remote sensing

MDPI

Article

Capsule Attention Network for Hyperspectral Image Classification

Nian Wang †, Aitao Yang †, Zhigao Cui *, Yao Ding, Yuanliang Xue and Yanzhao Su

Xi'an Research Institute of High-Tech, Xi'an 710025, China; nianwang04@outlook.com (N.W.); aitaoyang10@163.com (A.Y.); dingyao.88@outlook.com (Y.D.); xyl_507@outlook.com (Y.X.); syzlhh@163.com (Y.S.)

* Correspondence: cuizg10@126.com

† These authors contributed equally to this work.

Abstract: While many neural networks have been proposed for hyperspectral image classification, current backbones cannot achieve accurate results due to the insufficient representation by scalar features and always cause a cumbersome calculation burden. To solve the problem, we propose the capsule attention network (CAN), which combines an activity vector with an attention mechanism to improve HSI classification. In particular, we consider two attention mechanisms to improve the effectiveness of the activity vectors. First, an attention-based feature extraction (AFE) module is proposed to preprocess the spectral-spatial features of HSI data, which effectively mines useful information before the generation of the activity vectors. Second, we propose a self-weighted mechanism (SWM) to distinguish the importance of different capsule convolutions, which enhances the representation of the primary activity vectors. Experiments on four well-known HSI datasets have shown our CAN surpasses state-of-the-art (SOTA) methods on three widely used metrics with a much lower computational burden.

Keywords: HSI classification; capsule network; attention-based feature extraction; self-weighted mechanism

Citation: Wang, N.; Yang, A.; Cui, Z.; Ding, Y.; Xue, Y.; Su, Y. Capsule Attention Network for Hyperspectral Image Classification. *Remote Sens.* **2024**, *16*, 4001. https://doi.org/10.3390/rs16214001

Academic Editor: Salah Bourennane

Received: 7 August 2024
Revised: 18 October 2024
Accepted: 24 October 2024
Published: 28 October 2024

1. Introduction

Hyperspectral images (HSIs) contain a wealth of spectral–spatial information within hundreds of continuous spectra, holding promise for distinguishing different land covers at a fine-grained level, particularly for those which have extremely similar spectral signatures in RGB space [1]. Therefore, HSI classification has great potential to effectively achieve a series of high-level Earth observation tasks such as mineral analysis, land cover mapping, precision agriculture, mineral exploration and military monitoring, etc.

Early studies on HSI classification utilized various machine learning-based approaches as feature extractors. Classical methods include K-nearest neighbor [2], Markov random field [3], random forest [4], support vector machine [5] and Gaussian process [6]. These methods can quickly obtain classification results but cannot ensure accuracy since the man-made feature extractor limits the data representation and fitting ability. Thanks to the great success of deep learning, many neural networks have been proposed to achieve HSI classification in an end-to-end way. Neural networks are capable of cultivating the potentially valuable information hiding in the pluralistic data of HSI, and thus have become a 'hot topic' in HSI classification [7]. For neural networks, the key problem is how to design the architecture and feature extraction in a way that automatically cultivates high-level nonlinearity features. The Convolutional Neural Network (CNN) is a partularly valuable paradigm, which extracts features by stacking multiple convolutions that have a local receptive field. In an early work, 1-D CNN [8] was proposed to classify HSI by using the spectral signature while ignoring the spatial relation. Subsequently, 2-D CNN was proposed to effectively extract the abundant neighbor information of HSI. Moreover, 3-D CNN [9] can extract features by regarding HSI as various cubes, which takes the spatial

information and spectral signature into account simultaneously. Since HSI itself is 3-D data, 3-D CNN outperforms 1-D CNN and 2-D CNN in most cases. Although CNNs have a powerful ability to extract spatially structural and contextual information from HSI, this only works in short-range spatial relation building, and otherwise tends to cause pepper noise [10]. Some recent works [11–13] have sought to solve the problem by designing a network architecture and attention mechanism. Although some progress has been made, encoding the local information by scalar features limits the representation ability and causes information loss during the propagation of layers. Some methods, like residual network [14], dense network [15] and frequency combined network [16], can suppress the problem, but they cannot enable CNN-based models to achieve extremely accurate results and always result in a heavy computation burden [17].

Some recent works have helped improve HSI classification performance by designing particular network architectures, including autoencoders (AEs) [18,19], recurrent neural networks (RNNs) [20], graph convolutional networks (GCNs) [21–23], Transformers (TFs) [24–26], and Mamba [27]. Chen et al. [19] employed stacked AEs to semantically extract spatial–spectral features by considering the local contextual information of HSI. Since it is an unsupervised paradigm, the AE-based method cannot ensure high classification accuracy. Hang et al. [20] designed a cascaded RNN for HSI classification by taking advantage of RNNs to effectively model the sequential relations of neighboring spectral bands. However, RNN can only capture the sequences in short- and middle-dependencies. More recently, more attention has been paid to GCN and TF for HSI classification. GCN regards the pixels as vertices and models the relations of vertices as a graph structure. Therefore, in contrast to GNN, GCN has natural merit for capturing long-range spatial relations [21]. By embedding superpixel segmentation [28], this advantage is further enhanced, avoiding local pepper noise [29]. As a strong substitute for RNN, TF [30] has a powerful ability to build long-term dependencies using a self-attention technique, which has encouraged the development of a large number of TF-based HSI classification models.

However, GCN, TF and Mamba have inevitable limitations due to their intrinsic mechanisms. Although GCN extracts long-range spatial information that ensures the accuracy of non-adjacent patches, the propagation of graph data is still a challenging problem, which restricts the number of GCN layers used and limits deep semantic feature building [31,32]. For TF, it has the merit of building long-range dependency of tokens, and thus has shown huge success in dealing with large-scale datasets in natural language processing [33]. Although TF can be directly used for HSI to build a spectral sequence, its performance is limited by the length of tokens due to relatively fewer "inductive biases" [34]. In other words, in contrast to CNN, TF only builds 2-D information in the MLP layer of Encoder, whose effectiveness is mainly achieved by the self-attention mechanism used [35]. If the token sequence is small, TF will achieve relatively poor performance [36]. Therefore, the performance of TF-based HSI classification methods is limited since the HSI used is generally small- or medium-scale, in which the spectral signature only provides hundreds of tokens, far fewer than TF-based NLP models. On the other hand, although Mamba-based work [27] alleviated the memory burden by using a State-Space Model (SSM) to replace the self-attention mechanism of TF, the classification performance is still limited by the length of the spectral signature.

Many recent works have used composite models to solve the limitations of the current backbones. By "composite" is meant that at least two backbones are combined to compensate mutually for the limitations , thus providing a more exact representation to achieve high-quality classification. For example, CEGCN [37] incorporates the advantages of both CNN and GCN, to jointly cultivate pixel-level and superpixel-level spatial-spectral features to enhance feature representation. WFCG [38] involves a weighted feature fusion model with an attention mechanism, which combines CNN with a graph attention network for HSI classification. Moreover, AMGCFN [39] utilizes a novel attention-based fusion network, which aggregates rich information by merging multihop GCN and multiscale CNN. Moreover, GTFN [40] explores the combination of GCN and TF. By using GCN to

provide long-range spatial information, GTFN achieves SOTA performance after using TF for classification. A further approach, DSNet [41], enhances classification performance by introducing a deep AE unmixing architecture to the CNN. The above composite backbones have demonstrated performance improvement resulting from the synergistic effect of the two backbones. However, simply combining two kinds of backbones with a designed fusion strategy only solves the problem of insufficient representation and propagation to a limited extent but results in a huge computational burden.

In this paper, we resort to a more abstract feature (activity vector) and rethink HSI classification with a capsule network (CapsNet) [42]. As shown in Figure 1, the pipeline of CapsNet can be divided into three steps termed ConvUnit, PrimaryCap and ClassCap, For standard CapsNet, the ConvUnit contains one convolution, which extracts preliminary features from the data input and sends them to PrimaryCap. PrimaryCap consists of multiple convolutional capsule layers, which are used to yield primary activity vectors. These primary activity vectors are adaptively activated in the ClassCap using a dynamic routing mechanism, and thus obtain the class activity vectors for classification. Some recent works have explored HSI classification based on CapsNet. CNet [43] was the first CapsNet-based model for HSI classification. Subsequently, 3DCNet [44] was proposed, which splits HSI into diverse 3-D cubes as the input to CapsNet to solve the classification problem with limited training samples. More recently, some attention mechanisms have been used to improve the capsule network. For example, SA-CNet [45] encountered a limitation of the ConvUnit step, i.e., a single convolution cannot effectively address the spatial information. Therefore, a correlation matrix with trainable cosine distance was used to assign varying weights for the different patches of HSI. Moreover, ATT-CNet [46] addressed the HSI data using a light-weight channel-wise attention mechanism. It incorporates self-weighted dynamic routing by introducing the self-attention mechanism of Transformer into the activation of the activity vectors. By contrast, our proposed CAN seeks to achieve SOTA classification performance using low-level calculations, and has three obvious differences from recent attention-based capsule networks. First, we drop the spectral signatures by PCA rather than weight them, thus effectively reducing the feature dimension of HSI and improving efficiency. Second, to ensure effectiveness, we design a light-weight module (termed AFE) without distance estimation to adaptively weight each pixel. Moreover, we observed that treating different capsule convolutions equally limits the representation of primary activity vectors. Therefore, we innovatively propose to produce more representative primary activity vectors by adaptively weighting the capsule convolutions during the PrimaryCap step.

The main contributions of this research are as follows:

- We propose CAN for HSI classification with two attention components, termed AFE and SWM, which improves the representation of primary activity vectors by, respectively, weighting the pixel-wise features and capsule convolutions in an adaptive way. In contrast to standard CapsNet, our CAN only adds three more convolutions due to the used AFE and SWM, causing insignificant parameter increment. However, benefiting from these attention mechanisms, it can run on an extremely low spectral dimension (with low calculations) of HSI data while achieving much higher classification results.
- We propose AFE block to simultaneously mine spectral–spatial features with an efficient adaptive pixel weighting mechanism, which ensures the ability to gather useful information from HSI data.
- We propose SWM to distinguish the importance of different capsule convolutions, which effectively enhances the representation of primary activity vectors and thus benefits classification performance.
- Experiments on four well-known HSI datasets show our CAN surpasses SOTA methods. Moreover, it is shown that our CAN performs much more efficiently than other methods with an extremely lower computational burden.

The rest of this paper is organized as follows: Section 2 analyzes the limitations of CNN and introduces recent CapsNet based HSI classification works. Section 3 introduces CAN. Section 4 shows the experiments to verify the effectiveness. Section 5 concludes the paper and presents some outlooks for the future.

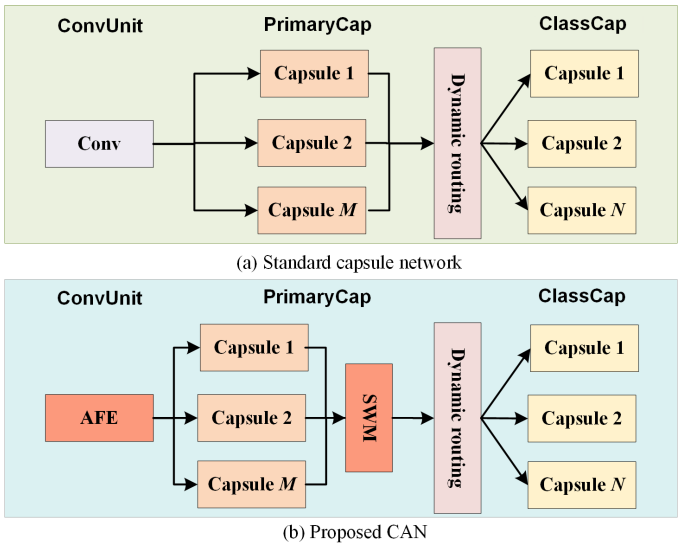

Figure 1. A graphic illustration to show the objectives. (**a**) The architecture of standard CapsNet. (**b**) The architecture of the proposed CAN, which provides more rich features in the ConvUnit step and takes the importance of capsules into account.

2. Related Work

2.1. Limitations of CNN for HSI Classification

CNN, as the classical neural network, is a pixel-wise method that extracts high-level semantic representation by stacking diverse convolutions to build the mapping relation. Formally, let the input HSI cube be $\boldsymbol{X} \in \mathbb{R}^{H\times W\times C}$; each convolution of the l-th layer computes the dot product between the input and weight matrix $\boldsymbol{W} \in \mathbb{R}^{H\times W\times C}$ as

$$\begin{aligned} f_{i,j,t}^{(l)} &= \left(\boldsymbol{X}^{(l)} * \boldsymbol{W}^{(l)}\right)_{i,j,t} \\ &= \sum_{\hat{i}=0}^{k-1}\sum_{\hat{j}=0}^{k-1}\sum_{\hat{t}=0}^{q-1} x_{(i\cdot s+\hat{i}),(j\cdot s+\hat{j}),(t\cdot s+\hat{t})}^{(l)} \cdot w_{\hat{i},\hat{j},\hat{t}}^{(l)} + b^{(l)}, \end{aligned} \tag{1}$$

where $k \times k \times q$ denotes the kernel size and s denotes the stride. $x_{\hat{i},\hat{j},\hat{t}}^{(l)}$ and $w_{\hat{i},\hat{j},\hat{t}}^{(l)}$ denote the input data and the weight of element $(\hat{i},\hat{j},\hat{t})$. $f_{\hat{i},\hat{j},\hat{t}}^{(l)}$ is the corresponding output after convolution. $b^{(l)}$ is the bias. Therefore, the obtained $\boldsymbol{F}^{(l)}$ will be an array of scalar values.

The size of the kernel and stride are essential for convolution. The kernel is a statistical property of the image, which can be considered as a stationary reflection of the pixel area, where the pixels are equally distributed with corresponding positions. The stride denotes the scan interval of the kernel. Therefore, convolution in essence numeralizes the local information of the input, and thus deeper convolution has a larger receptive field, which extracts more semantic features [47]. As the representation limitation of scalar features, CNN always needs a multiscale and deep network with a large number of convolutions and nonlinear layers to ensure the effectiveness, which creates difficulties for the feature propagation between long-range layers. Although some classical frameworks like ResNet or DenseNet ensure effective feature flow, the huge calculation burden still cannot be met on computing-constrained devices. More importantly, the limited representation of scalar presents a bottleneck to HSI classification.

Although extracting the spatial and spectral information simultaneously, 3-D CNN [9] still cannot achieve exact classification. As can be seen in Figure 2, 3-D CNN causes much pepper noise within the area that has obvious context relations. This error stems from the insufficient representation of the scalar feature for complex and mixed HSI information. Fortunately, it is easy to address such limitations by using CapsNet. Different from CNN, CapsNet establishes a vector feature transmission mechanism and outputs a series of class activity vectors, where the number of vectors strictly equals the number of land cover types. Therefore, instead of building pixel mapping in scalar form (for CNN), CapsNet aims to find a specific entity by representation of the activity vector, where the length of the vector denotes the likelihood that the current pixel belongs to this entity. The activity vector provides a more exact representation since it covers multi-dimensional features, where each dimensionality focuses on certain properties of the entity. These properties can include any type of parameter, such as position, orientation, size, hue, albedo, texture, etc. [42], which provides sufficient information to consider how the pixels are spatially arranged.

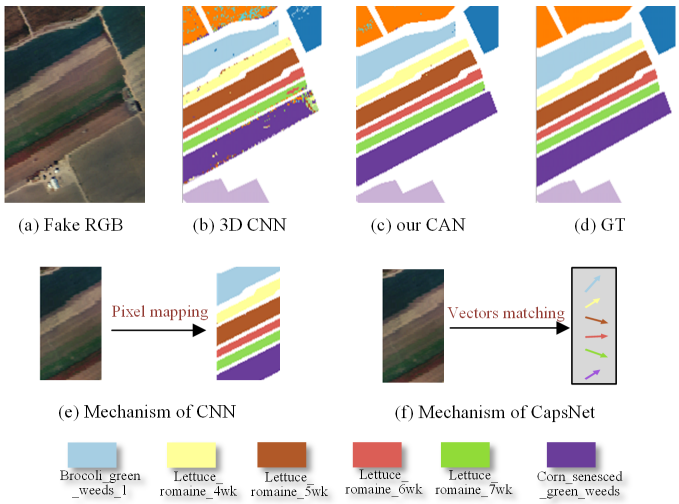

Figure 2. A graphic illustration to show the internal mechanism of CNN and CapsNet. Due to the pixel mapping achieved by the scalar feature, 3-D CNN creates much pepper noise. In contrast, CapsNet finds a series of activity vectors, where each vector represents one kind of entity in the input while the vector length determines the confidence. Since each vector covers rich information on the diverse properties of the entity with a controllable vector dimension, CapsNet yields a more abstract and exact representation pattern than CNN. In this way, our CAN obtains much better results that are very similar to GT.

2.2. CapsNet for HSI Classification

CapsNet provides a more abstract feature presentation for HSI data, and can achieve SOTA classification with a lower parameter and calculation burden than other backbones. There are some HSI classification models based on CapsNet. For example, an early work, CNet [43], extended CapsNet for HSI classification. 3DCNet [44] was proposed to split HSI into diverse 3-D cubes as the input and included a three-dimensional dynamic routing algorithm to exploit the spectral–spatial features during the ClassCap step. Moreover, some recent works have introduced attention mechanisms to improve CapsNet-based HSI classification. SA-CNet [45] includes a spatial attention mechanism in the ConvUnit step, where a correlation matrix with a trainable cosine distance function is used to assign varying weights for different cubes of HSI. ATT-CNet [46] improved CapsNet-based HSI classification models by using two attention mechanisms. First, in the ConvUnit step, it deals with the HSI data using a light-weight channel-wise attention mechanism, which highlights the useful spectral information. In the Classcap step, ATT-CNet uses the self-attention mechanism of Transformer for dynamic routing, which improves the effectiveness of some useful primary activity vectors. More recently, TSCCN [48] was proposed involving

a two-stream spectral-spatial convolutional capsule network, where an efficient structural information mining module (SIM) is designed to learn complementary cross-channel attention between two streams. Our proposed CAN aims to achieve SOTA classification with limited calculation by introducing a reasonable attention module into the ConvUnit and PrimaryCap steps, which is different from recent HSI classification works that combine an attention mechanism with CapsNet. In the ConvUnit step, different from ATT-CNet and SA-CNet, we first use PCA to drop the spectral dimension to one-fifth of the HSI data and then design a light-weight spatial attention mechanism (not relying on distance calculation like SA-CNet) to weight the HSI cubes. Moreover, this is the very first work to consider the attention of capsule convolutions during the generation of the primary activity vectors. The unique attention mechanism enables our CAN to achieve SOTA classification with far fewer calculations by only using one-fifth of the spectral dimensions of HSI data.

3. CAN

The pipeline of CAN can be divided into four steps termed Preprocessing, ConvUnit, PrimaryCap and ClassCap. As can be seen in Figure 3, the Preprocessing step extracts spectral-spatial information from HSI. Assuming the dimension of HSI is $H \times W \times B$, where H, W and B denote the values of height, width and band, respectively. First, we randomly extract a spectral-spatial cube with a patch size of 7. Afterwards, inspired by [19,24], we drop the spectral band from B to C using a classic dimensionality reduction method, PCA. By projecting into orthogonal space, PCA adaptively finds valuable spectral features (i.e., principal components). In our experiments, we set $C = 0.2B$, which reduces redundant information to improve efficiency while maintaining effectiveness. Moreover, the remaining three steps are the standard pipeline of CapsNet. However, we propose AFE for the ConvUnit step to weight the different pixels in an adaptive way. Moreover, a novel attention mechanism is designed in the PrimaryCap step to automatically distinguish the importance of capsule convolutions. We next detail the three steps.

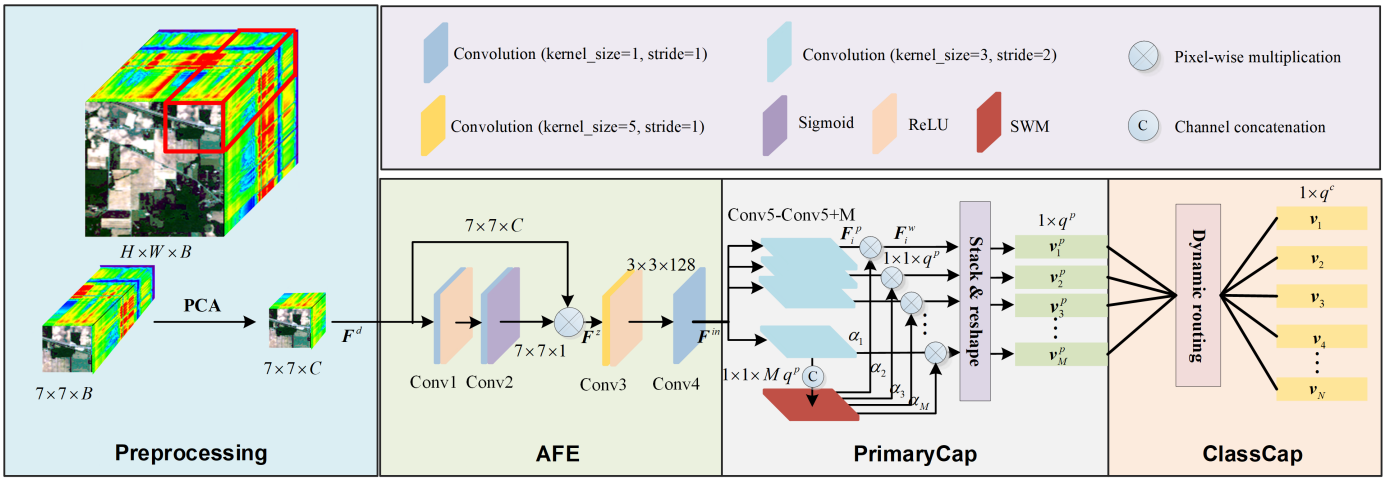

Figure 3. The overall framework of the proposed CAN.

3.1. AFE

Standard CapsNet utilizes a single convolution to process the input data, which uses all information equally and fails to gather useful features. Therefore, we propose AFE, a simple but effective module to quickly obtain high-quality features. Attention mechanisms are a widely studied topic in recent works [12,24,45,46]. The famous Squeeze-and-Excitation Network [49] (SENet) is in essence a channel-wise attention, which has been verified to be effective in improving the classification task. Moreover, CBAM [50] (Convolutional Block Attention Module) and DBDA [12] (Double-Branch Dual-Attention) were proposed to benefit from channel- and spatial-wise attention, which fuse the weighted features in a cascaded and parallel way, respectively. The proposed AFE is different from the above

works since it only uses spatial-wise attention, and no channel-wise attention is used. This design is because PCA discards most bands, which means the remaining spectral bands are all important and adding channel-wise attention only achieves a quite limited gain. Concretely, the output of prepossessing is first sent to a convolution (kernel size 1, stride 1) with the ReLU function to obtain the nonlinearity features. After that, another convolution (kernel size 1, stride 1) with the sigmoid function is used to yield a spatial-wise attention map by dropping the channel dimension to 1. This process can be mathematically defined as

$$\boldsymbol{M} = \text{Sigmoid}\Big(\text{Conv2}\Big(\text{ReLU}\Big(\text{Conv1}\Big(\boldsymbol{F}^d\Big)\Big)\Big)\Big), \tag{2}$$

where $\boldsymbol{F}^d \in \mathbb{R}^{7\times7\times C}$ denotes the output of preprocessing. Conv1 and Conv2 denote the first and second convolution, respectively. $\boldsymbol{M} \in \mathbb{R}^{7\times7\times1}$ denotes the adaptively learned pixel-wise attention map. As can be seen from the angle of spatial attention, the proposed AFE is also different from CBAM and DBDA. First, in contrast to the spatial attention in CBAM, the proposed AFE obtains the spatial attention map by two convolutions with a ReLU function rather than by one convolution. The addition of one convolution with the ReLU function enhances the ability to build nonlinear features, making the attention mechanism more efficient with negligible computational increment. Moreover, in contrast to the spatial attention in DBDA, the proposed AFE only learns the attention map in one branch rather than in three branches, which ensures efficiency. In particular, the convolution's kernel size of AFE is pixel-wise (1×1) rather than having a large kernel size (7×7 and 9×9) as used in CBAM and DBDA, respectively, which fits in pixel classification. After learning the attention map, we obtain the weighted features as

$$\boldsymbol{F}^z = \boldsymbol{M} \otimes \boldsymbol{F}^d, \tag{3}$$

where $\otimes$ denotes element-wise multiplication. $\boldsymbol{F}^z$ denotes the weighed features by pixel-wise attention. Hereto, we have obtained more representative features by distinguishing the importance of each pixel in an automatic way.

After spatial-wise attention, it is necessary to conduct nonlinearity mapping again to extract the high-level features and adjust the scale. To be specific, a convolution (kernel size 5, stride 1) with ReLU function is used to change the patch size from 7×7 to 3×3. To suppress the loss of spatial information, we correspondingly increase the channel dimension to 128. Finally, these features are sent to another convolution (kernel size 1, stride 1) to accomplish the feature preparation for the PrimaryCap step.

3.2. PrimaryCap

The PrimaryCap step yields primary activity vectors, which are the lowest level of multi-dimensional entities. Here, the concept of entity can be understood as land covers (i.e., classification objects) with multi-dimensional parts of interest that are expressed as the instantiation parameters [42]. These instantiation parameters are meaningful; for HSI, they represent the associated properties of the corresponding land-cover type. Therefore, in this sense, the generation of the capsule can be interpreted as being opposite to the rendering process in computer graphics, where the pattern is obtained by applying rendering to an object with corresponding instantiation parameters.

The PrimaryCap is an M-dimensionality convolutional capsule layer with q^p channels, i.e., it contains M capsule convolutions with the kernel size being 3 and the stride being 2. Therefore, in total, PrimaryCap obtains M capsules, where each output is a q^p dimensionality vector. This process can be mathematically defined as

$$\boldsymbol{F}_i^p = \text{conv}i(\boldsymbol{F}^{in}), \tag{4}$$

$$\boldsymbol{F}^p = \text{stack}(\boldsymbol{F}_i^p), \tag{5}$$

$$\boldsymbol{s}^p = \text{reshape}(\boldsymbol{F}^p), \tag{6}$$

where conv$i, i \in 5, \cdots, 5+M$ denote the M capsule convolutions. $\boldsymbol{F}_i^p \in \mathbb{R}^{q^p \times 1 \times 1}$ are the extracted features. Considering the outputs of all the M capsules, we stack them in dimension 0 to obtain $\boldsymbol{F}^p \in \mathbb{R}^{M \times q^p \times 1 \times 1}$. Afterwards, $\boldsymbol{F}^p$ is reshaped as $\boldsymbol{s}^p \in \mathbb{R}^{M \times q^p}$, where M denotes the number of activity vectors and q^p denotes the vector dimension. From another angle, M is also the number of parallel convolutions and q^p denotes the corresponding number of output channels. In our experiments, we set $M = 8$ and $q^p = 32$.

After obtaining the vectors, a squash operation should be conducted, which serves as the nonlinearity mapping operation of the vector version. The mathematical definition is

$$\boldsymbol{v}_i^p = \frac{\left\|\boldsymbol{s}_i^p\right\|^2}{1+\left\|\boldsymbol{s}_i^p\right\|^2} \frac{\boldsymbol{s}_i^p}{\left\|\boldsymbol{s}_i^p\right\|}, \tag{7}$$

where $\boldsymbol{v}_i^p$ denotes the obtained primary activity vectors after the squash operation. From another angle, Equation (7) can also be regarded as a weight operation of the vector. First, it is easy to see $\frac{\boldsymbol{s}_i^p}{\|\boldsymbol{s}_i^p\|}$ denotes the normalized vector of $\boldsymbol{s}_i^p$. Based on this, the larger the norm of $\boldsymbol{s}_i^p$, the more the norm of $\boldsymbol{v}_i^p$ gets closer to 1. While the less the norm of $\boldsymbol{s}_i^p$, the more the norm of $\boldsymbol{v}_i^p$ gets closer to 0. This shows Equation (7) is meaningful, which makes these vectors well recognized.

Equations (4)–(7) show the pipelines of the standard CapsNet. It is easy to see that the learned primary activity vectors are closely related to the M capsules. A natural idea is to automatically give the importance of these capsules and thus improve the representation of the primary activity vectors. Therefore, a simple but effective method, termed SWM, is designed to adaptively yield the weights. Concretely, we concatenate all the $\boldsymbol{F}_i^p$ in the channel dimension and then send them into SWM to generate the weights for all the capsules by a gated convolution. This process can be mathematically denoted as

$$\boldsymbol{F}^c = \text{concat}(\boldsymbol{F}_i^p), \tag{8}$$

$$\alpha_i = \text{gated\_conv}(\boldsymbol{F}^c), \tag{9}$$

where concat$(\cdot)$ denotes the concatenation operation in channel dimensionality. gated_conv is a convolution with kernel size of 3, with stride of 1, and thus it only changes the size of the channels. The input channel is $8q^p$ and the output channel is M, which can be regarded as the adaptive weight maps of M capsules. Formally, the weighted features can be defined as

$$\boldsymbol{F}_i^w = \alpha_i \otimes \boldsymbol{F}_i^p, \tag{10}$$

where $\boldsymbol{F}_w^i$ are the weighted features. Afterwards, Equations (5) and (6) are solved sequentially to obtain high-quality activity vectors. These activity vectors are further squashed by Equation (7) to obtain primary activity vectors for the ClassCap step.

It is easy to see the proposed SWM is meaningful. First, since the input of gated_conv is the concatenated features of all the capsules, it is theoretically explainable that the output M maps can give reasonable weights for the results of different capsules. Moreover, the SWM has two obvious advantages. On the one hand, the weights are data driven, free from manmade priors. On the other hand, in contrast to the standard capsule network, it only introduces one more convolution, and thus the added calculation quantity can be completely ignored.

3.3. ClassCap

The ClassCap step describes an information propagation rule between vectors. Following [42], we use dynamic routing to build such relations. As shown in Figure 4, the primary activity vectors are first transformed by a projection matrix $\boldsymbol{W}$ as

$$\boldsymbol{u}_i = \boldsymbol{W}_i \boldsymbol{v}_i^p, \tag{11}$$

where $\boldsymbol{u}_i$ denotes the projected vectors. These vectors are weighted by different weights and fused by element-wise addition. The process can be mathematically defined as

$$\boldsymbol{s}_n = \sum_{m=1}^{M} \boldsymbol{u}_m;\ \boldsymbol{u}_m = \boldsymbol{c}_m \boldsymbol{u}_i, \tag{12}$$

where $\boldsymbol{s}_n$ denotes the n-th weighed vector, where n denotes the number of class activity vectors (also the number of land cover types). Afterwards, the squashing operation (Equation (7)) is conducted on $\boldsymbol{s}_n$ to obtain the class activity vector $\boldsymbol{v}_n$. Assuming the total iteration number of dynamic routing is T, the weight is adaptively updated by

$$\boldsymbol{c}_m^{t+1} = \boldsymbol{c}_m^t + \boldsymbol{v}_n^t \cdot \boldsymbol{u}_m, \tag{13}$$

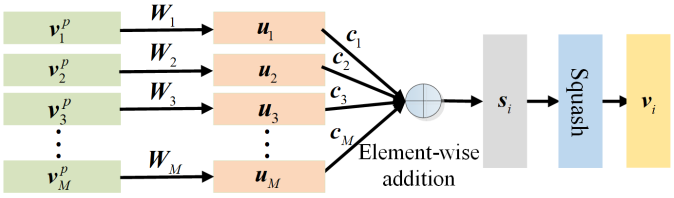

Figure 4. Graphical illustration of dynamic routing.

In Equation (13), $\boldsymbol{c}^{t+1}$ denotes the weight at the $t+1$ epoch, where $t \in 1, 2, \ldots, T$ denotes the index of iteration. According to [42], we set $T = 3$ in the experiments. $\cdot$ denotes the inner-product, which is used to estimate the similarity of the two vectors. Since $\boldsymbol{v}_n^t$ denotes the fused vector, the weighting mechanism is meaningful since the vector input with a more positive contribution will be incrementally assigned a larger weight. The whole process is summarized in Algorithm 1.

Algorithm 1: Dynamic routing.

Input: $v_1^p, v_2^p, \cdots, v_M^p$;
Calculate projected vectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_M$;
Initialize $\boldsymbol{c}_1^0 = 0, \boldsymbol{c}_2^0 = 0, \cdots, \boldsymbol{c}_{M=0}^0$;
For $t = 1$ to T **do**
(1) $\boldsymbol{c}^t, \boldsymbol{c}_2^t, \cdots, \boldsymbol{c}_M^t = \text{softmax}(\boldsymbol{c}_1^{t-1}, \boldsymbol{c}_2^{t-1}, \cdots, \boldsymbol{c}_M^{t-1})$;
(2) $\boldsymbol{s}_n = \boldsymbol{c}_1^t \boldsymbol{u}_1 + \boldsymbol{c}_2^t \boldsymbol{u}_2 + \cdots + \boldsymbol{c}_M^t \boldsymbol{u}_M$;
(3) $\boldsymbol{v}_n = \text{Squash}(\boldsymbol{s}_n)$;
(4) $\boldsymbol{c}_m^{t+1} = \boldsymbol{c}_m^t + \boldsymbol{v}_n^t \cdot \boldsymbol{u}_m$;

3.4. Loss Function

The margin loss and reconstruction loss are used to train the proposed CAN.

$$L = L_m + \theta L_r, \tag{14}$$

where L denotes the overall loss, and L_m and L_r denote the margin loss and reconstruction loss, respectively. θ is a trade-off for the loss, which is suggested as 0.0005 [42].

(1) Margin Loss: For a CapsNet, it is expected that the class capsule for entity c will have a long instantiation vector if, and only if, that entity is present in the current input. Therefore, the margin loss is used as

$$\begin{aligned} L_c =& T_c \max\left(0, \beta^+ - \|\boldsymbol{v}_c\|\right)^2 \\ &+ \lambda(1 - T_c) \max\left(0, \|\boldsymbol{v}_c\| - \beta^-\right)^2, \end{aligned} \tag{15}$$

where $T_c = 1$ if the current input is assigned to class c with margin coefficients $\beta^+ = 0.9$ and $\beta^- = 0.1$. λ stops the initial learning from shrinking the lengths of the activity vectors of all the class capsules, which is suggested as 0.5 [42].

(2) Reconstruction Loss: The reconstruction loss is used as a regularization to ensure exact pattern representation for each entity. If the current entity belongs to class capsule c, we first mask the outputs of all other class capsules and then only use the result of capsule c to reconstruct the input. The reconstruction network is composed by three fully connected layers as follows:

$$\hat{\boldsymbol{F}}_{in} = \mathrm{FC3}(\mathrm{FC2}(\mathrm{FC1}(\boldsymbol{v}_c))), \tag{16}$$

where FC denotes the fully connected layers. FC1 and FC2 are followed by a ReLU function; while FC3 is followed by a sigmoid function. The reconstruction loss is defined as

$$L_r = ||\boldsymbol{F}_{in} - \hat{\boldsymbol{F}}_{in}||_2^2, \tag{17}$$

where $||\cdot||_2^2$ denotes l_2 loss.

4. Experiments and Discussion

Section 4.1 introduces the used hyperspectral datasets. Section 4.2 introduces the experimental setting and implementation details. Section 4.3 shows the experimental results. Section 4.4 presents the ablation study.

4.1. Data Description

During the experiments, well-known hyperspectral datasets are used. They are Indian Pines (IP), Salina (SA), Pavia University (PU), UH2013 and UH2018.

The IP dataset is captured from scenes in northwestern Indiana by the AVIRIS sensor, and contains 145 × 145 pixels and 224 bands. By removing some interference bands, 200 bands remain. This dataset contains 10,249 labeled pixels with 16 feature categories in total. In our experiments, 435 samples are selected as training data and 9814 samples as testing data. The division of samples is shown in Table 1. As can be seen, our division strategy may result in the training samples being less than the testing samples (e.g., Class 9) due to the extremely uneven distribution of IP. The classification results of this class can verify the performance where there are sufficient training samples.

Table 1. Category information of IP dataset.

ClassID	Class Name	Training	Testing
1	Alfalfa	15	31
2	Corn-notill	30	1398
3	Corn-mintill	30	800
4	Corn	30	207
5	Grass-pasture	30	453
6	Grass-trees	30	700
7	Grass-pasture-mowed	15	13
8	Hay-windrowed	30	448
9	Oats	15	5
10	Soybeans-notill	30	942
11	Soybean-mintill	30	2425
12	Soybean-clean	30	563
13	Wheat	30	175
14	Woods	30	1235
15	Buildings-grass-trees-drivers	30	356
16	Stone-steel-towers	30	63
	Total	435	9814

The SA dataset is captured from scenes in the Salinas Valley, California by the AVIRIS sensor, which contains 512 × 217 pixels and 224 bands. By removing some interference bands, 204 bands remain. This dataset contains 54,129 labeled pixels with 16 feature categories in total. In our experiments, 480 samples are selected as training data and 53,649 samples as testing data. The division of samples is shown in Table 2.

The PU dataset is captured from scenes from Pavia university, and contains 610 × 340 pixels and 115 bands. By removing some interference bands, 103 bands remain with 9 feature categories and 42,776 labeled pixels. In our experiments, 270 samples are selected as training data and 42,506 samples as testing data. The division of samples is shown in Table 3.

The UH2013 dataset has a size of 349 × 1905, with 144 effective bands, 15 land cover classes, and a spatial resolution of 2.5 m. In our experiments, 450 samples are selected as training data and 14,579 samples as testing data. The division of samples is shown in Table 4.

UH2018 is a large-scale dataset, which has a size of 601 × 2384 with 50 effective bands and 20 land covers. In our experiments, 600 samples are selected as training data and 504,256 samples as testing data. The division of samples is shown in Table 5.

Table 2. Category information of SA dataset.

ClassID	Class Name	Training	Testing
1	Brocoli_green_weeds_1	30	1979
2	Brocoli_green_weeds_2	30	3696
3	Fallow	30	1946
4	Fallow_rough_plow	30	1364
5	Fallow_smooth	30	2648
6	Stubble	30	3929
7	Celery	30	3549
8	Grapes_untrained	30	11,241
9	Soil_vinyard_develop	30	6173
10	Corn_senesced_green_weeds	30	3248
11	Lettuce_romaine_4wk	30	1038
12	Lettuce_romaine_5wk	30	1897
13	Lettuce_romaine_6wk	30	886
14	Lettuce_romaine_7wk	30	1040
15	Vinyard_untrained	30	7238
16	Vinyard_vertical_trellis	30	1777
	Total	480	53,649

Table 3. Category information of PU dataset.

ClassID	Class Name	Training	Testing
1	Asphalt	30	6601
2	Meadows	30	18,619
3	Gravel	30	2069
4	Trees	30	3034
5	Metal Sheets	30	1315
6	Bare-soil	30	4999
7	Bitumen	30	1300
8	Bricks	30	3652
9	Shadows	30	917
	Total	270	42,506

Table 4. Category information of UH2013 dataset.

ClassID	Class Name	Training	Testing
1	Healthy grass	30	1221
2	Stressed grass	30	1224
3	Synthetic grass	30	667
4	Trees	30	1214
5	Soil	30	1212
6	Water	30	295
7	Residential	30	1238
8	Commercial	30	1214
9	Road	30	1222
10	Highway	30	1197
11	Railway	30	1205
12	Parking Lot 1	30	1203
13	Parking Lot 2	30	439
14	Tennis Court	30	398
15	Running Track	30	630
	Total	450	14,579

Table 5. Category information of UH2018 dataset.

ClassID	Class Name	Training	Testing
1	Healthy grass	30	9769
2	Stressed grass	30	32,472
3	Artificial turf	30	654
4	Evergreen trees	30	13,565
5	Deciduous trees	30	4991
6	Bare earth	30	4486
7	Water	30	236
8	Residential building	30	39,742
9	Non-residential building	30	223,722
10	Road	30	45,836
11	Sidewalks	30	33,999
12	Crosswalks	30	1488
13	Major thoroughfares	30	46,318
14	Highways	30	9835
15	Railways	30	6907
16	Paved parking lots	30	11,470
17	Unpaved parking lots	30	116
18	Cars	30	6517
19	Trains	30	5339
20	Stadium seats	30	6794
	Total	600	504,256

4.2. Experiment Settings

(1) Implementation Details of CAN: Our CAN was implemented on the PyTorch 1.12 framework using a PC with one RTX 2080Ti GPU. For the training of CAN, we set the patch size as 7×7, and used PCA to drop the band dimensionality as $1/5$ of the HSI input. The total epoch is set at 300 with an initial learning rate of 5×10^{-4}. After each ten epochs, the learning rate is decreased to nine-tenths of the latest learning rate. The batch size is set as 128, and the Adam [51] optimizer is utilized.

(2) Comparison With the SOTA Backbones: We compare our CAN with 13 representative deep learning HSI classification methods to comprehensively verify its effectiveness. These methods are as follows: the CNN-based methods, 3-D CNN [9], RSSAN [11], DBDA [12]; the GCN-based method MDGCN [22]; the capsule-based methods CNet [43], ATT-CNet [46], SA-CNet [45]; the TF-based methods SSFTT [24], GAHT [25], SF [26]; the Mamba-based work MHSI [27] and the composite methods WFCG [38], AMGCFN [39],

GTFN [40], and DSNet [41]. Among the composite methods, WFCG and AMGCFN combine the merits of CNN and GCN to improve HSI classification. GTFN combines the merits of GCN and TF to enhance classification performance. For all the comparative methods, we directly use the published model by the authors with the suggested parameter settings and training epochs. All the experiments were conducted on the same PC with one RTX 2080Ti GPU.

(3) Evaluation Metrics: Three widely used metrics, the Overall classification Accuracy (OA), the Average classification Accuracy (AA), and the Kappa coefficient are chosen to quantitatively verify the performance. Moreover, qualitative analysis is simultaneously carried out by visualizing the classification maps.

4.3. Experiment and Analysis

(1) Quantitative Analysis: Tables 6–9 show the quantitative results of the IP, SA, PU and UH2013 datasets, respectively. For the CNN-based methods, 3-D CNN obtains relatively lower results. By combining spectral–spatial attention, RSSAN improves the performance to some extent, but the results are still much lower than for other works. It is interesting that by designing a double-branch dual-attention mechanism, DBDA achieves competitive results. The results show that CNN-based methods are limited by the inadequate representation of scalar features, whose effectiveness relies on the design of the network architecture and attention mechanism. By contrast, MDGCN utilizes multi-scale dynamic graph convolution to improve classification, which achieves results that are near to those for CNN-based DBDA. Moreover, it seems that the performance of TF-based methods is limited to some extent. First, like 3-D CNN, SF simply uses the spectral–spatial information for Transformer, which obtains much lower results than the same type methods SSFTT and GAHT. For SSFTT, the performance improvement mainly results from using the Gaussian weighted feature tokenizer. For GAHT, a more complex group-aware hierarchical transformer is used, and the performance is usually better than SSFTT. All the results show that TF-based methods hit the same bottleneck as the CNN-based method, i.e., the baseline cannot achieve accurate classification due to insufficient feature representation and thus it relies on a well-designed architecture or attention mechanism. Examining the results of the composite models WFCG, AMGCFN, GTFN and DSNet, we can see that the metrics are generally higher than for those that only use one backbone. Interestingly, as can be seen in the 9th class of the IP dataset, we find that all the methods except WFCG achieve 100% accuracy. This demonstrates that WFCG is unstable and cannot ensure high accuracy under sufficient training samples. Moreover, combining GCN with TF (GTFN) achieves significant advantages when compared with methods that incorporate CNN and TF (WFCG and AMGCFN). However, by only using the standard CapsNet, CNet achieves SOTA results. Moreover, the attention-based CapsNet ATTCNet and SA-CNet obtain much better results than all the recent attention-based CNN methods, TF or composite models. The results show that CapsNet better fits HSI classification by using high-level abstract vector features to represent entity patterns. Consistently better than all other methods, our CAN obtains the best results on these four datasets by using a suitable attention mechanism to address HSI data and improve the primary activity vectors.

Table 6. Quantitative comparison on IP dataset. Numbers in bold denote the best value in comparison to others.

Class No.	CNN		GCN		TF		Composite Model				Mamba	CapsNet				
	3D CNN	RSSAN	DBDA	MDGCN	SSFTT	GAHT	SF	WFCG	AMGCFN	GTFN	DSNet	MHSI	CNet	ATT-CNet	SA-CNet	CAN
1	87.01 ± 0.0	31.18 ± 25.8	**100.0 ± 0.0**	97.91 ± 2.9	**100.0 ± 0.0**	98.92 ± 1.5	47.31 ± 13.5	82.33 ± 3.5	**100.0 ± 0.0**	**100.0 ± 0.0**	99.78 ± 0.1	56.14 ± 3.3	88.17 ± 3.1	92.63 ± 1.1	89.32 ± 0.9	94.19 ± 1.2
2	41.18 ± 6.8	60.94 ± 13.5	83.69 ± 0.4	74.61 ± 9.0	77.23 ± 2.9	80.31 ± 2.2	50.74 ± 5.0	85.99 ± 3.1	83.08 ± 10.1	80.78 ± 1.7	84.37 ± 2.2	90.30 ± 0.5	87.63 ± 1.4	92.36 ± 1.1	84.36 ± 0.8	**95.89 ± 1.0**
3	46.92 ± 8.7	64.75 ± 9.2	89.87 ± 0.0	84.39 ± 3.2	77.46 ± 10.0	84.83 ± 7.3	57.04 ± 6.7	79.80 ± 2.7	88.87 ± 7.2	91.79 ± 0.4	86.258 ± 1.3	91.45 ± 1.6	94.67 ± 0.6	96.47 ± 0.8	95.94 ± 0.7	**97.38 ± 0.6**
4	88.89 ± 2.4	90.02 ± 4.9	90.98 ± 0.9	97.42 ± 1.9	98.39 ± 1.3	99.03 ± 1.0	84.06 ± 6.3	75.29 ± 2.5	**100.0 ± 0.0**	98.71 ± 1.2	92.37 ± 1.0	91.02 ± 0.9	92.27 ± 0.7	92.27 ± 0.7	92.27 ± 0.7	98.65 ± 0.5
5	73.36 ± 6.9	83.81 ± 5.8	93.23 ± 0.9	95.05 ± 0.8	93.75 ± 4.0	91.69 ± 3.9	72.33 ± 6.6	99.34 ± 0.7	95.12 ± 5.0	96.70 ± 0.6	98.63 ± 1.3	91.02 ± 0.9	98.31 ± 0.9	**99.82 ± 0.0**	99.21 ± 0.3	99.74 ± 0.1
6	88.29 ± 4.1	90.95 ± 2.9	97.61 ± 0.3	99.42 ± 0.1	96.67 ± 1.3	96.62 ± 1.3	82.29 ± 0.9	98.68 ± 0.7	97.37 ± 1.3	96.71 ± 1.0	97.71 ± 0.2	99.46 ± 0.6	99.81 ± 0.0	99.87 ± 0.0	99.64 ± 0.2	**99.91 ± 0.0**
7	**100.0 ± 0.0**	**100.0 ± 0.0**	94.87 ± 3.6	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	72.02 ± 17.0	**100.0 ± 0.0**	94.87 ± 3.6	96.70 ± 0.6	98.63 ± 1.3	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
8	97.02 ± 1.0	99.26 ± 0.5	**100.0 ± 0.0**	**100.0 ± 0.0**	99.78 ± 0.3	99.85 ± 0.2	96.35 ± 1.2	99.48 ± 0.2	99.77 ± 0.1	99.85 ± 0.1	99.40 ± 0.3	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
9	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	66.13 ± 7.48	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
10	57.93 ± 3.7	78.91 ± 8.6	88.39 ± 1.1	91.63 ± 2.5	87.30 ± 2.7	86.13 ± 3.3	71.09 ± 4.6	69.28 ± 4.0	83.24 ± 6.5	96.04 ± 1.2	94.90 ± 0.7	86.03 ± 2.0	85.21 ± 0.8	96.08 ± 0.9	**96.54 ± 0.8**	95.86 ± 1.5
11	59.64 ± 3.7	56.78 ± 8.6	75.79 ± 0.7	75.21 ± 4.7	67.68 ± 7.5	75.77 ± 4.8	53.46 ± 9.3	93.44 ± 2.4	90.03 ± 6.3	90.38 ± 0.7	79.38 ± 1.4	93.67 ± 1.2	94.42 ± 0.8	96.32 ± 0.7	96.04 ± 0.6	**97.31 ± 0.5**
12	62.94 ± 5.6	65.78 ± 4.3	89.16 ± 0.8	93.30 ± 2.1	83.36 ± 2.7	88.93 ± 2.0	43.64 ± 4.4	82.08 ± 8.3	90.98 ± 5.1	90.70 ± 1.9	91.36 ± 1.7	92.67 ± 1.3	94.97 ± 1.1	96.33 ± 1.2	97.44 ± 0.9	**98.33 ± 0.7**
13	99.05 ± 0.2	98.48 ± 0.9	**100.0 ± 0.0**	**100.0 ± 0.0**	99.62 ± 0.5	99.43 ± 0.8	97.71 ± 1.6	98.89 ± 1.5	99.61 ± 0.5	99.62 ± 0.5	79.38 ± 1.4	96.67 ± 1.1	99.81 ± 0.2	**100.0 ± 0.0**	99.83 ± 0.1	**100.0 ± 0.0**
14	83.13 ± 0.6	96.06 ± 0.4	89.58 ± 0.0	94.67 ± 1.6	91.77 ± 1.6	90.31 ± 0.8	76.60 ± 5.1	97.64 ± 0.6	97.18 ± 0.4	99.43 ± 0.3	98.33 ± 0.6	97.96 ± 0.9	99.56 ± 0.1	99.45 ± 0.1	99.26 ± 0.3	**99.60 ± 0.1**
15	80.43 ± 6.9	79.78 ± 7.2	98.59 ± 0.3	**99.90 ± 0.1**	95.51 ± 3.5	95.51 ± 3.3	85.49 ± 3.3	82.96 ± 5.1	98.77 ± 0.8	99.16 ± 0.0	98.33 ± 0.7	94.32 ± 0.6	96.63 ± 0.4	97.39 ± 1.1	99.05 ± 0.3	99.49 ± 0.3
16	**100.0 ± 0.0**	98.41 ± 1.3	98.41 ± 0.0	**100.0 ± 0.0**	96.30 ± 5.2	**100.0 ± 0.0**	98.94 ± 1.5	94.53 ± 2.3	99.47 ± 0.7	98.33 ± 0.7	99.34 ± 0.5	99.47 ± 0.7	**100.0 ± 0.0**	99.95 ± 0.0	**100.0 ± 0.0**	**100.0 ± 0.0**
OA (%)	65.83 ± 1.7	73.78 ± 1.0	87.06 ± 0.3	86.77 ± 2.4	82.96 ± 1.2	85.96 ± 1.6	65.24 ± 2.0	87.68 ± 0.2	91.25 ± 4.8	94.00 ± 0.1	89.32 ± 0.8	92.76 ± 1.2	94.26 ± 0.5	96.49 ± 0.5	95.64 ± 0.8	**97.91 ± 0.4**
AA (%)	79.12 ± 1.2	80.94 ± 1.2	93.13 ± 0.0	93.97 ± 0.7	91.55 ± 0.9	92.96 ± 0.7	76.07 ± 0.9	86.12 ± 1.4	95.21 ± 5.4	96.33 ± 0.2	94.46 ± 0.8	92.34 ± 1.3	95.72 ± 0.9	97.82 ± 0.4	96.69 ± 0.8	**98.52 ± 0.2**
Kappa	61.38 ± 1.9	70.48 ± 0.9	85.32 ± 0.3	85.03 ± 2.7	80.71 ± 1.2	84.05 ± 1.8	61.06 ± 2.1	86.03 ± 0.2	90.03 ± 5.4	93.15 ± 0.1	87.24 ± 1.7	91.69 ± 1.4	93.41 ± 0.6	96.92 ± 0.7	95.36 ± 0.8	**97.60 ± 0.5**

Table 7. Quantitative comparison on SA dataset. Numbers in bold denote the best value in comparison to others.

Class No.	CNN		GCN		TF		Composite Model				Mamba	CapsNet				
	3D CNN	RSSAN	DBDA	MDGCN	SSFTT	GAHT	SF	WFCG	AMGCFN	GTFN	DSNet	MHSI	CNet	ATT-CNet	SA-CNet	CAN
1	98.77 ± 0.8	98.84 ± 0.2	99.86 ± 0.1	97.82 ± 3.0	**100.0 ± 0.0**	**100.0 ± 0.0**	94.74 ± 0.9	99.74 ± 0.3	**100.0 ± 0.0**	**100.0 ± 0.0**	97.87 ± 0.4	99.70 ± 0.2	99.82 ± 0.1	99.60 ± 0.2	99.84 ± 0.1	99.92 ± 0.0
2	98.70 ± 0.7	99.50 ± 0.0	99.91 ± 0.1	**100.0 ± 0.0**	99.49 ± 0.4	99.98 ± 0.0	98.82 ± 0.4	**100.0 ± 0.0**	99.72 ± 0.4	99.97 ± 0.0	99.86 ± 0.1	99.89 ± 0.1	99.84 ± 0.0	99.87 ± 0.1	99.81 ± 0.0	99.99 ± 0.0
3	90.00 ± 1.1	97.14 ± 1.8	98.33 ± 0.6	**100.0 ± 0.0**	90.10 ± 7.8	98.49 ± 0.9	92.17 ± 1.9	**100.0 ± 0.0**	99.95 ± 0.1	99.97 ± 0.0	98.45 ± 0.7	96.67 ± 0.6	99.93 ± 0.0	99.82 ± 0.0	99.84 ± 0.0	99.74 ± 0.1
4	98.75 ± 0.3	99.49 ± 0.4	97.80 ± 0.7	99.85 ± 0.1	99.76 ± 0.3	99.12 ± 0.7	96.26 ± 0.9	98.99 ± 0.3	98.53 ± 2.2	99.59 ± 0.1	99.93 ± 0.0	96.87 ± 0.7	99.41 ± 0.2	99.54 ± 0.0	99.72 ± 0.0	**99.77 ± 0.1**
5	95.20 ± 1.0	96.99 ± 1.2	98.74 ± 0.7	97.09 ± 0.0	97.13 ± 2.3	99.13 ± 0.4	89.41 ± 1.8	97.87 ± 0.7	97.25 ± 0.7	95.89 ± 1.3	97.82 ± 0.6	99.94 ± 0.0	99.74 ± 0.2	99.78 ± 0.1	99.62 ± 0.2	**99.84 ± 0.1**
6	99.40 ± 0.3	99.02 ± 0.6	99.99 ± 0.0	99.92 ± 0.0	99.71 ± 0.3	99.97 ± 0.0	99.20 ± 0.7	99.96 ± 0.0	99.59 ± 0.6	99.84 ± 0.0	99.75 ± 0.1	99.93 ± 0.0	99.96 ± 0.0	99.60 ± 0.0	99.96 ± 0.0	**99.99 ± 0.0**
7	99.38 ± 0.3	98.07 ± 1.6	99.84 ± 0.1	92.12 ± 3.1	99.03 ± 0.5	99.80 ± 0.1	97.91 ± 1.9	99.80 ± 0.2	99.99 ± 0.0	99.72 ± 0.2	99.68 ± 0.1	99.92 ± 0.0	99.97 ± 0.0	99.96 ± 0.0	99.97 ± 0.0	**100.0 ± 0.0**
8	72.55 ± 6.8	71.60 ± 1.7	85.29 ± 0.4	91.97 ± 3.0	73.95 ± 3.6	83.11 ± 3.1	73.07 ± 1.1	77.52 ± 7.7	83.51 ± 9.2	92.18 ± 0.8	82.48 ± 1.0	88.38 ± 0.9	87.47 ± 0.5	90.67 ± 0.3	91.55 ± 0.4	**94.35 ± 0.4**
9	93.30 ± 0.9	98.15 ± 0.9	99.54 ± 0.2	**100.0 ± 0.0**	98.59 ± 1.3	99.47 ± 0.1	93.77 ± 1.4	99.90 ± 0.1	99.96 ± 0.0	99.90 ± 0.1	99.93 ± 0.0	99.80 ± 0.2	99.90 ± 0.0	99.45 ± 0.2	99.62 ± 0.1	99.98 ± 0.0
10	88.24 ± 0.9	92.03 ± 2.8	94.42 ± 2.8	83.65 ± 1.6	94.28 ± 1.1	97.36 ± 1.8	91.39 ± 1.8	94.71 ± 2.0	99.61 ± 0.3	95.42 ± 0.5	96.44 ± 0.7	95.78 ± 0.9	95.54 ± 0.3	98.36 ± 0.7	96.39 ± 0.7	**99.61 ± 0.1**
11	93.96 ± 2.2	96.76 ± 1.8	97.54 ± 2.9	96.91 ± 0.0	98.65 ± 0.2	98.62 ± 1.1	92.36 ± 3.3	98.74 ± 0.5	99.33 ± 0.5	99.77 ± 0.2	99.28 ± 0.2	97.57 ± 0.3	99.13 ± 0.1	99.43 ± 0.1	99.27 ± 0.0	**99.83 ± 0.1**
12	99.33 ± 0.2	99.37 ± 0.7	98.47 ± 0.0	89.10 ± 4.0	99.95 ± 0.0	**100.0 ± 0.0**	99.05 ± 0.5	**100.0 ± 0.0**	99.91 ± 0.1	99.66 ± 0.2	99.82 ± 0.1	99.83 ± 0.1	99.62 ± 0.2	99.92 ± 0.0	99.64 ± 0.1	99.86 ± 0.1
13	99.06 ± 0.9	99.44 ± 0.6	99.06 ± 0.7	96.95 ± 0.0	97.86 ± 1.2	**99.96 ± 0.0**	99.40 ± 0.4	99.94 ± 0.0	99.85 ± 0.1	99.50 ± 0.4	94.95 ± 0.7	99.62 ± 0.2	99.85 ± 0.1	99.74 ± 0.2	99.88 ± 0.1	99.93 ± 0.1
14	96.73 ± 0.9	95.54 ± 2.7	98.59 ± 0.5	96.98 ± 0.1	98.91 ± 0.1	98.94 ± 0.2	96.60 ± 0.9	98.04 ± 0.7	99.23 ± 0.3	99.17 ± 0.8	94.36 ± 0.7	98.68 ± 0.6	99.58 ± 0.2	**99.66 ± 0.1**	99.50 ± 0.2	99.62 ± 0.2
15	71.35 ± 3.3	79.07 ± 3.1	71.66 ± 2.1	71.74 ± 2.7	81.10 ± 3.6	87.10 ± 2.8	81.58 ± 2.2	84.93 ± 7.1	89.77 ± 10.3	93.06 ± 0.6	65.36 ± 2.1	83.36 ± 0.7	83.35 ± 0.7	88.48 ± 0.5	90.03 ± 0.8	**93.53 ± 0.6**
16	92.61 ± 1.9	95.72 ± 2.5	99.19 ± 0.5	**100.0 ± 0.0**	96.96 ± 2.0	96.83 ± 2.6	93.72 ± 2.0	99.15 ± 0.4	99.90 ± 0.1	99.18 ± 0.3	98.93 ± 0.6	98.76 ± 0.5	99.04 ± 0.3	99.54 ± 0.2	99.16 ± 0.3	99.83 ± 0.1
OA (%)	87.59 ± 1.4	89.66 ± 0.7	92.18 ± 0.2	92.20 ± 0.7	90.67 ± 1.5	94.21 ± 0.3	88.80 ± 0.2	92.68 ± 1.0	94.87 ± 1.0	96.81 ± 0.2	91.01 ± 1.2	94.63 ± 0.5	94.77 ± 0.2	96.24 ± 0.4	95.82 ± 0.3	**97.73 ± 0.1**
AA (%)	92.96 ± 0.5	94.79 ± 0.7	96.14 ± 0.1	94.63 ± 0.4	95.34 ± 1.1	97.37 ± 0.0	93.09 ± 0.1	96.83 ± 0.4	97.10 ± 0.1	98.30 ± 0.0	95.77 ± 0.3	95.77 ± 0.3	97.70 ± 0.1	98.37 ± 0.2	97.92 ± 0.3	**99.05 ± 0.0**
Kappa	86.21 ± 1.6	88.53 ± 0.8	91.30 ± 0.2	91.30 ± 0.8	89.64 ± 1.6	93.56 ± 0.4	87.57 ± 0.2	91.86 ± 1.1	89.98 ± 0.7	94.01 ± 0.6	96.45 ± 0.2	96.45 ± 0.2	94.18 ± 0.2	96.47 ± 0.2	95.82 ± 0.4	**97.47 ± 0.1**

Table 8. Quantitative comparison on PU dataset. Numbers in bold denote the best value in comparison to others.

Class No.	CNN		GCN		TF			Composite Model			Mamba	CapsNet			
	3D CNN	RSSAN	DBDA	MDGCN	SSFTT	GAHT	SF	WFCG	AMGCFN	GTFN	DSNet	MHSI	CNet	ATT-CNet	SA-CNet	CAN
1	80.72 ± 4.5	77.98 ± 10.0	92.81 ± 0.9	88.96 ± 2.9	87.56 ± 3.4	92.12 ± 3.5	72.19 ± 6.6	98.79 ± 0.6	88.21 ± 3.4	89.62 ± 0.7	89.72 ± 0.8	94.75 ± 0.6	94.76 ± 0.3	95.85 ± 0.56	94.99 ± 0.4	**96.75 ± 0.4**
2	73.15 ± 6.1	87.64 ± 7.4	92.82 ± 0.3	90.73 ± 2.1	92.35 ± 6.2	91.19 ± 5.6	75.51 ± 3.0	98.31 ± 0.4	93.98 ± 1.1	97.24 ± 0.4	98.24 ± 0.3	97.94 ± 0.5	98.36 ± 0.5	98.39 ± 0.3	96.76 ± 0.8	**98.72 ± 0.1**
3	71.61 ± 2.9	59.55 ± 6.9	83.72 ± 0.8	84.98 ± 6.8	86.37 ± 2.6	87.26 ± 4.9	75.17 ± 8.8	80.41 ± 8.2	90.24 ± 1.3	87.62 ± 0.6	**94.55 ± 2.0**	84.52 ± 3.2	80.16 ± 0.8	82.72 ± 0.7	76.32 ± 0.8	88.14 ± 0.3
4	97.25 ± 0.8	96.84 ± 0.4	92.66 ± 0.3	97.40 ± 0.5	97.90 ± 0.4	97.73 ± 0.1	89.53 ± 2.2	94.74 ± 3.4	89.85 ± 4.2	94.91 ± 0.4	93.79 ± 0.8	93.77 ± 0.7	96.90 ± 0.2	95.43 ± 0.3	92.66 ± 0.5	**98.21 ± 0.1**
5	99.92 ± 0.1	99.34 ± 0.4	98.88 ± 0.1	99.39 ± 0.9	99.97 ± 0.0	**100.0 ± 0.0**	**100.0 ± 0.0**	99.89 ± 0.1	**100.0 ± 0.0**	98.71 ± 0.3	99.51 ± 0.3	99.54 ± 0.2	99.92 ± 0.1	98.64 ± 0.2	99.34 ± 0.1	99.97 ± 0.0
6	74.67 ± 6.9	68.59 ± 7.9	92.44 ± 0.2	93.88 ± 0.9	87.62 ± 2.0	96.03 ± 3.4	76.65 ± 0.6	80.61 ± 11.3	97.93 ± 1.7	**99.59 ± 0.1**	74.95 ± 1.6	87.28 ± 0.8	92.18 ± 0.9	96.28 ± 0.6	91.36 ± 0.7	95.59 ± 0.8
7	86.15 ± 4.7	86.33 ± 7.6	87.79 ± 0.5	**99.59 ± 0.6**	92.64 ± 3.1	95.36 ± 4.6	70.21 ± 6.5	85.77 ± 5.2	96.22 ± 4.1	98.97 ± 0.4	55.08 ± 2.4	95.76 ± 1.0	65.62 ± 2.1	82.36 ± 1.0	76.38 ± 2.1	80.62 ± 2.4
8	72.32 ± 1.6	74.10 ± 11.0	90.53 ± 0.3	94.28 ± 7.6	86.73 ± 3.0	91.65 ± 0.1	62.05 ± 7.8	88.81 ± 0.2	89.65 ± 2.3	91.59 ± 2.6	86.58 ± 1.3	89.87 ± 0.9	92.85 ± 0.2	92.45 ± 0.3	91.36 ± 0.6	**94.37 ± 0.1**
9	99.27 ± 0.6	99.64 ± 0.3	96.18 ± 0.3	99.85 ± 0.1	99.93 ± 0.1	99.82 ± 0.1	97.89 ± 1.2	98.61 ± 0.9	91.53 ± 4.7	96.07 ± 0.7	99.68 ± 0.1	99.92 ± 0.0	99.67 ± 0.0	**100.0 ± 0.0**	99.34 ± 0.1	**100.0 ± 0.0**
OA(%)	77.87 ± 1.3	82.61 ± 2.2	92.23 ± 0.2	92.10 ± 0.7	91.08 ± 2.6	92.81 ± 2.3	76.03 ± 0.7	92.98 ± 2.1	93.12 ± 0.6	95.14 ± 0.1	94.26 ± 0.3	93.57 ± 0.6	93.96 ± 0.3	95.23 ± 0.6	94.86 ± 0.4	**96.64 ± 0.2**
AA(%)	83.90 ± 0.6	83.33 ± 2.4	91.98 ± 0.1	94.38 ± 0.2	92.34 ± 0.9	94.57 ± 0.4	79.91 ± 0.8	91.77 ± 1.2	93.55 ± 0.9	94.58 ± 0.5	94.30 ± 0.5	91.28 ± 0.8	88.98 ± 0.5	93.46 ± 0.7	93.34 ± 0.6	**94.71 ± 0.3**
Kappa	71.82 ± 1.3	77.19 ± 2.9	89.80 ± 0.3	89.68 ± 1.0	88.32 ± 3.2	90.64 ± 2.8	69.35 ± 0.8	90.83 ± 2.7	90.96 ± 0.7	93.58 ± 0.2	92.38 ± 0.7	91.42 ± 0.4	91.94 ± 0.4	93.46 ± 1.2	92.68 ± 0.6	**95.53 ± 0.3**

Table 9. Quantitative comparison on UH2013 dataset. Numbers in bold denote the best value in comparison to others.

Class No.	CNN		GCN		TF			Composite Model			Mamba	CapsNet			
	3D CNN	RSSAN	DBDA	MDGCN	SSFTT	GAHT	SF	WFCG	AMGCFN	GTFN	DSNet	MHSI	CNet	ATT-CNet	SA-CNet	CAN
1	94.02 ± 6.5	90.72 ± 4.4	85.09 ± 2.4	87.36 ± 2.2	89.32 ± 2.6	92.55 ± 2.3	91.10 ± 3.8	94.27 ± 0.9	86.09 ± 1.3	88.36 ± 0.4	93.45 ± 0.4	93.61 ± 0.6	92.36 ± 0.7	93.50 ± 0.4	90.99 ± 0.8	**95.17 ± 0.6**
2	81.62 ± 8.9	97.49 ± 1.5	98.77 ± 0.8	86.25 ± 2.3	96.35 ± 2.2	98.88 ± 0.2	93.19 ± 2.5	92.48 ± 0.6	87.84 ± 9.2	87.16 ± 1.4	98.77 ± 0.6	98.53 ± 0.4	97.63 ± 0.4	98.43 ± 0.4	99.26 ± 0.4	**99.59 ± 0.3**
3	96.81 ± 1.1	**99.15 ± 0.6**	89.96 ± 1.4	87.62 ± 3.1	90.31 ± 2.5	97.36 ± 0.8	93.90 ± 4.6	89.72 ± 3.6	97.35 ± 0.2	88.68 ± 3.6	90.85 ± 0.7	94.15 ± 0.6	80.62 ± 0.6	96.60 ± 0.9	94.32 ± 0.6	96.25 ± 0.6
4	94.15 ± 3.0	95.09 ± 3.4	95.96 ± 1.6	95.70 ± 0.8	94.32 ± 0.7	94.23 ± 1.0	88.44 ± 2.6	95.42 ± 1.7	90.04 ± 1.9	94.21 ± 0.7	96.21 ± 0.3	97.03 ± 0.1	94.58 ± 0.8	96.52 ± 0.7	97.20 ± 0.8	**98.93 ± 0.3**
5	98.79 ± 1.1	94.39 ± 5.6	98.6 ± 1.1	98.68 ± 0.7	97.33 ± 0.7	99.39 ± 0.7	97.14 ± 0.6	98.82 ± 0.3	99.42 ± 0.7	93.84 ± 0.7	99.26 ± 0.4	94.58 ± 0.7	96.36 ± 0.7	98.00 ± 0.7	99.83 ± 0.0	**99.92 ± 0.0**
6	84.29 ± 3.6	89.83 ± 5.5	71.19 ± 1.4	92.88 ± 1.6	88.62 ± 1.0	**94.58 ± 5.2**	88.36 ± 3.9	88.69 ± 1.4	92.65 ± 0.6	90.56 ± 2.1	78.31 ± 0.8	76.95 ± 3.1	90.17 ± 0.9	85.76 ± 1.3	73.56 ± 3.1	93.22 ± 0.8
7	77.95 ± 5.2	84.41 ± 3.8	83.84 ± 2.3	92.33 ± 0.9	91.34 ± 3.1	84.30 ± 3.7	74.18 ± 1.9	90.47 ± 3.8	92.16 ± 2.5	92.47 ± 0.9	90.71 ± 1.6	95.10 ± 0.4	63.72 ± 2.5	95.96 ± 0.8	93.78 ± 1.2	**97.33 ± 0.2**
8	54.37 ± 3.1	71.17 ± 4.9	49.51 ± 4.1	86.18 ± 4.2	86.14 ± 3.2	82.18 ± 4.1	76.44 ± 4.5	87.82 ± 2.5	71.92 ± 3.5	84.63 ± 2.7	70.35 ± 3.7	72.16 ± 2.8	88.64 ± 0.8	90.36 ± 1.2	71.17 ± 1.1	**91.10 ± 0.4**
9	81.23 ± 1.6	83.42 ± 4.8	84.12 ± 2.6	87.64 ± 0.7	83.93 ± 1.4	83.77 ± 3.5	67.59 ± 3.7	90.60 ± 0.8	76.28 ± 4.5	85.06 ± 0.4	88.54 ± 0.8	91.57 ± 0.3	87.62 ± 0.7	90.82 ± 0.6	89.81 ± 0.8	**91.96 ± 0.4**
10	63.30 ± 3.4	68.64 ± 10.9	58.9 ± 3.4	84.18 ± 1.3	91.08 ± 2.6	89.75 ± 5.8	78.75 ± 3.6	90.92 ± 1.1	87.49 ± 14.8	88.36 ± 0.7	85.13 ± 2.3	82.79 ± 2.3	86.91 ± 0.8	**94.90 ± 0.5**	82.96 ± 0.4	93.96 ± 0.2
11	62.79 ± 2.9	66.25 ± 8.1	82.49 ± 2.4	87.04 ± 1.3	90.04 ± 1.6	77.48 ± 0.8	62.38 ± 6.0	92.73 ± 1.8	88.98 ± 4.4	92.16 ± 0.7	87.47 ± 0.3	89.79 ± 1.2	89.36 ± 0.6	92.27 ± 0.4	86.22 ± 2.3	**95.44 ± 0.6**
12	55.20 ± 9.1	79.30 ± 5.3	47.8 ± 0.7	86.18 ± 1.2	84.69 ± 2.8	88.03 ± 2.1	74.40 ± 5.5	86.83 ± 1.3	82.76 ± 12.0	90.51 ± 0.4	73.4 ± 6.8	82.71 ± 4.2	90.36 ± 0.4	88.69 ± 0.6	75.15 ± 0.9	**94.18 ± 0.4**
13	92.26 ± 2.1	92.71 ± 1.1	35.76 ± 4.3	68.48 ± 3.7	80.13 ± 0.9	**94.08 ± 1.4**	64.16 ± 5.6	88.81 ± 1.4	88.61 ± 0.6	83.36 ± 0.8	59.45 ± 4.7	66.97 ± 3.2	72.89 ± 0.7	76.08 ± 1.7	64.46 ± 2.7	78.59 ± 1.4
14	98.99 ± 0.7	69.93 ± 41.6	97.99 ± 0.1	96.62 ± 1.5	96.77 ± 1.9	**99.75 ± 0.2**	88.86 ± 1.4	97.14 ± 0.8	97.25 ± 0.7	98.36 ± 0.7	97.74 ± 0.8	98.74 ± 0.4	94.31 ± 0.8	92.75 ± 0.2	98.49 ± 0.7	99.25 ± 0.3
15	99.95 ± 0.0	93.02 ± 8.8	98.73 ± 0.4	95.48 ± 0.7	96.78 ± 2.1	**100.00 ± 0.0**	98.52 ± 0.9	96.92 ± 2.1	98.72 ± 0.4	98.17 ± 0.6	99.84 ± 0.1	99.84 ± 0.1	96.73 ± 1.0	99.84 ± 0.1	98.24 ± 0.4	**100 ± 0.0**
OA (%)	79.61 ± 1.3	84.36 ± 3.0	79.07 ± 1.6	86.23 ± 1.3	88.48 ± 1.3	90.84 ± 0.5	81.68 ± 0.6	92.63 ± 0.8	88.07 ± 2.0	92.74 ± 0.7	88.15 ± 1.2	90.2 ± 0.6	89.36 ± 0.6	92.66 ± 0.4	88.66 ± 0.8	**95.48 ± 0.4**
AA (%)	82.38 ± 1.3	85.03 ± 4.6	78.58 ± 1.4	88.37 ± 0.9	91.03 ± 0.1	92.13 ± 0.6	82.49 ± 0.1	86.14 ± 1.4	89.58 ± 1.6	91.63 ± 0.7	87.30 ± 1.3	89.33 ± 0.5	86.59 ± 0.8	92.90 ± 0.6	87.82 ± 1.4	**94.98 ± 0.5**
Kappa	77.97 ± 1.4	83.08 ± 3.3	77.36 ± 1.3	85.62 ± 1.6	87.73 ± 1.4	90.10 ± 0.6	80.22 ± 0.7	88.03 ± 0.8	87.09 ± 2.2	90.45 ± 0.9	87.18 ± 0.8	89.39 ± 0.7	88.49 ± 0.7	93.23 ± 0.5	87.73 ± 0.8	**95.11 ± 0.3**

(2) Visual Analysis: This section qualitatively analyzes the performance of different methods using visual maps. Figures 5–8 show the results of IP, SA, PU and UH2013, respectively. In all the datasets, the CNN-based method 3-D CNN, and the TF-based methods SSFTT and SF, do not work well, leading to a large amount of pepper noise and even large patches of false classification. Moreover, the CNN-based methods RSSAN and DBDA tend to cause many false results within or on the edges of entities. Although the GCN-based method MDGCN effectively builds the long-range spatial relation using a superpixels graph, some clumpy misclassification can be found. A similar problem can be seen in the maps of the TF-based method GAHT. The Mamba-based work MHSI causes obvious dotted noise. All the results show that the current backbones fail to achieve accurate classification due to inefficient feature representation. A composite model can be used to solve the problem to some extent. As can be seen, the CNN and GCN composite methods WFCG and AMGCFN produce better maps, where the false classification is effectively suppressed. AE and the CNN composite method DSNet result in some pieces of noise. By contrast, the GCN and TF composite method GTFN outperforms the other composite methods by using the long-range spectral dependencies (TF) and long-range spatial relations (GCN) simultaneously. Although some progress has been made, a large number of false classification can be found in the results of the composite methods. By contrast, CNet achieves competitive visual maps when compared to these composite methods. By using spatial attention for the HSI data, SA-CNet ameliorates the false classification to some extent. The visual maps of ATT-CNet are smoother than that of SA-CNet by introducing attention for the activation of the activity vectors. Better than all other methods, the proposed CAN achieves the most smooth visual map. In particular, for the IP dataset, we can see the visual map of CAN very closely approximates that of GT. For UH2013, the proposed CAN restores the details more exactly for some similar regions (e.g., the healthy grass and the stressed grass in the amplified areas of the red boxes) than the recent attention-based CapsNet models ATT-CNet and SA-CNet.

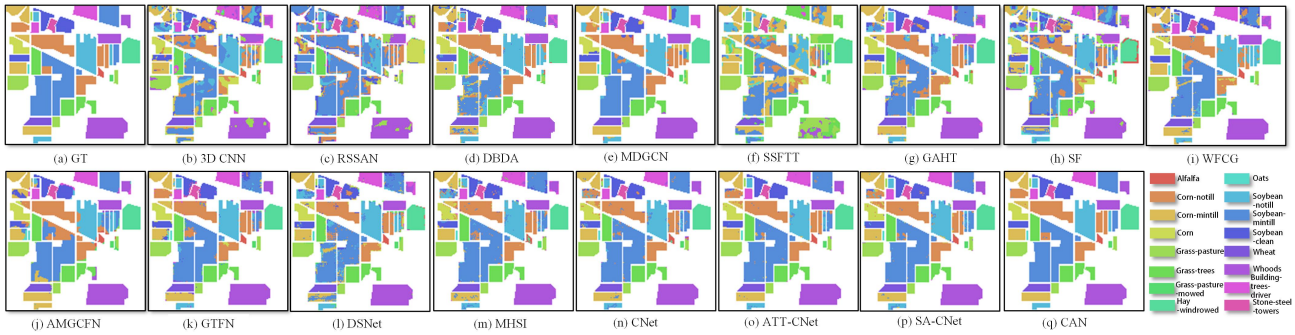

Figure 5. Visual maps of the results on the IP dataset.

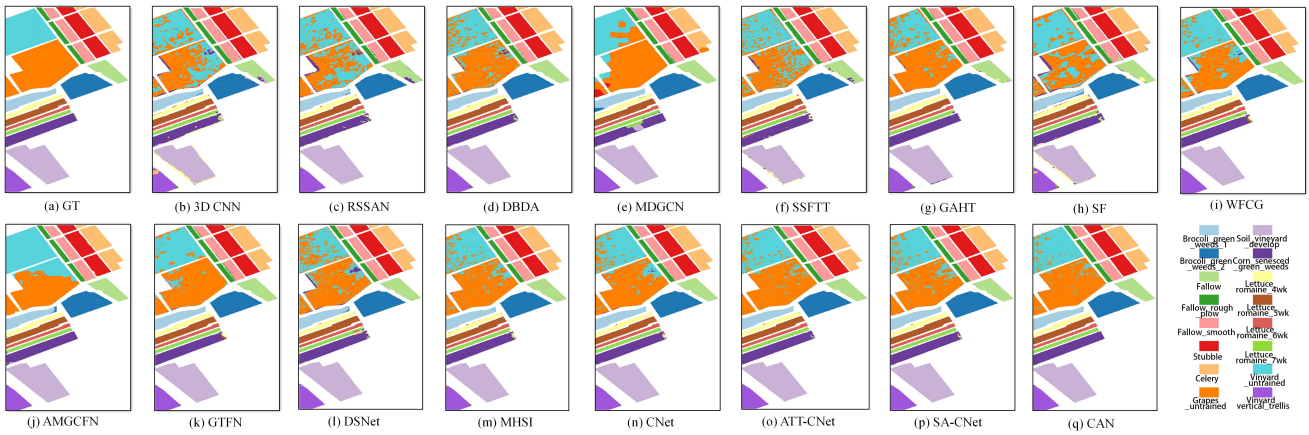

Figure 6. Visual maps of the results on the SA dataset.

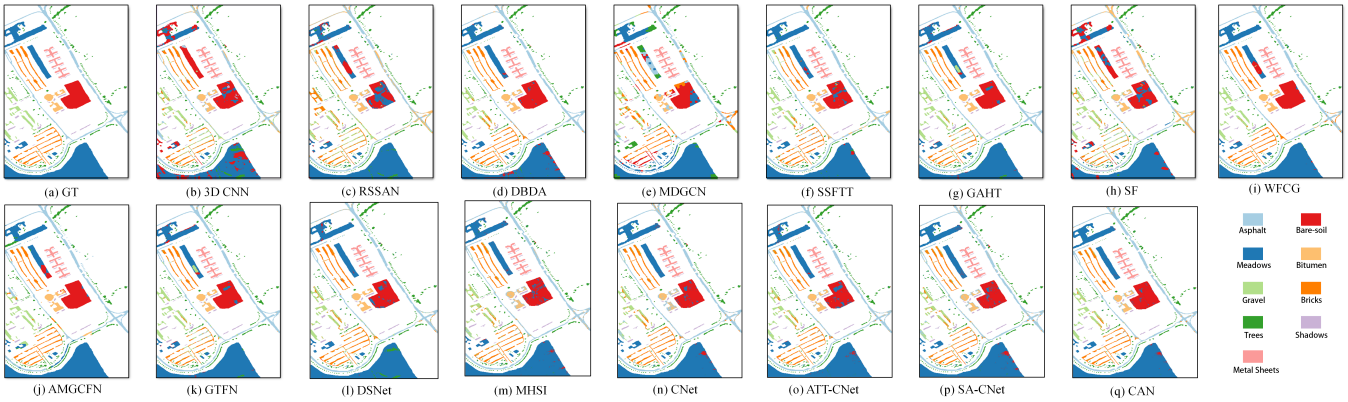

Figure 7. Visual maps of the results on the PU dataset.

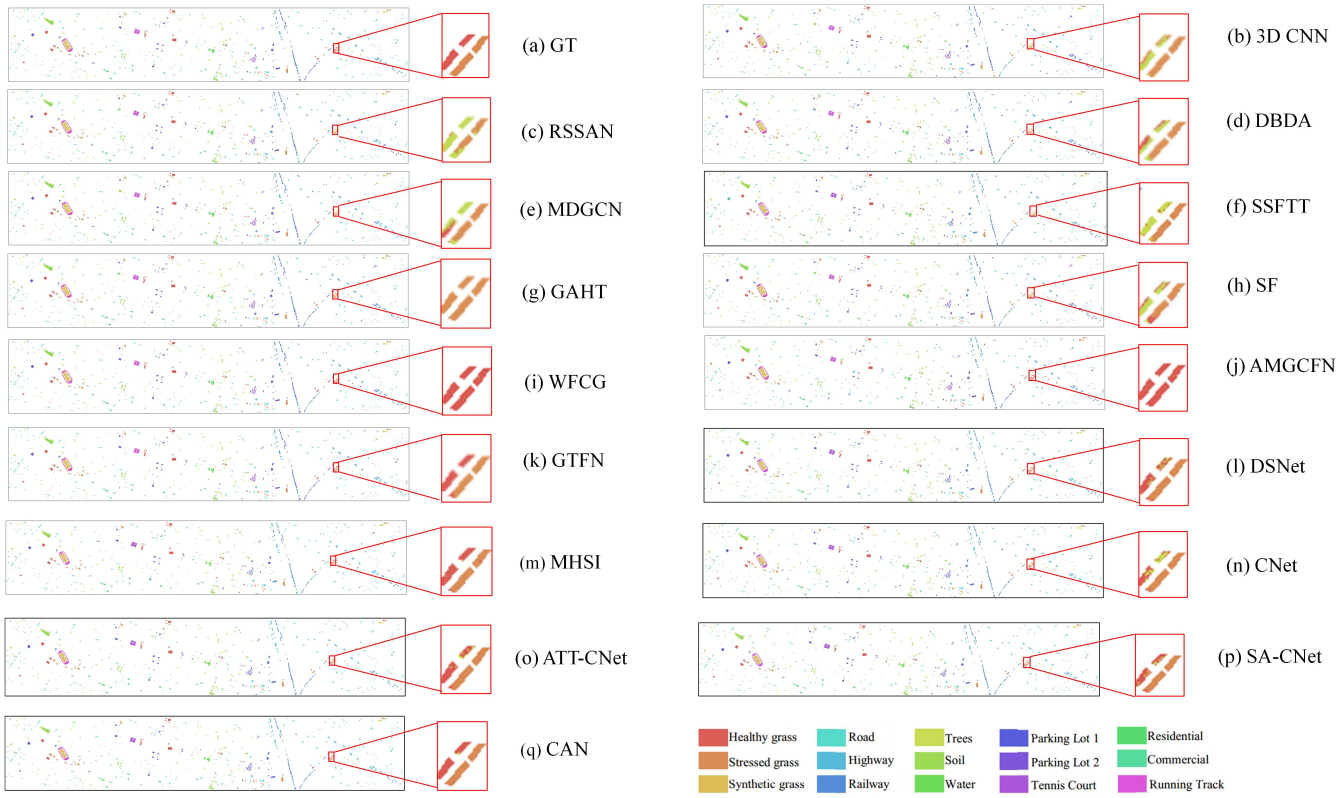

Figure 8. Visual maps of the results on the UH2013 dataset.

(3) Experiments on UH2018. To verify generalization of the ability to challenge a large-scale dataset, we tested on UH2018. The comparative methods include the composite model GTFN, the Mamba-based model MHSI, and the CapsNet-based models CNet, ATT-CNet, SA-CNet. All of these are quite recently proposed models that have shown strong competitiveness against our CAN according to the results on the previous four benchmarks. We report both quantitative and qualitative results. For the quantitative results, we run each method two times and report the average values. Table 10 reports the results. As can be seen, our CAN still achieves SOTA classification with all three metrics' results being much higher than those of the comparative methods. In particular, examining the results of Class 9, it is found that the CapsNet-based methods obtain much higher accuracy. Since we only use 30 samples for training and test the classification results on 223,722 samples, GTFN and MHSI obtain much lower accuracy for this class. The results show that the CapsNet-based methods have better ability to achieve accurate classification when the number of training samples is quite limited, which has been verified by previous work [44].

Moreover, we note that our CAN performs much better than other CapsNet-based models on Class 3 and Class 17, but it still fails to exactly classify Class 12 (i.e., Crosswalks).

Table 10. Quantitative comparison on UH2018 dataset. Numbers in bold denote the best value in comparison to others.

Class No.	GTFN	MHSI	CNet	ATT-CNet	SA-CNet	CAN
1	**93.28 ± 0.8**	90.47 ± 0.5	87.33 ± 0.8	89.09 ± 0.9	88.43 ± 0.8	91.02 ± 0.2
2	64.62 ± 1.5	72.36 ± 0.9	94.44 ± 0.2	94.39 ± 0.5	94.86 ± 0.0	**96 ± 0.5**
3	96.87 ± 0.4	**100 ± 0.0**	1.53 ± 0.3	76.3 ± 2.1	78.75 ± 0.6	95.57 ± 0.1
4	90.69 ± 0.5	92.69 ± 0.2	96.36 ± 0.9	**97.98 ± 0.5**	97.43 ± 0.0	97.88 ± 0.2
5	91.54 ± 0.4	**92.69 ± 0.1**	64.02 ± 1.7	74.79 ± 1.4	76.16 ± 0.5	85.53 ± 0.8
6	97.59 ± 0.1	98.32 ± 0.2	86.87 ± 0.8	96.19 ± 0.2	97.19 ± 0.8	**99.38 ± 0.2**
7	**100 ± 0.0**	95.62 ± 0.9	63.98 ± 0.8	67.37 ± 0.8	68.64 ± 0.7	78.39 ± 1.2
8	65.39 ± 1.5	75.64 ± 0.7	85.15 ± 0.2	88.17 ± 0.5	88.41 ± 0.5	**92.13 ± 0.2**
9	30.64 ± 0.8	49.14 ± 1.5	95.41 ± 0.4	96.87 ± 0.1	96.7 ± 0.4	**98.08 ± 0.1**
10	30.47 ± 0.9	49.26 ± 0.8	56.37 ± 1.6	67.45 ± 0.8	68.76 ± 1.2	**78.16 ± 0.9**
11	43.69 ± 0.4	53.69 ± 2.2	58.81 ± 1.4	66.28 ± 0.4	66.25 ± 1.5	**75.09 ± 1.6**
12	**71.21 ± 0.9**	68.69 ± 0.9	0 ± 0	0.07 ± 0.02	0 ± 0	1.61 ± 0.14
13	48.65 ± 1.4	62.47 ± 1.9	77.19 ± 0.3	83.42 ± 0.2	83.6 ± 0.2	**89.74 ± 0.2**
14	**97.79 ± 0.2**	92.46 ± 0.1	71.94 ± 1.7	83.5 ± 0.4	83.99 ± 0.1	91.08 ± 0.2
15	**98.36 ± 0.1**	96.98 ± 0.2	92.27 ± 0.2	94.83 ± 0.2	94.86 ± 0.4	97.96 ± 0.4
16	85.64 ± 0.0	87.36 ± 0.7	75.16 ± 0.7	82.55 ± 0.4	84.05 ± 0.3	**90.2 ± 0.2**
17	97.69 ± 0.1	**98.69 ± 0.2**	1.72 ± 0.4	4.31 ± 0.1	6.9 ± 0.2	68.97 ± 0.8
18	**81.48 ± 0.2**	75.36 ± 0.8	40.22 ± 0.8	65.97 ± 1.1	69.82 ± 0.5	80.28 ± 0.4
19	86.25 ± 0.4	82.76 ± 0.2	89.75 ± 0.2	93.95 ± 0.1	93.16 ± 0.1	**97.12 ± 0.2**
20	94.78 ± 0.9	92.42 ± 0.9	91.24 ± 0.3	**98.94 ± 0.2**	97.79 ± 0.3	98.75 ± 0.0
OA (%)	68.36 ± 1.2	72.69 ± 0.8	84.12 ± 0.5	88.3 ± 0.8	85.52 ± 0.8	**92.24 ± 0.7**
AA (%)	82.36 ± 0.8	84.63 ± 0.9	66.49 ± 0.7	80.02 ± 1.1	78.79 ± 0.7	**85.15 ± 0.5**
Kappa	62.45 ± 1.6	66.24 ± 0.7	79.07 ± 0.8	84.67 ± 0.9	80.96 ± 0.6	**89.86 ± 0.7**

We further show the visual maps in Figure 9. As can be seen, the CapsNet-based methods achieve much more exact results than other backbones for Class 9 (non-residential buildings). Moreover, in the amplified box, we find that the CapsNet-based methods all achieve a smoother map for Class 13 (major thoroughfares). However, they tend to cause false classification on complex areas where there is adjacent location of residential buildings and water. By using AFE and SWM to enhance the representation of HSI data and primary activity vectors, our CAN obtains a much better map for this region against the other attention-based CapsNet models ATT-CNet and SA-CNet.

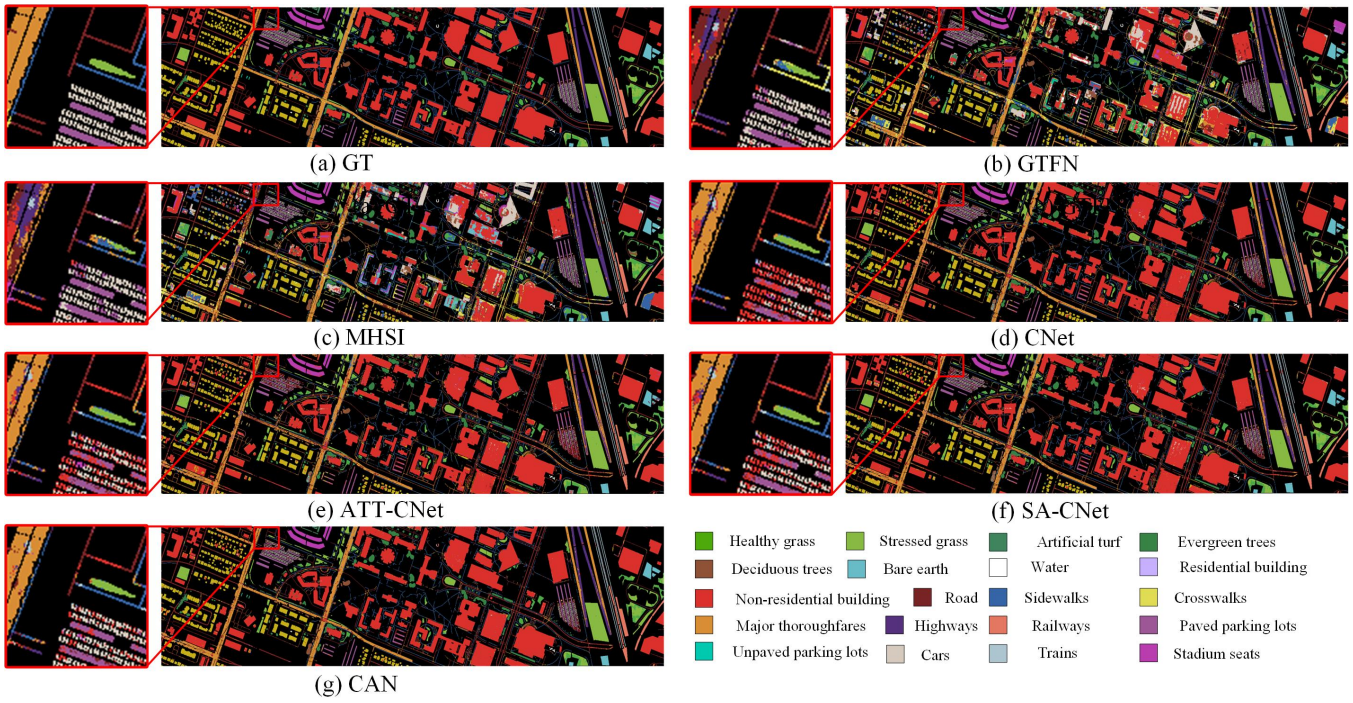

Figure 9. Visual maps of the results on the UH2018 dataset.

(4) Training Samples Analysis: To further verify the effectiveness and stability of the proposed CAN, we draw the curves of OA using different proportions of the training samples. As shown in Figure 10, to control the number of training samples, we set the diverse proportions of the training samples as 1%, 2%, 3%, 4%, 5% if the total samples are less than 20,000, otherwise 0.25%, 0.5%, 1%, 2%, 3%. Concretely, the proportions of the training samples for the rIP and UH2013 datasets were set as 1%, 2%, 3%, 4%, 5%. The proportions of the training samples for the SA, PU and UH2018 datasets were set as 0.25%, 0.5%, 1%, 2%, 3%. The results show that the OA of the different algorithms are generally increased as the training samples increase since the model can be adequately trained with more training samples. Among all the algorithms, CAN always acquires the best OA when using different proportions of samples for the training on these datasets, which demonstrates its excellent and stable performance. in particular, for the large-scale dataset UH2018, our CAN outperforms other methods by a large margin, which demonstrates its ability to achieve high classification accuracy for challenging data. More importantly, when the proportion of training samples are relatively small, our CAN provides a much higher OA than the competitors, which shows the stability of our CAN when the number of training samples is limited. All the results show that the proposed AFE and SWM effectively improve the quality of the activity vectors, providing an exact representation for each entity enabling CAN to obtain accurate classification even for extremely limited training conditions.

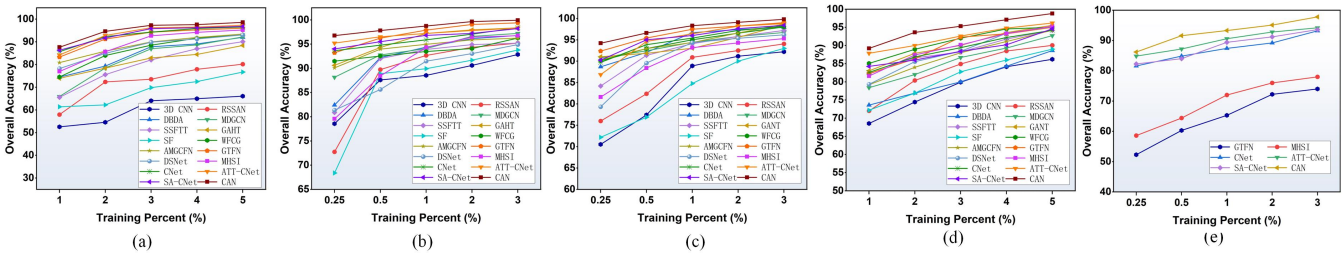

Figure 10. OA curves for different proportions of training samples on the datasets. (**a**) IP. (**b**) SA. (**c**) PU. (**d**) UH2013. (**e**) UH2018.

(5) Efficiency Analysis: To verify the efficiency, we recorded the time cost and computational burden (FLOPs) of the different methods. The results are reported in Table 11. As can be seen, recent works built a complex network to improve classification, resulting in a large time cost and computational burden. This disadvantage is aggravated when a multi-scale architecture (e.g., MDGCN) or composite mode is used (e.g., WFCG and AMGCFN). Based on an entirely different approach, capsule networks learn an exact representation using activity vectors, enabling them to achieve SOTA classification results with an extremely simple architecture. Therefore, CNet relies on few calculations with 0.34 G FLOPs. By combining CapsNet with an attention mechanism, ATT-CNet and SA-CNet increase the calculation quantity to some extent but this is still much lower than for most recent works. Better than all the other methods, the proposed CAN needs a much lower calculation quantity of 0.18 G FLOPs since it only uses 1/5 spectral signatures to achieve SOTA classification, which means our CAN needs much less time to yield the classification results, especially when tested on large-scale datasets (as can be seen in the time cost on UH 2018). Our CAN can achieve SOTA by using limited spectral signatures due to the unique attention design for HSI data and the primary activity vectors. The huge merits of accuracy and efficiency demonstrate that our CAN has wide industrial application prospects.

Table 11. Results of efficiency experiments. Numbers in bold denote the best value. Default unit is second. "m" and "h" denote minute and hour, respectively.

Methods	Running Time (s)										Complexity
	IP		SA		PU		UH2013		UH2018		(FLOPs)
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	
3D CNN	98.27	5.36	93.6	17.24	48.27	18.92	37.28	16.48	-	-	0.69 G
RSSAN	125.32	4.28	89.03	9.25	56.39	14.66	43.75	12.18	-	-	0.85 G
DBDA	582.36	14.36	270.31	34.1	82.15	10.24	69.32	8.26	-	-	1.54 G
MDGCN	528.25	3.27	239.48	2.98	119.47	4.18	82.32	3.27	-	-	6.80 G
SSFTT	98.26	2.68	74.24	8.17	59.23	9.22	48.29	5.63	-	-	2.59 G
GAHT	117.17	2.47	136.36	12.47	86.21	1.36	72.44	14.23	-	-	4.82 G
SF	460.78	42.36	490.58	240.14	210.51	178.36	176.24	106.34	-	-	1.50 G
WFCG	287.36	2.31	932.15	6.44	728.36	11.58	326.30	7.32	-	-	5.98 G
AMGCFN	153.22	5.34	184.12	6.92	132.75	13.62	78.65	11.22	-	-	6.32 G
GTFN	120.35	4.06	154.73	10.64	138.36	13.48	89.72	9.46	4.3 h	8.9 m	0.74 G
DSNet	104.23	3.72	136.62	8.74	121.03	11.46	101.75	7.32	-	-	0.72 G
MHSI	68.63	2.31	76.48	2.29	56.12	4.64	39.32	2.94	3.2 h	7.6 m	0.96 G
CNet	78.28	3.34	82.32	2.16	63.17	5.48	54.23	3.12	1.3 h	3.0 m	0.34 G
ATT-CNet	92.31	3.84	10.64	2.73	82.32	7.12	68.73	2.89	2.8 h	7.3 m	0.86 G
SA-CNet	83.73	3.06	76.13	3.29	76.08	6.92	62.38	2.74	2.1 h	6.1 m	0.64 G
CAN	**53.72**	**2.12**	**65.32**	**1.89**	**43.32**	**3.48**	**48.26**	**2.04**	**52 m**	**2.5 m**	**0.18 G**

4.4. Ablation Experiment and Model Analysis

(1) Ablation Experiment for Overall Framework: For our CAN, three modules termed PCA, AFE and SWM are proposed to reduce the calculation and enhance the classification performance of CapsNet. To explore the contribution of them, this part reports an ablation experiment on the IP dataset. The results are reported in Table 12. As can be seen, using the standard CapsNet achieved high results on all the three metrics, only slightly lower than some recent SOTA methods. By distinguishing the contribution of pixels, the proposed AFE effectively improves the feature representation of HSI data, which increases all the metrics especially when using PCA to achieve classification with a limited spectral dimension. More importantly, the proposed SWM makes an appealing improvement by enhancing the primary activity vectors with a self-taught mechanism. By using both of them, the proposed method achieves somewhat better results. In particular, it can be seen that although PCA dramatically reduces the calculation of the proposed CAN, it drops the performance to some extent. Fortunately, with the proposed AFE and SWM, our proposed method obtains the second-best results, only slightly lower than those generated without using PCA. Therefore, we prefer to use PCA for accelerating our proposed method. All the three components enable our proposed CAN to achieve SOTA performance with extremely little calculation.

Table 12. Results of ablation experiment.

Method	PCA	AFE	SWM	OA (%)	AA (%)	KAPPA
CAN	×	×	×	92.87	94.62	93.06
	×	✓	×	94.04	96.24	94.47
	×	×	✓	97.03	97.54	96.86
	✓	×	×	90.22	92.16	90.17
	✓	✓	×	93.42	95.68	93.26
	✓	×	✓	96.73	97.48	96.54
	×	✓	✓	**98.15**	**98.73**	**98.04**
	✓	✓	✓	97.91	98.52	97.60

(2) The Study of AFE: The proposed AFE plays an important role in improving the classification performance. Therefore, we explore its effectiveness in this part. First, we compare AFE with the spatial attention mechanism used in DBDA [12] and CBAM [50]. The results are reported in Table 13. As can be seen, using AFE achieves the best results. Compared with DBDA, it it is seen that using multi-branches to obtain the attention map is not necessary. Compared with CBAM, our AFE effectively improves the OA from 96.74 to 97.91 since the introduced convolution and ReLU enhance the feature-building ability. Moreover, like CBAM, we also performed an experiment using channel-wise attention (CA) before AFE. The results show a quite limited gain. A main reason is that PCA has discarded most bands, which means that the remaining bands are all important and it is not necessary to weight them.

Table 13. Results by comparing AFE with other attention mechanisms.

Methods	DBDA	CBAM	AFE	CA + AFE
OA	97.64	96.74	97.91	97.98

The kernel size of AFE was studied further. As can be seen in Table 14, the results show that using a smaller kernel generates better OA. A very possible reason is that HSI classification is a pixel classification task. Therefore, using 1×1 kernel better weights the pixels.

Table 14. Results by changing the kernel size of AFE.

Kernel_Size	1×1	3×3	5×5	7×7
OA	97.91	97.45	97.07	96.98

(3) The Study of AWM: The proposed AWM is designed to adaptively weight different capsule convolutions, thus generating primary activity vectors with more effective representation. In this part, we show that using AWM makes the learned features more discriminative. We use *t*-sne to show the feature distribution with (w/o) SWM. The results are shown in Figure 11. As can be seen, after using SWM, the proposed CAN learns more compact features on all the datasets, which results in improved classification performance. Therefore, weighting the capsule convolutions is a feasible approach to improving the performance of CapsNet.

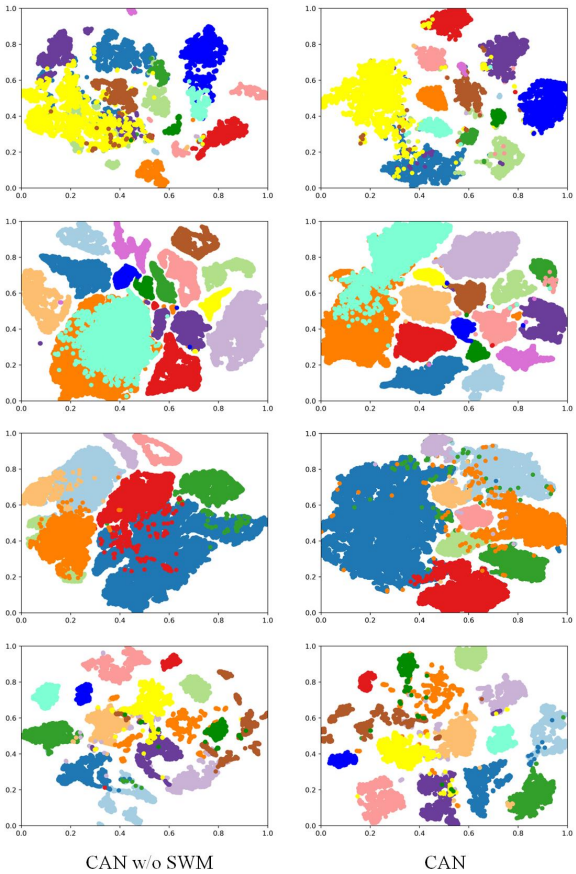

Figure 11. The visual feature distributions using *t*-sne. From the first row to the fourth row, the results of AP, SA, PU, UH2013, respectively, are shown.

(4) Intermediate Result Analysis: How fast can the proposed CAN achieve satisfactory classification results is an interesting question. To analyze the problem, the OA curves are drawn with the recorded values after each 10 epochs. As can be seen in Figure 12,

for the SA dataset, our CAN stably obtains a high OA value after 100 epochs; while for the IP, PU and UH2013 datasets, it obtains a high OA value after 200 epochs. These intermediate results show that our CAN is highly efficient, being free from large iterations for data training. In particular, although the UH2018 dataset is large, we find that our CAN improves the performance faster than when tested on PU and UH2013, with nearly 80% OA after 50 epochs. Therefore, we set the total training epoch as 300. Compared with the SOTA HSI classification methods, this represents an obvious advantage since many other methods [24,26,27,41] need more than 500 epochs for training, and some (e.g., [22]) need even more than 1000 epochs.

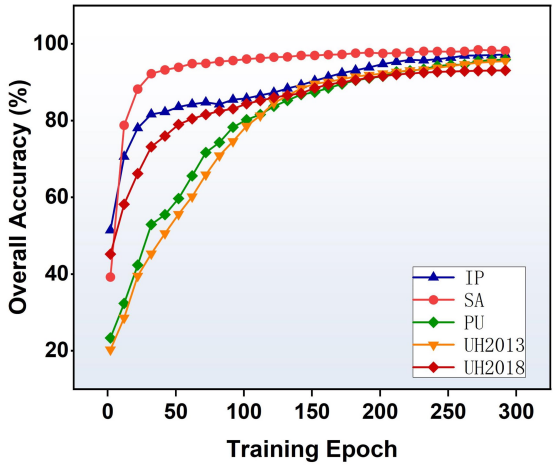

Figure 12. OA curves on the datasets.

(5) Sensitivity Analysis of the Vector Dimension: For our CAN, the dimensions of the primary activity vector and the class activity vector are key variables, which should be set in advance. To analyze the sensitivity of these two variables, a grid searching strategy is used. The results are shown in Figure 13. As can be seen, the classification performance (OA) is not sensitive to the setting of these two vector dimensions. However, readers should note that the dimension of the primary activity vector (q_p) should set a larger value than that of the class activity vector (q_c). Otherwise the performance might be dropped to a low value. $q_p < q_c$ should be regarded as incorrect settings since more property features cannot be produced out of thin air when conveying the information of the activity vectors.

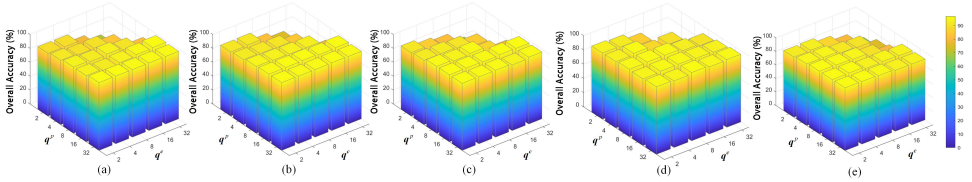

Figure 13. Sensitivity analysis for the dimensions of the primary activity vector (q^p) and the class activity vector (q^c) on the datasets. (**a**) IP. (**b**) SA. (**c**) PU. (**d**) UH2013. (**e**) UH2018.

5. Conclusions

This paper proposes CAN, a capsule attention network for HSI classification. By using activity vectors, our CAN yields more abstract and exact representations for entities. To fully cultivate the advantage of CapsNet, AFE is proposed to extract rich and useful features from spatial-spectral input. Furthermore, SWM is subtly designed to better yield primary activity vectors by self-weighting all the capsule convolutions, which ensures high-quality class activity vectors for entity representation. Experiments on the HSI datasets showed that our CAN surpasses 13 SOTA methods by a large margin with a much lower time cost and computational burden.

Author Contributions: Conceptualization, N.W.; methodology, N.W.; validation, N.W. and A.Y.; formal analysis, N.W. and A.Y.; writing—original draft, N.W. and Z.C.; writing—review and editing, Z.C., Y.D., Y.X. and Y.S.; supervision, Y.S.; project administration, Y.S.; funding acquisition, Z.C. and Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Natural Science Foundation of Shaanxi Province under Grant 2020JQ-298 and 2023-JC-YB-501.

Data Availability Statement: The original data presented in the study are openly available and we also provide them with the code of CAN. The code will be published at https://github.com/NianWang-HJJGCDX/CAN.git (accessed on 23 October 2024).

Acknowledgments: The authors would like to thank D. Landgrebe at Purdue University for providing the free downloads of the Indian Pines and Salinas datasets, and P. Gamba from the University of Pavia for providing the Pavia University dataset.

Conflicts of Interest: The authors declare no conflicts of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; nor in the decision to publish the results.

References

1. Zhang, H.; Liu, H.; Yang, R.; Wang, W.; Luo, Q.; Tu, C. Hyperspectral Image Classification Based on Double-Branch Multi-Scale Dual-Attention Network. *Remote Sens.* **2024**, *16*, 2051. [CrossRef]
2. Guo, Y.; Han, S.; Li, Y.; Zhang, C.; Bai, Y. K-nearest neighbor combined with guided filter for hyperspectral image classification. In *Procedia Computer Science*; Elsevier: Amsterdam, The Netherlands, 2018; pp. 159–165.
3. Zhang, B.; Li, S.; Jia, X.; Gao, L.; Peng, M. A daptive Markov random field approach for classification of hyperspectral imagery. *IEEE Geosci. Remote Sens. Lett.* **2011**, *8*, 973–977. [CrossRef]
4. Amini, S.; Homayouni, S.; Safari, A.; Darvishsefat, A.A. Object-based classification of hyperspectral data using random forest algorithm. *Geo-Spat. Inf. Sci.* **2018**, *21*, 127–138. [CrossRef]
5. Melgani, F.; Bruzzone, L. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Trans. Geosci. Remote Sens.* **2004**, *42*, 1778–1790. [CrossRef]
6. Li, W.; Prasad, S.; Fowler, J.E. Hyperspectral image classification using Gaussian mixture models and Markov random fields. *IEEE Geosci. Remote Sens. Lett.* **2013**, *11*, 153–157. [CrossRef]
7. Islam, M.R.; Islam, M.T.; Uddin, M.P.; Ulhaq, A. Improving Hyperspectral Image Classification with Compact Multi-Branch Deep Learning. *Remote Sens.* **2024**, *16*, 2069. [CrossRef]
8. Hu, W.; Huang, Y.; Wei, L.; Zhang, F.; Li, H. Deep convolutional neural networks for hyperspectral image classification. *J. Sens.* **2015**, *2015*, 258619. [CrossRef]
9. Ben Hamida, A.; Benoit, A.; Lambert, P.; Ben Amar, C. 3-D Deep Learning Approach for Remote Sensing Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 4420–4434. [CrossRef]
10. Jia, S.; Jiang, S.; Zhang, S.; Xu, M.; Jia, X. Graph-in-Graph Convolutional Network for Hyperspectral Image Classification. *IEEE Trans. Neural Networks Learn. Syst.* **2024**, *35*, 1157–1171. [CrossRef]
11. Zhu, M.; Jiao, L.; Liu, F.; Yang, S.; Wang, J. Residual spectral–spatial attention network for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2020**, *59*, 449–462. [CrossRef]
12. Li, R.; Zheng, S.; Duan, C.; Yang, Y.; Wang, X. Classification of hyperspectral image based on double-branch dual-attention mechanism network. *Remote Sens.* **2020**, *12*, 582. [CrossRef]
13. Ding, Y.; Zhang, Z.; Zhao, X.; Cai, W.; Yang, N.; Hu, H.; Huang, X.; Cao, Y.; Cai, W. Unsupervised self-correlated learning smoothy enhanced locality preserving graph convolution embedding clustering for hyperspectral images. *IEEE Geosci. Remote Sens.* **2022**, *60*, 5536716. [CrossRef]
14. Zhang, X.; Shang, S.; Tang, X.; Feng, J.; Jiao, L. Spectral partitioning residual network with spatial attention mechanism for hyperspectral image classification. *IEEE Geosci. Remote Sens.* **2021**, *60*, 5507714. [CrossRef]
15. Ahmad, M.; Khan, A.M.; Mazzara, M.; Distefano, S.; Roy, S.K.; Wu, X. Hybrid dense network with attention mechanism for hyperspectral image classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2022**, *15*, 3948–3957. [CrossRef]
16. Yang, A.; Li, M.; Wu, Z.; He, Y.; Qiu, X.; Song, Y.; Du, W.; Gou, Y. CDF-net: A convolutional neural network fusing frequency domain and spatial domain features. *IET Comput. Vision* **2023**, *17*, 319–329. [CrossRef]
17. Xue, Y.; Jin, G.; Shen, T.; Tan, L.; Wang, N.; Gao, J.; Wang, L. SmallTrack: Wavelet pooling and graph enhanced classification for UAV small object tracking. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 5618815. [CrossRef]
18. Ding, Y.; Zhang, Z.; Zhao, X.; Cai, Y.; Li, S.; Deng, B.; Cai, W. Self-Supervised Locality Preserving Low-Pass Graph Convolutional Embedding for Large-Scale Hyperspectral Image Clustering. *IEEE Geosci. Remote Sens.* **2022**, *60*, 5536016. [CrossRef]
19. Chen, Y.; Lin, Z.; Zhao, X.; Wang, G.; Gu, Y. Deep Learning-Based Classification of Hyperspectral Data. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2014**, *7*, 2094–2107. [CrossRef]

20. Hang, R.; Liu, Q.; Hong, D.; Ghamisi, P. Cascaded Recurrent Neural Networks for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 5384–5394. [CrossRef]
21. Hong, D.; Gao, L.; Yao, J.; Zhang, B.; Plaza, A.; Chanussot, J. Graph Convolutional Networks for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2021**, *59*, 5966–5978. http://arxiv.org/abs/2008.02457. [CrossRef]
22. Wan, S.; Gong, C.; Zhong, P.; Du, B.; Zhang, L.; Yang, J. Multiscale Dynamic Graph Convolutional Network for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2020**, *58*, 3162–3177. [CrossRef]
23. Wu, K.; Zhan, Y.; An, Y.; Li, S. Multiscale Feature Search-Based Graph Convolutional Network for Hyperspectral Image Classification. *Remote Sens.* **2024**, *16*, 2328. [CrossRef]
24. Sun, L.; Zhao, G.; Zheng, Y.; Wu, Z. Spectral–Spatial Feature Tokenization Transformer for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5522214. [CrossRef]
25. Mei, S.; Song, C.; Ma, M.; Xu, F. Hyperspectral Image Classification Using Group-Aware Hierarchical Transformer. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5539014. [CrossRef]
26. Hong, D.; Han, Z.; Yao, J.; Gao, L.; Zhang, B.; Plaza, A.; Chanussot, J. SpectralFormer: Rethinking Hyperspectral Image Classification with Transformers. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5518615. [CrossRef]
27. Li, Y.; Luo, Y.; Zhang, L.; Wang, Z.; Du, B. MambaHSI: Spatial-Spectral Mamba for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 5524216. [CrossRef]
28. Achanta, R.; Shaji, A.; Smith, K.; Lucchi, A.; Fua, P.; Süsstrunk, S. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 2274–2282. [CrossRef]
29. Zhao, H.; Zhou, F.; Bruzzone, L.; Guan, R.; Yang, C. Superpixel-level global and local similarity graph-based clustering for large hyperspectral images. *IEEE Trans. Geosci. Remote Sens.* **2021**, *60*, 5519316. [CrossRef]
30. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
31. Wang, X.; Zhu, M.; Bo, D.; Cui, P.; Shi, C.; Pei, J. AM-GCN: Adaptive Multi-channel Graph Convolutional Networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Long Beach, CA, USA, 6–10 August 2020; pp. 1243–1253.
32. Li, G.; Muller, M.; Thabet, A.; Ghanem, B. Deepgcns: Can gcns go as deep as cnns? In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9267–9276.
33. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the NaacL-HLT, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
34. He, K.; Fan, H.; Wu, Y.; Xie, S.; Girshick, R. Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Washington, DC, USA, 14–19 June 2020; pp. 9729–9738.
35. Xue, Y.; Jin, G.; Shen, T.; Tan, L.; Wang, N.; Gao, J.; Wang, L. Consistent Representation Mining for Multi-Drone Single Object Tracking. *IEEE Trans. Circuits Syst. Video Technol.* **2024**. [CrossRef]
36. Grill, J.B.; Strub, F.; Altché, F.; Tallec, C.; Richemond, P.; Buchatskaya, E.; Doersch, C.; Avila Pires, B.; Guo, Z.; Gheshlaghi Azar, M.; et al. Bootstrap your own latent-a new approach to self-supervised learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 21271–21284.
37. Liu, Q.; Xiao, L.; Yang, J.; Wei, Z. CNN-enhanced graph convolutional network with pixel-and superpixel-level feature fusion for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2020**, *59*, 8657–8671. [CrossRef]
38. Dong, Y.; Liu, Q.; Du, B.; Zhang, L. Weighted Feature Fusion of Convolutional Neural Network and Graph Attention Network for Hyperspectral Image Classification. *IEEE Trans. Image Process.* **2022**, *31*, 1559–1572. [CrossRef] [PubMed]
39. Zhou, H.; Luo, F.; Zhuang, H.; Weng, Z.; Gong, X.; Lin, Z. Attention multi-hop graph and multi-scale convolutional fusion network for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 5508614.
40. Yang, A.; Li, M.; Ding, Y.; Hong, D.; Lv, Y.; He, Y. GTFN: GCN and transformer fusion with spatial-spectral features for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 6600115. [CrossRef]
41. Han, Z.; Yang, J.; Gao, L.; Zeng, Z.; Zhang, B.; Chanussot, J. Dual-Branch Subpixel-Guided Network for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2024**. [CrossRef]
42. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic routing between capsules. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 3859–3869.
43. Paoletti, M.E.; Haut, J.M.; Fernandez-Beltran, R.; Plaza, J.; Plaza, A.; Li, J.; Pla, F. Capsule networks for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2018**, *57*, 2145–2160. [CrossRef]
44. Kumar, D.; Kumar, D. A spectral–spatial 3D-convolutional capsule network for hyperspectral image classification with limited training samples. *Int. J. Inf. Technol.* **2023**, *15*, 379–391. [CrossRef]
45. Xiaoxia, Z.; Xia, Z. Attention based Deep Convolutional Capsule Network for Hyperspectral Image Classification. *IEEE Access* **2024**, *12*, 56815–56823. [CrossRef]
46. Paoletti, M.E.; Moreno-Álvarez, S.; Haut, J.M. Multiple Attention-Guided Capsule Networks for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–20. [CrossRef]
47. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
48. Zhai, H.; Zhao, J. Two-Stream spectral-spatial convolutional capsule network for Hyperspectral image classification. *Int. J. Appl. Earth Obs. Geoinf.* **2024**, *127*, 103614. [CrossRef]

49. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141.
50. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
51. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.