Deep Fuzzy Graph Convolutional Networks for PolSAR Imagery Pixelwise Classification

Hongying Liu, *Member, IEEE*, Tianwen Zhu, Fanhua Shang, *Senior Member, IEEE*, Yuanyuan Liu, *Member, IEEE*, Derui Lv, and Shuyuan Yang, *Senior Member, IEEE*

***Abstract*—Pixelwise classification plays an important role for image interpretation. The remote sensing images, especially polarimetric synthetic aperture radar (PolSAR), have provided wide applications for both military and civilian users regardless of weather or lighting conditions. However, the classification of heterogeneous imagery is still challenging. In this article, we propose a novel deep fuzzy graph convolutional network (DFGCN) for pixelwise classification of PolSAR imagery. Inspired by the fuzzy logic, the imagery (i.e., a reflection of the backscattering of the ground) is represented by a fuzzy graph, whose node is associated with a feature vector, and fuzzy weights denoting the similarity degrees between nodes are defined by using both feature distance and spatial relation. We also construct a new graph residual module stacking multiple residual layers to extend the depth of our fuzzy graph network. Moreover, we present a graph shrinking strategy to reduce the graph size for training and predicting the labels of large-scale PolSAR imagery, which can cut down the computational cost. The experimental results on various real-world PolSAR datasets indicate that DFGCN dramatically enhances classification accuracies on both heterogeneous and homogeneous regions with limited labeled pixels compared with the state-of-the-art methods.**

***Index Terms*—Fuzzy logic, graph convolutional network (GCN), pixel classification, polarimetric synthetic aperture radar (PolSAR).**

I. Introduction

PIXELWISE classification is a fundamental task that aims at categorizing every pixel in the image to different semantic classes based on the features of the image. In terms of the utilization of labeled data, the classification methods mainly fall into three types —supervised, unsupervised, and semisupervised methods [1]. The conventional supervised methods include K-nearest neighbor (KNN), support vector machine (SVM), decision tree, Naive Bayes, linear discriminant analysis, and so on. K-means, mean-shift, watershed, Ncut, and level sets belong to unsupervised classification methods. In addition, the commonly used semisupervised classification methods include cotraining, self-training, semisupervised SVMs, and graph-based methods [2]. More recently, deep learning-based methods have been widely applied to various image classification tasks due to the strong ability in learning discriminative representation.

For remote sensing images, the pixelwise classification aims to categorize each pixel to the known classes, which is an important task for image interpretation. Many unsupervised methods have been proposed for the categorization. The classical unsupervised classification based on the scattering properties of the observed earth surface was proposed by Cloude and Pottier [3], and Lee and Pottier [4]. Several works based on analyzing the statistics of the covariance or coherency matrix for classification of the heterogeneous PolSAR images have been proposed in [5]–[9], which achieve sound performance. Based on the edge maps computed from the statistics of PolSAR data, Liu *et al.* [10] utilized the Ncut algorithm to generate superpixels, and then obtained the final classification results by iterated clustering on a few undecided superpixels. Guo *et al.* [11] proposed a new superpixel generation method for PolSAR image classification, named as fuzzy superpixel (FS), where the fuzzy logic is utilized for defining the class-undetermined superpixels, namely, fuzzy superpixels. Then the pixels in the fuzzy superpixel are assigned to the corresponding superpixel by computing the degree of membership. The final classification is conducted by SVM on fuzzy superpixels. The aforementioned algorithms have been proposed to reduce the influence of speckle noise and decrease computational time. More recently, many deep neural networks have been proposed for PolSAR classification. In 2014, Xie *et al.* [12] exploited a stack auto-encoder network to learn the features of PolSAR images, overcoming the difficulty of manually extracting features. Bi *et al.* [13] designed an energy function for unsupervised classification by combining a Softmax regression model with a Markov random field smoothness constraint. In addition, based on the Wishart distribution of PolSAR data, Liu *et al.* [14] stacked the restricted Boltzmann machines, and proposed a Wishart deep belief network (W-DBN) for modeling the data and classification. However, their performances are not always satisfied for the classification of heterogeneous terrains.

Manuscript received August 7, 2020; revised October 11, 2020 and November 3, 2020; accepted November 13, 2020. Date of publication December 1, 2020; date of current version January 6, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61976164, Grant 61876220, and Grant 61876221, in part by the Project supported the Foundation for Innovative Research Groups of the National Natural Science Foundation of China under Grant 61621005, in part by the Major Research Plan of the National Natural Science Foundation of China under Grant 91438201 and Grant 91438103, in part by the Program for Cheung Kong Scholars and Innovative Research Team in University under Grant IRT_15R53, in part by the Fund for Foreign Scholars in University Research and Teaching Programs under the 111 Project B07048, in part by the Science Foundation of Xidian University under Grant 10251180018 and Grant 10251180019, in part by the National Science Basic Research Plan in Shaanxi Province of China under Grant 2019JQ-657 and Grant 2020JM-194, and in part by the Key Special Project of China High Resolution Earth Observation System-Young Scholar Innovation Fund. *(Corresponding authors: Tianwen Zhu; Fanhua Shang.)*

The authors are with the School of Artificial Intelligence, Xidian University, Xi'an 710071, China (e-mail: hyliu@xidian.edu.cn; 17794242838@163.com; fhshang@xidian.edu.cn; yyliu@xidian.edu.cn; drlv@stu.xidian.edu.cn; syyang@xidian.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/JSTARS.2020.3041534, provided by the authors.

Digital Object Identifier 10.1109/JSTARS.2020.3041534

TABLE I
COMPARISON OF GCN, DEEPGCN, AND DFGCN

Methods	Graph weights	Graph learning layers	Other layers	Training with graphs
GCN [21]	Binary	Graph convolutional layers only	Softmax	Pre-constructed graph
DeepGCN [22]	Binary	Graph residual layers only	Max pooling & MLP	Dynamic KNN graph
DFGCN (ours)	Fuzzy	Graph convolutional & residual layers	Softmax	Graph shrinking strategy

In addition, a lot of supervised methods have also been applied to PolSAR classification, such as boosting, random forest, and the sparse coding classifiers [15]–[17]. For instance, a convolutional neural network (CNN) [18] is used in the classification of PolSAR data. Mohammadimanesh *et al.* [19] proposed a fully convolutional network for which stacks of convolutional filters are used as encoders and decoders for feature extraction and classification. Bi *et al.* [20] proposed a supervised CNN, which is incrementally fine-tuned by incorporating the actively annotated the most informative candidate pixels, and reduced the annotation cost. However, most of these supervised methods require large numbers of labeled samples to learn the category information. This is costly for remote sensing imageries since field investigation costs plenty of manpower and material resources.

Recently semisupervised feature extraction and classification methods [23]–[29] attract more attention. These methods utilize a few labeled samples and many unlabeled samples for classification. [23] constructed an undirected graph, where each element from PolSAR data is represented using a vertex, and some nearest neighbors of each vertex are connected by edges. The label information can propagate to the unlabeled vertexes through edges. Though it achieves sound results, this graph-based method is restricted by the explicit graph-based regularization, and can not learn the various structure information. A graph-based deep CNN for semisupervised label propagation was also presented in [30] for PolSAR categorization. Liu *et al.* [31] utilized sparse learning and proposed a deep neural network based on sparse manifold regularization (DSMR) for feature extraction and classification.

More recently, graph convolutional networks (GCNs) [21], [32], which are graph structured learning networks, are proposed and widely used for predicting individual relations in social networks, enhancing predictions of recommendation system and segmenting large point clouds. However, conventional GCN [21] uses binary weight between connected vertices for graph construction, and it is confined within only a few layers (three or four layers) due to vanishing gradient, which limits its real-world applications. Then deeper GCN (DeepGCN) [22] has been proposed with residual connections. It constructs and updates a dynamic KNN graph for each graph learning layer, uses max pooling for feature aggregation of each vertex, and adopts a multilayer perceptron (MLP) for class prediction. And the experiments on point clouds and biological networks confirmed its effectiveness. Nevertheless, the construction of a delicate graph, which can more accurately model the heterogenous terrains for classification, is still pending.

To address these issues, we propose a novel deep fuzzy GCN (DFGCN) for PolSAR pixelwise classification. In our DFGCN, each element or pixel from PolSAR datasets is represented using a vertex. Since each pixel is a combination of backscattered reflections from many surrounding objects, inspired by the fuzzy logic, we construct a fuzzy graph with fuzzy weights computed from both physical properties and spatial distances, whose value represents the similarity degree between connected pixels. That is, we use the generated fuzzy graph in our DFGCN instead of common graphs used in [21], [22], and [32]. Please refer to Table I for detailed comparison. Moreover, to extract intuitive features from PolSAR imagery, we further design a new graph residual module (GResM) followed by the backbone of our network (i.e., few graph convolutional layers) to extend the depth of DFGCN. For large-scale data, we present a novel graph shrinking strategy to reduce a graph matrix to a smaller one for both training and testing. Finally, the Softmax function is utilized to calculate the class label for each pixel.

It is clear that DFGCN first utilizes the intrinsic property of the graph convolution computation, that is, it has nonrigidly deformable shapes across neighboring pixels/reception field, which is different from the patch-based convolution in CNNs, and this is crucial to the division of heterogenous regions, which may has different number of mixed pixels. Then we propose a fuzzy graph to represent the relation between pixels, which helps more accurate description for the division of pixels. And with our deepened network which can reach tens or hundreds of layers, it can extract more intuitive features for pixelwise classification. Moreover, with our shrinking strategy, the pixelwise classification can be greatly sped up.

The main contributions of this article are summarized as follows.

1) We show that GCN is powerful for representation and pixelwise classification of PolSAR imageries.
2) We propose a new DFGCN with both graph convolutional and GResMs for PolSAR classification. To the best of our knowledge, the combination of fuzzy logic with GCNs is the first attempt to the pixelwise classification of PolSAR data. In particular, both the physical property and spatial information of each pixel are considered to construct a fuzzy graph.
3) A new GResM is presented for extending the depth of our network, which can extract intuitive features for real-world imagery. Experimental results confirm its effectiveness compared with its counterparts.
4) We also present a new graph shrinking strategy for training and testing of large-scale imageries, which are effective for real-world large-scale data.

The remainder of the article is organized as follows. In Section II, the related work on GCNs is briefly introduced. In Section III, our DFGCN is proposed and described in details. Section IV presents experimental results and discussions. Finally, the conclusion is drawn in Section V.

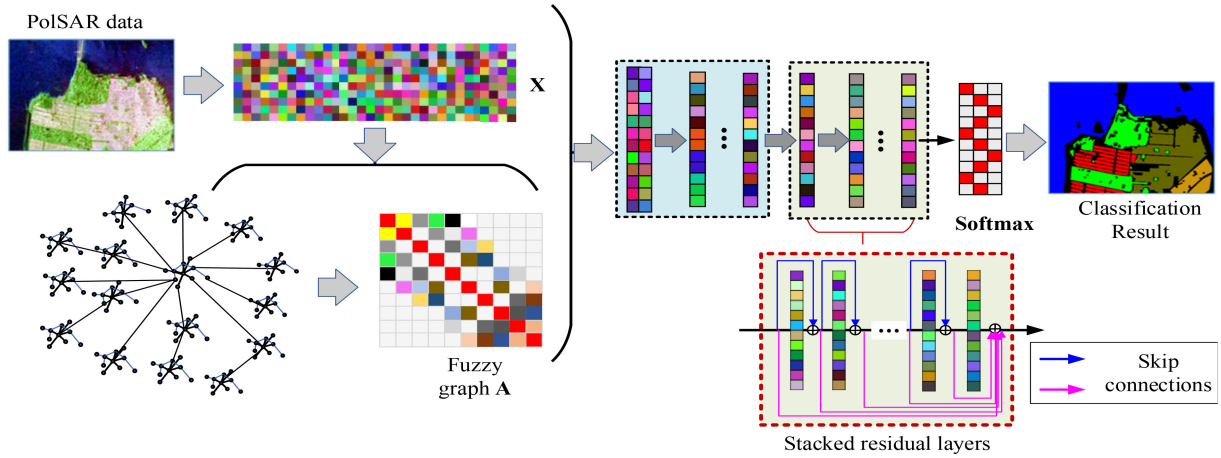


Fig. 1. Flowchart and architecture of our deep GCN which includes fuzzy graph, graph convolutional, and GResMs.

II. Related Work

This section briefly introduces the GCNs [21]. For input data $\mathbf{X} \in \mathbb{R}^{N \times T}$, where T is the number of features and N is the number of samples, it is represented by a graph with N nodes, and the connection relationships between nodes are denoted by an adjacency matrix $\mathbf{G} \in \mathbb{R}^{N \times N}$. A multilayer GCN is formulated as

$$\mathbf{H}^{(l+1)} = f(\mathbf{G}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) \tag{1}$$

where $l \geq 0$, $\mathbf{H}^{(l)}$ denotes the output of the lth layer, $\mathbf{W}^{(l)}$ is the trainable weight matrix in the lth layer, and $f(\cdot)$ is an activation function. Here, $\mathbf{H}^{(0)} = \mathbf{X}$, and $\mathbf{W}^{(0)}$ is the weight matrix connecting input data and the first layer. In fact, the normalized adjacency matrix is used in many real-world applications. For more details, readers please refer to [21].

III. DFGCNs for PolSAR Classification

In this section, we propose a novel DFGCN with fuzzy graph, graph convolutional, and GResMs for PolSAR classification, whose flowchart is shown in Fig. 1. The architecture of DFGCN mainly includes the following three parts: inputs, the backbone network, and the output layer, where the inputs mainly include a fuzzy graph matrix and extracted features $\mathbf{X}$, and the backbone network includes both a graph convolutional module (GCM) and a GResM. Their details are described as follows.

A. Fuzzy Graph Construction

In conventional GCNs [21], we usually use an undirected graph, and each edge between two vertices carries a binary weight (i.e., 0 or 1). However, the binary weights are not appropriate in many real-world applications [33]. For remote sensing images, even with high spatial resolution, each pixel may be a combination of backscattered reflections from many surrounding objects. For example, as shown in Fig. 2, for a mixed pixel p_0, it is likely to be similar to a neighboring pixel p_1, and it may resemble the neighboring pixel p_2 or p_3 to some extent. That is, it has similarities to the surrounding pixels with different probabilities. To address this problem, inspired by the fuzzy logic, we design a graph with fuzzy weights and call it as a fuzzy graph. The fuzzy weight, whose value is in the interval [0, 1], on the edge represents the degree of similarity between the neighboring two pixels. The construction of the fuzzy weights for our graph is depicted in details below.

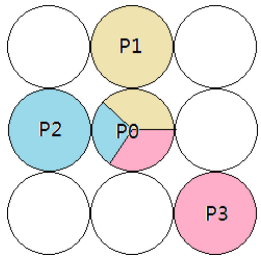


Fig. 2. Fuzzy similarity of neighboring pixels.

We observe that the pixel similarity is related to both physical properties and spatial distance. The two factors are considered in our method. Thus, the similarity is calculated by the distance metric, and physical properties of data can be represented by the feature vectors. We represent the PolSAR data as: $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, and each single pixel i is denoted by its feature vector $\mathbf{x}_i \in \mathbb{R}^T$, where T is the number of features and N denotes the number of pixels or samples.

As it is known, a number of signatures (e.g., the elements of coherency matrix or covariance matrix, the elements of the Yamaguchi decomposition or Freeman–Durden decomposition) can be used as features for PolSAR data. More detailed descriptions can be found in [4]. It is indicated that the coherency matrix or the covariance matrix follows a complex Wishart distribution. Therefore, the revised Wishart distance [34], [35] d_{W} is utilized as a metric to define the distance between two features $\mathbf{x}_i$ and $\mathbf{x}_j$ as follows:

$$d_{\mathrm{W}}(\mathbf{x}_i, \mathbf{x}_j) = \ln\left(\frac{|\mathbf{\Sigma}_i|}{|\mathbf{\Sigma}_j|}\right) + \mathrm{Tr}\left(\mathbf{\Sigma}_i^{-1}\mathbf{\Sigma}_j\right) - q \tag{2}$$

where $\mathbf{\Sigma}_i$ is the mean covariance matrix for pixel centered at i, q is a constant, $|\cdot|$ and $\mathrm{Tr}(\cdot)$ are the determinant and the trace of a matrix. For instance, the setting $q = 3$ is used for the reciprocal SAR and $q = 4$ is for the bistatic SAR.

As for other features (e.g., Huynen decomposition, Freeman–Durden decomposition), their distributions may be not known or still under study, we can utilize the common Euclidean distance for a metric. The distance d_{E} between two features $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as

$$d_{\mathrm{E}}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2. \tag{3}$$

In order to utilize both of the above distances in the same scale, they are normalized as follows:

$$d'_{\mathrm{W}}(\mathbf{x}_i, \mathbf{x}_j) = \frac{d_{\mathrm{W}}(\mathbf{x}_i, \mathbf{x}_j) - \min(d_{\mathrm{W}}(:,:))}{\max(d_{\mathrm{W}}(:,:)) - \min(d_{\mathrm{W}}(:,:))} \tag{4}$$

$$d'_{\mathrm{E}}(\mathbf{x}_i, \mathbf{x}_j) = \frac{d_{\mathrm{E}}(\mathbf{x}_i, \mathbf{x}_j) - \min(d_{\mathrm{E}}(:,:))}{\max(d_{\mathrm{E}}(:,:)) - \min(d_{\mathrm{E}}(:,:))}. \tag{5}$$

where $d'_{\mathrm{W}}, d'_{\mathrm{E}} \in [0, 1]$. As all the features contribute to the classification, the final distance between features $\mathbf{x}_i$ and $\mathbf{x}_j$ from samples i and j should consider both the Wishart and Euclidean distances, and is defined as follows:

$$d_R(\mathbf{x}_i, \mathbf{x}_j) = \lambda d'_{\mathrm{W}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \lambda) d'_{\mathrm{E}}(\mathbf{x}_i, \mathbf{x}_j) \tag{6}$$

where $\lambda \in [0, 1]$ is a coefficient to balance the contribution between the two distances. That is, the spatial distance is also considered in our method. As it is known that in remote sensing imageries, the pixels are the backscattered reflection of the ground surface. And there are high similarities between neighboring pixels. Therefore, this local spatial information can be utilized for feature extraction.

Suppose that the coordinate for the pixel i in PolSAR imagery is represented by (r^i, c^i), $i = 1, 2, \ldots, N$. The spatial distance between pixels i and j is defined as

$$d_O(i, j) = \sqrt{(c^i - c^j)^2 + (r^i - r^j)^2}. \tag{7}$$

For neighboring pixels which have smaller physical distances, they are more likely to belong to the same class if their feature distance is closer, while they should be farther if their feature distance is not closer. Based on this observation, we define a weighted feature distance D as follows:

$$D(\mathbf{x}_i, \mathbf{x}_j) = \log(d_O(i, j)) \cdot d_R(\mathbf{x}_i, \mathbf{x}_j) \tag{8}$$

where $\cdot$ represents the element product. Obviously, by using this logarithmic weighting, the smaller distance of the pixels within a local region would be more significant.

Based on the defined distances, we define a fuzzy graph $\mathbf{A} \in \mathbb{R}^{N \times N}$ between pixels, and thus, construct a KNN graph, which is also a sparse graph. That is, for pixel $\mathbf{x}_i$, we only select the most similar K pixels according to the following equation, and let those of others be zero.

When the distance between pixels i and j is small, the similarity should be large, and vice versa. Each entry of the graph matrix $\mathbf{A}$ (namely the weight of $\mathbf{A}$) is defined as

$$\mathbf{A}_{ij} = \begin{cases} \exp(-\frac{D^2(\mathbf{x}_i, \mathbf{x}_j)}{\delta_i \delta_j}), & \mathbf{x}_j \in N_K(\mathbf{x}_i) \\ 0, & \mathbf{x}_j \notin N_K(\mathbf{x}_i) \end{cases} \tag{9}$$

where $N_K(\mathbf{x}_i)$ denotes a set of K nearest neighbors for $\mathbf{x}_i$, δ_i and δ_j are two scaling factors. δ_i is defined as

$$\delta_i = D(\mathbf{x}_i, \mathbf{x}_j^K) \tag{10}$$

where $\mathbf{x}_i^K$ denotes the Kth nearest neighbor feature of $\mathbf{x}_i$, i.e., the farthest sample in the set of nearest neighbors. Using the scaling factors, the similarity can adaptively vary with the statistics of the nearest neighbors, therefore the fuzzy weights can more precisely represent the relation between pixels.

Note that the set of nearest neighbors contains both the labeled samples and the unlabeled samples calculated by our defined distance metric. Moreover, for graph construction, we can use the Kd-Tree algorithm [36] for a fast search of the nearest neighbors and enhance the computational efficiency.

B. Deep Fuzzy Graph Convolution and Residual Modules

The backbone of our DFGCN network is M layers including two parts: the GCM and the GResM. The GCM consists of ξ conventional graph convolutional layers, which are effective for feature extraction. And the GResM consists of multiple stacked residual learning layers, which use skip connections to avoid gradients vanishing and make the network deeper. Let $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ be a graph matrix with added self-connections, where $\mathbf{I}_N$ is an identity matrix. Then we compute a symmetric normalized graph matrix $\widehat{\mathbf{A}}$ as follows:

$$\widehat{\mathbf{A}} = \widetilde{\mathbf{E}}^{-\frac{1}{2}} \widetilde{\mathbf{A}} \widetilde{\mathbf{E}}^{-\frac{1}{2}} \tag{11}$$

where $\widetilde{\mathbf{E}}$ is the diagonal degree matrix of $\widetilde{\mathbf{A}}$, i.e., $\widetilde{\mathbf{E}}_{ii} = \sum_j \widetilde{A}_{ij}$.

With the input feature matrix $\mathbf{X} \in \mathbb{R}^{T \times N}$, $\mathbf{H}_0 = \mathbf{X}$ is for the input layer. And the layers of our GCM are expressed as

$$\mathbf{H}_{l+1} = \sigma(\widehat{\mathbf{A}}\, \mathbf{H}_l\, \mathbf{W}_l) \tag{12}$$

where $l = 0, 1, \ldots, (\xi - 1)$, $\mathbf{H}_l$ is the output of the lth layer, $\mathbf{W}_l$ is the trainable weight matrix connecting the lth and $(l + 1)$th layers, and $\sigma(\cdot)$ denotes an activation function such as the function $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$.

Unlike traditional GCNs [32], the GCM is followed by a graph residual learning module. That is, the graph residual learning starts from the $(\xi + 1)$th layer

$$\mathbf{H}_{l+1} = \mathbf{H}_l + \mathbf{H}_{l+1}^{res} \tag{13}$$

where $l = \xi, (\xi + 1), \ldots, (M - 2)$, and $\mathbf{H}_{l+1}^{res} = \sigma(\mathbf{H}_l, \mathbf{W}_l, \widehat{\mathbf{A}})$ is a residual graph representation. As shown in Fig. 1, the final residual layer receives the information from all the previous $(M - \xi - 1)$ layers through skip connections, and then the final residual layer for our network is defined as follows:

$$\mathbf{H}_M = \sigma(\mathbf{H}_{M-1}, \mathbf{W}_{M-1}, \widehat{\mathbf{A}}) + \sum_{i=1}^{M-\xi-1} \mathbf{H}_{M-i}. \tag{14}$$

The output layer of our network includes a Softmax function for classification, i.e., $\mathbf{Z} = \mathrm{Softmax}(\mathbf{H}_M)$. For our DFGCN, cross-entropy is used as a loss function for classification. This function measures the performance of the classification model, whose output is a probability value in the interval $[0, 1]$. The cross-entropy loss decreases as the predicted probability is close to the actual label. The loss function of our DFGCN is defined as follows:

$$L_{ss} = -\sum_{m \in G_L} \sum_{q=1}^{Q} \mathbf{Y}_{mq} \ln \mathbf{Z}_{mq} \tag{15}$$

where G_L represents a collection of labeled samples, Q is the number of classes, and $\mathbf{Y}$ corresponds to the ground-truth labels of all training samples.

When our DFGCN network has been trained, and the weight parameters have been determined, we can predict the categories for the unlabeled samples using our trained network. The detailed process of our DFGCN network for classification is briefly summarized in Algorithm 1.

C. Deep Fuzzy GCN for Large-Scale Data

As shown in Algorithm 1, the graph matrix $\widehat{\mathbf{A}}$ is used in all the layers of DFGCN. However, most of the datasets acquired by remote sensing systems are in very large scale, since they are back-scattered observation to the broad area. Thus, the number of nodes in the graph matrix is very large, and the per-epoch calculation time becomes enormous and even there is a memory overflow problem. To address this issue and improve the training and testing efficiency of our algorithm, we propose a graph shrinking strategy, which can be used in both training and testing processes, and extend our method for fast training on large-scale PolSAR datasets.

The key idea of our new graph shrinking strategy is to reduce the size of the graph matrix as follows. In our training process, the index of all the samples $\mathbf{X}$ is defined as: $Idx = [Idx^{\text{label}}, Idx^{\text{unlabel}}]$, where $Idx^{\text{label}} = [Idx_1, Idx_2, \ldots, Idx_L]$ is the index of the L labeled samples, while that of the unlabeled ones is $Idx^{\text{unlabel}} = [Idx_{L+1}, Idx_{L+2}, \ldots, Idx_{L+U}]$. In the first hidden layer, we can extract the row elements from $\widehat{\mathbf{A}}$ for the transition of the next layer, i.e.,

$$\widehat{\mathbf{A}}_1^{\text{train}} = \widehat{\mathbf{A}}[Idx^{\text{label}}, :] \tag{16}$$

and $\widehat{\mathbf{A}}_1^{\text{train}} \in \mathbb{R}^{L \times N}$. Since the graph matrix $\widehat{\mathbf{A}}$ contains the relationship of both labeled and unlabeled samples, this neighborhood relation is also preserved in $\widehat{\mathbf{A}}_1^{\text{train}}$. Moreover, we can further reduce the size of the graph matrix used in other layers. For the second hidden layer, the graph matrix $\widehat{\mathbf{A}}_2^{\text{train}} \in \mathbb{R}^{L \times L}$ is defined as follows:

$$\widehat{\mathbf{A}}_2^{\text{train}} = \widehat{\mathbf{A}}_1^{\text{label}}[:, Idx^{\text{label}}]. \tag{17}$$

We can continue to deepen the network, and $\widehat{\mathbf{A}}_l^{\text{train}}$ in subsequent layers (i.e., $l \geq 3$) is also defined as

$$\widehat{\mathbf{A}}_l^{\text{train}} = \widehat{\mathbf{A}}[Idx^{\text{label}}, Idx^{\text{label}}]. \tag{18}$$

From the above steps, we can reduce $\widehat{\mathbf{A}}_l^{\text{train}}$ to the size of $L \times L$, where L is the number of the labeled samples. It is clear that the number of labeled samples is usually much smaller than that of unlabeled samples in real-world data. In this way, both space and time complexities of our network are greatly decreased. Note that both the labeled samples and the unlabeled samples are used in our training process.

Similarly, in the testing process, we can also reduce the size of the graph matrix for the first hidden layer as follows:

$$\widehat{\mathbf{A}}_1^{\text{test}} = \widehat{\mathbf{A}}[Idx^{\text{unlabel}}, :]. \tag{19}$$

And similar formulations can be derived from (17) and (18) as follows: $\widehat{\mathbf{A}}_2^{\text{test}} = \widehat{\mathbf{A}}_1^{\text{test}}[:, Idx^{\text{unlabel}}]$ and $\widehat{\mathbf{A}}_l^{\text{test}} = \widehat{\mathbf{A}}[Idx^{\text{unlabel}}, Idx^{\text{unlabel}}]$. Therefore, for large-scale data, we can replace the graph matrixes used in different layers (e.g., $\widehat{\mathbf{A}}_1$, $\widehat{\mathbf{A}}_2$ and $\widehat{\mathbf{A}}_l$) with much smaller graph matrices (e.g., $\widehat{\mathbf{A}}_1^{\text{train}}$, $\widehat{\mathbf{A}}_2^{\text{train}}$ and $\widehat{\mathbf{A}}_l^{\text{train}}$) in the training process of Algorithm 1. Then in the testing process, we can also substitute $\widehat{\mathbf{A}}_1$, $\widehat{\mathbf{A}}_2$, and $\widehat{\mathbf{A}}_l$ with $\widehat{\mathbf{A}}_1^{\text{test}}$, $\widehat{\mathbf{A}}_2^{\text{test}}$, and $\widehat{\mathbf{A}}_l^{\text{test}}$, respectively.

Note that with our shrinking strategy, both space and time complexities for training have been cut down. The space complexity is reduced from $O(\sum_{l=1}^{M} N_l T + T^2)$ to $O(\sum_{l=1}^{M} R_l T + T^2)$, and the time complexity is reduced from $O(\sum_{l=1}^{M} N_l K T + N_l T^2)$ to $O(\sum_{l=1}^{M} R_l K T + R_l T^2)$, where N_l and R_l are the number of total samples and labeled samples in layer l, and when $l \geq 3$, $R_l = 1\% N_l$ in this article.

D. Comparison of Existing GCNs

It is well known that conventional GCNs have been successfully applied in various machine learning and pattern recognition tasks. However, they are limited to shallow architectures typically with 3 or 4 layers because of vanishing gradients during training. In contrast, we can make our DFGCN deeper (e.g., 16 or more layers) by using a new GResM, which is used to relieve the problems of vanishing gradients and exploit the intuitive features from input data. Unlike existing residual modules [37], [38], dense modules [38], [39] and residual-dense modules [40], the last layer in the proposed GResM stacked by multiple graph residual learning layers receives the feature maps of all preceding layers in this module, as shown in Fig. 1. Although our GResM is different from general residual-dense modules, it enjoys the advantages of both residual connection modules and dense connectivity modules.

The comparison of architectures between different networks are listed in Table I. In GCN [21] and DeepGCN [22], the graph is constructed using only binary weights, where the Euclidean distance is computed, while the graph in DFGCN is constructed with fuzzy weights, which are calculated by using both feature distance and spatial information as in (9). Moreover, for the network architectures, GCN only adopts some graph convolutional layers, and DeepGCN only uses the graph residual layers. In contrast, our DFGCN uses both graph convolutional layers and graph residual layers. Additionally, to speed up the computation, DeepGCN designs a max aggregation on the relative features (MR) from the neighborhood, and our DFGCN proposes a graph shrinking strategy to reduce the graph matrix. In fact, the MR in DeepGCN is still limited by the dilated graph convolutions which construct a new graph in each convolutional computation. Our DFGCN only utilizes the pooled fuzzy graph for each convolution which is more faster. The GCN does not use any speedup approaches.

IV. Experimental Results and Discussion

In this section, we evaluate the performance of DFGCN on real-world PolSAR imagery from different remote sensing systems, e.g., the Flevoland dataset from AIRSAR and the San Francisco dataset from RADARSAT-2, and Oberpfaffenhofen dataset from E-SAR system. We also compare DFGCN with five state-of-the-art algorithms including one region-based method, FS [11], and four pixel-based methods (i.e., W-DBN [14] with

Algorithm 1: DFGCN for PolSAR Imagery Classification.

Input: The feature matrix $\mathbf{X} \in \mathbb{R}^{T\times N}$ of N PolSAR samples, where T is the number of features. $\mathbf{X}_L = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{L}$ are L labeled samples and the rest are unlabeled samples $\mathbf{X}_U = \{\mathbf{x}_i\}_{i=L+1}^{N}$. The label $\mathbf{y} \in \mathbb{Z}^{Q\times 1}$ is a vector of class labels, and Q is the number of terrain classes.

Initialize: Initialize the weight matrices $\mathbf{W}_0$, $\mathbf{W}_1$, $\mathbf{W}_2$, $\cdots$, $\mathbf{W}_{M-1}$ with random values.

Construct the fuzzy graph $\mathbf{A}$ according to (9);

Compute the symmetric normalized matrix $\widehat{\mathbf{A}}$ according to (11);

Input the feature matrix $\mathbf{X}$ and graph matrix $\widehat{\mathbf{A}}$ into the network and calculate the cross-entropy loss of the labeled samples according to (15);

Update the weights with an optimization algorithm and obtain learned weight matrices $\mathbf{W}_0, \mathbf{W}_1, \ldots, \mathbf{W}_{M-1}$;

Calculate the Softmax function and predict the labels for unlabeled samples;

Output: Labeled matrix $\mathbf{Y} \in \mathbb{Z}^{Q\times(N-L)}$.

unsupervised pretraining, the supervised CNN [18], the semisupervised methods DSMR [31] and GCN [21]). Moreover, in order to alleviate the influence from the speckle noise of the PolSAR datasets, the Lee filtering [41] with 5 × 5 window size is applied to all the datasets for preprocessing. We conducted all the experiments on a PC with Intel Xeon E5 core clocked at 2.6 GHz and a GPU with 11 GB memory. Moreover, we run all the methods 20 times and report both the average overall accuracy (OA) and standard deviation.

A. Discussion on Architectures

In this section, we discuss the architecture of our DFGCN including GResM and GCM, and analyze the performance of DFGCN with different number of graph residual and convolutional layers. We conduct some experiments on a sub-imagery of the Flevoland dataset with size of 200 × 100 and 6 classes including bare soil, potatoes, beet, forest, wheat, and peas. For our DFGCN, the graph residual layer starts from its third layer. Moreover, we compare the architectures of DFGCN with GCN [21] and DeepGCN [22], as listed in Table I. Note that DeepGCN denotes ResGCN in [22], which usually has better performance than DenseGCN in [22], and thus, we did not report the results of DenseGCN in this article.

The OA results of the three GCNs (i.e., GCN, DeepGCN, and DFGCN) with different number of layers are reported in Fig. 3, where the number of the hidden layers is chosen in the range [2, 16]. It is clear that our DFGCN consistently attains much better OA results than other networks in all the settings. When DFGCN has only two hidden layers, its OA result is 92.35%, while those of GCN and DeepGCN are 87.03% and 84.21%, respectively. This illustrates the importance of our fuzzy graph, which makes full use of both feature and spatial information and plays an important role in DFGCN. Moreover, the OA results of our DFGCN become slightly better when the number of layers increases, while those of GCN drop dramatically when the number of graph convolutional layers is greater than four. The almost same phenomena has also been witnessed in the literature [42], [43], which means that most state-of-the-art traditional GCN algorithms are no deeper than three or four layers. The main reason may be that the gradients become very small (i.e., vanishing gradient), when the number of the hidden layers is greater than four. In contrast, DFGCN uses the graph residual layers after the two graph convolutional layers, and thus, it can still perform very robust even when it is deeper than 15 layers. The results in Fig. 3 show that the OA results of DeepGCN descend slowly as the number of layers increases from 2 to 16. Although DeepGCN is constructed by using a graph residual block in each hidden layer, its OA results are much worse than those of DFGCN, which indicates the feasibility of DFGCN and the importance of GCM. The results also indicate that the setting of $\xi = 2$ and $M = 16$ is an appropriate choice for our DFGCN on this dataset, and the parameter setting is also referred for other datasets in subsequent experiments.

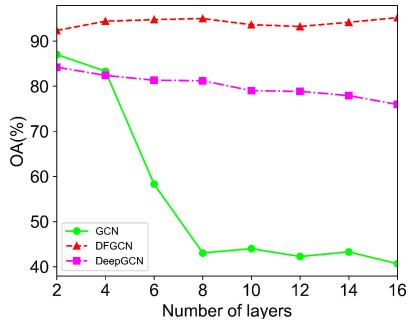


Fig. 3. Comparison of the three architectures (i.e., GCN [21], DeepGCN [22], and DFGCN) with different number of layers.

TABLE II
ABLATION STUDIES OF DIFFERENT MODULES IN OUR DFGCN WITH DIFFERENT NUMBER OF HIDDEN UNITS

Number of hidden units	8	16	32	64
PlainG + GCM(2)	64.28	77.82	87.03	86.32
PlainG + GCM(4)	57.32	66.58	83.31	88.17
PlainG + GCM(8)	45.68	45.85	43.05	49.49
PlainG + GCM(12)	45.84	45.18	42.26	40.66
PlainG + GCM(16)	41.39	41.40	40.66	38.06
FuzzyG + GCM(2)	86.69	89.58	91.62	93.10
FuzzyG + GCM(4)	91.76	93.34	94.31	93.47
FuzzyG + GCM(8)	65.47	91.16	89.09	89.80
FuzzyG + GCM(12)	57.05	74.20	75.20	71.77
FuzzyG + GCM(16)	49.64	73.24	69.01	66.12
PlainG + GCM(1) + GResM(1)	67.00	77.57	86.64	85.84
PlainG + GCM(2) + GResM(2)	69.51	85.11	88.88	90.49
PlainG + GCM(2) + GResM(6)	56.02	76.23	87.90	90.22
PlainG + GCM(2) + GResM(10)	39.57	54.44	66.80	74.16
PlainG + GCM(2) + GResM(14)	27.54	37.62	60.99	64.59
FuzzyG + GCM(1) + GResM(1)	88.85	89.89	92.35	**93.59**
FuzzyG + GCM(2) + GResM(2)	94.47	94.14	**94.82**	94.80
FuzzyG + GCM(2) + GResM(6)	80.78	93.34	**94.99**	94.71
FuzzyG + GCM(2) + GResM(10)	69.70	90.88	**93.25**	92.21
FuzzyG + GCM(2) + GResM(14)	52.73	85.88	**95.20**	91.66

Bold represents the best results.

B. Ablation Studies

In this section, we conduct comprehensive ablation studies to show the effect of different components [i.e., fuzzy graph (FuzzyG), GCM, and GResM] in our architecture. The experimental results on the subdataset of Flevoland are reported in Table II. One of the compared modules is the plain graph

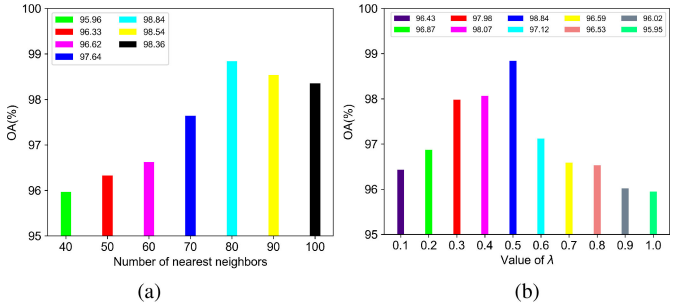


Fig. 4. Classification accuracy versus the number of nearest neighbors (NNs, left) and the parameter λ (right) on the Flevoland dataset.

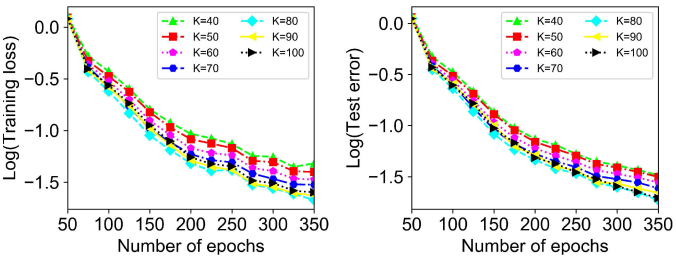


Fig. 5. Training error and test error of our DFGCN with different epochs on the Flevoland dataset.

(PlainG), which is constructed with binary weights and is used as the input for the GCM. We study the OA results of the deep networks with different parameters, e.g., the number of layers (e.g., 2, 4, 8, 12, 16) and the number of hidden units (e.g., 8, 16, 32, 64). Moreover, we also study the following cases: our fuzzy graph (FuzzyG) with GCM, PlainG with GCM and GResM, and our FuzzyG with GCM and GResM. All the results indicate several conclusions.

1) The fuzzy graph greatly enhances the OA results, as shown in the first five rows and the second five rows in Table II, while the OA results of PlainG+GCM in almost all the hidden neuron settings become worse as the depth increases, which further means that most traditional GCN algorithms are no deeper than three or four layers. The comparison shows that the fuzzy graph, which is constructed using both physical features and spatial information, brings rich information for feature learning.
2) The graph residual module improves the OA results, when it is used after the second hidden layer, as shown in the first five rows and the third five rows. Especially, when the number of hidden units is 32 or 64, the GResM improves the learning ability of the network.
3) The GCM together with the GResM and fuzzy graph dramatically enhance the performance of our network, which is verified by the last five rows in Table II, where FuzzyG+GCM(2)+GResM(2) denotes the network with two graph convolutional layers and two graph residual layers.

C. Parameter Settings

The parameters of all the methods are listed below.

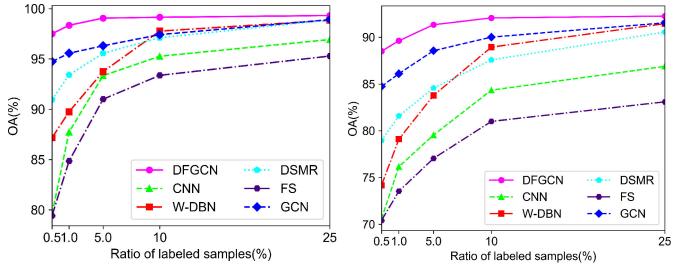


Fig. 6. Classification accuracies of all the methods with varying ratios of labeled samples.

1) FS [11]: Here the number of superpixels K is selected in the interval $[500, 3000]$, and the compactness of the superpixels $mpol$ is chosen among the interval $[20, 60]$.
2) W-DBN[14]: W-DBN has two hidden layers, whose node numbers are set to 50 and 100, respectively. The learning rate is set to 0.01, the thresholds τ_0 is chosen in the interval [0.95, 0.99], ρ_0 is set to 0, and the window size is set to 3 or 5.
3) CNN [18]: The CNN includes two convolution layers, two max-pooling layers and one fully connected layer. The sizes of the filters in two convolutional layers are 3×3 and 2×2, respectively, and the pooling size is 2×2. The momentum parameter is 0.9, and the weight decay rate is set to 5×10^{-4}.
4) DSMR [31]: We choose the number of nearest neighbors among the interval [10,20], and the regularization parameter λ in the interval $[1 \times 10^3, 1 \times 10^4]$. The weight decay rate β is chosen in the interval $[1 \times 10^{-4}, 1 \times 10^{-3}]$.
5) GCN [21]: Three hidden layers are used in the GCN network and their node numbers are set to 64, 32, and 15, respectively. The adjacency matrix A is a binary, symmetric matrix, and the number of nearest neighbors is chosen in the interval [50,100]. The optimization method is Adam, the learning rate is 0.1, the weight reduction factor is 5×10^{-6}, and the dropout rate is 0.5.
6) Our DFGCN: The number of nearest neighbors is chosen in the interval [60,100]. The chosen optimization method is Adam and its learning rate is 0.05. The weight reduction factor is 5×10^{-6}, and the dropout rate is set to 0.5.

Other parameters are manually tuned according to specific classification tasks and datasets. Note that for each dataset, 1% samples from each class are randomly selected as labeled samples, and an additional 5% labeled samples are used as validation set to determine some parameters and avoid overfitting. For all the methods, 41 features frequently used in the literature are used as the inputs for PolSAR imagery classification as multiple features can enhance the diversity of the training data. More details about the features can be found in the Supplementary Materials. Moreover, the Flevoland dataset commonly used in the literature is taken as an example to indicate how to determine the following parameters, and the complete results on this dataset are also included in Supplementary Materials.

1) Number of Nearest Neighbors K: As the number of nearest neighbors K increases, the classification accuracy of DFGCN varies, as shown in the left of Fig. 4(a). If the number of nearest neighbors K is too small (e.g., $K = 20$), the local spatial

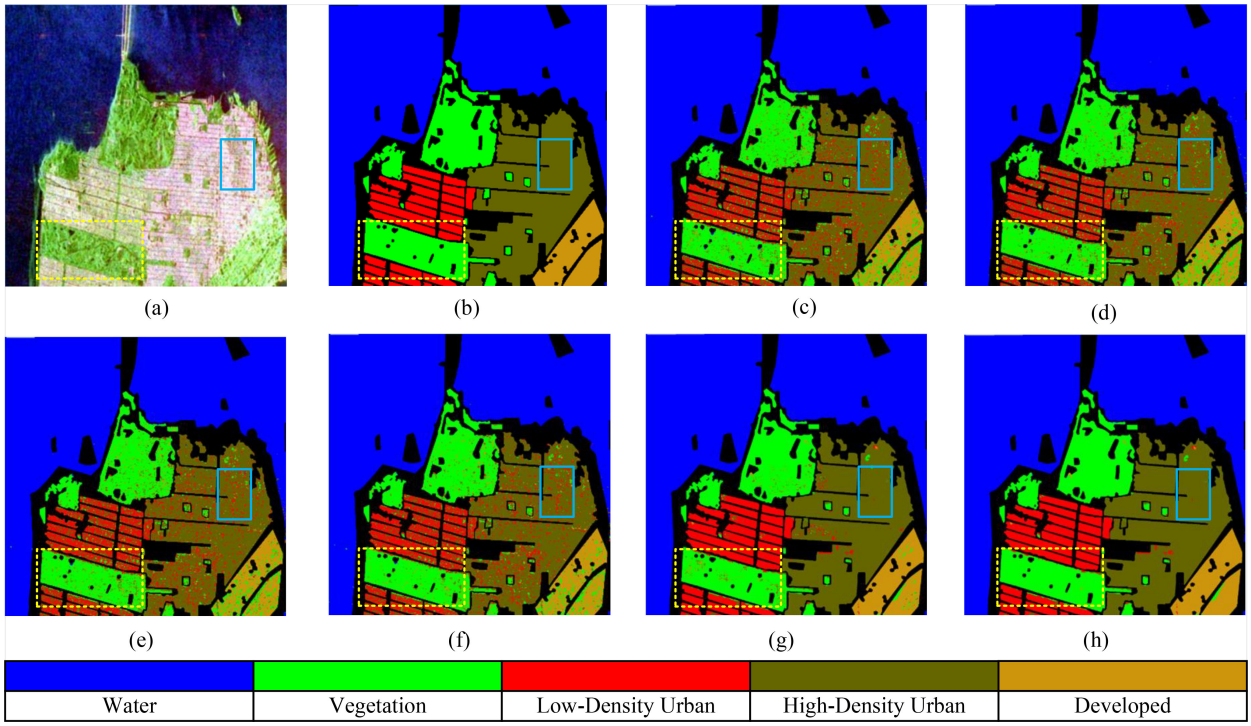


Fig. 7. Classification results of DFGCN and other algorithms on the San Francisco dataset. (a) Pauli RGB image. (b) Ground truth. (c) FS [11]. (d) W-DBN[14]. (e) CNN [18]. (f) DSMR [31]. (g) GCN [21]. (h) DFGCN (ours).

information can not be used sufficiently, and then the classification accuracy falls down, which is verified by Fig. 4(a). On the contrary, if K is too large (e.g., $K = 100$), many surrounding pixels are likely to be included in one class (i.e., the neighboring pixels may not be divided correctly), and then the classification accuracy decreases. This is verified by the results shown in Fig. 4(a). Moreover, the training error and test error vary with different epochs, as demonstrated in Fig. 5. The experimental results indicate that DFGCN can obtain a higher accuracy when K is set to 80.

2) Coefficient λ of Distance Weighting: Moreover, we also study that the classification accuracies vary with the change of the coefficient λ, as shown in Fig. 4(b). When λ is greater than 0.5, the classification accuracy of DFGCN becomes worse. For instance, the accuracy in the case of $\lambda = 0.4$ is superior to that of $\lambda = 0.6$. That is, the Wishart distance is less dominant than the Euclidean distance. In particular, when λ equals to 0.5, we can obtain the highest accuracy. Therefore, the coefficient λ is set to 0.5 for this dataset. As the Flevoland dataset mainly contains several crops which are more homogenous in structure, the Wishart distribution for most of the pixels may not be evident. The higher classification accuracy is obtained when the value of λ is 0.5 or slightly lower than 0.5. For other datasets which are more heterogenous, the value of λ can be set as 0.5 or slightly larger than 0.5.

D. Results on San Francisco Dataset

This dataset is *C*-band, single-look, and full-polarimetric SAR data acquired by RADARSAT-2. It is the bay of San Francisco with the golden gate bridge of size 1300 $\times$ 1300. This dataset provides a coverage of both natural targets and man-made targets, especially with heterogeneous regions. Its Pauli RGB image is illustrated in Fig. 7(a), and the ground truth image is obtained by referring to [44] shown in Fig. 7(b). It has five terrain types, including water, vegetation, low-density urban areas, high-density urban areas, and developed areas. The classification results of all the methods are shown in Fig. 7(c)–(h). It is clear that our DFGCN shows the best visual effect in the high-density urban, low-density urban, and developed areas. Especially, for the vegetation area mixed with lakes which is heterogeneous shown in dashed rectangular, there exist much less misclassified pixels than those of other methods, and the homogeneous region shown in solid rectangular is more uniform than that of other methods.

The classification accuracies of all the methods are reported in Table III. The OA of DFGCN is the highest (i.e., 98.35%), and it is 15.10%, 10.62%, 10.03%, and 5.95% higher than those of FS, CNN, W-DBN, and DSMR, respectively. The results indicate that the advantage of semisupervised DFGCN. Moreover, for both low-density urban area and developed area, the classification accuracies of DFGCN are 36.00% and 29.65% higher than those of FS, respectively. It demonstrates the role of the utilization of neighboring pixels with any shape for graph convolution in the pixel based methods. For the high-density urban area, the classification accuracy of DFGCN is 5.05% higher than that of GCN due to the spatially weighted fuzzy graph matrix and the deep architecture proposed in our work. We also report the OAs of all the algorithms with different ratios of labeled samples in Fig. 6(a), which further verifies that our DFGCN significantly outperforms other methods in terms of accuracy.

TABLE III
CLASSIFICATION ACCURACIES OF ALL THE METHODS WITH 1% LABELED SAMPLES ON THE SAN FRANCISCO DATASET

Methods	Water	Vegetation	Low-Density Urban	High-Density Urban	Developed	OA
FS [11]	90.63±1.776	79.28±1.675	61.33±1.987	80.13±1.888	68.21±1.811	83.25±1.803
W-DBN [14]	99.77±2.780	89.56±2.913	57.64±2.761	83.72±2.661	65.21±2.703	89.75±2.770
CNN [18]	98.56±2.257	81.71±2.164	53.24±2.538	85.03±2.252	62.35±2.290	87.73±2.267
DSMR [31]	99.89±1.464	91.63±1.634	69.34±1.656	87.78±1.552	74.91±1.589	92.40±1.529
GCN [21]	98.93±1.469	92.03±1.730	92.23±1.489	91.21±1.349	95.09±1.147	95.57±1.469
DFGCN (ours)	**99.99±1.455**	**96.36±1.422**	**97.33±1.140**	**96.26±1.433**	**97.86±1.355**	**98.35±1.414**

Bold represents the best results.

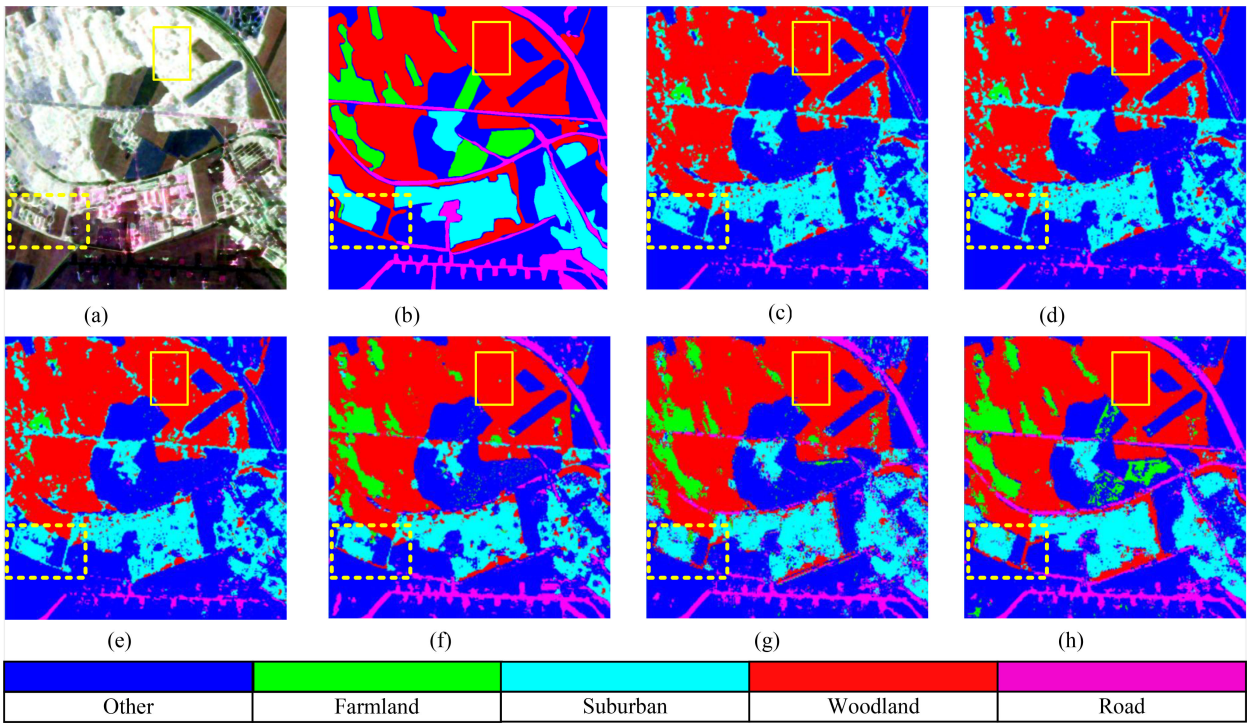


Fig. 8. Classification results of DFGCN and other algorithms on the Oberpfaffenhofen dataset. (a) Pauli RGB image. (b) Ground truth. (c) FS [11]. (d) W-DBN[14]. (e) CNN [18]. (f) DSMR [31]. (g) GCN [21]. (h) DFGCN (ours).

TABLE IV
CLASSIFICATION ACCURACIES OF ALL THE METHODS WITH 1% LABELED SAMPLES ON THE OBERPFAFFENHOFEN DATASET

Methods	Other	Farmland	Suburban	Woodland	Road	OA
FS [11]	83.22±1.876	75.15±1.883	78.11±1.506	35.22±1.783	35.01±1.780	72.54±1.813
W-DBN [14]	88.50±2.817	79.96±2.622	81.05±2.584	50.62±2.478	51.39±2.538	79.11±2.683
CNN [18]	88.45±2.303	76.34±2.148	83.51±2.315	36.89±2.186	35.33±2.213	76.16±2.246
DSMR [31]	89.34±1.655	86.08±1.753	80.21±1.607	53.73±1.655	59.20±1.656	81.58±1.680
GCN [21]	90.16±1.419	89.71±1.422	89.18±1.401	73.52±1.298	70.01±1.442	86.90±1.410
DFGCN (ours)	**93.35±1.449**	**92.19±1.181**	**93.64 ±1.390**	**75.83±1.250**	**73.43±1.210**	**90.12±1.333**

Bold represents the best results.

E. Results on Oberpfaffenhofen Dataset

This is an *L*-band, multilook PolSAR imagery collected by the German Aerospace Center's E-SAR system. The size of the original image is 1300 × 1200 pixels. In order to show the details for the classification of heterogeneous region, we cropped a subregion with 740 × 750 pixels. Its PauliRGB image is shown in Fig. 8(a), and its ground truth map is attained by referring to [17] shown in Fig. 8(b). It contains five types of terrain, which are farmland, suburban, woodland, road land, and others. The classification results of all the algorithms are shown in Fig. 8(c)–(h). Though it is challenging for the classification on this miscellaneous dataset, it is obvious that DFGCN yields much better results than other algorithms. The road and the narrow woodland around the suburban area marked in dashed box are much clearer than those of other methods. And the homogenous woodland marked in solid box is more uniform. The classification accuracies of all the methods and networks are listed in Table IV. The OA result of DFGCN is much better than those of the other methods, and is 17.58%, 13.96%, 12.58%, 8.54% higher than those of FS, CNN, W-DBN, and DSMR,

respectively. It probably benefits from the graph convolution which extracts intuitive features for classification. Moreover, for the suburban, the classification accuracy of DFGCN is 4.46% higher than that of GCN, which verified that both our fuzzy graph and residual module can significantly improve classification accuracy. Similarly, Fig. 6(b) shows the overall accuracies of all the methods vary with the ratio of labeled samples. Our DFGCN obtains the highest accuracy with all the ratios of labeled samples compared with those of GCN, DSMR, W-DBN, CNN, and FS. It can be seen when the labeled samples are only 0.5%, our DFGCN obtains the OA result greater than 87%. When the ratio of the labeled samples becomes 10%, our DFGCN gains the higher than 92% OA. It benefits from our defined fuzzy graph and the proposed GResM in the proposed method.

V. Conclusion

In this article, a novel DFGCN is proposed for classification of PolSAR imagery. The fuzzy logic is utilized to construct a fuzzy graph, whose nodes are pixels connecting to the nearest neighbors including both few labeled and large numbers of unlabeled ones, and the weights are calculated adaptively using both physical properties and spatial information of the pixels. By using graph convolutional layers and stacked multiple residual layers, our DFGCN can learn intuitive representations of input PolSAR data and performs semisupervised classification with only a few labeled samples. The experimental results on various real-world datasets demonstrate the superior performance of our network compared with state-of-the-art ones. The advantages of our DFGCN have also been verified by the experiments. 1) The key idea of using fuzzy logic to construct deep graph CNNs can greatly enhance the classification accuracy for both homogeneous and heterogeneous terrains. 2) The exploration of the nearest neighbors contributes to semisupervised learning, which dramatically cut down the number of labeled samples. This provides inspirations for the design of semisupervised deep neural networks for other researchers. It is noticed that the selection of some of the parameters is by experiment and experience. In future work, more sophisticated methodologies, such as random search [45] and gradient-based search [46], will be studied to find the optimal values for the parameters. Furthermore, more datasets with heterogeneous will be used to evaluate the proposed DFGCN.

References

[1] C. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer, 2006.

[2] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach. Learn.*, vol. 109, no. 2, pp. 1573–0565, 2019.

[3] S. R. Cloude and E. Pottier, "An entropy based classification scheme for land applications of polarimetric SAR," *IEEE Trans. Geosci. Remote Sens.*, vol. 35, no. 1, pp. 68–78, Jan. 1997.

[4] J.-S. Lee and E. Pottier, *Polarimetric Radar Imaging: from Basics to Applications*. Boca Raton, FL, USA: CRC Press, 2009.

[5] G. Vasile, J.-P. Ovarlez, F. Pascal, and C. Tison, "Coherency matrix estimation of heterogeneous clutter in high-resolution polarimetric SAR images," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 4, pp. 1809–1826, Apr. 2010.

[6] L. Pallotta, C. Clemente, A. De Maio, and J. J. Soraghan, "Detecting covariance symmetries in polarimetric SAR images," *IEEE Trans. Geosci. Remote Sens.*,vol. 55, no. 1, pp. 80–95, Jan. 2017.

[7] L. Pallotta, A. D. Maio, and D. Orlando, "A robust framework for covariance classification in heterogeneous polarimetric SAR images and its application to L-band data," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 1, pp. 104–119, Jan. 2019.

[8] L. Pallotta and D. Orlando, "Polarimetric covariance eigenvalues classification in SAR images," *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 5, pp. 746–750, May 2019.

[9] T. Eltoft and A. P. Doulgeris, "Model-based polarimetric decomposition with higher order statistics," *IEEE Geosci. and Remote Sens. Lett.*, vol. 16, no. 6, pp. 992–996, Jun. 2019.

[10] B. Liu, H. Hu, H. Wang, K. Wang, X. Liu, and W. Yu, "Superpixel-based classification with an adaptive number of classes for polarimetric SAR images," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 2, pp. 907–924, Feb. 2013.

[11] Y. Guo, L. Jiao, S. Wang, S. Wang, F. Liu, and W. Hua, "Fuzzy superpixels for polarimetric SAR images classification," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 5, pp. 2846–2860, Oct. 2018.

[12] H. Xie, S. Wang, K. Liu, S. Lin, and B. Hou, "Multilayer feature learning for polarimetric synthetic radar data classification," in *Proc. IEEE Geosci Remote Sens. Symp.*, 2014, pp. 2818–2821, 2014.

[13] H. Bi, J. Sun, and Z. Xu, "Unsupervised PoISAR image classification using discriminative clustering," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 6, pp. 3531–3544, Jun. 2017.

[14] F. Liu, L. Jiao, B. Hou, and S. Yang, "POL-SAR image classification based on Wishart DBN and local spatial information," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 6, pp. 1–17, Jun. 2016.

[15] X. She, J. Yang, and W. Zhang, "The boosting algorithm with application to polarimetric SAR image classification," in *Proc. 1st Asian Pacific Conf. Synthetic Aperture Radar*, 2007, pp. 779–783.

[16] T. Zou, W. Yang, D. Dai, and H. Sun, "Polarimetric SAR image classification using multifeatures combination and extremely randomized clustering forests," *EURASIP J. Adv. Signal Proc.*, vol. 2010, pp. 1–9, 2009.

[17] C. He, S. Li, Z. Liao, and M. Liao, "Texture classification of PolSAR data based on sparse coding of wavelet polarization textons," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 8, pp. 4576–4590, Aug. 2013.

[18] Y. Zhou, H. Wang, F. Xu, and Y.-Q. Jin, "Polarimetric SAR image classification using deep convolutional neural networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 12, pp. 1935–1939, Dec. 2016.

[19] F. Mohammadimanesh, B. Salehi, M. Mahdianpari, E. Gill, and M. Molinier, "A new fully convolutional neural network for semantic segmentation of polarimetric SAR imagery in complex land cover ecosystem," *ISPRS J. Photogrammetry Remote Sens.*, vol. 155, pp. 223–236, 2019.

[20] H. Bi, F. Xu, Z. Wei, Y. Xue, and Z. Xu, "An active deep learning approach for minimally supervised PolSAR image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 11, pp. 9378–9395, Nov. 2019.

[21] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.

[22] G. Li, M. Muller, A. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs go as deep as CNNs?," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, vol. 1, pp. 9267–9276.

[23] H. Liu, Y. Wang, S. Yang, S. Wang, J. Feng, and L. Jiao, "Large polarimetric SAR data semi-supervised classification with spatial-anchor graph," *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.*, vol. 9, no. 4, pp. 1439–1458, Apr. 2016.

[24] H. Liu *et al.*, "Semisupervised feature extraction with neighborhood constraints for polarimetric SAR classification," *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.*, vol. 9, no. 7, pp. 3001–3015, Jul. 2016.

[25] H. Liu, S. Yang, S. Gou, D. Zhu, R. Wang, and L. Jiao, "Polarimetric SAR feature extraction with neighborhood preservation-based deep learning," *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.*, vol. 10, no. 4, pp. 1456–1466, Apr. 2017.

[26] H. Liu, S. Yang, S. Gou, P. Chen, Y. Wang, and L. Jiao, "Fast classification for large polarimetric SAR data based on refined spatial-anchor graph," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 9, pp. 1589–1593, Sep. 2017.

[27] H. Liu, S. Yang, S. Gou, S. Liu, and L. Jiao, "Terrain classification based on spatial multi-attribute graph using polarimetric SAR data," *Appl. Soft Comput.*, vol. 68, pp. 24–38, 2018.

[28] H. Liu, Z. Wang, F. Shang, S. Yang, S. Gou, and L. Jiao, "Semi-supervised tensorial locally linear embedding for feature extraction using PolSAR data," *IEEE J. Sel. Top. Signal Proc.*, vol. 12, no. 6, pp. 1476–1490, Dec. 2018.

[29] H. Liu, F. Wang, S. Yang, B. Hou, L. Jiao, and R. Yang, "Fast semisupervised classification using histogram-based density estimation for large-scale polarimetric SAR data," *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 12, pp. 1844–1848, Dec. 2019.

[30] H. Bi, J. Sun, and Z. Xu, "A graph-based semisupervised deep learning model for PolSAR image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 4, pp. 2116–2132, Apr. 2019.

[31] H. Liu, F. Shang, S. Yang, M. Gong, T. Zhu, and L. Jiao, "Sparse manifold-regularized neural networks for polarimetric SAR terrain classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 3007–3016, Aug. 2020.
[32] J. Zhou *et al.*, "Graph neural networks: A review of methods and applications," 2018, *arXiv:1812.08434*.
[33] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, pp. 395–416, 2007.
[34] S. N. Anfinsen, R. Jenssen, and T. Eltoft, "Spectral clustering of polarimetric SAR data with Wishart-derived distance measures," in *Proc. POLinSAR*, 2007, vol. 7, pp. 1–9.
[35] P. R. Kersten, J. S. Lee, and T. L. Ainsworth, "Unsupervised classification of polarimetric synthetic aperture radar images using fuzzy clustering and em clustering," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 3, pp. 519–527, Mar. 2005.
[36] D. Otair, "Approximate k-nearest neighbour based spatial clustering using Kd tree," *Int. J. Database Manag. Syst.*, vol. 5, no. 1, pp. 97–108, 2013.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[38] G. Li, M. Müller, A. Thabet, and B. Ghanem, "Can GCNS go as deep as CNNs?," in *Proc. IEEE Conf. Comput. Vis.*, 2019, pp. 9266–9275.
[39] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2261–2269.
[40] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2472–2481.
[41] J.-S. Lee, M. Grunes, and G. de Grandi, "Polarimetric SAR speckle filtering and its implication for classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 37, no. 5, pp. 2363–2373, Sep. 1999.
[42] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. 32nd Conf. Artif. Intell.*, 2018, vol. 1, pp. 3538–3545.
[43] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," in *IEEE Trans. Neural Networks Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2020.2978386.
[44] S. Uhlmann and S. Kiranyaz, "Integrating color features in polarimetric SAR image classification," *Remote Sens. Land Resour.*, vol. 52, no. 4, pp. 2197–2216, 2017.
[45] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 281–305, 2012.
[46] Y. Bengio, "Gradient-based optimization of hyperparameters," *Neural Comput.*, vol. 12, no. 8, pp. 1889–1900, 2000.

Hongying Liu (Member, IEEE) received the B.E. and M.S. degrees in computer science and technology from the Xi'an University of Technology, Xi'an, China, in 2006 and 2009, respectively, and the Ph.D. degree in engineering from Waseda University, Tokyo, Japan in 2012.

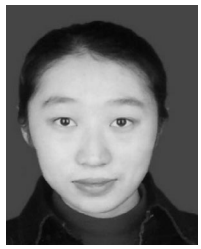
Currently, she is a Faculty Member with the School of Artificial Intelligence, and also with the Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, Xidian University, Xi'an, China. Her major research interests include image processing, intelligent signal processing, machine learning, etc.

Tianwen Zhu received his B.S. in software engineering from Lanzhou University of Technology, Lanzhou, China in 2019. He is currently working toward the M.Sc. degree in computer science and technology with the School of Artificial Intelligence, Xidian University, Xi'an, China.

His research interests include machine learning and distributed system, etc.

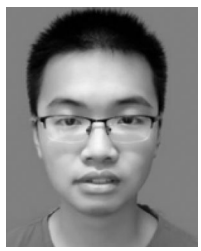
Fanhua Shang (Senior Member, IEEE) received the Ph.D. degree in circuits and systems from Xidian University, Xi'an, China, in 2012.

He is currently a Professor with the School of Artificial Intelligence, Xidian University. Prior to joining Xidian University, he was a Research Associate with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. From 2013 to 2015, he was a Postdoctoral Research Fellow with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. From 2012 to 2013, he was a Postdoctoral Research Associate with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA. His current research interests include machine learning, data mining, pattern recognition, and computer vision.



Yuanyuan Liu (Member, IEEE) received the Ph.D. degree in pattern recognition and intelligent system from Xidian University, Xi'an, China, in 2013.

She is currently a Professor with the School of Artificial Intelligence, Xidian University. Prior to joining Xidian University, she was a Postdoctoral Research Fellow with the Department of Computer Science and Engineering, The Chinese University of Hong Kong (CUHK). From 2013 to 2014, she was a Postdoctoral Research Fellow with the Department of Systems Engineering and Engineering Management. Her current research interests include machine learning, pattern recognition, and image processing.



Derui Lv is currently working toward the B.S. degree in intelligence science and technology with the School of Artificial Intelligence, Xidian University, Xi'an, China.

His major research interests include machine learning and image processing, etc.

Shuyuan Yang (Senior Member, IEEE) received the B.A. degree in electrical engineering from Xidian University, Xi'an, China, in 2000, the M.S. and Ph.D. degrees in circuit and system from Xidian University, Xi'an, China, in 2003 and 2005, respectively.

She is currently a Professor with the School of Artificial Intelligence, Xidian University. Her main current research interests are machine learning and multiscale geometric analysis, etc.