still some shortcomings in the existing CapsuleNets. First, the semantic features fed into primarycaps are extracted by a simple convolutional layer, which may not be efficient in representing and transferring hierarchical structures among capsules, compared with relatively high-level semantic features. Second, although the 3D conv-capsulenet uses local connections and shared weights to fully use the spectral-spatial information, the window size is fixed and cannot be adjusted according to the different shapes of input images. Third, the traditional neural networks use pooling layers to enlarge the receptive field while reducing the model complexity. However, some detailed features might disappear after several pooling operations.

To overcome these defects, a novel adaptive spatial pattern capsule network (ASPCNet) has been proposed that consists of the following major steps: First, relatively high-level features obtained by adaptive spatial pattern convolution operations (ASPConv) are fed into primarycaps. Next, adaptive spatial pattern-based conv-capsule operations (ASPCaps) are proposed during the dynamic routing process, adaptively adjusting the convolutional kernel sampling locations. Finally, the class label of pixels can be determined according to the fully connected capsule layer. The contributions of this paper are described as follows:

(1) A novel adaptive spatial pattern (ASP) unit is constructed by adaptively rotating the sampling location on the basis of dilated convolutions. Instead of using the downsampling strategy, the cascaded ASPConv layers can simultaneously expand the receptive field and avoid checkerboard effects.
(2) A series of ASPConvs are used for feature extraction before the primarycaps layer. The obtained relatively high-level semantic features can offer better services for hierarchical relationship representation among capsules regarding the transmission efficiency and representation ability.
(3) ASPCaps is developed into a dynamic routing process that can adaptively adjust the shape of local connections according to HSI's complex spatial contexts to ensure that the model has a strong generalization ability.
(4) Experiments conducted on several real hyperspectral datasets demonstrate that the proposed ASPCNet can always obtain better results and more accurate classification maps than state-of-the-art methods.

The remainder of this paper is organized as follows. Section 2 describes the related works. Section 3 presents the details of the proposed ASPCNet approach. In Section 4, the experimental results and discussions are provided. Finally, conclusions are given in Section 5.

2. Related Works

2.1. Deformable CNN

As typical supervised learning methods, CNNs use image patches centered on labeled pixels as the training patches for a model training and then the well-trained model can be used for HSIC; see Fig. 1.

Specifically, CNNs can learn different types of natural features through a stack of layers that includes convolutional layers, pooling layers, and fully connected layers [51]. As the layers go deeper, the model features become more precise and reliable [52]. However, the convolution kernels of classical CNNs are fixed, which is not suitable for representing complex image features. Therefore, the deformable CNNs (DCNNs) [53] were conceived in the computer vision field to overcome the problem. In more recent years, various DCNNs and its variants methods [54,55] have been proposed to significantly boost image classification performance. Later, the ideal of deformable neural networks was introduced into HSIC [56], which can effectively extract HSI features, especially for its complex spatial structures. As we can see from Fig. 2, there is a contrast between the sampling locations of regular convolution and sampling locations of deformable convolution, in which the deformable convolution is present in the shapes of a rhombus and a circle, showing better coverage.

Fig. 1. The architecture of the supervised learning-based classification of hyperspectral images.

Fig. 2. Illustration of the differences between different sampling locations on the University of Pavia dataset. (a) Sampling locations based on regular convolution and (b) Sampling locations based on deformable convolution.

2.2. CapsuleNet

In CapsuleNets, specific types of functional neurons are grouped together to form capsules, and individual neurons represent one of various attributes of a specific entity in an image [46]. A CapsuleNet can discriminate the consistency of the contextual information of the capsules and therefore has a better ability to model the spatial positions of visual features of images than a CNN.

Original CapsuleNet includes only one convolution layer and one fully connected capsule layer. To deepen the method and allow it to solve problems due to the stacking of multicapsule layers, many CapsuleNet-based expansions have been proposed that can achieve better classification performance [50,57–60]. Motivated by the core ideas of CNNs, Zhang et al. proposed a novel architecture named cnn-capsulenet to discriminate scene classes effectively without any data-augmentation operation [59]. Moreover, Xu et al. designed multiple kernels with parallel convolution to extract image features in a multi-scale way, reducing the redundancy of parameters and achieving high accuracies for HSIC [60]. These works have demonstrated the excellent performance of CapsuleNet.

Different from previous works, this paper is implemented based on the merits of both DCNN and CapsuleNet, and a more flexible operator named ASP unit is developed into CapsuleNet to better represent and transfer the hierarchical structures among capsules. The details of the proposed ASPCNet approach are shown in Section 3.

3. The Proposed Approach

Assume an input HSI $\mathbf{X}$ with the size of $H \times W \times D$, and $\mathbf{X} \in \mathbb{R}^{H \times W \times D}$. Each training pixel $\mathbf{X}^k \in \mathbb{R}^{1 \times 1 \times D}, k = 1, 2, \ldots, K$, where D and K represent the dimension of HSI and the number of training samples, respectively. Suppose that T represents the total class in HSI. For each pixel $\mathbf{X}^k$, the truth label of $\mathbf{X}^k$ can be represented as $\mathbf{t}^k$, which is actually a vector of length t with value "1" at the position of the correct label and "0" elsewhere. Different from the natural image classification, which inputs whole images, the HSI classification based on CNN uses image patches centered on labeled pixels as the input samples. Here, we conduct a standard preprocessing step, i.e., the principal component analysis (PCA) algorithm, for HSI dimension reduction. Therefore, HSI's band channels shrink from D to d. Let $\mathbf{X}^k \in \mathbb{R}^{1 \times 1 \times d}$ be the center pixel of dimension reduction images, the image patch can be represented as $\mathbf{Y}^k \in \mathbb{R}^{m \times m \times d}$ of the window size $m \times m$, and $\{\mathbf{Y}^k, \mathbf{t}^k\}_{k=1}^{K}$ represents the input samples. Since the ASP units are the core components of the proposed ASPCNet, in Section 3.1, we first give a detailed description of the ASP unit and ASP-based convolution operation (ASPConvs) in the task of HSIC. Then, we develop the ASP unit into the process of the original capsule neural network

Fig. 3. The framework of the proposed ASPCNet for HSIC.

(ASPCaps) in Section 3.2. Finally, the overall architecture of the proposed ASPCNet is given in Section 3.3 and Fig. 3.

3.1. ASPConvs

Given an image patch (centered on labeled pixel) $\mathbf{Y}^k \in \mathbb{R}^{m\times m\times d}$ with a window size of $m \times m$, the sampling locations of the input image can be represented as $s_1 = \{(0,0),(0,1),\ldots,(m-1, m-1)\}$. The traditional convolution operation consists of two steps: 1) sampling using a series of regular grids $\Re$ of size $p \times q$ slide over the input image patch and 2) the summation of sampled values weighted by $\mathbf{W}$ for different convolution kernels. A larger size of regular grids $\Re$ can be replaced by multiple smaller sizes of $\Re$ with the same receptive field. As the convolution network model goes deeper, the pooling layers are supposed to shrink the model's parameters. Unfortunately, the principle behind the pooling operation is to preserve distinctive and representative features of each pooling area. This means that some details tend to disappear after several pooling operations, which is extremely unfavorable for classification accuracy. In addition, the sampling locations of the general convolution layer are fixed; this configuration cannot effectively extract the spatial information according to the different structures of input image patches. Therefore, to overcome these issues, the ASP unit is proposed in this paper, which simultaneously expands the receptive field and rotates the sampling location, i.e., integrating dilated and deformation operations into one unit (see Fig. 4). The details are shown as follows:

First, we can introduce the dilated convolution to implement the larger receptive field with fewer parameters. For instance, two 3×3 filters along with a 2-dilation rate can be equally replaced by one 9×9 filter with a 1-dilation rate, which can prevent the loss of the spatial relative relationship of pixels caused by the downsampling operation. Grid $\Re\prime$ defines the locations of the modified dilated convolution kernels, which are described below. Suppose the dilation rate h_{dr}=3, $\Re' = \{(x,y)|(-3,-3),(-3,-1),\ldots(1,3),(3,3)\}$. Here, h_{dr} can be changed freely in this model according to the different shapes of images; its effect is analyzed in Section 4.4.3. For any dilation rate h_{dr}, its size can be calculated as

$$p_{dr} = p + (p-1)(h_{dr}-1), \tag{1}$$

$$q_{dr} = q + (q-1)(h_{dr}-1), \tag{2}$$

where p_{dr} and q_{dr} refer to the height and width of a kernel of dilated convolution, respectively. There are only no more than $p\times q$ nonzero values in a $p_{dr}\times q_{dr}$ area. However, without a brilliantly designed dilated convolution structure [61], multiple cascaded dilated convolutional layers result in a griding effect while hurting the continuity of the kernel due to the checkerboard processing method. Based on this observation, a deformable convolution operation is combined with dilated convolution to adaptively adjust the sampling location of the grid kernel. Therefore, the new unit called the ASP unit is promoted, which can directly choose the grid location in an adaptive way to search for the most relevant pixels in a local area and achieve a larger receptive field, fewer parameters, and more spatial information.

Fig. 4. Illustration of the process of the ASPConv operation. The pixel marked by the orange box is the central pixel. After ASPConv is applied to the input images, the sampling locations of regular convolutional filters can be transferred to the pixels in the purple boxes.

Specifically, given one location (x,y) of the grid $\Re'$ and its value $\Re'(x,y)$ which corresponds to two values $\Delta x, \Delta y$ in the offset field. In this way, the adaptive spatial unit is generated to fuse the information of similar neighboring pixels, which can be obtained by

$$\Re'_{\text{new}}(x,y) = \mathbf{Y}^k(\Re'(x,y) + (\Delta x, \Delta y)) = \Re'(x', y'), \tag{3}$$

where x' and y' refer to fractional locations and can be represented as $x' = \min(\max(0, x+\Delta x), p-1)$, and $y' = \min(\max(0, y+\Delta y), q-1)$. The value of $\Re'(x', y')$ is calculated according to the values of the surrounding integer locations via bilinear interpolation.

Finally, the adaptive spatial feature $\mathbf{u}(x,y)$ of location (x,y) can be extracted as follows

$$\mathbf{u}(x,y) = \sum_{i=1}^{Len(\Re')} w_i \cdot \Re'_{\text{new}}(x_i, y_i) \cdot \Delta m_i, \tag{4}$$

where w_i represents the corresponding weight of the kernel. Δm_i refers to the weight of the ith sampling pixel, and if the sample pixel does not meet the sampling rules, Δm_i tends toward zero [54]. $Len(\cdot)$ represents the length of the vector $\Re'$. We provide a visual explanation of the proposed ASPConv module via gradient guided backpropagation in Fig. 5. Focusing on the framed area, we can draw the conclusion that the proposed method can always adjust the shapes of the region of interest in adaptive rules according to the real objects. We also package the ASP unit into a simple and easy-to-use network layer that can replace the traditional convolutional layer and can be placed anywhere[1].

3.2. ASPCaps

To create a deep and robust CapsuleNet with fewer parameters, there are some considerations: 1) Some robust capsule layers should be staked. 2) Location connections and shared routing matrices should be added to a dynamic routing algorithm. 3) More adaptive spatial information should be learned through the location connection. Based on these considerations, we develop a novel deep CapsuleNet named ASPCNet with a special ASP unit. Specifically, an ASP-based conv-capsule named ASPCaps is introduced during the dynamic routing process to replace the general conventional convolution layer. The detailed description is shown below.

After the ASPConvs module operation, the relatively high-level features $\mathbf{u}$ can be obtained as the input features to transfer into the primarycaps layer. The ASPCaps operation has the ability to explore the relationship among capsules. Fig. 6 shows an illustration of the ASPCaps operation. Here, we divide the previous layer feature $\mathbf{u}$ into multiple capsules, where $\mathbf{u} = \{\mathbf{u}_i, i = 1,\ldots,I\}$. Supporting the posterior layer feature $\mathbf{v}$ with j capsules, the detailed transmission process is described in the following. For each capsule during the dynamic routing process, all the capsules in their receptive fields make a prediction through the transform matrix. Here, given a pixel at location (x,y), $\mathbf{u}_i^{(x+p)(y+q)}$ represents the output of the capsule, which is the i-th capsule in the last capsule layer at position $(x+p, y+q)$, and $\mathbf{W}_{ij}^{pq}$ represents the shared transform

[1] ASPConvs can be found on the author's homepage (https://github.com/Cimy-wang).

Fig. 5. Two image patches with the same region of interest. (a) and (b) Pseudo color images and (c) and (d) Visualization features of contributing scores. Different colors indicate the proportion of contribution in the feature learning process, where blue indicates higher contribution scores and red indicates lower contribution scores.

Fig. 6. Illustration of the process of the ASPCaps module, which vividly describes the transfer process from previous features **u** to posterior features **v**.

matrix between the ith capsule of layer **u** and the jth capsule of layer **v**, which possesses adaptive rotation rules based on ASPConv. I is the number of capsules in the last capsule layer.

$$\mathbf{u}_{j|i}^{(x+p)(y+q)} = \mathbf{W}_{ij}^{pq}\mathbf{u}_i^{(x+p)(y+q)}, \tag{5}$$

where $\mathbf{u}_{j|i}^{(x+p)(y+q)}$ denotes the jth prediction feature in the high feature layer. As all the "prediction vectors" $\mathbf{u}_{j|i}$ are obtained, the weighted sum of all capsules of $(x+p)(y+q)$ can serve as the input of the capsule.

$$\mathbf{s}_j^{xy} = \sum_i \sum_p \sum_q \mathbf{c}_{ij}^{pq}\mathbf{u}_{j|i}^{(x+p)(y+q)}, \tag{6}$$

where $\mathbf{s}_j^{xy}$ is the input vector of capsule j and $\mathbf{c}_{ij}^{pq}$ denotes the coupling coefficients, which are obtained by the softmax function and updated by the dynamic routing algorithm as follows

$$\mathbf{c}_{ij}^{pq} = \frac{\exp\left(\mathbf{b}_{ij}^{pq}\right)}{\sum_{j=1}^{J}\mathbf{b}_{ij}^{pq}}, \tag{7}$$

where J is the total number of capsules of posterior layer **v**. $\mathbf{b}_{ij}^{pq}$ is initialized to 0 before the training begins and is determined by the dynamic routing algorithm. In the dynamic routing algorithm, the coefficient $\mathbf{b}_{ij}^{pq}$ is iteratively refined by measuring the agreement between the "prediction vector" $\mathbf{u}_{j|i}^{(x+p)(y+q)}$ and $\mathbf{v}_j^x$. If agreement is reached to a great extent, capsule $\mathbf{u}_{j|i}^{(x+p)(y+q)}$ make a good prediction for capsule $\mathbf{v}_j^{xy}$. Then, the coefficient $\mathbf{b}_{ij}^{pq}$ is significantly increased. In our network, the agreement is quantified as the inner product between two vectors $\mathbf{u}_{j|i}^{(x+p)(y+q)}$ and $\mathbf{v}_j^{xy}$. This agreement is added to $\mathbf{b}_{ij}^{pq}$ as follows

$$\mathbf{b}_{ij}^{pq} \leftarrow \mathbf{b}_{ij}^{pq} + \mathbf{u}_{j|i}^{(x+p)(y+q)}\mathbf{v}_j^{xy}. \tag{8}$$

Finally, the input vector is squashed by a nonlinear function (i.e., the *squash* function) to generate the output of the capsule. The detailed equations are listed below

$$\mathbf{v}_j^{xy} = \mathrm{Squash}\left(\mathbf{s}_j^{xy}\right). \tag{9}$$

3.3. Architecture of the ASPCNet

In the following, we describe the whole framework of HSIC based on ASPCNet, which is shown in Fig. 3. The whole architecture can be divided into several steps: 1) Dimension reduction by using PCA. 2) Primary feature extraction by using ASPConvs modules. 3) Hierarchical structure extraction by using ASPCaps modules. 4) Image patch-based classification. Here, the proposed ASPCNet consists of four blocks: ASPConv 1, ASPConv 2, ASPCaps 1, and ASPCaps 2, which are shown in Fig. 7. Significantly, each ASPConv block consists of three layers, i.e., an ASP layer, a convolution layer, and a rectified linear unit activation function. For details on the architecture of the ASPCNet and the configuration of their hyperparameters, we refer the reader to the APPENDIX section.

After learning the deep ASPCNet features **v**, we use a fully connected capsule layer to obtain digital capsule $\mathbf{v}'$, which is of size $T \times 16$. Then, the Euclidean norm (i.e., $\|\ell_2\|$) is used to obtain the probability vector $\mathbf{v}''$ for each test patch with size $T \times 1$, where each value is mapped to [0, 1]. Once the deep network is well-trained, for a test patch $\mathbf{Y}^{test} \in \mathbb{R}^{m\times m\times d}$, the class label of its center pixel can be determined based on the maximum probability

$$\mathrm{Class}\left(\mathbf{Y}^{test}\right) = \arg\max_{t=1,2,\ldots,T}\mathbf{v}_t''. \tag{10}$$

Here, we use the margin loss [46] as the loss function in this paper, since it can increase the probability of true classes compared with that of the traditional cross-entropy loss. For each capsule j in the last capsule layer, its loss function L_j can be calculated as follows

$$L_j = T_j \cdot \max\left(0, n^+ - \|\mathbf{v}_j\|\right)^2 + \lambda(1 - T_j) \cdot \max\left(0, \|\mathbf{v}_j\| - n^-\right)^2, \tag{11}$$

where T_j = 1 when class j is actually present and equals to 0 otherwise. We set n^+ = 0.9 and n^- = 0.1 as the lower bound and the upper bound for the correct class and the wrong class[46], respectively. λ is a hyper-parameter that controls the effect of gradient backpropagation in the initial phase during training. The total loss of the model is the sum of the loss of all the output capsules of the last layer.

Fig. 7. Network architecture of the proposed ASPCNet model.

4. Experiments results and analysis

In this section, three experiments are performed on the Salinas, University of Pavia, and Houston datasets. The details of these three datasets are described as follows:

4.1. Datasets

(1) *Salinas*: The first dataset is the Salinas image dataset, which was acquired by AVIRIS over the Salinas Valley in southern California. The images have 224 bands and 512×217 pixels with a spatial resolution of 3.7 m. According to the reference classification map shown in Fig. 8, there are 16 different classes labeled with different colors.
(2) *University of Pavia*: The second dataset is called the University of Pavia dataset and was photographed by a Reflective Optics System Imaging Spectrometer (ROSIS-3) in the range of 0.46–0.86 μm over the University of Pavia. The scenes contain 115 bands and 610×340 pixels with a spatial resolution of 1.3 m per pixel. After 12 fairly noisy bands are removed, the experiments are performed using only 103 bands. Fig. 9(a-c) shows a false color composite of the University of Pavia image, the reference classification map, and the color labels, and there are 9 different classes.
(3) *Houston*: The last image set, namely the Houston images, was captured by an ITRES-CASI 1500 sensor, and shows the University of Houston campus and its neighboring field. The images contain 349×1905 pixels at a spatial resolution of 2.5 m per pixel and 144 bands in the range of 0.38–1.05 μm. Fig. 10 shows the false-color composite of the Indian Pines image and the corresponding reference data with 15 different classes.

4.2. Compared Methods

In this subsection, we compare the proposed method to other methods, i.e., a SVM [8], extreme learning machine (ELM) [62], EPF [21], deep convolution neural network (DeepCNN), CapsuleNet [49], deformable convolution neural network (DHCNet) [56], and spectral-spatial fully convolutional network (SSFCN) [41]. The implementation details of each method included in the experiments are summarized as follows.

Fig. 8. The Salinas dataset. (a) Three-band color composite of the Salinas image. (b) and (c) the corresponding reference data.

Fig. 9. The University of Pavia dataset. (a) Three-band color composite of the University of Pavia image. (b) and (c) the corresponding reference data.

Fig. 10. The Houston dataset. (a) Three-band color composite of the Houston image. (b) and (c) the corresponding reference data.

(1) SVM: Classification is performed with the original spectral features via an SVM classifier with an RBF kernel [8]. This algorithm is implemented using the LIBSVM library [8] and the parameters of the RBF-SVM are chosen using fivefold cross-validation.
(2) ELM: Original spectral classification is performed with ELM [62]. This is a simple machine learning algorithm with only one hidden layer and one output layer and the default parameters of the ELM in [62] are adopted.
(3) EPF: This is a spectral-spatial classification method with the first 20 principal components determined via guided filter-based EPF. We use the default parameters of a filtering size r and blur degree ϵ to 3 and 0.2, respectively [21], in the implementations.
(4) DeepCNN: Spatial classification is performed via a 3D-CNN method with the first 20 principal components. The architecture is shown in Table 1. For each pixel, a patch size of 27×27 is extracted as the input of the network. This algorithm is trained using the Adam optimizer with a learning rate of 5e-4.

Table 1
The structure of the DeepCNN method. d represents the image dimension, and T is the total class of HSIs. The stride of Convolution Layer and Maxpooling Layer are set to 1 and 2, respectively.

Layer Name	Input Shape	Kernel Size	Output Shape
Input Layer	(27, 27, d)	-	(27, 27, d)
Conv Layer	(27, 27, d)	(3×3×d, 32)	(27, 27, 32)
Conv Layer	(27, 27, 32)	(3×3×32, 32)	(27, 27, 32)
Conv Layer	(27, 27, 32)	(3×3×32, 64)	(27, 27, 64)
Conv Layer	(27, 27, 64)	(3×3×64, 64)	(27, 27, 64)
Maxpooling	(27, 27, 64)	(2×2)	(14, 14, 64)
Conv Layer	(14, 14, 64)	(3×3×64, 128)	(14, 14, 128)
Conv Layer	(14, 14, 128)	(3×3×128, 128)	(14, 14, 128)
Conv Layer	(14, 14, 128)	(3×3×128, 256)	(14, 14, 256)
Conv Layer	(14, 14, 256)	(3×3×256, 256)	(14, 14, 256)
Maxpooling	(14, 14, 256)	(2×2)	(7, 7, 256)
FC Layers	(12544)	-	(1280)
FC Layers	(1280)	-	(128)
FC Layers	(128)	-	(T)

(5) CapsuleNet: The original capsule network-based HSIC method with the original spectral information [49]. The architecture adopts the Adam optimizer with a learning rate equal to 0.001, 100 training epochs, and a patch size of 15×15.
6) DHCNet: This is a deformable convolution neural network for HSIC with the first 3 principal components. According to the implementations [56], the patch size, the momentum of the batch normalization layer, and the training epoch are set to 29×29, 0.9, and 1500, respectively.
(7) SSFCN: This is a spectral-spatial FCN with conditional random fields. The detailed architecture and default parameters of the network follow the implementations in [41]. The network is trained with a learning rate of 5e-4 using the Adam optimizer.
(8) ASPCNet: This is the proposed spectral-spatial ASPC neural network. The detailed architecture of the network is shown in Table 2. We set the batch size and training epoch to 96 and 200, respectively. The momentum of the batch normalization layer is set to 0.9. The network is trained using the Adam optimizer with a learning rate of 5e-4, beta_1 of 0.9, beta_2 of 0.999, and epsilon of 1e-8.

All of the methods are carried out on a desktop computer with a Windows 10 OS, an Intel i7-10700F processor with 32 GB of RAM, and a single NVIDIA GTX2070 GPU. DeepCNN, CapsuleNet, DHCNet, SSFCN, and ASPCNet are constructed by using the *Keras* framework, CUDA 10, and Python 3.6.5 as the programming language. The SVM, ELM, and EPF methods are carried out in MATLAB R2020a without GPU acceleration.

Table 2
The structure of the proposed ASPCNet method. d represents image dimension, and T is the total class of HSI. The stride of Convolution Layers are described in Appendices A.

Layer Name	Input Shape	Kernel Size	Output Shape
Input Layer	(25, 25, d)	-	(25, 25, d)
ASP Layer 1	(25, 25, d)	(3×3×d, 128)	(25, 25, 128)
Conv Layer 1	(25, 25, 128)	(1×1×128, 128)	(13, 13, 128)
ASP Layer 2	(13, 13, 128)	(3×3×128, 256)	(13, 13, 256)
Conv Layer 2	(13, 13, 256)	(1×1×256, 256)	(7, 7, 256)
BN Layer	(7, 7, 256)	-	(7, 7, 256)
ConvertToCaps	(7, 7, 256)	-	(7, 7, 256, 1)
ASPCaps Layer 1	(7, 7, 256, 1)	(3×3×256, 32, 4)	(7, 7, 32, 4)
ASPCaps Layer 2	(7, 7, 32, 4)	(3×3×128, 32, 4)	(7, 7, 32, 4)
FlattenCaps	(7, 7, 32, 4)	-	(1568, 4)
DigitalCaps	(1568, 4)	(T, 1568, 16, 4)	(T, 16)
CapsToScalars	(T, 16)	-	(T, 1)

4.3. Quantitative Metrics

To effectively evaluate the classification performance of different methods, three objective indexes, i.e., the overall accuracy (OA), average accuracy (AA), and Kappa coefficient (Kappa) are adopted in the experiments. Specifically, the OA value represents the percentage of all test pixels that are correctly classified. the AA value measures the mean of all class accuracies. Taking the uncertainty factors of the classification into consideration, the Kappa value is proposed, which represents the percentage of correctly classified pixels corrected by the degree of agreement.

4.4. Analysis of the influences of different parameters

In this subsection, a detailed analysis of the influence of three parameters (i.e., the number of dimensions d, patch size $m \times m$ and dilation rate h_{dr}) on the image classification performance is conducted on three real datasets.

4.4.1. Analysis of the influence of the number of dimensions

First, we analyze the influence of the number of dimensions (the principal components) of HSI under different settings. Fig. 11 shows the influence of parameter d on three different datasets, i.e., the Salinas, University of Pavia, and Houston datasets. As seen in Fig. 11(a), for the Salinas dataset, the classification performance of the ASPCNet method increases first and then decreases slightly when $d > 15$ as parameter d increases because the first several components include most of the spatial information. With an increasing number of components, there is much extra even noisy information among the HSI bands, and this situation is even worse for a complex dataset such as the University of Pavia (see Fig. 11(b)). In addition, a similar phenomenon can be seen on the Houston datasets in Fig. 11(c). Therefore, $d = 15$ is set as the default parameter among all datasets.

4.4.2. Analysis of the influence of the patch size

Second, we discuss the influence of the patch size m of OA under different settings. Experiments are conducted on the three datasets. For instance, Fig. 12(a) denotes different patch sizes on the Salinas dataset, showing the OA values versus different settings of patch size m varying from 19×19 to 35×35 with an interval of 2. As seen, as the parameter m increases, the OA increases first and declines later. Experimental results indicate that it can extract more local spatial features at first. As m increases (e.g., 35×35), it may introduce extra information as interference. The same conclusion is reached for the other two datasets. Therefore, in consideration of the trade-off between classification performance and the number of parameters, a patch size of 25×25 is adopted as the default parameter.

4.4.3. Analysis of the influence of the dilation rate

The third experiment is conducted on three datasets to evaluate the classification performance under different dilation rate settings. In general, a CNN uses a pooling operation to increase the receptive field. Unfortunately, some materials may "disappear" after several pooling layers. Dilated convolutions can avoid this problem in an elegant way, in which the dilation rate h_{dr} exerts control. Here, Fig. 13 shows the influence of the dilation rate h_{dr} on the three different datasets. As we can see from Fig. 13(a), as the dilation rate h_{dr} increases, OA increases first when $h_{dr} < 3$ and then decreases when $h_{dr} > 3$ on the Salinas dataset. For the University of Pavia and Houston datasets, shown in Fig. 13(b) and (c), the proposed ASPCNet can also obtain the highest accuracies when h_{dr} equals 3. Therefore, for the Salinas, University of

Fig. 11. The effects of the number of dimension d based on the proposed method for three real datasets: (a) Salinas, (b) University of Pavia, (c) Houston.

Fig. 12. The effects of the patch size $m \times m$ based on the proposed method for three real datasets: (a) Salinas, (b) University of Pavia, (c) Houston.

Fig. 13. The effects of the dilation rate h_{dr} based on the proposed method for three real datasets: (a) Salinas, (b) University of Pavia, (c) Houston.

Pavia and Houston datasets, the default parameters are set as h_{dr} = 3.

Finally, the impact of the training epoch on the accuracies and losses of training and testing are analyzed as well. Here, the proposed ASPCNet method can quickly converge with little oscillation. For example, it can always reach the maximum training accuracies when the epoch $\in$ [25,30]. Therefore, if we want to improve the training efficiency in practical engineering applications, we can use the "*early-stopping*" operation; that is, when the training accuracies reach the local maximum and there are no better training accuracies in the subsequent training epochs, the training processing is automatically stopped.

4.5. Classification Results

In this section, the proposed method named ASPCNet is compared with different well-known classification methods, including traditional machine learning-based methods such as SVM [8], ELM [62], and EPF [21] and state-of-the-art (SOTA) deep learning-based methods such as DeepCNN, CapsuleNet [49], DHCNet [56], and SSFCN [41], on the Salinas, University of Pavia and Houston datasets. To evaluate the aforementioned three quantitative metrics,

Table 3
Numbers of training and testing samples on the Salinas dataset.

Class Name	Total	Training	Testing
Weeds 1	2009	20	1989
Weeds 2	3726	20	3706
Fallow	1976	20	1956
Fallow P	1394	20	1374
Fallow S	2678	20	2658
Stubble	3959	20	3939
Celery	3579	20	3559
Graps	11271	20	11251
Soil	6203	20	6183
Corn	3278	20	3258
Lettuce 4wk	1068	20	1048
Lettuce 5wk	1927	20	1907
Lettuce 6wk	916	20	896
Lettuce 7wk	1070	20	1050
Vinyard U	7268	20	7248
Vinyard T	1807	20	1787
Total	54129	320	53809

Fig. 14. Classification maps obtained by eight methods on the Salinas dataset: (a) Ground Truth Map, (b) SVM (84.33%), (c) ELM (87.78%), (d) EPF (90.88%), (e) DeepCNN (87.12%), (f) CapsuleNet (91.50%), (g) DHCNet (92.10%), (h) SSFCN (93.85%), and (i) ASPCNet (96.65%).

i.e., OA, AA and the Kappa, each experiment is repeated approximately 20 times for each classification method.

The first experiment is conducted on the Salinas dataset. Here, 20 labeled training samples per class are randomly chosen as the training patches to train the parameters of the network and the rest of the samples are viewed as test samples, as seen in Table 3. The classification performance and the corresponding classification maps obtained by the nine methods are exhibited in Table 6 and Fig. 14. The numbers in parentheses indicate the standard variances, and the best results of each class are highlighted in bold typeface. The SVM classifier obtains the worst classification accuracy on the Salinas dataset, which indicates that only using spectral information is not enough for complex HSIs. By taking the spatial information of HSI into consideration, traditional machine learning-based methods such as EPF can achieve better results with 89.94%. Meanwhile, image patch-based classifiers such as DeepCNN and CapsuleNet, which both obtain about 90% classification accuracies, indicate that image patch-based classifiers truly play important role in extracting spatial information of HSIs. As seen from Fig. 14(e)-(f), there are some misclassified samples in the CapsuleNet and DHCNet results, especially for the grapes untrained class, the Brocoli green weeds 2 class, and the Vinyard untrained class, which demonstrates that there is still improvement space. Moreover, the SSFCN is a more efficient method that can avoid patch extraction and was a new SOTA in 2020. By contrast, the proposed ASPCNet uses a new framework by developing the adaptive spatial unit into the original CapsuleNet, which can better adapt to the complex spatial characteristics of the HSI. In addition, ASPCNet can obtain the highest classification accuracies in terms of OA, AA, and Kappa.

Table 4
Numbers of training and testing samples on the University of Pavia dataset.

Class Name	Total	Training	Testing
Asphalt	6631	20	6611
Meadows	18649	20	18629
Gravel	2099	20	2079
Trees	3064	20	3044
Metal sheets	1345	20	1325
Bare soil	5029	20	5009
Bitumen	1330	20	1310
Self-Blocking Bricks	3682	20	3662
Shadows	947	20	927
Total	42776	180	42596

The second experiment is performed on the University of Pavia dataset. First, 20 samples per class are selected randomly as the training samples and the rest are used as test samples (see Table 4). Table 7 shows the classification performance of the different methods, and Fig. 15 displays the corresponding classification maps. As seen, the proposed ASPCNet achieves the highest accuracies on four classes and the highest OA, AA, and Kappa with a smaller variance, which shows the outstanding performance of ASPCNet. Moreover, as we can see from Fig. 15, ASPCNet provides more homogeneous classification maps with clear object edges, even for the University of Pavia dataset, with complex structures.

Then, similar conclusions can be observed on the Houston dataset, for which training samples are shown in Table 5 and the classification results and maps are displayed in Table 8 and Fig. 16, respectively. The classification results of ASPCNet are consistently better than those of the compared spectral-spatial classification methods, and the experimental results from the three real datasets indicate the robustness of the proposed ASPCNet method.

Last, experiments conducted on three datasets indicate the effectiveness of the proposed method with a larger number of

Fig. 15. Classification maps obtained by eight methods on the University of Pavia dataset: (a) Ground Truth Map, (b) SVM (76.39%), (c) ELM (78.85%), (d) EPF (84.17%), (e) DeepCNN (80.93%), (f) CapsuleNet (83.91%), (g) DHCNet (86.63%), (h) SSFCN (89.82%), and (i) ASPCNet (92.59%).

Table 5
Numbers of training and testing samples on the Houston dataset.

Class Name	Total	Training	Testing
Grass-healthy	1251	20	1231
Grass-stressed	1254	20	1234
Grass-synthetic	697	20	677
Tree	1244	20	1224
Soil	1242	20	1222
Water	325	20	305
Residential	1268	20	1248
Commercial	1244	20	1224
Road	1252	20	1232
Highway	1227	20	1207
Railway	1235	20	1215
Parking Lot-1	1233	20	1213
Parking Lot-2	469	20	449
Tennis Court	428	20	408
Running Track	660	20	640
Total	15029	300	14729

training samples. Here, the number of the training samples is 40 for each class, which is selected randomly. Table 9 shows the OA, AA, and Kappa obtained by different methods of three datasets. As seen from Table 9, whatever compared with traditional machine learning-based methods or deep learning-based methods, the proposed ASPCNet method can always obtain a better result.

4.6. Comparison of Different Feature Fusion Steps

To measure the performance of different components of the proposed ASPCNet, an experiment is performed on the University of Pavia dataset with 40 training samples per class. The experimental results are shown in Table 10. As seen, Dil, Def, and Dil-Def refer to learning the primary convolution feature using the dilated convolution, deformable convolution, and Dil-Def convolution, respectively. Note that Dil-Def and ASPConv are different. The Dil-Def convolution means that the network conducts dilated convolution first and then deformable convolution later (two layers totally), whereas ASPConv aims to integrate the dilation and deformation operations into one unit (only a single layer). Caps and ConvCaps refer to learning digitalCaps features using the original capsule network and conv-capsule network with digital feature extraction.

In the proposed ASPCNet method, we create an ASP unit by taking advantage of both dilated and deformable convolution operations, which can simultaneously expand the receptive field and rotate the sampling location, and effectively avoid grid effects.

Table 6
Classification accuracies (%) on the Salinas dataset among different methods. Number in parenthesis indicates the standard variance of the repeated experiments. The best results of each class are highlighted in **Bold** typeface.

Classes		SVM	ELM	EPF	DeepCNN	CapsuleNet	DHCNet	SSFCN	ASPCNet
CA	1	99.47(0.67)	99.89(0.22)	**100.0(0.00)**	94.71(3.25)	94.59(4.19)	93.20(3.31)	99.89(0.22)	99.89(0.04)
	2	99.10(0.41)	98.94(0.50)	**100.0(0.00)**	85.28(6.41)	81.90(6.16)	96.45(2.34)	99.99(0.02)	**100.0(0.00)**
	3	89.22(2.51)	94.68(1.22)	92.98(1.08)	99.62(0.54)	99.90(0.14)	99.50(4.21)	99.55(0.57)	**100.0(0.00)**
	4	97.22(0.56)	98.58(0.21)	97.53(0.24)	99.99(0.03)	99.98(0.03)	97.56(1.46)	**100.0(0.00)**	99.55(0.44)
	5	98.43(1.13)	96.97(1.57)	98.83(0.17)	98.40(1.42)	**98.93(0.69)**	94.62(1.96)	96.14(2.68)	98.65(1.30)
	6	**100.0(0.00)**	99.99(0.01)	**100.0(0.00)**	98.93(1.46)	98.32(1.62)	95.88(4.28)	97.42(2.99)	95.80(0.28)
	7	99.43(0.37)	99.59(0.12)	**99.93(0.07)**	95.74(2.39)	96.69(2.64)	91.62(4.37)	97.98(3.18)	99.26(1.30)
	8	71.35(3.33)	77.71(2.80)	81.09(5.26)	60.83(22.2)	78.66(3.29)	85.27(5.07)	83.81(8.21)	**87.30(4.60)**
	9	99.11(0.47)	99.31(0.13)	99.25(0.51)	99.64(0.50)	99.44(0.56)	98.53(7.42)	98.45(1.76)	**99.83(0.14)**
	10	75.61(9.50)	91.60(2.87)	86.10(8.06)	99.77(0.27)	**99.98(0.01)**	86.87(2.14)	98.34(2.35)	98.22(0.20)
	11	84.89(6.10)	93.00(1.11)	93.55(6.16)	98.74(0.82)	98.22(0.53)	96.39(1.52)	97.82(1.30)	**99.06(0.86)**
	12	95.86(1.24)	95.29(0.66)	99.20(0.85)	99.16(1.33)	98.62(1.49)	98.61(8.65)	99.33(0.49)	**99.64(0.39)**
	13	93.22(3.27)	93.93(1.97)	97.94(2.70)	98.60(1.99)	98.14(2.39)	98.52(7.93)	94.31(9.72)	**98.62(0.76)**
	14	90.26(7.19)	92.97(2.94)	97.76(1.40)	99.43(0.83)	**99.94(0.09)**	93.35(3.62)	98.70(1.78)	99.56(0.54)
	15	52.67(2.47)	59.45(3.91)	69.22(8.51)	85.97(10.6)	80.91(11.1)	82.74(6.32)	85.64(7.37)	**90.38(4.53)**
	16	95.99(2.08)	99.21(0.57)	99.42(0.65)	98.40(1.55)	98.13(1.69)	93.85(2.61)	98.25(2.24)	**100.0(0.00)**
OA		83.96(1.04)	87.74(0.92)	89.94(2.17)	88.06(4.15)	90.88(2.85)	91.45(2.03)	93.62(1.09)	**96.37(1.13)**
AA		90.12(0.32)	93.20(0.30)	94.61(0.73)	94.58(1.56)	95.15(1.79)	93.79(1.35)	96.60(0.99)	**97.86(0.43)**
Kappa		82.20(1.14)	86.37(1.01)	88.82(2.39)	86.80(4.54)	89.86(3.17)	90.50(2.26)	92.91(1.20)	**95.97(1.26)**

Table 7
Classification accuracies (%) on the University of Pavia dataset among different methods. Number in parenthesis indicates the standard variance of the repeated experiments. The best results of each class are highlighted in **Bold** typeface.

Classes		SVM	ELM	EPF	DeepCNN	CapsuleNet	DHCNet	SSFCN	ASPCNet
CA	1	93.62(3.11)	89.77(3.01)	**95.83(4.00)**	60.51(12.6)	77.23(2.42)	72.21(7.80)	75.29(5.04)	77.04(3.61)
	2	90.16(1.88)	90.91(2.37)	95.93(2.15)	84.68(4.80)	83.17(9.20)	88.16(1.96)	93.44(4.96)	**98.95(0.46)**
	3	54.47(5.38)	59.26(3.93)	69.64(13.0)	73.06(19.9)	73.29(3.70)	76.87(14.4)	92.54(2.71)	**93.37(2.35)**
	4	69.89(9.88)	62.87(3.08)	81.44(7.22)	76.11(14.5)	92.83(5.72)	92.76(5.09)	70.95(3.49)	**93.55(2.15)**
	5	91.76(4.26)	97.64(1.21)	96.53(5.24)	96.69(3.04)	91.74(5.45)	93.25(5.05)	92.59(8.88)	**99.77(0.09)**
	6	47.86(4.19)	54.53(4.23)	61.98(9.74)	93.57(4.24)	91.72(4.32)	84.17(11.1)	**99.68(0.24)**	96.24(3.25)
	7	54.76(6.65)	49.52(3.15)	79.71(10.4)	94.56(5.79)	90.41(2.52)	95.32(2.07)	**99.32(0.91)**	98.34(1.09)
	8	75.65(11.0)	77.20(1.21)	86.16(3.68)	63.16(11.6)	67.00(8.12)	89.99(6.75)	**95.87(1.96)**	86.95(2.20)
	9	99.83(0.11)	**99.96(0.09)**	97.85(1.60)	80.32(6.56)	99.11(1.13)	94.51(2.62)	67.05(3.87)	95.26(2.05)
OA		75.40(1.38)	76.76(1.02)	85.87(3.93)	79.52(4.59)	83.19(4.66)	87.20(2.71)	89.52(2.32)	**92.61(0.80)**
AA		75.33(0.61)	75.74(0.76)	85.01(4.22)	80.29(4.27)	85.61(2.15)	87.58(2.16)	87.41(1.52)	**93.26(0.67)**
Kappa		68.84(1.68)	70.41(1.30)	81.83(4.94)	73.68(5.65)	78.53(5.54)	81.33(3.55)	86.27(2.88)	**90.17(1.04)**

Table 8
Classification accuracies (%) on the Houston dataset among different methods. Number in parenthesis indicates the standard variance of the repeated experiments. The best results of each class are highlighted in **Bold** typeface.

Classes		SVM	ELM	EPF	DeepCNN	CapsuleNet	DHCNet	SSFCN	ASPCNet
CA	1	90.18(4.29)	91.37(3.71)	92.08(4.82)	83.47(3.47)	90.17(4.82)	84.77(3.88)	86.47(7.73)	**93.56(4.26)**
	2	91.45(5.12)	**94.30(3.85)**	91.52(6.24)	86.42(6.81)	79.66(6.24)	90.05(4.01)	86.52(3.03)	90.34(7.44)
	3	96.94(4.07)	**99.91(0.07)**	98.22(3.14)	92.68(5.24)	99.41(3.14)	96.37(4.25)	99.38(0.41)	96.79(1.82)
	4	98.01(1.87)	98.13(1.33)	**98.99(0.36)**	82.60(5.83)	78.19(0.36)	81.45(4.59)	82.68(6.34)	80.03(3.70)
	5	92.94(5.04)	93.81(2.15)	95.68(5.47)	99.21(1.46)	98.69(5.47)	98.94(1.41)	99.05(1.17)	**99.92(0.09)**
	6	91.41(6.44)	**99.79(0.28)**	92.90(6.79)	91.93(7.59)	93.77(6.79)	92.41(4.69)	94.75(3.82)	92.69(5.80)
	7	77.41(4.28)	84.98(1.68)	79.90(5.00)	70.89(13.7)	93.07(5.00)	68.24(6.47)	84.47(5.42)	**93.16(5.30)**
	8	77.77(4.08)	86.11(3.45)	85.46(3.28)	67.94(7.34)	74.59(3.28)	84.81(1.94)	81.86(4.84)	**87.53(7.95)**
	9	71.72(4.53)	74.69(1.87)	79.23(5.28)	54.44(6.22)	74.03(5.28)	77.81(6.10)	**80.03(2.20)**	75.90(6.99)
	10	75.15(4.55)	70.33(4.53)	87.79(7.12)	90.54(5.09)	98.26(7.12)	91.73(10.5)	98.92(0.67)	**99.08(4.45)**
	11	73.92(4.16)	69.96(3.50)	81.62(6.45)	79.92(19.5)	77.78(6.45)	91.55(4.43)	90.86(4.12)	**93.49(3.36)**
	12	71.99(2.39)	69.72(2.42)	79.77(5.13)	82.49(4.57)	83.18(5.13)	88.99(4.75)	89.45(4.10)	**90.02(7.00)**
	13	44.15(11.4)	52.68(11.6)	62.88(14.5)	89.34(7.82)	89.31(14.5)	92.71(3.11)	**96.44(3.08)**	90.79(3.24)
	14	79.99(8.65)	93.19(3.09)	83.61(10.3)	**100.0(0.00)**	**100.0(0.00)**	**100.0(0.00)**	**100.0(0.00)**	**100.0(0.00)**
	15	99.24(0.64)	99.04(0.88)	99.72(0.55)	96.03(5.02)	**100.0(0.00)**	97.87(1.73)	98.75(1.11)	98.71(1.67)
OA		82.23(1.62)	84.05(1.15)	87.03(1.85)	83.13(3.23)	86.92(1.85)	88.02(1.30)	88.52(1.16)	**89.94(1.41)**
AA		82.15(1.43)	85.30(1.11)	87.29(1.70)	84.53(2.59)	88.74(1.70)	89.18(1.39)	91.11(0.94)	**91.93(1.11)**
Kappa		80.79(1.74)	82.75(1.24)	85.98(1.99)	80.69(3.49)	85.87(1.99)	86.50(1.41)	88.89(1.26)	**89.48(1.53)**

(a) (b) (c) (d) (e) (f) (g) (h) (i)

Fig. 16. Classification maps obtained by eight methods on the Houston dataset: (a) Ground Truth Map, (b) SVM (82.21%), (c) ELM (84.52%), (d) EPF (87.47%), (e) DeepCNN (90.12%), (f) CapsuleNet (86.57%), (g) DHCNet (87.66%), (h) SSFCN (88.10%), and (i) ASPCNet (90.02%).

Table 9
Classification accuracies (%) of different methods on three datasets with 40 training samples per class. Number in parenthesis indicates the standard variance of the repeated experiments. The best results of each class are highlighted in **Bold** typeface.

Datasets	Index	SVM	ELM	EPF	DeepCNN	CapsuleNet	DHCNet	SSFCN	ASPCNet
Salinas	OA	85.86(1.29)	89.89(0.45)	91.25(2.33)	92.24(4.39)	93.00(2.54)	94.09(0.72)	94.64(2.26)	**96.97(2.13)**
	AA	90.68(0.40)	93.59(0.21)	94.72(0.60)	94.29(2.15)	95.61(1.40)	96.26(0.28)	96.55(1.12)	**97.73(0.65)**
	Kappa	84.31(1.42)	88.74(0.50)	90.28(2.56)	91.31(4.96)	92.18(2.85)	93.42(0.81)	94.02(2.52)	**96.81(2.35)**
University of Pavia	OA	82.14(1.39)	84.50(0.57)	92.88(2.72)	92.67(2.85)	93.22(1.97)	95.29(1.91)	96.73(0.91)	**97.84(0.65)**
	AA	79.99(0.99)	81.50(0.78)	92.39(2.18)	92.36(1.79)	92.42(1.58)	95.30(1.25)	96.66(0.67)	**97.40(0.52)**
	Kappa	77.09(1.67)	79.83(0.66)	90.73(3.48)	90.39(3.67)	91.07(2.53)	95.23(2.49)	94.86(1.20)	**97.13(0.86)**
Houston	OA	87.47(0.72)	88.71(0.50)	91.79(1.10)	88.98(1.47)	92.96(1.50)	93.57(0.54)	94.38(0.58)	**95.42(0.57)**
	AA	87.27(1.30)	89.54(0.46)	91.34(1.69)	90.90(1.10)	94.11(1.31)	94.60(0.63)	95.27(0.44)	**96.03(0.55)**
	Kappa	86.45(0.77)	87.79(0.53)	91.12(1.19)	88.08(1.59)	92.39(1.62)	93.04(0.59)	93.93(0.62)	**95.04(0.61)**

Table 10
Classification accuracies (%) of different steps of the ASPCNet on the University of Pavia dataset with 40 training samples per class.

Indexes	ASPCaps-based			ASPConvs-based		ASPCNet
	Dil	Def	Dil-Def	Caps	ConvCaps	
OA	91.35	92.88	93.58	94.31	94.99	**97.81**
AA	88.78	92.33	93.28	95.13	95.45	**97.14**
Kappa	88.62	90.62	91.53	94.32	95.42	**97.23**

Fig. 17. Classification accuracies (%) of different number of ASPConv and ASPCaps blocks of the ASPCNet on the University of Pavia dataset with 20 training samples per class.

Note that we develop the ASP unit into a traditional CapsuleNet during the primarycaps and digitalcaps process. Based on the experimental results observations, the Dil, Def, and Dil-Def operations lead to lower OAs of 91.35%, 92.88%, and 93.58%, respectively. Especially, the classification performance obtained by ASPCNet is better than that of Dil-Def, which indicates that the ASP unit is not a simple combination of the Dil and Def operations, like stacking one to another. By contrast, the ASP unit combines the Dil and Def operations into a unit, improving the classification performance by adaptively enlarging the receptive field and adapting the shapes according to the complex features of HSIs. Based on the ASPConvs modules, we compare the output classifier using traditional Caps, ConvCaps, and ASPCaps. According to the experimental results, we find that the proposed ASPCNet can obtain the highest result of the three methods, approximately 97.81% (traditional caps and ConvCaps can only obtain 94.31% and 94.99%, respectively). The results demonstrate that the ASPCaps located in the Caps layer can help improve classification performance.

4.7. Comparison of Different Number of ASPConv and ASPCaps Blocks

To explore the optimal numbers of ASPConvs and ASPCaps blocks of the proposed network, another experiment is performed on the University of Pavia dataset with 20 training samples per class. The experimental results are shown in Fig. 17 and Table 11. As seen from Fig. 17, the classification accuracies increase as the number of ASPConv and ASPCaps blocks increase. Moreover, the importance of a more significant number of the ASPCaps block to classification performance is much higher than that of the ASPConv block. From the bottom line of Fig. 17, we can see that there is an over-fitting phenomenon in the proposed network when only increasing the number of ASPCaps without any ASPConv block, which further indicates the significance of the ASPConv block in extracting relatively high-level features. In addition, as the number of the ASPConv and ASPCaps blocks increases, the parameters and training time of ASPCNet increase sharply as well (see Table 11). Here, we choose ASPConv = 2 and ASPCaps = 2 as the default parameter to keep a trade-off between classification accuracy and training time. To sum up, readers can choose the most suitable architectures of the ASPCNet according to their real application.

Table 12
Computational time (in Seconds) of different methods.

Methods	Datasets		
	Salinas	University of Pavia	Houston
SVM	19.68	16.78	12.68
ELM	4.21	1.71	0.98
EPF	21.49	18.67	13.87
DeepCNN	40.44	38.63	30.47
CapsuleNet	38.91	33.72	27.64
DHCNet	53.76	48.06	36.89
SSFCN	140.29	114.98	98.11
DASPNet	57.26	42.95	33.48

4.8. Computing Times of Different Methods

The time consumption (in seconds) of the training processes of the proposed ASPCNet and the other compared methods on the three typical hyperspectral datasets are reported in Table 12. The experimental environments are described at the end of Section 4.2, and training samples are shown in Table 3–5.The traditional machine learning methods (SVM, ELM, and EPF) and some related deep learning methods (DeepCNN, CapsuleNet, DHCNet, and SSFCN) are compared with the proposed ASPCNet. As observed, the proposed ASPCNet method requires more computational time than all the traditional machine learning methods, and the proposed method's main computational cost is due to the dynamic routing algorithm in CapsuleNet. When compared with DHCNet and SSFCN, the ASPCNet method needs less time on the same benchmark. To make the network more efficient and better able to be transformed for other application fields, searching for a lightweight neural network will be a focus of our future research.

Table 11
Various comparison indicators with different number of ASPConv and ASPCaps blocks on the University of Pavia dataset.

The Architecture of Backbone	OA (%)	Kappa (%)	Total Parameters	Time (s)
ASPConv * 1 + ASPCaps * 2	90.37	87.47	1,075,742	38.66
ASPConv * 1 + ASPCaps * 3	90.93	87.98	1,254,457	43.75
ASPConv * 2 + ASPCaps * 2	**92.61**	**90.17**	**1,630,393**	**42.95**
ASPConv * 2 + ASPCaps * 3	92.83	90.53	1,809,108	48.63
ASPConv * 3 + ASPCaps * 2	92.99	91.14	1,850,836	44.63
ASPConv * 3 + ASPCaps * 3	93.58	92.38	2,029,551	53.34

5. Conclusions

In this paper, we develop an adaptive spatial pattern capsule network (ASPCNet) for image classification, in which a unique convolutional unit called the ASP unit is used to extract features. On the research basis of the conv-capsulenet, the ASP units are introduced twice into the CapsuleNet during the classification process. Initially, instead of using a shallow convolutional layer, the proposed ASPCNet uses ASPConvs to extract the relatively high-level features before they are fed into primaycaps, making it easier to transfer hierarchical relationships between low-level and high-level capsules. Furthermore, considering the fixed sampling location of the convolutional kernels, ASPCaps is further introduced to this model, making full use of contextual information adaptively and meeting the requirements of HSIC tasks of complex structures. In addition, the experimental results on three real HSIs demonstrate the superiority of the proposed ASPCNet over several compared methods in terms of the visual qualities of the classification map and quantitative metrics.

CRediT authorship contribution statement

Jinping Wang: Conceptualization, Methodology, Software, Data curation, Writing - original draft. **Xiaojun Tan:** Validation, Funding acquisition. **Jianhuang Lai:** Visualization, Investigation, Formal analysis. **Jun Li:** Supervision, Writing - review & editing.

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.

Acknowledgment

This work was supported by the Key-Area Research and Development Program of Guangdong Province under Grant (2020B0909030005 and 2020B090921003) and Southern Marine Science and Engineering Guangdong Laboratory (Zhuhai) under Grant SML2020SP011. The authors would like to thank the Editor-in-Chief, the anonymous Associate Editor, and reviewers for their insightful comments and suggestions that have greatly improved this paper.

The authors would like to thank the Editor-in-Chief, the anonymous Associate Editor, and reviewers for their insightful comments and suggestions that have greatly improved this paper. .

Appendix A. Proof of the Section III-C Part

A.1. Hyperparameters for ASPConvs 1, 2

Each ASPConv block consists of three layers of an ASP layer, a convolution layer, and a rectified linear unit (ReLU) activation function. In ASPConv Block 1, for the ASPConv layer, the kernel contains 3×3 nonzero weights, 128 filters, stride = 1, and padding = same. For the convolution layer, there are 128 filter kernels of size 1×1, stride = 2, and padding = same. After the convolution operation, ReLU activation functions are followed. As a practice, a padding operation is conducted to ensure that the feature maps remain the same size before and after convolutions. In ASPConv Block 2, for the ASP layer, the kernel contains 3×3 nonzero weights, 256 filters, stride = 1, and padding = same. For the convolution layer, the kernel contains 1×1 nonzero weights, 256 filters, stride = 2, and padding = same. After the convolution operation, the ReLU activation functions and batch normalization (BN) layers are followed.

A.2. Hyperparameters for ASPCaps 1, 2

There are two ASPCaps blocks. For ASPCaps Block 1, the kernel contains 3×3 nonzero weights, 32×4 filters, stride = 1, and padding = same. After the ASPC operation, ReLU activation functions are followed. For ASPCaps Block 2, the kernel contains 3×3 nonzero weights, 32×4 filters, stride = 1, and padding = same.

References

[1] H. Sun, X. Zheng, X. Lu, A supervised segmentation network for hyperspectral image classification, IEEE Trans. Image Process (2021).
[2] S. Jia, S. Jiang, Z. Lin, N. Li, M. Xu, S. Yu, A survey: Deep learning for hyperspectral image classification with few labeled samples, Neurocomputing 448 (2021) 179–204.
[3] B. Du, Y. Zhang, L. Zhang, D. Tao, Beyond the sparsity-based target detector: A hybrid sparsity and statistics-based detector for hyperspectral images, IEEE Trans. Image Process. 25 (11) (2016) 5345–5357.
[4] R. Qureshi, M. Uzair, K. Khurshid, H. Yan, Hyperspectral document image processing: Applications, challenges and future prospects, Pattern Recognit. 90 (2019) 12–22.
[5] X.-H. Han, Y. Zheng, J. Sun, Y.-W. Chen, Hyperspectral reconstruction with redundant camera spectral sensitivity functions, ACM Trans. Multimedia Comput. Commun. Appl. 16 (2) (2020) 1–15.
[6] L. Wang, M. Jia, D. Yin, J. Tian, A review of remote sensing for mangrove forests: 1956–2018, Remote Sens. Environ. 231 (2019) 1–15.
[7] C.-C. Liu, T.-W. Hsu, H.-L. Wen, K.-H. Wang, Mapping pure mangrove patches in small corridors and sandbanks using airborne hyperspectral imagery, Remote Sens. 11 (5) (2019) 592.
[8] F. Melgani, L. Bruzzone, Classification of hyperspectral remote sensing images with support vector machines, IEEE Trans. Geosci. Remote Sens. 42 (8) (2004) 1778–1790.
[9] J. Li, J.M. Bioucas-Dias, A. Plaza, Spectral–spatial hyperspectral image segmentation using subspace multinomial logistic regression and markov random fields, IEEE Trans. Geosci. Remote Sens. 50 (3) (2011) 809–823.
[10] P. Ghamisi, J. Plaza, Y. Chen, J. Li, A.J. Plaza, Advanced spectral classifiers for hyperspectral images: A review, IEEE Geosci. Remote Sens. Magazine 5 (1) (2017) 8–32.
[11] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin, B.A. Johnson, Deep learning in remote sensing applications: A meta-analysis and review, ISPRS J. Photogrammetry Remote Sens. 152 (2019) 166–177.
[12] K. Tan, H. Wang, L. Chen, Q. Du, P. Du, C. Pan, Estimation of the spatial distribution of heavy metal in agricultural soils using airborne hyperspectral imaging and random forest, J. Hazardous Mater. 382 (2020) 120987.
[13] M. Dalponte, H.O. Ørka, T. Gobakken, D. Gianelle, E. Næsset, Tree species classification in boreal forests with hyperspectral data, IEEE Trans. Geosci. Remote Sens. 51 (5) (2013) 2632–2645.
[14] F. Zhou, R. Hang, Q. Liu, X. Yuan, Hyperspectral image classification using spectral-spatial lstms, Neurocomputing 328 (2019) 39–47.
[15] Y. Gu, T. Liu, X. Jia, J.A. Benediktsson, J. Chanussot, Nonlinear multiple kernel learning with multiple-structure-element extended morphological profiles for hyperspectral image classification, IEEE Trans. Geosci. Remote Sens. 54 (6) (2016) 3235–3247.
[16] P. Ghamisi, J.A. Benediktsson, M.O. Ulfarsson, Spectral-spatial classification of hyperspectral images based on hidden markov random fields, IEEE Trans. Geosci. Remote Sens. 52 (5) (2014) 2565–2574.
[17] B. Tu, X. Zhang, X. Kang, G. Zhang, J. Wang, J. Wu, Hyperspectral image classification via fusing correlation coefficient and joint sparse representation, IEEE Geosci. Remote Sens. Lett. 15 (3) (2018) 340–344.
[18] E. Pan, X. Mei, Q. Wang, Y. Ma, J. Ma, Spectral-spatial classification for hyperspectral image based on a single gru, Neurocomputing 387 (2020) 150–160.
[19] X. Kang, S. Li, J.A. Benediktsson, Feature extraction of hyperspectral images with image fusion and recursive filtering, IEEE Trans. Geosci. Remote Sens. 52 (6) (2014) 3742–3752.
[20] J.A. Benediktsson, J.A. Palmason, J.R. Sveinsson, Classification of hyperspectral data from urban areas based on extended morphological profiles, IEEE Trans. Geosci. Remote Sens. 43 (3) (2005) 480–491.
[21] X. Kang, S. Li, J.A. Benediktsson, Spectral-spatial hyperspectral image classification with edge-preserving filtering, IEEE Trans. Geosci. Remote Sens. 52 (5) (2014) 2666–2677.
[22] X. Tu, X. Shen, P. Fu, T. Wang, Q. Sun, Z. Ji, Discriminant sub-dictionary learning with adaptive multiscale superpixel representation for hyperspectral image classification, Neurocomputing 409 (2020) 131–145.
[23] B. Tu, J. Wang, X. Kang, G. Zhang, X. Ou, L. Guo, Knn-based representation of superpixels for hyperspectral image classification, IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens. 11 (11) (2018) 4032–4047.
[24] C. Xiang, W. Zou, C. Xu, Cimask: Segmenting instances by class-specific semantic feature extraction and instance-specific attribute discrimination, Neurocomputing 464 (2021) 164–174.

[25] H. Ning, B. Zhao, Y. Yuan, Semantics-consistent representation learning for remote sensing image-voice retrieval, IEEE Trans. Geosci. Remote Sens. (2021).
[26] C. Gong, J. Han, P. Zhou, D. Xu, Learning rotation-invariant and fisher discriminative convolutional neural networks for object detection, IEEE Trans. Image Process. 28 (2018) 265–278.
[27] H. Ning, X. Zheng, X. Lu, Y. Yuan, Disentangled representation learning for cross-modal biometric matching, IEEE Trans. Multimedia PP (99) (2021) 1–1.
[28] W. Zhang, X. Lu, X. Li, Similarity constrained convex nonnegative matrix factorization for hyperspectral anomaly detection, IEEE Trans. Geosci. Remote Sens. 57 (99) (2019) 4810–4822.
[29] M. Zhao, L. Yan, J. Chen, Hyperspectral image shadow compensation via cycle-consistent adversarial networks, Neurocomputing (2021).
[30] S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi, J.A. Benediktsson, Deep learning for hyperspectral image classification: An overview, IEEE Trans. Geosci. Remote Sens. 57 (9) (2019) 6690–6709.
[31] J. Yue, S. Mao, M. Li, A deep learning framework for hyperspectral image classification using spatial pyramid pooling, Remote Sens. Lett. 7 (9) (2016) 875–884.
[32] P. Zhou, J. Han, G. Cheng, B. Zhang, Learning compact and discriminative stacked autoencoder for hyperspectral image classification, IEEE Trans. Geosci. Remote Sens. 57 (7) (2019) 4823–4833.
[33] Y. Chen, X. Zhao, X. Jia, Spectral-spatial classification of hyperspectral data based on deep belief network, IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens. 8 (6) (2015) 2381–2392.
[34] L. Mou, P. Ghamisi, X.X. Zhu, Deep recurrent neural networks for hyperspectral image classification, IEEE Trans. Geosci. Remote Sens. 55 (7) (2017) 3639–3655.
[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proc. Int. Conf. Neural Inf. Process. Syst., 2014, pp. 2672–2680.
[36] H. Lee, H. Kwon, Going deeper with contextual cnn for hyperspectral image classification, IEEE Trans. Image Process. 26 (10) (2017) 4843–4855.
[37] B. Fang, Y. Li, H. Zhang, J.C.-W. Chan, Hyperspectral images classification based on dense convolutional networks with spectral-wise attention mechanism, Remote Sens. 11 (2) (2019) 159.
[38] B. Pan, X. Xu, Z. Shi, N. Zhang, H. Luo, X. Lan, Dssnet: A simple dilated semantic segmentation network for hyperspectral imagery classification, IEEE Geosci. Remote Sens. Lett. (2020) 1–5 (to be pubilshed).
[39] R.R. Devaram, D. Allegra, G. Gallo, F. Stanco, Hyperspectral image classification via convolutional neural network based on dilation layers, in, in: Int. Conf. Image Analysis Process, Springer, 2019, pp. 378–387.
[40] K. Pooja, R.R. Nidamanuri, D. Mishra, Multi-scale dilated residual convolutional neural network for hyperspectral image classification, in: Proc. IEEE 10th Workshop Hyperspectral Imag. Signal Process., Evol. Remote Sens., 2019, pp. 1–5.
[41] Y. Xu, B. Du, L. Zhang, Beyond the patchwise classification: Spectral-spatial fully convolutional networks for hyperspectral image classification, IEEE Trans. Big Data 6 (3) (2020) 492–506.
[42] G. Cheng, Z. Li, J. Han, X. Yao, L. Guo, Exploring hierarchical convolutional features for hyperspectral image classification, IEEE Trans. Geosci. Remote Sens. 56 (11) (2018) 6712–6722.
[43] X. Li, M. Ding, A. Pižurica, Deep feature fusion via two-stream convolutional neural network for hyperspectral image classification, IEEE Trans. Geosci. Remote Sens. 58 (4) (2020) 2615–2629.
[44] J. Wang, J. Li, Y. Shi, J. Lai, X. Tan, AM^{3}Net: Adaptive mutual-learning-based multimodal data fusion network, IEEE Trans. Circuits Syst. Video Technol. (2022), Early Access. DOI:10.1109/TCSVT.2022.3148257.
[45] C. Mu, Z. Guo, Y. Liu, A multi-scale and multi-level spectral-spatial feature fusion network for hyperspectral image classification, Remote Sens. 12 (1) (2020) 125.
[46] S. Sabour, N. Frosst, G.E. Hinton, Dynamic routing between capsules, in: Proc. Int. Conf. Neural Inf. Process. Syst., 2017, pp. 3856–3866.
[47] C. Xiang, L. Zhang, Y. Tang, W. Zou, C. Xu, Ms-capsnet: A novel multi-scale capsule network, IEEE Signal Proce. Lett. 25 (12) (2018) 1850–1854.
[48] C. Xiang, Z. Wang, S. Tian, J. Liao, W. Zou, C. Xu, Matrix capsule convolutional projection for deep feature learning, IEEE Signal Proce, Lett, 2020.
[49] 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.
[50] K. Zhu, Y. Chen, P. Ghamisi, X. Jia, J.A. Benediktsson, Deep convolutional capsule network for hyperspectral image spectral and spectral-spatial classification, Remote Sens. 11 (3) (2019) 223.
[51] W. Song, S. Li, L. Fang, T. Lu, Hyperspectral image classification with deep feature fusion network, IEEE Trans. Geosci. Remote Sens. 56 (6) (2018) 3173–3184.
[52] J. Yue, W. Zhao, S. Mao, H. Liu, Spectral–spatial classification of hyperspectral images using deep convolutional neural networks, Remote Sens. Lett. 6 (6) (2015) 468–477.
[53] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, Y. Wei, Deformable convolutional networks, in: Proc. IEEE Int. Conf. Comput. Vis. (ICCV), 2017.
[54] X. Zhu, H. Hu, S. Lin, J. Dai, Deformable convnets v2: More deformable, better results, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2019.
[55] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, J. Dai, Deformable detr: Deformable transformers for end-to-end object detection, 2020, arXiv preprint arXiv:2010.04159.
[56] J. Zhu, L. Fang, P. Ghamisi, Deformable convolutional neural networks for hyperspectral image classification, IEEE Geosci. Remote Sens. Lett. 15 (8) (2018) 1254–1258.
[57] J. Rajasegaran, V. Jayasundara, S. Jayasekara, H. Jayasekara, S. Seneviratne, R. Rodrigo, Deepcaps: Going deeper with capsule networks, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 10725–10733.
[58] P. Arun, K.M. Buddhiraju, A. Porwal, Capsulenet-based spatial–spectral classifier for hyperspectral images, IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens. 12 (6) (2019) 1849–1865.
[59] W. Zhang, P. Tang, L. Zhao, Remote sensing image scene classification using cnn-capsnet, Remote Sens. 11 (5) (2019) 494.
[60] Q. Xu, D.Y. Wang, B. Luo, Faster multiscale capsule network with octave convolution for hyperspectral image classification, IEEE Geosci. Remote Sens. Lett. (2020).
[61] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, G. Cottrell, Understanding convolution for semantic segmentation, in: Proc. Winter Conf. Appl. Comput. Vis. (WACV), IEEE, 2018, pp. 1451–1460.
[62] R. Moreno, F. Corona, A. Lendasse, M. Graña, L.S. Galvão, Extreme learning machines for soybean classification in remote sensing hyperspectral images, Neurocomputing 128 (2014) 207–216.

Jinping Wang received the M.S. degree in information and communication engineering from Hunan Institute of Science and Technology in 2020. She is currently working toward the Ph.D. degree in School of Intelligent Systems Engineering from Sun Yat-sen University. Her research interests include computer vision, pattern recognition, and image processing.

Xiaojun Tan received the Ph.D. degree from Sun Yat-sen University, China, in 2005. He was a lecturer with the Engineering School, Sun Yat-sen University, from 2005 to 2010, and also an associate professor, from 2010 to 2017. Since 2010, he has been a Chief Engineer with the Electric Vehicle Research Center, Dongguan-SYSU Institute. He is currently an associate professor with the School of Intelligent Systems Engineering, Sun Yat-sen University. His research interests include computer vision, control networks, smart driving, and battery management systems.

Jianhuang Lai received the M.S. degree in applied mathematics in 1989, and the Ph.D. degree in mathematics from SunYat-sen University, China, in 1999. He is currently a Professor with the School of Data and Computer Science, Sun Yat-sen University. His current research interests are in the areas of digital image processing, pattern recognition, multimedia communication, and wavelet and its applications. He has authored over 100 scientific papers in the international journals and conferences on image processing and pattern recognition, e.g., the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the IEEE TRANSACTIONS ON NEURAL NETWORKS, the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS–PARTB, Pattern Recognition, ICCV, CVPR, and ICDM.

Jun Li received the B.E. degree from Tsinghua University, the M.S. degree from Asian Institute of Technology, and the Ph.D. degree in Civil Engineering from Nagoya University. He is currently an associate professor at the School of Intelligent Systems Engineering, Sun Yat-Sen University. His research interests include system optimization and machine learning.