remote sensing

MDPI

Article

An Efficient Graph Convolutional RVFL Network for Hyperspectral Image Classification

Zijia Zhang [1,2], Yaoming Cai [3], Xiaobo Liu [4], Min Zhang [5] and Yan Meng [1,2,*]

[1] School of Artificial Intelligence, Hubei University, Wuhan 430062, China; zijiazhang@hubu.edu.cn
[2] Key Laboratory of Intelligent Sensing System and Security, Hubei University, Ministry of Education, Wuhan 430062, China
[3] School of Information and Safety Engineering, Zhongnan University of Economics and Law, Wuhan 430073, China; caiyaom@zuel.edu.cn
[4] School of Automation, China University of Geosciences, Wuhan 430074, China; xbliu@cug.edu.cn
[5] Department of Land Surveying and Geo-Informatics, The Hong Kong Polytechnic University, Hong Kong, China; lsgi-min.zhang@polyu.edu.hk
* Correspondence: mengyan@hubu.edu.cn

Abstract: Graph convolutional networks (GCN) have emerged as a powerful alternative tool for analyzing hyperspectral images (HSIs). Despite their impressive performance, current works strive to make GCN more sophisticated through either elaborate architecture or fancy training tricks, making them prohibitive for HSI data in practice. In this paper, we present a Graph Convolutional RVFL Network (GCRVFL), a simple but efficient GCN for hyperspectral image classification. Specifically, we generalize the classic RVFL network into the graph domain by using graph convolution operations. This not only enables RVFL to handle graph-structured data, but also avoids iterative parameter adjustment by employing an efficient closed-form solution. Unlike previous works that perform HSI classification under a transductive framework, we regard HSI classification as a graph-level classification task, which makes GCRVFL scalable to large-scale HSI data. Extensive experiments on three benchmark data sets demonstrate that the proposed GCRVFL is able to achieve competitive results with fewer trainable parameters and adjustable hyperparameters and higher computational efficiency. In particular, we show that our approach is comparable to many existing approaches, including deep CNN models (e.g., ResNet and DenseNet) and popular GCN models (e.g., SGC and APPNP).

Keywords: graph convolutional network; graph-level classification; hyperspectral image; RVFL network

Citation: Zhang, Z.; Cai, Y.; Liu, X.; Zhang, M.; Meng, Y. An Efficient Graph Convolutional RVFL Network for Hyperspectral Image Classification. *Remote Sens.* **2024**, *16*, 37. https://doi.org/10.3390/rs16010037

Academic Editor: Paul Scheunders

Received: 8 October 2023
Revised: 17 December 2023
Accepted: 18 December 2023
Published: 21 December 2023

1. Introduction

Hyperspectral images (HSIs) are characterized by rich spectral and spatial information that consists of hundreds of narrow spectral bands. This enables us to distinguish physical objects at the pixel level. Over recent decades, hyperspectral imaging has become one of the most important earth observation technologies [1]. HSI classification has also become an active interdisciplinary topic that centralizes artificial intelligence and remote sensing [2,3].

Early HSI classification methods such as support vector machine (SVM) [4] focus on either handcrafted shallow-level features or linear separable problems, usually leading to poor robustness. Recently, representation learning centralized by deep neural networks (DNN) has garnered great success in the remote sensing community [5–7]. In particular, convolutional neural networks (CNN) [8] have become one of the most representative learning paradigms for HSI classification and building extraction [9,10] due to their powerful ability to automatically extract spectral and spatial information.

We broadly divide convolution-based models into Euclidean models and non-Euclidean models. Euclidean models (e.g., CNN) define convolution on the Euclidean space, and thus

they only can handle regular inputs such as images, texts, and sequences. The state-of-the-art Euclidean convolutional models include residual network (ResNet) [11–13], dense network (DenseNet) [14–16], attention mechanism [17–19], and so on. The non-Euclidean models aim at generalizing convolution into the graph domain. These types of models are also referred to as graph convolutional networks (GCN) [20,21]. Since Euclidean data can be regarded as special cases of graph data, GCN is a more general learning paradigm than CNN, especially learning for structural information. Based on this insight, many works have attempted to revisit HSI classification from the point of view of graph representation learning.

Utilizing GCN for HSI classification offers several advantages. Firstly, it serves as a viable alternative to CNN, providing a more efficient approach for aggregating information from neighboring pixels. Secondly, GCN demonstrates robustness to spectral and spatial variations in noise, owing to its consideration of long-range interdependencies. Thirdly, GCN comes with a complete theoretical guarantee, incorporating spectral graph theory, which proves valuable for model analysis and interpretation. In general, there are two important problems when applying GCN to HSI classification. The first problem is how to convert the Euclidean HSI data into graph data. A common idea is to construct a similarity graph for the whole sample set (e.g., the *k*NN graph) [22], which leads to a node-level transductive semi-supervised learning problem. However, the computation of a pixel-level graph is usually prohibitive for large-scale HSI data due to its high computational space requirements and time complexity. To alleviate the drawback, many studies adopt a superpixel-level graph [23–25], which can greatly reduce the number of nodes. Nonetheless, such a process is limited not only by the absence of superpixel-level labels but also by the reliability of superpixel segmentation. Furthermore, Hong et al. [26] introduced MiniGCN by constructing a graph for a batch of HSI samples, enabling supervised training. Additionally, Zhang et al. [27] introduced a local regions graph for HSI classification, significantly reducing the graph's size.

The second problem is how to efficiently train a GCN model. The vast majority of existing GCN models rely on gradient descent. This often leads to local optimal solutions and vanishing gradient issues, requiring either elaborate architecture or fancy learning tricks. In addition, GCN usually suffers from the over-smoothing issue, in which node feature representations become indistinguishable in different classes, which worsens as the model depth increases. In other words, unnecessarily complex design may aggravate over-fitting when training data are limited.

Unlike the gradient-based neural networks which need iterative parameter updating, randomized neural networks [28,29], represented by random vector functional link networks (RVFL) [30], provide a simple but efficient learning scheme known as pseudoinverse learning [31]. In an RVFL network, the hidden layers serve as an important part, but their parameters can be generated randomly and kept fixed during training. This signifies that RVFL only needs to concentrate on the optimization of its task-specific output layer. Over the past decade, the RVFL network has made great progress with the explosion of artificial intelligence techniques. Its variants, such as extreme learning machine, have been widely used for HSI classification [32–34] due to their ease of use, training, and implementation.

However, RVFL networks are greatly limited by their representational ability when handling complex and variable HSI data, leading to poorer results compared to those from sophisticated CNNs and GCNs. This is because RVFL ignores important inductive biases in both architecture and learning mechanisms. On the one hand, nearly all previous RVFL approaches are designed for Euclidean data, thus failing to deal with structured information efficiently. On the other hand, RVFL uses fully connected architecture without parameter sharing, which treats inputs as equally important components and is computationally inefficient. Many studies have demonstrated that proper use of inductive biases makes learning algorithms more robust and effective. For example, the success of CNN and GCN can mainly be attributed to their local information aggregation mechanism (regular local receptive field in CNN [8] or irregular neighborhood aggregation in GCN [20]).

Based on these insights, we propose a simple yet effective RVFL network for HSI classification. We refer to the proposed approach as a Graph Convolutional RVFL network (GCRVFL). The core idea behind the approach is to generalize the classic RVFL to the non-Euclidean domain. Specifically, we introduce graph convolution into RVFL layers, thus deriving random graph convolution and a closed-form solution. In order to process large-scale HSI, we recast HSI classification as a graph-level classification task, resulting in an inductive GCRVFL model. Unlike previous transductive GCN-based HSI classification methods, our approach has better generalization ability and higher efficiency for large HSI. To this end, we construct HSI patches as graphs, where every pixel on a patch is seen as nodes over a graph, and edges are determined by distances between nodes. Furthermore, we introduce a global pooling operation into GCRVFL to generate graph-level representations. The proposed approach extends RVFL to the graph domain without losing its simplicity in training and implementation.

The rest of the paper is structured as follows. We first briefly review graph representation learning and RVFL networks in Section 2. Next, we introduce the details of the proposed GCRVFL method in Section 3. Finally, in Section 4, we systematically qualitatively and quantitively assess the proposed method, following with a conclusion in Section 5.

2. Related Works

2.1. Notations

In this paper, we use boldface lowercase italicized symbols (e.g., $\boldsymbol{x}$), boldface uppercase roman symbols (e.g., $\mathbf{X}$), regular italicized symbols (e.g., N), and calligraphy symbols (e.g., $\mathcal{G}$) to represent vectors, matrices, scalars, and sets, respectively. A graph structure is expressed as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ indicates the node set of the graph with $v_i \in \mathcal{V}$ and $|\mathcal{V}| = N$, $\mathcal{E}$ represents the edge set with $(v_i, v_j) \in \mathcal{E}$. We use $\mathbf{A} \in \mathbb{R}^{N \times N}$ to denote the adjacency matrix of $\mathcal{G}$. The diagonal degree matrix of $\mathcal{G}$ is defined as $\mathbf{D} \in \mathbb{R}^{N \times N}$, where $D_{ii} = \sum_j A_{ij}$. In the graph, $\mathbf{L} = \mathbf{D} - \mathbf{A}$ indicates the graph Laplacian matrix, and the corresponding normalized version is $\mathbf{L}_{sym} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$. Moreover, $\mathbf{X}^{\mathrm{T}}$ denotes the transpose of matrix $\mathbf{X}$, and $\mathbf{I}_N$ denotes an identity matrix with the size of N.

2.2. GCN

GCN [20,35] is a powerful graph representation learning architecture that extends CNN [8] to the graph domain. The existing GCN models can be divided into two main classes, i.e., spectral GCNs and spatial GCNs. Kift et al. [36] proposed the vanilla GCN based on simplifying the approximation of the graph Laplacian using the Chebyshev expansion method [37]. Technically, a typical GCN consists of a rectified linear unit activation (ReLU) and a softmax classifier, which can be written as:

$$\mathbf{H} = \mathrm{Softmax}\big(\tilde{\mathbf{A}}\ \mathrm{ReLU}\big(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_0\big)\mathbf{W}_1\big). \tag{1}$$

Here, $\tilde{\mathbf{A}} = (\mathbf{D} + \mathbf{I}_N)^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I}_N)(\mathbf{D} + \mathbf{I}_N)^{-\frac{1}{2}}$ denotes a renormalized adjacency matrix with self-loops; $\mathbf{W}_1$ and $\mathbf{W}_2$ are trainable filter parameter matrices, which are optimized using stochastic gradient descent. GCN models often suffer from the over-smoothing problem when increasing the model depth, which means node feature representations become indistinguishable in different classes [38]. Thus, most GCN models usually adopt a shallow structure. In general, the vanilla GCN achieves its best performance with a two-layer structure.

2.3. RVFL

RVFL is one of the most classical random neural networks, which is a special form of the feedforward neural network [28]. The classical RVFL contains two stages: random mapping and ridge regression. Mathematically, given input data $\mathbf{X} \in \mathbb{R}^{N \times m}$ consisting of N samples each with m features, the random mapping can be expressed as

$$\mathbf{H} = \sigma(\mathbf{X}\mathbf{W} + \boldsymbol{b}), \tag{2}$$

where $\mathbf{H} \in \mathbb{R}^{N\times L}$ denotes the L-dimensional nonlinear mapping matrix of N samples, σ is an activation function, $\mathbf{W} \in \mathbb{R}^{m\times L}$ represents a random hidden weight matrix, and $\boldsymbol{b} \in \mathbb{R}^{L}$ indicates a bias vector. Due to RVFL containing parameter-free direct links between the input layer and the output layer, the hidden layer output of RVFL can be considered the concatenation of the random mapping and the initial input, i.e., $\tilde{\mathbf{H}} \in \mathbb{R}^{N\times(L+m)} = \text{concat}(\mathbf{H}, \mathbf{X})$.

Then, the ridge regression layer of RVFL takes the hidden layer output $\tilde{\mathbf{H}}$ as input and predicts the corresponding labels. Thus, the output layer can be expressed in the following form:

$$\mathbf{Y} = \mathbf{H}\boldsymbol{\beta}. \tag{3}$$

Here, $\boldsymbol{\beta} \in \mathbb{R}^{L\times C}$ represents the output weight matrix with C classes, and $\mathbf{Y} \in \mathbb{R}^{N\times C}$ denotes the prediction matrix. In order to obtain the unique optimal solution $\boldsymbol{\beta}$, Equation (3) can be solved as follows:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{Y}, \tag{4}$$

where $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalized inverse of the matrix $\mathbf{H}$. We note that RVFL overcomes the problems of low training efficiency by avoiding iterative parameter tuning, which is the difference from the general gradient-based training method.

3. Materials and Method

3.1. Overall Framework

Before elaborating on the details, we introduce the overall framework of the proposed approach. Given HSI data, $\mathbf{X}_{cube} \in \mathbb{R}^{W\times H\times m}$, composed of m spectral bands and $N = W \times H$ pixels, our goal is to assign each pixel a certain land cover type. To this end, we first convert the HSI classification task into a graph-level classification problem. Then, we use a GVRVFL network to efficiently aggregate spectral and structural information between nodes.

The overall framework of our approach is shown in Figure 1. The core idea is to generalize the classic RVFL to the non-Euclidean domain by using graph convolution operations, such that it can naturally aggregate the neighborhood structure. Our GCRVFL consists of a random graph convolutional layer and a graph embedding layer followed by a global pooling operation. In the following, we focus on the two main concerns: (1) how to represent samples into a series of graphs, and (2) how to process graphs using the proposed GCRVFL.

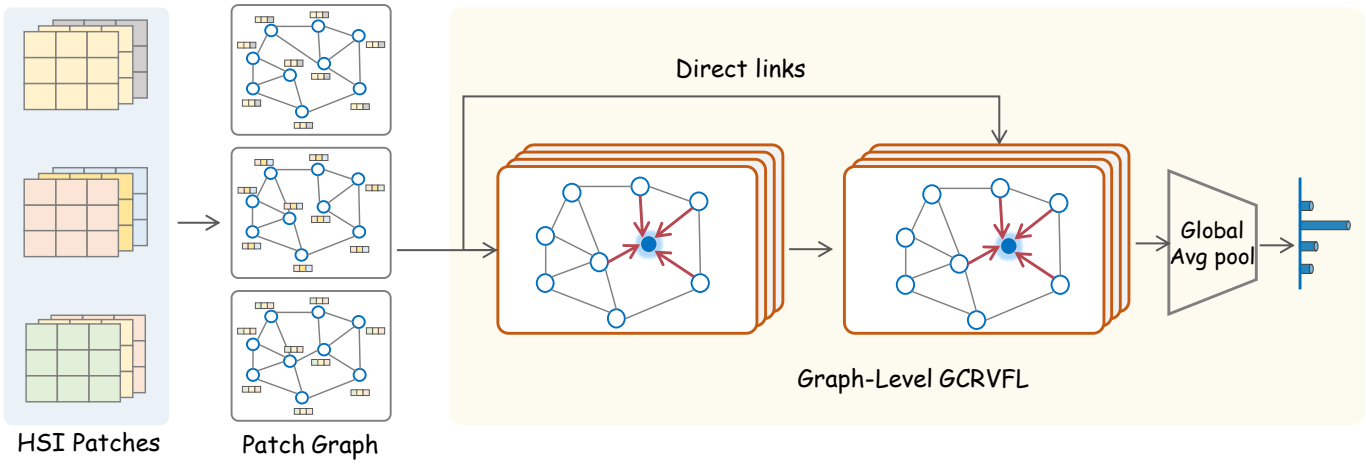

Figure 1. Overview of the proposed GCRVF approach. We treat HSI classification as a graph-level classification problem, where each HSI patch is converted into a series of patch graphs. We use these graphs to train an efficient graph-level GCRVFL network by extending RVFL into the graph domain.

3.2. Graph Construction

Existing GCN-based HSI classification methods focus on the node-level transductive learning task, thus having poorer generalization ability and requiring an exponential increase in the memory space. To avoid this problem, we aim to construct a local graph on each HSI patch separately instead of constructing one global graph of the whole HSI. Specifically, we take $s \times s$ small patches neighboring each central pixel as the sample set, denoted as $\mathcal{X} = \{\mathbf{X}_i \in \mathbb{R}^{s\times s\times m}\}_{i=1}^{N}$. Then, we transform each HSI patch $\mathbf{X}_i$ into a graph representation $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ by constructing a K-nearest neighbors (KNN) graph. As illustrated in Figure 2, we regard $n = s \times s$ pixels in $\mathbf{X}_i$ as the node set $\{v_j\} \in \mathcal{V}_i$ over a graph, where each node contains an m-dimensional node feature vector, and the edge (neighborhood relationships) set $\mathcal{E}_i$ is determined by node similarity. Formally, we define the adjacent matrix $\mathbf{A}^{(i)}$ of the graph $\mathcal{G}_i$ as

$$A_{jk}^{(i)} = \begin{cases} 1, & \text{if } v_k \in \mathcal{N}_K(v_j) \text{ or } v_j \in \mathcal{N}_K(v_k) \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Here, $\mathcal{N}_K(\cdot)$ denotes neighborhood set with size of K. For example, $\mathcal{N}_K(v_j)$ indicates K neighbors of node v_j. In this paper, we determine the neighborhoods by computing the Euclidean distance between node feature vectors. As a result, the sample set is converted into a set of graphs, i.e., $\mathcal{X} = \{\mathcal{G}_i\}_{i=1}^{N}$, while the pixel-wise classification task is transformed into a graph-wise classification task that requires a classifier with the capability of handling structured data. For this reason, the traditional RVFL networks fail in this case.

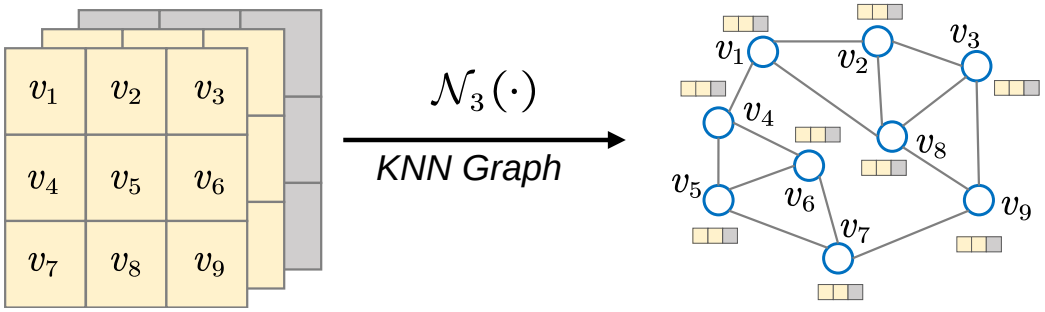

Figure 2. Example of constructing patch graph with patch size of 3×3 and $K = 3$.

3.3. Random Graph Convolution

Now, we are ready to describe the GCRVFL network by following the same notations of RVFL and GCN. The key to the GCRVFL method is to extend the classical RVFL in the graph domain while maintaining its backbone. Suppose we have $N_{\mathcal{T}}$ training graphs. For convenience, we use a bigger graph $\mathcal{G}$ to indicate all training graphs, where every single graph is an isolated subgraph of $\mathcal{G}$ and the total number of nodes is $n^{N_{\mathcal{T}}}$. Further, the adjacent matrix of $\mathcal{G}$ can be denoted as a block diagonal matrix, i.e., $\mathbf{A} = \text{diag}([\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_N])$, and the node feature matrix is $\mathbf{X} = \left[\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_{N_{\mathcal{T}}}\right]^{\mathrm{T}}$. This enables us to process graphs in a batch-wise manner.

Parallel to the random mapping in the classical RVFL, the goal of the random graph convolution is to embed input graph $\mathcal{G}$ into a matrix, denoted as $\mathbf{H}$. We let $\mathbf{W} \in \mathbb{R}^{m\times L}$ be a filter parameter matrix containing L filters. All elements of $\mathbf{W}$ are randomly generated according to a random probability distribution and kept fixed during training. We define the random graph convolution as follows:

$$\mathbf{H} = \sigma(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}), \tag{6}$$

where $\tilde{\mathbf{A}} = (\mathbf{D} + \mathbf{I}_N)^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I}_N)(\mathbf{D} + \mathbf{I}_N)^{-\frac{1}{2}}$ is the normalized adjacent matrix, and σ represents a nonlinear function. In this paper, we use ReLU as σ, which is defined as $\sigma(\boldsymbol{x}) = max(0, \boldsymbol{x})$.

Similar to the classic RVFL, we concatenate the random graph embedding with the initial node features along the channel dimension as the hidden layer output, i.e., $\mathbf{H} = \text{concat}(\mathbf{H}, \mathbf{X})$. This process is known as direct links, and ensures that the final output is determined by both graph embedding and the initial input. According to the skip connection that is frequently used to build deep neural networks [39], the operation behaves like a regularizer to simplify the layer-by-layer information passing, thereby reducing the risk of over-fitting and over-smoothing.

3.4. Graph Convolutional Regression

In order to obtain a graph-level prediction, we generalize the regression classifier of RVFL into the graph domain. We call the resulting regression model graph convolutional regression. The graph convolutional regression is defined as

$$\mathbf{Y} = \rho\left(\tilde{\mathbf{A}}\mathbf{H}\right)\boldsymbol{\beta}. \tag{7}$$

Here, ρ is a global pooling function, which is permutation-invariant, such as an average or sum over nodes feature. In this paper, we use a global average pooling, denoted by $\rho(\mathcal{G}_i) = \frac{1}{n}\sum_j \boldsymbol{x}_j$. The only trainable parameter matrix in our GCRVFL model is $\boldsymbol{\beta}$. We solve $\boldsymbol{\beta}$ by rewriting Equation (7) as the following optimization problem:

$$\arg\min_{\beta} \frac{1}{2}\left\|\hat{\mathbf{H}}\boldsymbol{\beta} - \mathbf{Y}\right\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_F^2, \tag{8}$$

where $\hat{\mathbf{H}} = \rho\left(\tilde{\mathbf{A}}\mathbf{H}\right)$ is the graph-level representation of the input graphs and λ denotes a regularization coefficient. According to the least squares theorem, Equation (8) has a globally optimal closed-form solution.

By computing the partial derivative of Equation (8) with respect to $\boldsymbol{\beta}$, we derive

$$0 = \hat{\mathbf{H}}^{\mathrm{T}}\hat{\mathbf{H}}\boldsymbol{\beta} + \lambda\boldsymbol{\beta} - \hat{\mathbf{H}}^{\mathrm{T}}\mathbf{Y}. \tag{9}$$

This further obtains the closed-form solution as

$$\boldsymbol{\beta}^* = \left(\hat{\mathbf{H}}^{\mathrm{T}}\hat{\mathbf{H}} + \lambda\mathbf{I}_{N_{\mathcal{T}}}\right)^{-1}\hat{\mathbf{H}}^{\mathrm{T}}\mathbf{Y}. \tag{10}$$

For an unlabeled HSI patch $\mathbf{X}_{\mathcal{U}}$, we predict its label by GCRVFL as $\boldsymbol{y} = \hat{\boldsymbol{h}}\boldsymbol{\beta}^*$. We provide the step-by-step pseudo code of the proposed approach in Algorithm 1. It can be seen that there is no iterative operation in our method, making it efficient and easy to train, implement, and apply.

Algorithm 1: GCRVFL

Input: HSI cube: $\mathbf{X}_{cube}$, K, L, and λ.

1 Construct patch graphs according to Equation (5);
2 Randomly generate graph filters $\mathbf{W}$;
3 Calculate random graph embedding $\mathbf{H} = \text{concat}\left(\sigma\left(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}\right), \mathbf{X}\right)$;
4 Calculate graph-level representation $\hat{\mathbf{H}} = \rho\left(\tilde{\mathbf{A}}\mathbf{H}\right)$;
5 Solve graph convolutional regression $\boldsymbol{\beta}^* = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \breve{}\mathbf{I}_{\mathbf{N}_{\mathcal{T}}}\right)^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{Y}$;
6 Predict test labels.

Output: Labels of test set.

3.5. Connection to Existing Methods

In this section, we discuss the connections between our method and existing methods. Their visual comparison is provided in Figure 3.

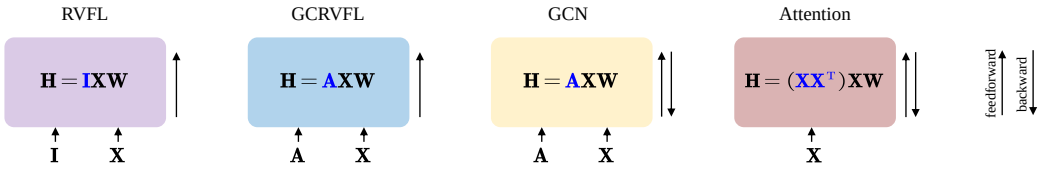

Figure 3. Connections between RVFL, GCRVFL, and Attention. These methods hold a similar feedforward process, but adopt different prior structures (i.e., **I**, **A**, and **XX**) and different optimization schemes (closed-form solution or backpropagation).

3.5.1. GCRVFL vs. RVFL

Our GCRVFL is a generalized version of the classic RVFL network in the graph domain. Thus, GCRVFL inherits all the advantages of RVFL, including its fast learning speed and extremely simple architecture. However, our GCRVFL greatly expands the capability of RVFL. More precisely, the classic RVFL can only handle grid data, while our design enables RVFL to deal with structural data. This is mainly beneficial in the graph convolution operations. Since the weights between nodes are shared, GCRVFL does not significantly increase the complexity compared to RVFL. Furthermore, because of the robustness of the graph structure, GCRVFL is more robust than the classic RVFL.

3.5.2. GCRVFL vs. GCN

Both GCRVFL and GCN follow the spectral graph convolution theory. Furthermore, the graph convolution used in both of them plays a low-pass filter role [40]. However, there are two main differences between GCRVFL and GCN. First, GCRVFL trains parameters by computing a closed-form solution, while GCN needs a greedily iterative gradient descent with necessary training tricks, e.g., batch normalization [36] and an elaborate optimizer. Thus, GCRVFL simplifies the training of GCN by using fewer trainable parameters and hyperparameters. Second, GCRVFL includes an entirely random graph convolutional layer. Thus, it usually needs sufficient hidden neurons to avoid overfitting and random noise.

3.5.3. GCRVFL vs. Attention Mechanism

From the point of view of the attention mechanism, our GCRVFL can be regarded as a spatial attention-induced RVFL network. For each input HSI patch, the matrix manipulation of $\tilde{\mathbf{A}}_i$ multiplied by $\mathbf{X}_i$, i.e., $\tilde{\mathbf{A}}_i\mathbf{X}_i$, is equivalent to a spatial attention process, where $\tilde{\mathbf{A}}_i = \mathbf{X}_i\mathbf{X}_i^T$. That is, $\tilde{\mathbf{A}}_i$ is a normalized attention score matrix, which indicates the pair-wise importance between input pixels. Due to the sparsity of $\tilde{\mathbf{A}}_i$, those trivial pixels are filtered while embedding in GCRVFL is being calculated. This property of GCRVFL is important for HSI classification, especially for the patches consisting of mixed-class pixels. It should be noted that our attention score matrix can be precomputed, making it more efficient than a trainable attention mechanism.

3.6. Data Sets

We selected three HSI data sets to assess GCRVFL. Notably, utilizing patched samples from a single HSI introduces overlaps between training and test sets, leading to potential information leakage during training. To mitigate this concern, we employed non-overlapping public partitions as recommended in [6]. The chosen data sets are as follows:

- Houston 2013: This data set was acquired by the ITRES CASI-1500 sensor over the University of Houston campus and the neighboring urban area [41]. The Houston imagery consists of 349 × 1905 samples and the spatial resolution is 2.5 m by pixel. It has 144 spectral bands in the 380 nm to 1050 nm region, and contains 15 classes with 15,029 labeled samples. Table 1 lists 15 challenging land-cover and land-use categories as well as the number of training and testing samples. Figure 4a shows a false-color image and the map of training and testing samples.
- Indian Pines 2010: This data set was captured by the ProSpecTIR sensor over Purdue University, Indiana in 2010, and includes a variety of different crops. The Indian

Pines 2010 imagery consists of 445 × 750 samples and the spatial resolution is 2 m by pixel. This scene contains 360 spectral bands ranging from 400 to 2450 nm, and is composed of 16 classes with 198,074 labeled samples. Table 2 shows 16 different land cover categories and the number of training and testing samples. Figure 4b depicts the false-color scene and the corresponding visualization of training and testing samples.

- Salinas: This data set was collected by AVIRIS sensors over Salinas Valley, California. The Salinas imagery consists of 512 × 217 samples and the spatial resolution is 3.7 m by pixel. It has 224 spectral bands in the 400 nm to 2500 nm region, and includes 16 classes with 54,129 labeled samples. Table 3 presents the number of training and testing samples with 16 different classes; those samples' visualizations are exhibited in Figure 4c.

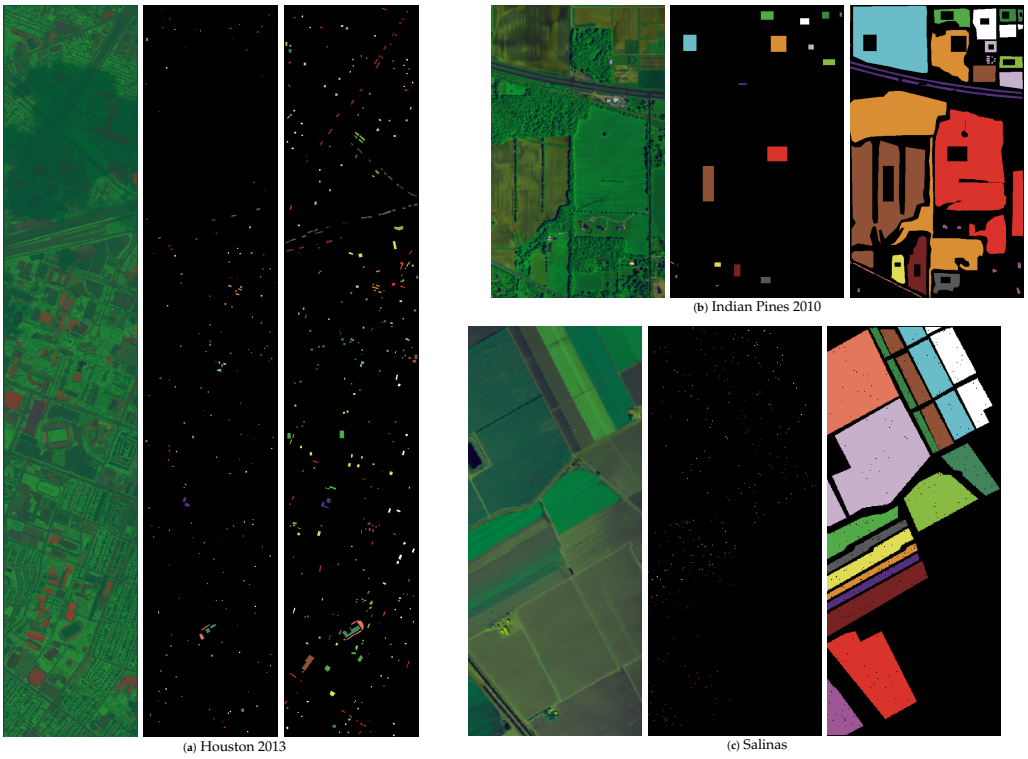

Figure 4. Visualization of the (**a**) Houston 2013, (**b**) Indian Pines 2010, and (**c**) Salinas data sets. For each data set, the first, second, and third images denote the false-color image, the training map, and the testing map, respectively. Note that the Houston 2013 and Salinas images were rotated 90 degree for better presentation.

Table 1. Land-Cover classes of the Houston 2013 data set with the standard training and testing sets for each class.

Class No.	Class Name	Training	Testing
1	Healthy Grass	198	1053
2	Stressed Grass	190	1064
3	Synthetic Grass	192	505
4	Tree	188	1056
5	Soil	186	1056
6	Water	182	143
7	Residential	196	1072
8	Commercial	191	1053
9	Road	193	1059
10	Highway	191	1036
11	Railway	181	1054

Table 1. *Cont.*

Class No.	Class Name	Training	Testing
12	Parking Lot1	192	1041
13	Parking Lot2	184	285
14	Tennis Court	181	247
15	Running Track	187	473
Total		2832	12,197

Table 2. Land-Cover classes of the Indian Pines 2010 data set with the standard training and testing sets for each class.

Class No.	Class Name	Training	Testing
1	Corn_high	726	2661
2	Corn_mid	465	1275
3	Corn_low	66	290
4	Soy_bean_high	324	1041
5	Soy_bean_mid	2548	35,317
6	Soy_bean_low	1428	27,782
7	Residues	368	5427
8	Wheat	182	3205
9	Hay	1938	48,107
10	Grass/Pasture	496	5048
11	Cover_crop_1	400	2346
12	Cover_crop_2	176	1988
13	Woodlands	1640	46,919
14	Highway	105	4758
15	Local road	52	450
16	Buildings	40	506
Total		10,954	187,120

Table 3. Land-Cover classes of the Salinas data set with the standard training and testing sets for each class.

Class No.	Class Name	Training	Testing
1	Brocoli_green_weeds_1	20	1989
2	Brocoli_green_weeds_2	20	3706
3	Fallow	20	1956
4	Fallow_rough_plow	20	1374
5	Fallow_smooth	20	2658
6	Stubble	20	3939
7	Celery	20	3559
8	Grapes_untrained	20	11,251
9	Soil_vinyard_develop	20	6183
10	Corn_senesced_green_weeds	20	3258
11	Lettuce_romaine_4wk	20	1048
12	Lettuce_romaine_5wk	20	1907
13	Lettuce_romaine_6wk	20	896
14	Lettuce_romaine_7wk	20	1050
15	Vinyard_untrained	20	7248
16	Vinyard_vertical_trellis	20	1787
Total		320	53,809

4. Experiments

In this section, we first conduct extensive experiments for analyzing the hyper-parameters involved in GCRVFL. Subsequently, we assess the performance of our proposed GCRVFL, comparing it with numerous classic and state-of-the-art approaches, across three

HSI data sets. Each method undergoes evaluation ten times with distinct random seeds, and the reported metric is the average classification accuracy. To ensure a thorough performance evaluation, we employ three widely recognized evaluation metrics: overall accuracy (OA), average accuracy (AA), and the kappa coefficient (Kappa).

4.1. Analysis of Hyper-Parameter Sensitivity

4.1.1. Impact of the Number of Hidden Neurons L

The number of hidden neurons is one of the most important hyper-parameters in neural networks. For this purpose, we exhibit the OA changing trends of GCRVFL with the different number of hidden neurons L on the Houston 2013, Indian Pines 2010, and Salinas data sets (Figure 5). In this experiment, we vary the number of hidden neurons from 2^3 to 2^{10} with an interval of a power of two. Figure 5 illustrates that, on the whole, GCRVFL tends to achieve better accuracy as the number of hidden neurons increases, although the changing trend of OA on Indian Pines 2010 is relatively slow and gentle. However, performance is degraded when the number of hidden neurons is larger than 2^{10}. This phenomenon reveals that it was the dense hidden layer with limited labeled samples that caused the over-fitting problem of the model.

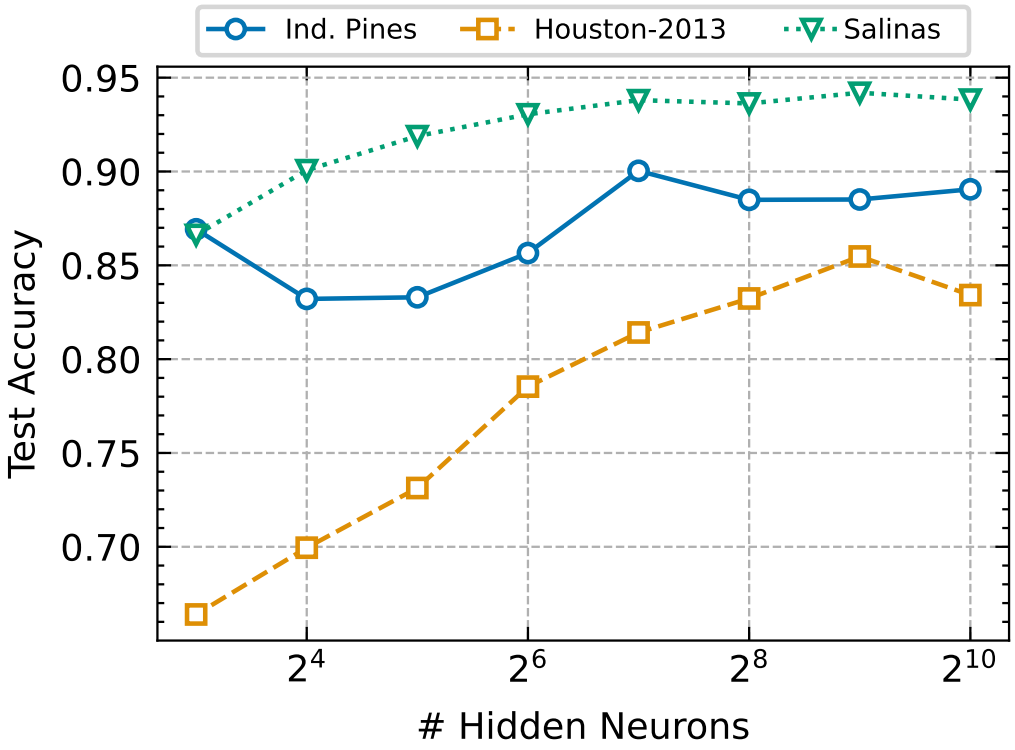

Figure 5. Influence of the hyper-parameter L on the Houston 2013, Indian Pines 2010 (i.e., Ind. Pines), and Salinas data sets.

4.1.2. Impact of λ

Figure 6 exhibits the impact of hyper-parameter λ on the Houston 2013, Indian Pines 2010, and Salinas data sets. We find that the regulation coefficient has a certain influence on HSI classification on the whole. More specifically, GCRVFL is insensitive to regulation coefficient λ when its value is less than 2^{-1}, whereas the classification accuracy of GCRVFL degrades on the Houston 2013 and Salinas data sets when λ is more than 2^{-1}. This is because too large of a value for the regularization coefficient λ leads to the problem of under-fitting for GCRVFL. Although the performance trend of GCRVFL on the Indian Pines 2010 data set appears to be gentle when λ varies in the range of $[2^{-8}, 2^{-7}, \cdots, 2^{4}]$, experience shows that λ should be chosen within $[2^{-5}, 2^{-1}]$.

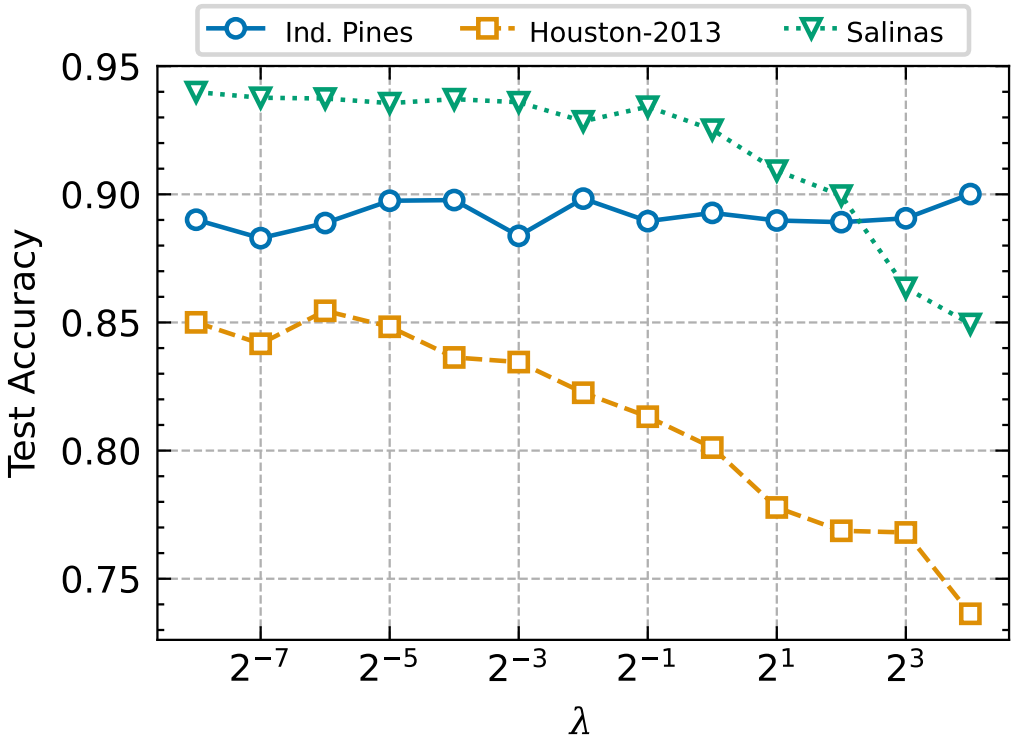

Figure 6. Influence of the hyper-parameter λ on the Houston 2013, Indian Pines 2010 (i.e., Ind. Pines), and Salinas data sets.

4.1.3. Impact of the Number of Neighbors K

The number of neighbors is also a crucial hyper-parameter for HSI classification performance. We explore the performance trend of our proposed GCRVFL with different numbers of the nearest neighbors K on the Salinas data set. Figure 7 shows the experimental results as a violin plot, where $K = [0, 3, 5, \cdots, 30, 40]$. In Figure 7, the white node denotes the median of classification accuracy, the black bar represents the first quartile to third quartile data (i.e., 25∼75%), and the upper and lower bound of the violin plot indicate the maximum and minimum value of obtained results. As can be seen, varying parameter K yields a slight performance improvement. When K is selected as 20, GCRVFL achieves the best accuracy. However, when K is greater than 30, the performance of GCRVFL begins to decline. The reason for this deterioration of performance is that the constructed graph collects noise edges, resulting in the inaccurate judgment of the information. For HSI data, parameter K signifies the property of the constructed graph, thus its value should not be selected to be too large.

4.1.4. Impact of Activation Function

The results of GCRVFL with different activation functions on the three HSI data sets are listed in Table 4. We use Sigmoid, Tahn, and ReLu as activation functions, and also carry out the experiment without an activation function. From the value of the three metrics in Table 4, it can be clearly seen that GCRVFL with the ReLu function generally outperforms the models using other activation functions by a substantial margin. It can also be seen that the classification performance of GCRVFL without using any activation function is the worst on all three HSI data sets. Therefore, ReLu is chosen as an activation function in all our experiments.

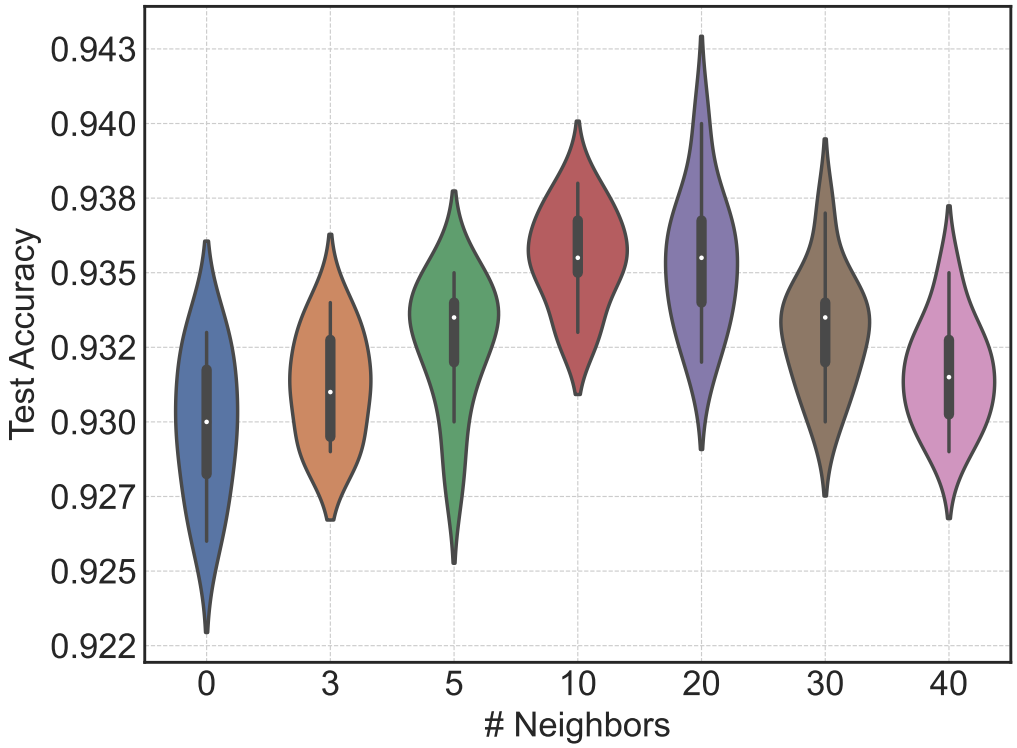

Figure 7. Influence of hyper-parameter K on the Salinas data set.

Table 4. Performance comparison of GCRVFL under different activation functions.The best results are in bold.

Data Set	Metrics	Sigmoid	Tahn	None	ReLU
Houston	OA	56.63 ± 0.02	76.26 ± 0.55	51.94 ± 4.02	**84.78 ± 1.24**
	AA	60.10 ± 0.02	79.65 ± 0.50	56.31 ± 3.38	**87.16 ± 1.00**
	Kappa × 100	53.50 ± 0.00	74.30 ± 0.57	48.68 ± 4.10	**83.52 ± 1.35**
Ind. Pines	OA	79.61 ± 0.00	80.81 ± 0.11	79.48 ± 0.00	**89.21 ± 0.32**
	AA	48.70 ± 0.01	61.90 ± 0.51	52.06 ± 0.00	**87.34 ± 0.05**
	Kappa × 100	74.70 ± 0.00	76.43 ± 0.12	74.80 ± 0.00	**86.80 ± 0.40**
Salinas	OA	74.92 ± 4.28	85.76 ± 1.57	73.21 ± 2.08	**92.46 ± 0.06**
	AA	77.60 ± 1.52	90.84 ± 0.80	72.81 ± 1.59	**96.51 ± 0.03**
	Kappa × 100	72.03 ± 4.54	84.13 ± 1.76	70.10 ± 2.16	**91.60 ± 0.10**

4.1.5. Impact of the Size of Patch

The size of the input image patch has important significance for neural networks. We conduct the experiment on the three HSI data sets with different input patch sizes. As depicted in Figure 8, GCRVFL tends to obtain better test accuracy results on the whole when the input patch size grows. However, the classification performance falls off on the Houston 2013 data set when the input patch size is set to 11 × 11. This is due to the fact that a larger image patch inevitably contains pixels of other classes, resulting in some negative influence on HSI classification. Furthermore, valuable spatial information can not be captured efficiently when the patch size is rather small. Therefore, for the appropriate use of spatial information, we set the size of the input image patch to 7 × 7 on the three HSI data sets in our experiments.

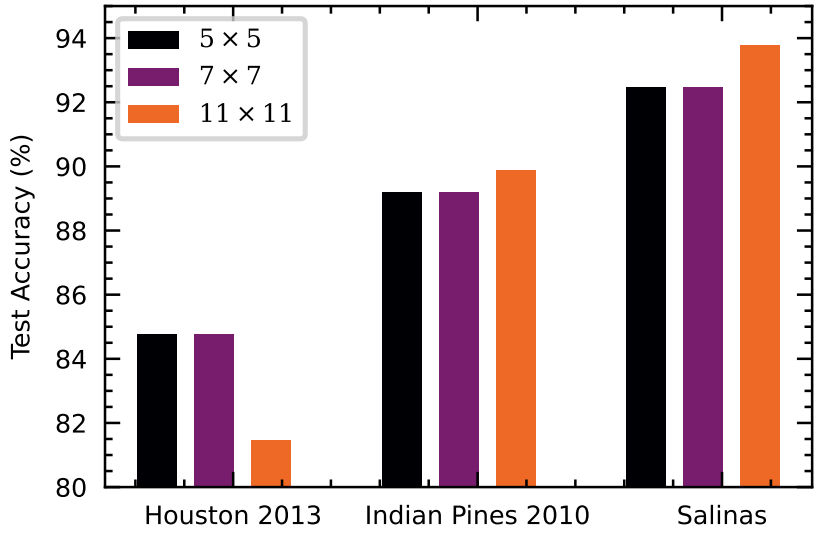

Figure 8. Influence of the size of patch on the Houston 2013, Indian Pines 2010, and Salinas data sets.

4.1.6. Influence of the Number of Principal Components

In this section, we explore the performance of our proposed model with different numbers of principal components (PCs). In Figure 9a, we show the variance ratio of the top 10 PCs on the three data sets. We can see that the first PC provides over 65% variance ratio for all data sets, while top 2 PCs reach 95%. It preserves a 99.9% variance ratio when using 10 PCs. As illustrated in Figure 9b, PC-3, PC-6, and PC-10 denote the principal components that we preserved. When ten principal components are employed to condense the spectral information, our proposed model obtains the best results on the three HSI data sets. If the number of principal components selected is lower than ten, the performance of our proposed model tends to be decreased. This is because the model cannot exploit the discriminant and nonlinear features effectively, and thus some valuable spectral information is ignored. As a result, in our experiments, ten principal components are chosen in our model for better performance.

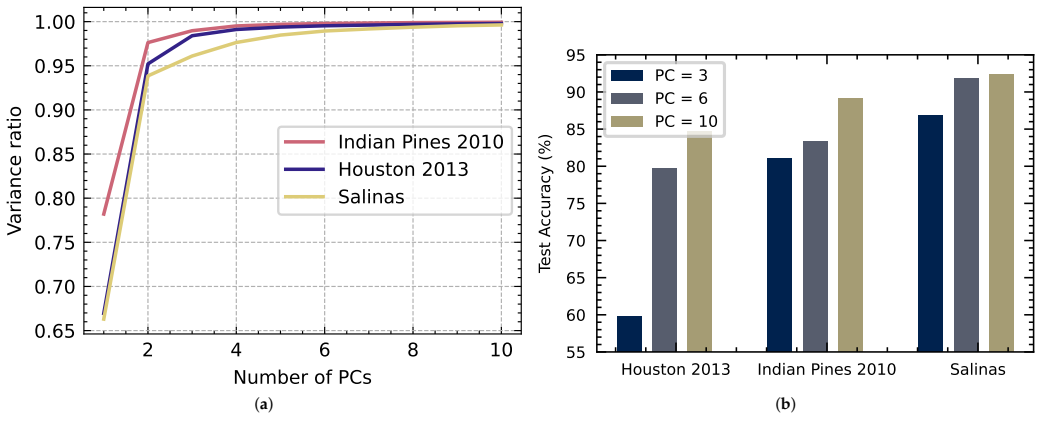

Figure 9. Influence of different principal components on the Houston 2013, Indian Pines 2010, and Salinas data sets. (**a**) Variance ratio of top 10 PCs. (**b**) Classification with varying PCs.

4.2. Main Results

4.2.1. Baselines and Setup

We compare the proposed GCRVFL model with nine existing baselines: three classical classification methods—random forest (RF) [42], support vector machine (SVM) [43], and RVFL [44]; three regular convolution based methods—CNN-2D [45], ResNet [39], and

DenseNet [46]; and three popular graph convolutional learning methods—GCN [36], SGC [40], and APPNP [47]. The parameter configurations for these methods are detailed below.

- RF: 200 decision trees are used in a RF classifier.
- SVM: The implementation of scikit-learn is used, where the radial basis function (RBF) kernel is adopted. Fivefold cross-validation is utilized to select the two hyperparameters, σ and λ. The kernel width σ varies in the range of $\left[2^{-4}, 2^{-3}, \cdots, 2^{4}\right]$, and the regularization coefficient λ varies in the range of $\left[10^{-2}, 10^{-1}, \cdots, 10^{4}\right]$.
- RVFL: The number of neurons is set as 512, and the regularization coefficient λ is tuned by grid searching in the range of $\left[10^{-4}, 10^{-1}, \cdots, 10^{3}\right]$.
- CNN-2D: Two 2D convolutional blocks are used followed by a fully connected layer with 512 neurons and a softmax classifier. Each convolutional block involves a 2D conventional layer, a batch normalization layer, a max-pooling layer, and a ReLU activation layer. The filters used in the convolutional layers are $3 \times 3 \times 64$ and $3 \times 3 \times 128$, respectively.
- ResNet: The network consists of two residual blocks, each containing two convolutional layers with $3 \times 3 \times 64$ kernels. The same classification head as that of CNN-2D is used.
- DenseNet: Two dense blocks are used in the model. Each block consists of two convolutional layers with $3 \times 3 \times 64$ kernels.
- GCN: A graph convolutional layer is adopted with 512 neurons followed by a ReLu activation in our case. The learning rate is set as 0.002, and the number of training epochs is set to 500.
- SGC: A simplified convolution is used with a feature propagation of 5 steps, followed by a softmax classifier.
- APPNP: For the APPNP, the teleport probability α of PageRank is set to 0.1, the number of power iteration steps is set to 5, and the other network configurations are the same as those of GCN and SGC.

We configure the hyperparameters for the proposed GCRVFL based on the sensitivity analysis conducted in the previous subsection. The specific settings for the three data sets are provided in Table 5. All baseline methods are implemented using Python 3.5 and assessed on hardware comprising an Intel i5-6500 3.20 GHz CPU with 8 GB RAM and an NVIDIA GeForce 1050 GPU with 4 GB RAM.

Table 5. Settings of hyperparameters in GCRVFL.

Data Sets	Houston 2013	Indian Pines 2010	Salinas
L	512	512	512
λ	0.005	0.05	0.005
K	5	5	5
s	7	7	7

In data preprocessing, we perform PCA to reduce spectral bands into 10 by preserving at least 99% of the cumulative percentage of variance (CPV). We construct spectral–spatial samples by using a neighborhood with an input size of 7×7 pixels for all HSI data sets. All samples are standardized by scaling feature values into $[0, 1]$ before use.

4.2.2. Quantitative Results

The first experiment is conducted on the Houston 2013 data set. The quantitative results obtained by different algorithms are shown in Table 6. For each row, results in bold font indicate the best performance. As we can observe, the classical classifiers (e.g., RF, SVM, and RVFL) achieve similar performance. Benefiting from the powerful ability of deep learning techniques, CNN-2D, ResNet, DenseNet, GCN, SGC, and APPNP exhibit significantly higher classification accuracy than the three conventional classifiers. However, SGC shows the worst classification performance because it removes the nonlinear

transformation and compresses the weight matrix between convolution layers. Based on evaluation metrics OA, AA, and Kappa, the proposed GCRVFL achieves the best results. In particular, the proposed model achieves 100% on the Soil, Tennis Court, and Running Track classes. Furthermore, GCRVFL greatly improves RVFL by 6.8, 6.82, and 7.37 percentage points for OA, AA, and Kappa evaluation metrics, respectively. These results demonstrate that graph convolution is beneficial to improving the learning capability of the classical RVFL on HSI classification to a great extent.

The second experiment is carried out on the Indian Pines 2010 data set. Table 7 shows the quantitative results obtained by different methods. For the traditional classifiers, RVFL has slight advantages over RF but nevertheless is inferior to SVM. There is a similar trend between CNN-2D, ResNet, DenseNet, GCN, and APPNP in classification performance. The five algorithms based on deep learning are superior to the three traditional classifiers. GCRVFL outperforms all competitors in terms of OA, AA, and Kappa. Specifically, GCRVFL has significant performance improvement on those challenging classes with few training samples (e.g., Local road and Buildings). These results demonstrate that global optimal solutions obtained by GCRVFL promote the model's excellent learning ability, which is its principal difference from comparison algorithms that can only obtain local approximate solutions. Moreover, graph convolution is efficient for capturing the spectral–spatial information of HSI.

The third experiment is performed on the Salinas data set. The quantitative results obtained by different models are summarized in Table 8. Despite using fewer training samples, GCRVFL also achieves the best results among all competitors in terms of three evaluation metrics compared to other deep learning models. It should be noted that RF obtains better results than ResNet and DenseNet. This phenomenon indicates that the deep learning model requires more labeled samples in the training procedure to achieve better performance, while the RF based on ensemble learning is less dependent on the proportion of training samples. GCN, SGC, and APPNP also show good classification performance, which signifies that the efficient feature extraction ability of graph convolution on the data will be helpful for HSI classification.

4.2.3. Qualitative Comparison of Different Methods

In order to further evaluate the classification results, we visualize the classification maps obtained by different approaches on the Houston 2013, Indian Pines 2010, and Salinas data sets, as depicted in Figures 10–12. In general, traditional classifiers (e.g., RF, SVM, and RVFL) yield poor classification maps on the three data sets due to more salt and pepper noise, while the other deep learning-based models generate smoother maps. More specifically, the classification maps obtained by APPNP have fewer noisy points than CNN-2D, ResNet, and DenseNet on the Indian Pines 2010 and Salinas data sets. Taken overall, our proposed GCRVFL obtains smoother and cleaner maps with desirable edge details in comparison with other approaches. This indicates that GCRVFL's graph convolution technique makes it easier to extract the features of HSI data; furthermore, the acquisition of a global optimal solution contributes to achieving robust representations, thereby generating a more accurate classification map. Moreover, due to the characteristic of the spatial attention mechanism, some trivial pixels or mixed-class pixels are filtered when calculating the embedding of GCRVFL. This is also an important reason why the classification maps obtained by GCRVFL perform better among all competitors.

Table 6. Classification results by different methods on the Houston 2013 data set. The best results are in bold.

Class No.	RF	SVM	RVFL	CNN-2D	ResNet	DenseNet	GCN	SGC	APPNP	GCRVFL
1	82.05 ± 0.00	82.01 ± 0.05	82.71 ± 0.12	82.68 ± 0.36	82.89 ± 0.21	82.99 ± 0.17	**83.10 ± 0.00**	82.86 ± 0.05	**83.10 ± 0.00**	81.90 ± 0.92
2	83.58 ± 0.04	83.46 ± 0.16	83.00 ± 0.33	83.52 ± 1.14	83.87 ± 0.99	84.33 ± 0.76	83.78 ± 0.49	83.55 ± 0.47	**85.01 ± 0.05**	83.53 ± 1.35
3	**99.74 ± 0.09**	99.74 ± 0.13	99.60 ± 0.18	96.24 ± 1.47	95.56 ± 1.59	94.61 ± 1.92	98.06 ± 0.34	93.86 ± 0.00	97.23 ± 0.00	99.72 ± 0.10
4	86.99 ± 0.04	87.07 ± 0.29	**90.16 ± 0.47**	87.92 ± 1.61	85.15 ± 3.95	86.78 ± 4.02	84.02 ± 0.80	82.77 ± 1.33	86.93 ± 0.09	89.03 ± 0.82
5	97.60 ± 0.12	97.62 ± 0.18	97.94 ± 0.10	99.70 ± 0.23	99.77 ± 0.31	99.90 ± 0.20	99.83 ± 0.09	99.01 ± 0.14	99.86 ± 0.05	**100.00 ± 0.00**
6	95.80 ± 0.57	**96.15 ± 0.78**	94.41 ± 0.83	94.55 ± 1.43	95.17 ± 1.51	94.83 ± 2.41	95.24 ± 0.28	90.21 ± 0.00	95.10 ± 0.00	95.80 ± 0.00
7	**82.56 ± 0.80**	82.26 ± 0.62	70.68 ± 1.12	72.36 ± 2.60	78.71 ± 4.29	77.93 ± 2.82	75.17 ± 1.24	70.43 ± 0.09	76.49 ± 0.93	74.59 ± 1.16
8	41.09 ± 0.43	42.01 ± 1.51	48.16 ± 7.04	64.71 ± 9.08	**75.95 ± 12.37**	68.93 ± 13.56	71.36 ± 1.44	48.58 ± 0.43	69.37 ± 0.14	48.36 ± 2.14
9	72.18 ± 0.89	72.08 ± 0.63	65.49 ± 0.87	74.04 ± 1.87	74.27 ± 3.30	76.51 ± 3.64	81.70 ± 1.33	46.55 ± 1.51	81.40 ± 1.89	**89.86 ± 2.83**
10	55.02 ± 2.58	54.39 ± 1.11	53.36 ± 2.84	59.11 ± 7.75	63.30 ± 14.31	55.73 ± 4.55	49.15 ± 1.29	62.50 ± 0.14	48.46 ± 0.29	**93.32 ± 5.58**
11	88.96 ± 0.24	**89.58 ± 0.72**	85.72 ± 1.18	78.93 ± 1.62	76.12 ± 1.99	79.22 ± 3.76	77.93 ± 0.27	55.03 ± 0.38	77.80 ± 0.66	83.68 ± 1.77
12	84.05 ± 1.51	85.30 ± 1.18	81.92 ± 2.03	91.19 ± 3.16	93.99 ± 2.03	**95.31 ± 1.51**	93.18 ± 0.68	63.45 ± 0.34	94.00 ± 0.53	85.42 ± 6.28
13	73.22 ± 0.17	72.67 ± 1.14	57.09 ± 1.28	81.54 ± 3.14	77.30 ± 3.55	81.30 ± 2.52	76.77 ± 0.68	69.65 ± 0.18	74.74 ± 1.40	**82.18 ± 3.41**
14	99.19 ± 0.00	99.11 ± 0.40	97.04 ± 0.45	98.91 ± 0.60	99.07 ± 1.43	98.46 ± 2.01	98.79 ± 0.00	97.37 ± 0.20	99.19 ± 0.00	**100.00 ± 0.00**
15	96.97 ± 0.10	97.12 ± 0.14	97.82 ± 0.30	96.49 ± 1.89	95.60 ± 1.76	95.62 ± 2.45	99.03 ± 0.22	92.18 ± 0.85	97.46 ± 0.42	**100.00 ± 0.00**
OA	79.69 ± 0.23	79.84 ± 0.13	77.98 ± 0.53	81.41 ± 1.31	82.98 ± 1.61	82.47 ± 1.06	81.93 ± 0.15	72.20 ± 0.23	82.08 ± 0.11	**84.78 ± 1.24**
AA	82.60 ± 0.16	82.71 ± 0.15	80.34 ± 0.38	84.13 ± 1.12	85.12 ± 1.27	84.83 ± 0.98	84.47 ± 0.17	75.87 ± 0.13	84.41 ± 0.17	**87.16 ± 1.00**
Kappa × 100	77.97 ± 0.25	78.15 ± 0.14	76.15 ± 0.57	79.93 ± 1.40	81.60 ± 1.72	81.09 ± 1.12	80.50 ± 0.17	70.15 ± 0.25	80.65 ± 0.15	**83.52 ± 1.35**

Table 7. Classification results by different methods on the Indian Pines 2010 data set. The best results are in bold.

Class No.	RF	SVM	RVFL	CNN-2D	ResNet	DenseNet	GCN	SGC	APPNP	GCRVFL
1	87.51 ± 0.29	85.23 ± 0.00	91.33 ± 0.55	90.74 ± 1.70	85.54 ± 2.41	82.79 ± 5.31	**94.10 ± 0.04**	80.91 ± 5.22	94.29 ± 0.11	90.77 ± 1.22
2	90.82 ± 0.17	91.14 ± 0.00	65.12 ± 3.48	99.39 ± 0.28	99.68 ± 0.60	97.59 ± 3.29	**100.00 ± 0.00**	99.96 ± 0.04	**100.00 ± 0.00**	**100.00 ± 0.00**
3	97.24 ± 0.00	95.52 ± 0.00	97.24 ± 0.00	97.66 ± 1.14	99.07 ± 1.40	99.89 ± 0.16	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
4	71.89 ± 0.25	72.53 ± 0.00	48.25 ± 0.97	86.36 ± 0.84	86.41 ± 2.72	84.73 ± 2.45	**93.23 ± 0.34**	85.64 ± 2.83	92.60 ± 0.19	83.77 ± 0.19
5	72.76 ± 0.55	81.02 ± 0.00	**89.18 ± 0.26**	78.08 ± 1.38	80.71 ± 3.85	77.33 ± 4.02	75.10 ± 0.08	71.01 ± 1.22	76.58 ± 1.08	79.94 ± 0.82
6	87.13 ± 0.08	90.32 ± 0.00	74.53 ± 0.46	81.92 ± 1.51	92.04 ± 4.85	94.84 ± 3.17	**96.08 ± 0.01**	83.95 ± 4.28	94.33 ± 1.71	95.84 ± 1.45
7	53.06 ± 0.60	45.92 ± 0.00	29.32 ± 1.80	71.72 ± 1.45	**77.52 ± 2.69**	76.66 ± 4.54	75.26 ± 0.06	77.39 ± 2.89	75.47 ± 0.21	74.66 ± 0.15
8	25.34 ± 0.09	25.55 ± 0.00	23.57 ± 0.05	24.29 ± 0.33	25.95 ± 1.59	23.35 ± 2.54	23.31 ± 0.09	**41.05 ± 8.56**	23.49 ± 0.09	24.99 ± 0.00
9	71.13 ± 0.51	84.53 ± 0.00	85.89 ± 0.20	85.84 ± 0.76	82.88 ± 5.25	82.18 ± 1.62	86.00 ± 0.05	80.52 ± 4.17	86.07 ± 0.56	**86.73 ± 0.08**
10	84.50 ± 0.45	81.38 ± 0.01	60.95 ± 0.83	94.85 ± 0.73	89.22 ± 5.66	92.21 ± 2.25	95.04 ± 0.19	89.86 ± 2.46	95.00 ± 0.80	**97.89 ± 0.23**
11	66.98 ± 0.37	81.84 ± 0.00	39.23 ± 1.53	93.15 ± 0.46	94.06 ± 1.34	**95.68 ± 1.89**	86.68 ± 0.79	88.51 ± 3.01	86.40 ± 1.41	92.86 ± 0.11
12	**100.00 ± 0.00**	99.40 ± 0.00	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	99.52 ± 0.23	**100.00 ± 0.00**	**100.00 ± 0.00**
13	96.66 ± 0.10	96.51 ± 0.00	94.82 ± 0.39	95.52 ± 1.48	92.97 ± 3.65	96.43 ± 1.26	93.56 ± 0.50	90.08 ± 0.75	93.98 ± 0.11	**97.97 ± 0.17**

Table 7. *Cont.*

Class No.	RF	SVM	RVFL	CNN-2D	ResNet	DenseNet	GCN	SGC	APPNP	GCRVFL
14	91.15 ± 0.14	86.07 ± 0.00	89.17 ± 0.09	98.62 ± 0.62	97.84 ± 1.60	98.85 ± 0.76	97.08 ± 0.34	98.81 ± 0.85	98.32 ± 0.25	**99.88 ± 0.01**
15	95.26 ± 0.28	43.56 ± 0.00	72.22 ± 1.19	98.00 ± 1.17	97.22 ± 3.04	94.15 ± 7.65	99.89 ± 0.11	99.33 ± 0.44	99.33 ± 0.00	**100.00 ± 0.00**
16	41.70 ± 1.70	1.38 ± 0.00	5.40 ± 0.99	21.34 ± 6.12	14.80 ± 10.54	5.01 ± 1.62	13.83 ± 1.38	13.54 ± 2.47	15.42 ± 1.78	**72.13 ± 4.15**
OA	80.42 ± 0.19	85.36 ± 0.00	82.82 ± 0.13	85.60 ± 0.39	86.15 ± 2.20	86.58 ± 0.94	86.74 ± 0.11	81.94 ± 0.21	86.92 ± 0.09	**89.21 ± 0.32**
AA	77.07 ± 0.21	72.62 ± 0.00	66.64 ± 0.39	82.34 ± 0.41	82.24 ± 0.81	81.36 ± 0.38	83.07 ± 0.10	81.25 ± 0.52	83.20 ± 0.03	**87.34 ± 0.05**
Kappa × 100	76.03 ± 0.21	82.00 ± 0.00	78.77 ± 0.17	82.54 ± 0.48	83.21 ± 2.59	83.67 ± 1.14	83.85 ± 0.15	78.25 ± 0.35	84.10 ± 0.10	**86.80 ± 0.40**

Table 8. Classification results by different methods on the Salinas data set. The best results are in bold.

Class No.	RF	SVM	RVFL	CNN-2D	ResNet	DenseNet	GCN	SGC	APPNP	Ours
1	98.70 ± 0.49	95.53 ± 0.88	93.27 ± 1.11	97.87 ± 2.11	98.54 ± 1.27	98.58 ± 1.29	99.21 ± 0.53	94.10 ± 5.19	99.66 ± 0.40	**99.87 ±0.03**
2	98.11 ± 1.82	91.68 ± 3.53	90.83 ± 2.69	94.27 ± 7.20	98.06 ± 2.30	97.54 ± 1.73	**99.96 ± 0.05**	98.05 ± 1.38	99.83 ± 0.17	99.53 ± 0.34
3	88.66 ± 5.50	82.43 ± 2.14	75.89 ± 2.87	97.48 ± 1.59	99.18 ± 0.62	97.99 ± 2.09	**99.69 ± 0.32**	90.49 ± 5.67	99.60 ± 0.42	99.31 ± 0.08
4	99.29 ± 0.35	98.66 ± 0.54	95.68 ± 2.46	99.05 ± 1.07	98.94 ± 0.60	99.23 ± 0.47	99.07 ± 0.51	97.44 ± 1.73	99.24 ± 0.25	**99.34 ± 0.22**
5	96.31 ± 0.88	87.98 ± 3.10	85.01 ± 3.45	93.12 ± 3.66	95.76 ± 0.87	96.06 ± 1.09	97.86 ± 1.88	94.06 ± 6.34	**98.21 ± 0.99**	96.82 ± 0.36
6	99.28 ± 0.48	99.59 ± 0.27	96.75 ± 2.28	99.30 ± 0.93	99.95 ± 0.09	99.84 ± 0.14	99.88 ± 0.22	**99.99 ± 0.01**	99.93 ± 0.12	99.86 ± 0.14
7	98.93 ± 0.23	98.83 ± 0.34	96.62 ± 1.16	96.94 ± 2.58	**99.76 ± 0.19**	98.77 ± 1.10	98.95 ± 0.97	95.18 ± 6.38	99.15 ± 0.69	99.13 ± 0.70
8	67.26 ± 5.90	57.85 ± 3.45	56.01 ± 4.84	71.46 ± 7.33	58.73 ± 14.83	58.56 ± 12.21	74.55 ± 9.95	71.68 ± 7.38	**83.78 ± 9.81**	79.67 ± 0.08
9	99.14 ± 0.36	95.18 ± 1.85	93.42 ± 2.26	98.39 ± 1.63	99.93 ± 0.11	99.28 ± 0.69	99.73 ± 0.34	96.84 ± 2.94	99.95 ± 0.06	**99.98 ± 0.02**
10	79.96 ± 3.17	73.66 ± 4.10	71.12 ± 5.94	89.71 ± 4.45	92.55 ± 3.05	89.68 ± 3.54	92.01 ± 2.37	88.95 ± 5.51	**92.55 ± 1.72**	92.17 ± 0.21
11	92.48 ± 2.04	83.53 ± 2.64	82.16 ± 3.10	96.76 ± 3.23	98.97 ± 1.28	98.95 ± 1.07	99.64 ± 0.63	98.13 ± 1.52	**99.41 ± 0.64**	99.33 ± 0.38
12	98.12 ± 1.43	81.92 ± 7.26	77.71 ± 5.22	97.59 ± 3.10	98.79 ± 1.42	98.65 ± 0.76	99.70 ± 0.31	96.62 ± 2.76	**99.76 ± 0.28**	99.06 ± 0.52
13	97.81 ± 0.49	92.17 ± 3.42	81.18 ± 5.10	97.43 ± 2.56	98.19 ± 1.97	98.88 ± 0.56	98.75 ± 1.05	96.27 ± 2.79	**99.75 ± 0.16**	99.55 ± 0.00
14	90.93 ± 2.20	89.22 ± 2.14	78.51 ± 8.84	97.02 ± 2.09	99.60 ± 0.27	98.61 ± 0.47	99.79 ± 0.19	95.60 ± 3.62	**99.83 ± 0.13**	99.52 ± 0.48
15	59.64 ± 7.55	52.80 ± 6.67	51.44 ± 4.78	69.44 ± 6.35	57.27 ± 13.08	58.48 ± 12.33	**84.58 ± 5.13**	61.54 ± 12.91	72.57 ± 9.43	82.12 ± 0.59
16	93.72 ± 2.22	95.99 ± 0.98	90.65 ± 2.48	98.38 ± 0.79	97.87 ± 1.20	97.81 ± 0.85	98.25 ± 0.62	97.34 ± 1.08	98.42 ± 0.70	**98.85 ± 0.48**
OA	84.86 ± 0.74	79.13 ± 0.92	76.51 ± 0.98	87.62 ± 1.74	84.51 ± 1.69	84.25 ± 1.06	91.74 ± 1.68	86.10 ± 1.47	92.17 ± 1.34	**92.46 ± 0.06**
AA	91.15 ± 0.75	86.06 ± 0.39	82.27 ± 0.62	93.39 ± 0.85	93.26 ± 0.65	92.93 ± 0.28	96.35 ± 0.39	92.02 ± 1.05	96.35 ± 0.39	**96.51 ± 0.03**
Kappa × 100	83.16 ± 0.81	76.82 ± 1.04	73.90 ± 1.08	86.24 ± 1.92	82.78 ± 1.83	82.48 ± 1.12	90.84 ± 1.84	84.54 ± 1.64	91.28 ± 1.45	**91.60 ± 0.10**

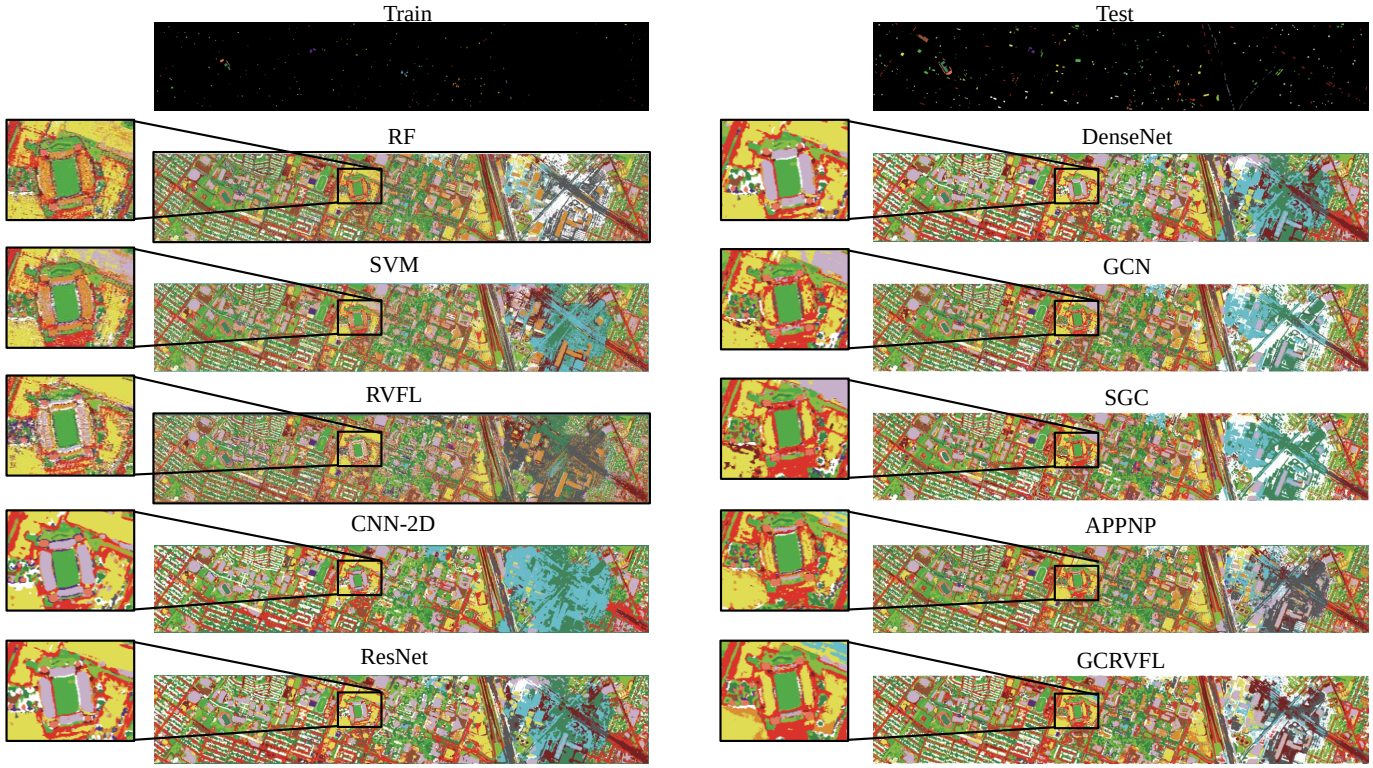

Figure 10. The classification maps obtained by different models on the Houston 2013 data set.

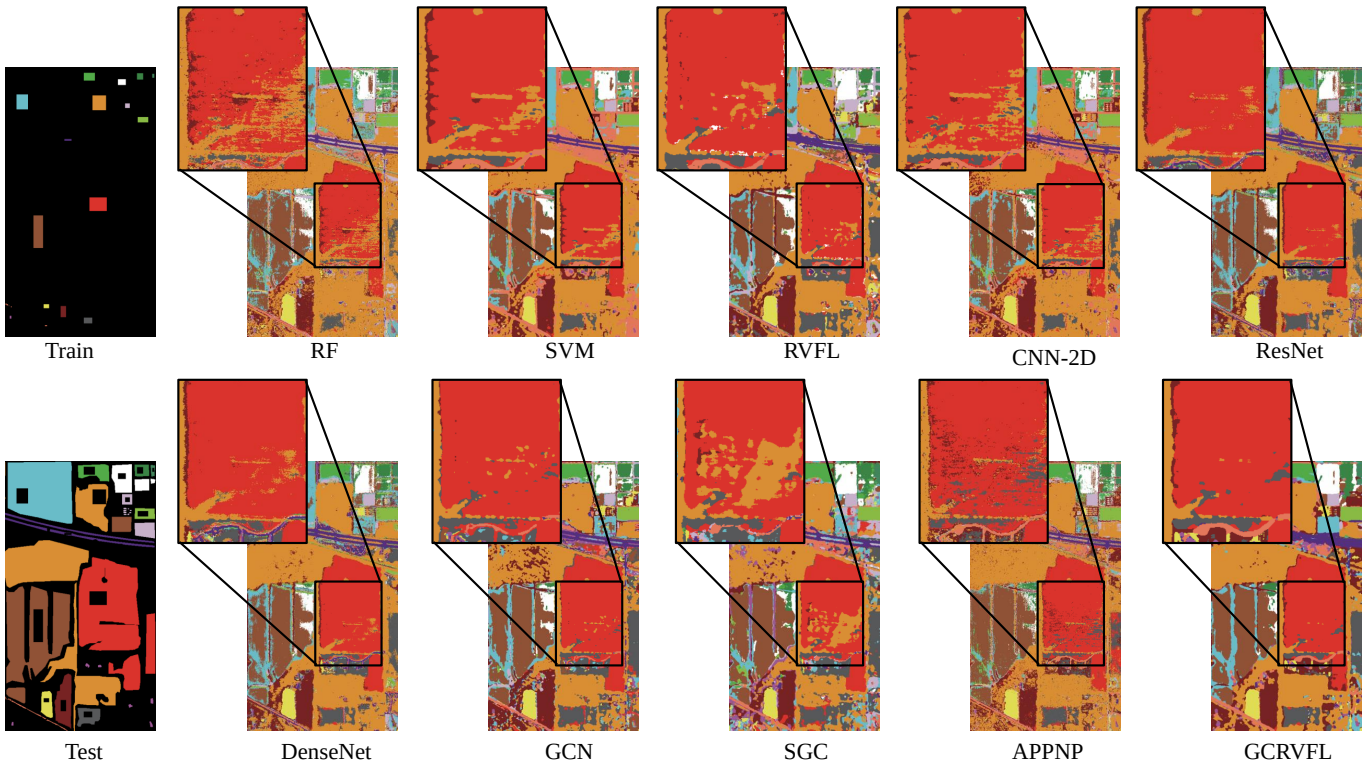

Figure 11. The classification maps obtained by different models on the Indian Pines 2010 data set.

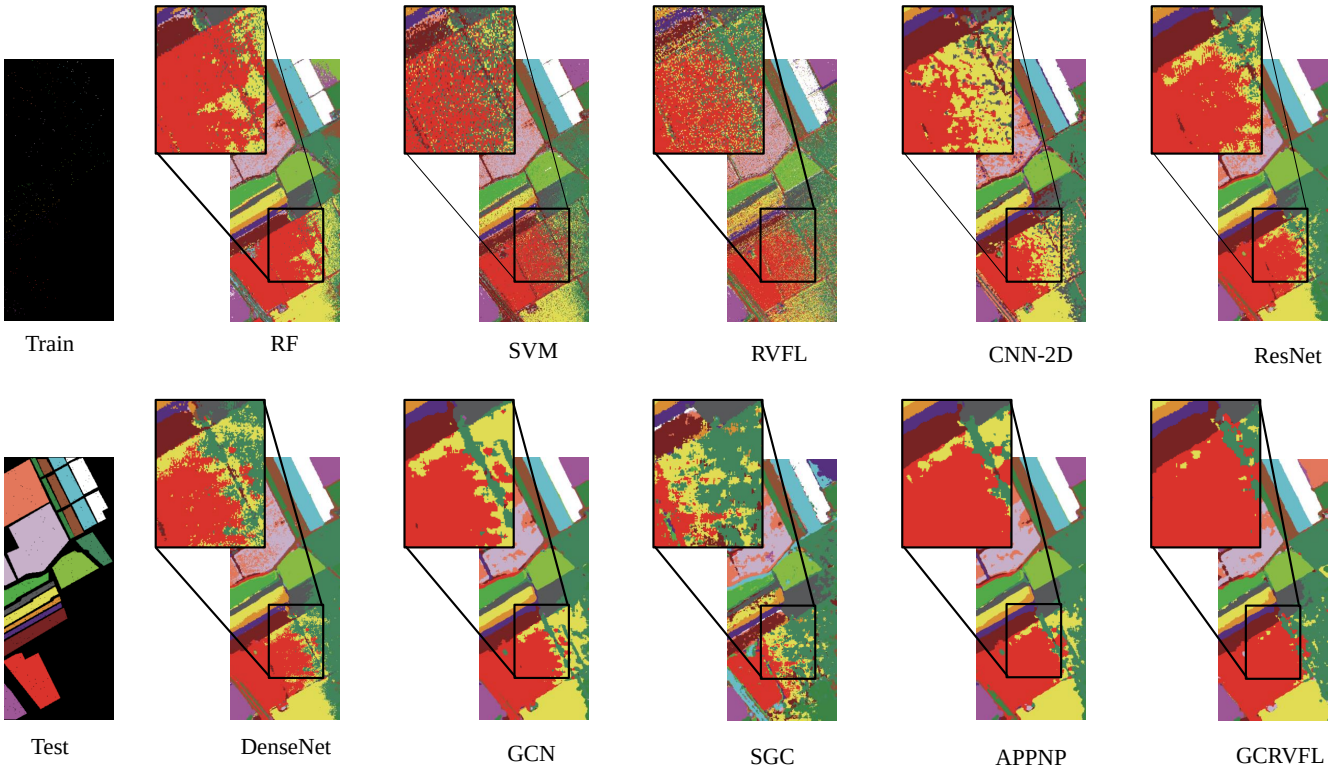

Figure 12. The classification maps obtained by different models on the Salinas data set.

4.2.4. Comparison with the State of the Art

To demonstrate the effectiveness of GCRVFL, we conducted experiments on the Houston 2013 data set, comparing its performance with five state-of-the-art methods: RNN [48], MiniGCN [26], ViT [5], and SpectralFormer [6]. The results are reported in Table 9. Remarkably, our proposed GCRVFL achieves competitive results despite its surprisingly simpler architecture compared to state-of-the-art methods. State-of-the-art models, such as ViT, often employ elaborate architectures that can be powerful when abundant training data are available. However, these models may suffer from overfitting on the Houston data set, necessitating the introduction of complex training tricks. For instance, SpectralFormer adopts cross-layer adaptive fusion. In contrast, our GCRVFL features a straightforward architecture with diverse neurons. This design not only provides comparable performance but also proves to be more user friendly and efficient for practical applications.

Table 9. Performance comparison with state-of-the-art methods on Houston 2013 data set.

Class No.	GCRVFL	RNN	MiniGCN	ViT	Spectral Former
1	81.90	82.34	98.39	82.81	83.48
2	83.53	94.27	92.11	96.62	95.58
3	99.72	99.60	99.60	99.80	99.60
4	89.03	97.54	96.78	99.24	99.15
5	100.00	93.28	97.73	97.73	97.44
6	95.80	95.10	95.10	95.10	95.10
7	74.59	83.77	57.28	76.77	88.99
8	48.36	56.03	68.09	55.65	73.31
9	89.86	72.14	53.92	67.42	71.86
10	93.32	84.17	77.41	68.05	87.93
11	83.68	82.83	84.91	82.35	80.36
12	85.42	70.61	77.23	58.50	70.70
13	82.18	69.12	50.88	60.00	71.23

Table 9. *Cont.*

Class No.	GCRVFL	RNN	MiniGCN	ViT	Spectral Former
14	100.00	98.79	98.38	98.79	98.79
15	100.00	95.98	98.52	98.73	98.73
OA	84.78	83.23	81.71	80.41	86.14
AA	87.16	85.04	83.09	82.50	87.48
Kappa × 100	83.52	81.83	80.18	78.76	84.97

4.3. Training Time

To showcase the efficiency advantage of our proposed GCRVFL compared to the baselines, we summarize the training times of the different methods in Table 10. GCRVFL's time complexity is comparable to other GCN-based methods, yet it utilizes a closed-form solution for training. The complexities of these methods are highly dependent on the scale of the input graph. In contrast, gradient-based deep methods like CNN-2D, ResNet, and DenseNet, which involve a large number of parameters and coupled operations, are challenging to compute in terms of time complexities as they far exceed those in GCRVFL. It is worth noting that we precompute all the adjacency matrices for the training set to avoid repetitive calculations before training. As shown in Table 10, our proposed GCRVFL exhibits the shortest training time, outpacing other deep learning-based models (e.g., CNN-2D, ResNet, DenseNet, GCN, SGC, and APPNP), especially in comparison to the three graph-based models. This efficiency arises because our proposed model avoids iteratively updating the output layer parameters, instead obtaining them through closed-form solutions. Additionally, the random generation of hidden parameters contributes to the overall faster training speed.

Table 10. The training time of different methods on three HSI data sets (in seconds). The last row denotes the time complexity of different methods, where C_{cnn}, C_{ResNet}, and $C_{DenseNet}$ indicate the complexity of the corresponding base block and $T = s \times s$.

Data Set	RF	SVM	RVFL	CNN-2D	ResNet	DenseNet	GCN	SGC	APPNP	GCRVFL
Houston 2013	1.202	0.085	0.068	41.429	74.707	74.213	505.971	183.757	446.336	0.846
Indian Pines 2010	4.346	0.496	0.244	141.581	248.961	287.809	1812.658	562.704	1428.152	3.852
Salinas	0.650	0.028	0.016	9.456	9.086	10.683	145.545	45.995	134.625	0.341
Time Complexity	$O(N\log(N))$	$O(N^2)$	$O(N \cdot L)$	$O(N \cdot Depth \cdot C_{cnn})$	$O(N \cdot Depth \cdot C_{ResNet})$	$O(N \cdot Depth \cdot C_{DenseNet})$	$O(T^2 \cdot L)$	$O(T^3)$	$O(Depth \cdot T^2)$	$O(T^2 \cdot L)$

5. Conclusions and Future Work

In this paper, we proposed a simple yet efficient HSI classification technique termed GCRVFL. The presented model not only inherits the performance advantage of GCN, but also simplifies GCN by extending the classic RVFL network into the graph domain, where the closed-form solution of the output layer simplifies the GCN model. In order to enable GCRVFL to process large-scale HSI, we consider HSI classification as a graph-level classification task, instead of a node-level task. Experimental results demonstrate that our presented GCRVFL can achieve more accurate performance while consuming less training time and significantly reducing training costs in comparison with many competitors.

In future work, we plan to explore a compelling research direction by extending GCRVFL to deeper architectures for hyperspectral image (HSI) classification. Additionally, there is significant potential in applying GCRVFL to various remote sensing tasks, including but not limited to building extraction.

Author Contributions: Conceptualization, Z.Z. and Y.C.; funding acquisition, Y.M. and Y.C.; investigation, Y.C. and X.L.; project administration, M.Z. and Y.M.; supervision, X.L.; visualization, Z.Z.; writing—original draft, Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by the Knowledge Innovation Program of Wuhan-Shuguang Project under Grant No. 2023020201020414, the Open Research Fund Program of LIESMARS under Grant No. 22S04, the National Natural Science Foundation of China under Grant No. 61973285, the Hubei Provincial Natural Science Foundation of China under Grant No. 2023AFB415, and the National Postdoctoral Researcher Program of China under Grant No. GZC20233138.

Data Availability Statement: The data presented in this study are available in this article.

Acknowledgments: The authors would like to thank Melba Crawford for providing the Indian Pines 2010 Data and the National Center for Airborne Laser Mapping, the Hyperspectral Image Analysis Laboratory at the University of Houston, and the IEEE GRSS Image Analysis and Data Fusion Technical Committee for providing the Houston data sets.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Salcedo-Sanz, S.; Ghamisi, P.; Piles, M.; Werner, M.; Cuadra, L.; Moreno-Martínez, A.; Izquierdo-Verdiguier, E.; noz Marí, J.M.; Mosavi, A.; Camps-Valls, G. Machine learning information fusion in Earth observation: A comprehensive review of methods, applications and data sources. *Inf. Fusion* **2020**, *63*, 256–272. [CrossRef]
2. Ghamisi, P.; Maggiori, E.; Li, S.; Souza, R.; Tarablaka, Y.; Moser, G.; De Giorgi, A.; Fang, L.; Chen, Y.; Chi, M.; et al. New Frontiers in Spectral-Spatial Hyperspectral Image Classification: The Latest Advances Based on Mathematical Morphology, Markov Random Fields, Segmentation, Sparse Representation, and Deep Learning. *IEEE Geosci. Remote Sens. Mag.* **2018**, *6*, 10–43. [CrossRef]
3. Yang, J.; Du, B.; Zhang, L. From center to surrounding: An interactive learning framework for hyperspectral image classification. *ISPRS J. Photogramm. Remote Sens.* **2023**, *197*, 145–166. [CrossRef]
4. Okwuashi, O.; Ndehedehe, C.E. Deep support vector machine for hyperspectral image classification. *Pattern Recognit.* **2020**, *103*, 107298. [CrossRef]
5. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv* **2021**, arXiv:2010.11929.2021.
6. 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]
7. Jia, N.; Tian, X.; Gao, W.; Jiao, L. Deep Graph-Convolutional Generative Adversarial Network for Semi-Supervised Learning on Graphs. *Remote Sens.* **2023**, *15*, 3172. [CrossRef]
8. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J.; et al. Recent advances in convolutional neural networks. *Pattern Recognit.* **2018**, *77*, 354–377. [CrossRef]
9. Guo, M.; Liu, H.; Xu, Y.; Huang, Y. Building Extraction Based on U-Net with an Attention Block and Multiple Losses. *Remote Sens.* **2020**, *12*, 1400. [CrossRef]
10. Meng, Y.; Chen, S.; Liu, Y.; Li, L.; Zhang, Z.; Ke, T.; Hu, X. Unsupervised Building Extraction from Multimodal Aerial Data Based on Accurate Vegetation Removal and Image Feature Consistency Constraint. *Remote Sens.* **2022**, *14*, 1912. [CrossRef]
11. Wang, F.; Jiang, M.; Qian, C.; Yang, S.; Li, C.; Zhang, H.; Wang, X.; Tang, X. Residual Attention Network for Image Classification. In Proceedings of the The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
12. Oskouei, A.G.; Balafar, M.A.; Motamed, C. RDEIC-LFW-DSS: ResNet-based deep embedded image clustering using local feature weighting and dynamic sample selection mechanism. *Inf. Sci.* **2023**, *646*, 119374. [CrossRef]
13. Zhan, L.; Li, W.; Min, W. FA-ResNet: Feature affine residual network for large-scale point cloud segmentation. *Int. J. Appl. Earth Obs. Geoinf.* **2023**, *118*, 103259. [CrossRef]
14. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269.
15. Li, B.; Xiao, C.; Wang, L.; Wang, Y.; Lin, Z.; Li, M.; An, W.; Guo, Y. Dense nested attention network for infrared small target detection. *IEEE Trans. Image Process.* **2022**, *32*, 1745–1758. [PubMed] [CrossRef]
16. Li, Z.; Yan, C.; Sun, Y.; Xin, Q. A densely attentive refinement network for change detection based on very-high-resolution bitemporal remote sensing images. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 4409818. [CrossRef]
17. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.u.; Polosukhin, I. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; pp. 5998–6008.

18. Cai, Y.; Liu, X.; Cai, Z. BS-Nets: An End-to-End Framework for Band Selection of Hyperspectral Image. *IEEE Trans. Geosci. Remote Sens.* **2020**, *58*, 1969–1984. [CrossRef]
19. Zhao, C.; Qin, B.; Feng, S.; Zhu, W.; Sun, W.; Li, W.; Jia, X. Hyperspectral image classification with multi-attention transformer and adaptive superpixel segmentation-based active learning. *IEEE Trans. Image Process.* **2023**, *32*, 3606–3621. [PubMed] [CrossRef]
20. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Yu, P.S. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 4–24. [PubMed] [CrossRef]
21. Ding, Y.; Guo, Y.; Chong, Y.; Pan, S.; Feng, J. Global Consistent Graph Convolutional Network for Hyperspectral Image Classification. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 5501516. [CrossRef]
22. Cai, Y.; Zhang, Z.; Cai, Z.; Liu, X.; Jiang, X.; Yan, Q. Graph Convolutional Subspace Clustering: A Robust Subspace Clustering Framework for Hyperspectral Image. *IEEE Trans. Geosci. Remote Sens.* **2020**, *59*, 4191–4202. [CrossRef]
23. 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]
24. Ding, Y.; Zhao, X.; Zhang, Z.; Cai, W.; Yang, N.; Zhan, Y. Semi-Supervised Locality Preserving Dense Graph Neural Network With ARMA Filters and Context-Aware Learning for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5511812. [CrossRef]
25. Yin, J.; Liu, X.; Hou, R.; Chen, Q.; Huang, W.; Li, A.; Wang, P. Multiscale Pixel-Level and Superpixel-Level Method for Hyperspectral Image Classification: Adaptive Attention and Parallel Multi-Hop Graph Convolution. *Remote Sens.* **2023**, *15*, 4235. [CrossRef]
26. 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. [CrossRef]
27. Zhang, X.; Chen, S.; Zhu, P.; Tang, X.; Feng, J.; Jiao, L. Spatial Pooling Graph Convolutional Network for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5521315. [CrossRef]
28. Zhang, L.; Suganthan, P. A comprehensive evaluation of random vector functional link networks. *Inf. Sci.* **2016**, *367–368*, 1094–1105. [CrossRef]
29. Zhang, Z.; Cai, Y.; Gong, W. Evolution-Driven Randomized Graph Convolutional Networks. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *52*, 7516–7526. [CrossRef]
30. Gao, R.; Du, L.; Suganthan, P.N.; Zhou, Q.; Yuen, K.F. Random vector functional link neural network based ensemble deep learning for short-term load forecasting. *Expert Syst. Appl.* **2022**, *206*, 117784. [CrossRef]
31. Malik, A.K.; Ganaie, M.A.; Tanveer, M.; Suganthan, P.N.; Initiative, A.D.N.I. Alzheimer's Disease Diagnosis via Intuitionistic Fuzzy Random Vector Functional Link Network. *IEEE Trans. Comput. Soc. Syst.* **2022**, 1–12. [CrossRef]
32. Cai, Y.; Zhang, Z.; Yan, Q.; Zhang, D.; Banu, M.J. Densely Connected Convolutional Extreme Learning Machine for Hyperspectral Image Classification. *Neurocomputing* **2020**, *434*, 21–32. [CrossRef]
33. Zhou, Y.; Wei, Y. Learning Hierarchical Spectral-Spatial Features for Hyperspectral Image Classification. *IEEE Trans. Cybern.* **2016**, *46*, 1667–1678. [CrossRef]
34. Cao, F.; Yang, Z.; Ren, J.; Ling, W.; Zhao, H.; Sun, M.; Benediktsson, J.A. Sparse Representation-Based Augmented Multinomial Logistic Extreme Learning Machine With Weighted Composite Features for Spectral-Spatial Classification of Hyperspectral Images. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 6263–6279. [CrossRef]
35. Zhang, Z.; Cui, P.; Zhu, W. Deep Learning on Graphs: A Survey. *IEEE Trans. Knowl. Data Eng.* **2020**, *56*, 6263–6279. [CrossRef]
36. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017.
37. Defferrard, M.; Bresson, X.; Vandergheynst, P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Proceedings of the Advances in Neural Information Processing Systems*; Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2016; Volume 29, pp. 3844–3852.
38. Chen, D.; Lin, Y.; Li, W.; Li, P.; Zhou, J.; Sun, X. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 3438–3445.
39. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
40. Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; Weinberger, K. Simplifying Graph Convolutional Networks. In Proceedings of the International Conference on Machine Learning, Long Beach , CA, USA, 9–15 June 2019; pp. 6861–6871.
41. Debes, C.; Merentitis, A.; Heremans, R.; Hahn, J.; Frangiadakis, N.; van Kasteren, T.; Liao, W.; Bellens, R.; Pižurica, A.; Gautama, S.; et al. Hyperspectral and LiDAR Data Fusion: Outcome of the 2013 GRSS Data Fusion Contest. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2014**, *7*, 2405–2418. [CrossRef]
42. Mekha, P.; Teeyasuksaet, N. Image Classification of Rice Leaf Diseases Using Random Forest Algorithm. In Proceedings of the 2021 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunication Engineering, Cha-am, Thailand, 3–6 March 2021, pp. 165–169. [CrossRef]
43. Fauvel, M.; Chanussot, J.; Benediktsson, J.A. A spatial-spectral kernel-based approach for the classification of remote-sensing images. *Pattern Recognit.* **2012**, *45*, 381–392. [CrossRef]

44. Igelnik, B.; Pao, Y.H. Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Trans. Neural Netw.* **1995**, *6*, 1320–1329. [CrossRef] [PubMed]
45. Chen, Y.; Jiang, H.; Li, C.; Jia, X.; Ghamisi, P. Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 6232–6251. [CrossRef]
46. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. *arXiv* **2018**, arXiv:1608.06993
47. Klicpera, J.; Bojchevski, A.; Günnemann, S. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
48. 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]