Deep Recurrent Neural Networks for Hyperspectral Image Classification

Lichao Mou, *Student Member, IEEE*, Pedram Ghamisi, *Member, IEEE*, and Xiao Xiang Zhu, *Senior Member, IEEE*

***Abstract*— In recent years, vector-based machine learning algorithms, such as random forests, support vector machines, and 1-D convolutional neural networks, have shown promising results in hyperspectral image classification. Such methodologies, nevertheless, can lead to information loss in representing hyperspectral pixels, which intrinsically have a sequence-based data structure. A recurrent neural network (RNN), an important branch of the deep learning family, is mainly designed to handle sequential data. Can sequence-based RNN be an effective method of hyperspectral image classification? In this paper, we propose a novel RNN model that can effectively analyze hyperspectral pixels as sequential data and then determine information categories via network reasoning. As far as we know, this is the first time that an RNN framework has been proposed for hyperspectral image classification. Specifically, our RNN makes use of a newly proposed activation function, parametric rectified tanh (PRetanh), for hyperspectral sequential data analysis instead of the popular tanh or rectified linear unit. The proposed activation function makes it possible to use fairly high learning rates without the risk of divergence during the training procedure. Moreover, a modified gated recurrent unit, which uses PRetanh for hidden representation, is adopted to construct the recurrent layer in our network to efficiently process hyperspectral data and reduce the total number of parameters. Experimental results on three airborne hyperspectral images suggest competitive performance in the proposed mode. In addition, the proposed network architecture opens a new window for future research, showcasing the huge potential of deep recurrent networks for hyperspectral data analysis.**

***Index Terms*— Convolutional neural network (CNN), deep learning, gated recurrent unit (GRU), hyperspectral image classification, long short-term memory (LSTM), recurrent neural network (RNN).**

I. Introduction

IN THE past few decades, the analysis of hyperspectral imagery acquired by remote sensors has attracted considerable attention in the remote sensing community, as such data are characterized in hundreds of continuous observation bands throughout the electromagnetic spectrum with high spectral resolution [1]. With this rich spectral information, different land cover categories can potentially be precisely differentiated. To benefit from this type of data, supervised hyperspectral image classification plays a significant role and has been investigated in many applications, including urban development [2]–[5], the monitoring of land changes [6]–[9], scene interpretation [10]–[13], and resource management [14], [15].

Numerous types of supervised classification models have been discussed in the literature, including decision trees [16], random forests [17], [18], and support vector machines (SVMs) [19], [20]. Among them, the random forest [18] develops multiple trees from randomly sampled subspaces of input hyperspectral pixel vectors and then combines the outputs via voting or a maximum *a posteriori* rule. In contrast, SVM, a supervised machine learning technique, has achieved great success in various applications and is considered a stable and efficient algorithm for hyperspectral image classification tasks. An SVM seeks to separate two-class data by learning an optimal decision hyperplane that can best separate the training samples in a kernel-included high dimensional feature space. Some strategies, such as one-against-all and one-against-one, enable the use of original binary SVM for multiclass classification. In addition, some extensions of the SVM model in hyperspectral image classification have been presented to improve the classification performance [21], [22].

When the ratio of the number of available training samples and the number of spectral bands is unbalanced, theoretical and practical problems may arise and the hyperspectral image classification becomes an ill-posed problem. For example, while keeping the number of available training samples constant, the classification accuracy will decrease when the dimension of input feature vectors becomes large [23], [24].

In recent years, deep learning has made promising achievements in the machine learning field [25]–[29]. It attempts to learn hierarchical representations from raw data and is capable of learning simple concepts first and then successfully building up more complex concepts by merging the simpler ones. In remote sensing, convolutional neural networks (CNNs) have been shown to be successful for hyperspectral data classification [30]–[32]. Hu *et al.* [30] presented a CNN that contains an input layer, a convolutional layer, a max-pooling layer, a fully connected layer, and an output layer for hyperspectral image classification. The CNN has been employed to classify hyperspectral data directly in the spectral domain. Makantasis *et al.* [31] presented a deep learning-based classification method that hierarchically constructs high-level features automatically. In particular, their model exploits

Manuscript received August 26, 2016; revised October 29, 2016; accepted November 15, 2016. Date of publication April 28, 2017; date of current version June 22, 2017. This work was supported in part by the China Scholarship Council, in part by the Alexander von Humboldt Foundation, in part by the Helmholtz Association through the framework of the Young Investigators Group SiPEO under Grant VH-NG-1018, and in part by the European Research Council (ERC) under the European Unions Horizon 2020 Research and Innovation Programme under Grant ERC-2016-StG-714087 (Acronym: So2Sat). *(Corresponding author: Xiao Xiang Zhu.)*

The authors are with the Remote Sensing Technology Institute, German Aerospace Center, 82234 Wessling, Germany, and also with Signal Processing in Earth Observation, Technical University of Munich, 80333 Munich, Germany (e-mail: lichao.mou@dlr.de; pedram.ghamisi@dlr.de; xiao.zhu@dlr.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TGRS.2016.2636241

a CNN to encode the spectral and spatial information of pixels and a multilayer perceptron to conduct the classification task. Chen *et al.* [32] proposed a regularized 3-D CNN-based feature extraction model to extract efficient spectral-spatial features for hyperspectral image classification. In addition, Chen *et al.* [33] proposed a hybrid framework based on stacked autoencoders for the classification of the hyperspectral data.

All of the supervised models for hyperspectral images described earlier are vector-based methodologies.[1] It should be noted that these vector-based approaches can lead to information loss when representing hyperspectral pixels, which intrinsically have a sequence-based data structure. To the best of our knowledge, almost all advanced spectral classifiers, such as SVM, random forest, and CNN-based classification, are vector-based approaches, which consider hyperspectral data to be a collection of pixel vectors and perform the classification procedure in feature space: each pixel is considered a point in an orderless d-dimensional feature space in which d represents the number of dimensions (bands) [1]. However, hyperspectral data can be seen as a set of orderly and continuing spectra sequences in the spectral space. Analyzing hyperspectral imagery from a sequential perspective has not been addressed so far. Our motivation in this paper is to explore the representation of hyperspectral pixels via the sequential perspective instead of classifying hyperspectral data in the feature space.

In this paper, we make use of a recurrent neural network (RNN) to characterize the sequential property of a hyperspectral pixel vector for the classification task. An RNN [34]–[36] is a network that uses recurrent connections between neural activations at consecutive time steps; such a network uses hidden layers or memory cells to learn the states that model the underlying dynamics of the input sequence for sequential data over time. RNNs have gained significant attention for solving many challenging problems involving sequential data analysis, such as language modeling [37], machine translation [38], and speech recognition [39], [40]. Since the temporal variability of a sequential signal, such as a language sentence, is similar to the spectral variability of a hyperspectral pixel, the same idea can be applied to hyperspectral pixel vectors. The RNN exploits a recurrent procedure to characterize spectral correlation and band-to-band variability, where the network parameters are determined by training with available samples. In this context, we propose a novel RNN with a specially designed activation function and modified gated recurrent unit (GRU) to solve the multiclass classification for hyperspectral imagery. This paper contributes to the literature in three major respects.

1) We represent and process the pixels of hyperspectral images via a sequential perspective instead of taking them as feature vectors to capture the intrinsic sequence-based data structure of hyperspectral pixels. This enables us to take full advantage of the sequential property of hyperspectral data, e.g., spectral correlation and band-to-band variability.
2) An RNN with GRUs is proposed for a hyperspectral image classification task. To the best of our knowledge, this is the first use of the recurrent network model for the problem of hyperspectral image classification.
3) We introduce a new activation function, parametric rectified tanh (PRetanh), which generalizes the rectified unit for the deep RNN and then modifies the proposed activations of GRUs. With this new activation function, fairly high learning rates can be used to train the network without the risk of divergence.

The remainder of this paper is organized as follows. An introduction to RNNs is briefly given in Section II. The details of the proposed RNN architecture for hyperspectral image classification, including a novel activation function and modified GRU, are described in Section III. The network setup, experimental results, and a comparison with state-of-the-art approaches are provided in Section IV. Finally, Section V concludes this paper.

II. Background on Recurrent Neural Networks

An RNN [34], [35] is a class of artificial neural network that extends the conventional feedforward neural network with loops in connections. Unlike a feedforward neural network, an RNN is able to process the sequential inputs by having a recurrent hidden state whose activation at each step depends on that of the previous step. In this manner, the network can exhibit dynamic temporal behavior.

Given a sequence data $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$, where $\mathbf{x}_i$ is the data at ith time step, an RNN updates its recurrent hidden state $\mathbf{h}_t$ by

$$\mathbf{h}_t = \begin{cases} 0, & \text{if } t = 0 \\ \varphi(\mathbf{h}_{t-1}, \mathbf{x}_t), & \text{otherwise} \end{cases} \tag{1}$$

where φ is a nonlinear function, such as a logistic sigmoid function or hyperbolic tangent function. Optionally, the RNN may have an output $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T)$. For some tasks, such as hyperspectral image classification, we need only one output, i.e., $\mathbf{y}_T$.

In the traditional RNN model, the update rule of the recurrent hidden state in (1) is usually implemented as follows:

$$\mathbf{h}_t = \varphi(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1}) \tag{2}$$

where $\mathbf{W}$ and $\mathbf{U}$ are the coefficient matrices for the input at the present step and for the activation of recurrent hidden units at the previous step, respectively.

In fact, an RNN can model a probability distribution over the next element of the sequence data, given its present state $\mathbf{h}_t$, by capturing a distribution over sequence data of variable length. Let $p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ be the sequence probability, which can be decomposed into

$$p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T) = p(\mathbf{x}_1) \cdots p(\mathbf{x}_T | \mathbf{x}_1, \ldots, \mathbf{x}_{T-1}). \tag{3}$$

Then, each conditional probability distribution can be modeled with a recurrent network

$$p(\mathbf{x}_t | \mathbf{x}_1, \ldots, \mathbf{x}_{t-1}) = \varphi(\mathbf{h}_t) \tag{4}$$

[1]Here, vector-based approaches refer to those that consider the input to be vectors. Although CNN-based models consider the inherent relationship of the inputs during the process, such models are still categorized as vector-based models in this paper.

where $\mathbf{h}_t$ is obtained from (1) and (4). Our motivation in this paper is apparent here: a hyperspectral pixel acts as sequential data instead of a feature vector, and so a recurrent network can be adopted to model the spectral sequence.

As an important branch of the deep learning family, RNNs have recently shown promising results in many machine learning and computer vision tasks. However, it has been observed that it is difficult to train the RNNs to deal with long-term sequential data, as the gradients tend to vanish. To address this issue, one common approach is to design a more sophisticated recurrent unit.

Long short-term memory (LSTM) [41], [42] is a special type of recurrent hidden unit, capable of learning long-term dependences. LSTM was initially introduced in [41]. Since then, a number of minor modifications to the original version have been made [42], [43]. A recurrent layer with traditional recurrent hidden units is shown in (2), which simply calculates a weighted linear sum of inputs and then applies a nonlinear function. In contrast, an LSTM-based recurrent layer creates a memory cell $\mathbf{c}_t$ at step t. The activation of the LSTM units can be computed by

$$\mathbf{h}_t = \mathbf{o}_t \tanh(\mathbf{c}_t) \tag{5}$$

where $\tanh(\cdot)$ is the hyperbolic tangent function and $\mathbf{o}_t$ is the output gate that determines the part of the memory content that will be exposed. The output gate is updated by

$$\mathbf{o}_t = \sigma(\mathbf{W}_{oi}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t) \tag{6}$$

where $\sigma(\cdot)$ is a logistic sigmoid function and $\mathbf{W}$ terms denote weight matrices; e.g., $\mathbf{W}_{oi}$ is the input–output weight matrix and $\mathbf{W}_{oc}$ represents the memory-output weight matrix.

The memory cell $\mathbf{c}_t$ is updated by adding new content of memory cell $\tilde{\mathbf{c}}_t$ and discarding part of the present memory content

$$\mathbf{c}_t = \mathbf{i}_t \odot \tilde{\mathbf{c}}_t + \mathbf{f}_t \odot \mathbf{c}_{t-1} \tag{7}$$

where $\odot$ is an elementwise multiplication, and the new content of memory cell $\tilde{\mathbf{c}}_t$ is obtained by

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_{ci}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1}). \tag{8}$$

Input gate $\mathbf{i}_t$ modulates the extent to which the new memory information is added to the memory cell. The degree to which content of the existing memory cell is forgotten is decided by the forget gate $\mathbf{f}_t$. The equations that calculate these two gates are as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i i\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1}) \tag{9}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f i\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1}). \tag{10}$$

Fig. 1 shows the graph model of LSTM.

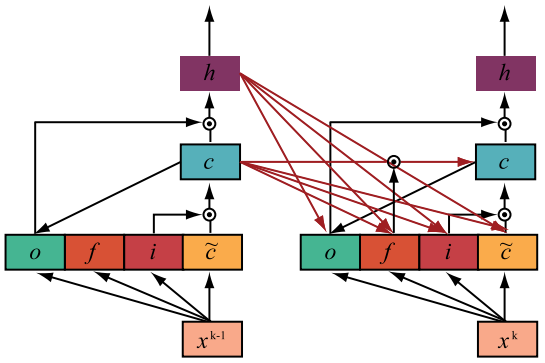


Fig. 1. Graphic model of LSTM. i, f, o, and c are the input gate, forget gate, output gate, and memory cell, respectively. The new memory cell content is denoted by $\tilde{c}$.

III. Proposed Recurrent Network for Hyperspectral Image Classification

In the main procedure of the proposed recurrent network, as shown in Fig. 2, the input of the network is a hyperspectral pixel $\mathbf{x}$, where the kth spectral band is denoted as x^k. The output is a label that indicates the category the pixel belongs to. The entire classification map can be obtained by applying the network to all pixels in the image. The flowchart of our RNN can be summarized as follows.

1) First, the value of the existing spectral band x^k is fed into the input layer.
2) Then, the recurrent layer receives x^k and calculates the hidden state information for the current band; it also restores that information in the meantime.
3) Subsequently, the value of the next band x^{k+1} is input to the recurrent layer simultaneously with the state information of x^k, and the activation at spectral band $k+1$ is computed by a linear interpolation between proposal activation and the activation of the previous band k.
4) Finally, the RNN predicts the label of the input hyperspectral pixel by looping through the entire hyperspectral pixel sequence.

Two important factors affect the performance of RNN: the activation function and the structure of the recurrent unit. In Section III-A and Section III-B, we will discuss our innovative contributions on these two factors in detail.

A. Parametric Rectified tanh

Recently, rectified linear activation functions, such as the rectified linear unit (ReLU) [25], have become a common approach to training deep convolutional networks. They have been proposed to alleviate the vanishing gradient problem and speed up the learning process by identifying positive values; however, this leads to a nonbounded output. We have utilized the proposed activation function instead of the existing ones for several reasons.

1) To train an RNN in our task, the vanishing gradient problem is not a concern, as modern recurrent network models, such as LSTM and GRU, have already been designed to tackle this issue. By using gates, LSTMs and the GRUs help preserve the errors that can be back-propagated through sequence and layers. By maintaining a more constant error, they allow recurrent networks to continue to learn over many bands of hyperspectral pixels without the risk of the vanishing gradient.
2) In our experiments, the recurrent network often runs into numerical problems when a rectified linear function like

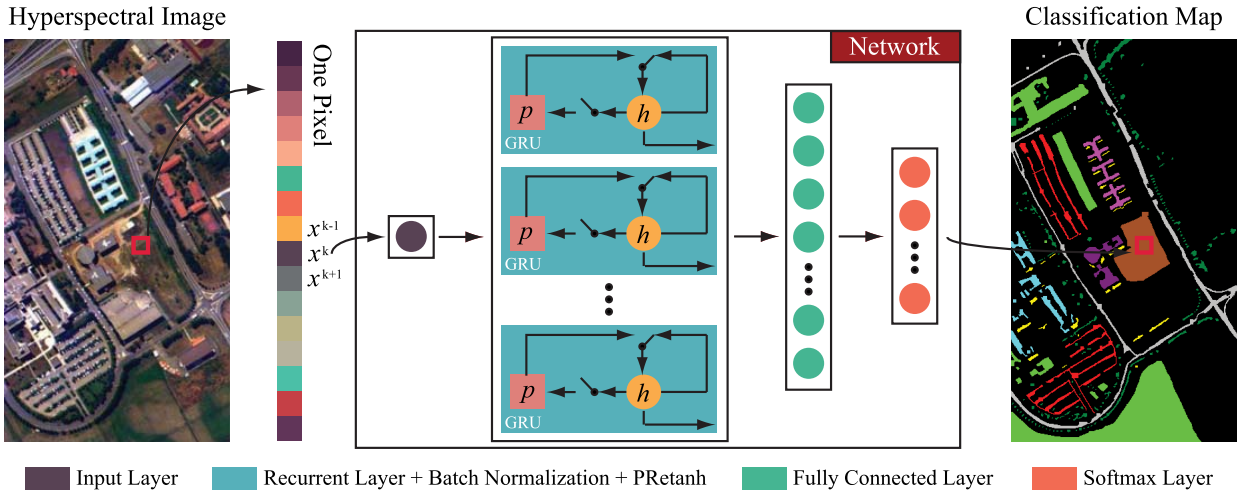


Fig. 2. Overview of our pipeline. First, the value of existing spectral band x^k is fed into the input layer. Then, the recurrent layer receives x^k and calculates the hidden state information for the current band; it also restores that information in the meantime. Next, the value of the next band x^{k+1} is input to the recurrent layer simultaneously with the state information of x^k, and the activation at spectral band $k+1$ is computed by a linear interpolation between proposal activation and the activation of previous band k. Finally, the RNN can predict the label of the input hyperspectral pixel by looping through the entire hyperspectral pixel sequence.

ReLU is used as an activation functions for the network output, given that gradients often need to be truncated often (and ReLU cannot dampen them like the bounded activation functions, such as tanh).

3) Traditional bounded activation functions, such as sigmoid and tanh, are always likely to generate some nonzero values, resulting in dense representations, while sparse representations seem to be better than dense representations in terms of representation learning.

Thus, to train a valid recurrent network for the hyperspectral image classification, we designed the new activation function PRetanh, which has two major advantages: 1) producing a bounded output and 2) promoting sparsity adaptively.

Definition: In this section, we introduce a newly defined activation function—PRetanh. It is defined as

$$f(h_i) = \begin{cases} \tanh(h_i), & \text{if } h_i > 0 \\ \lambda_i \tanh(h_i), & \text{if } h_i \leq 0 \end{cases} \tag{11}$$

where h_i is the input of the nonlinear activation f on the ith channel and $0 \leq \lambda_i \leq 1$ is a coefficient that can control the range of the negative part. The subscript i means that PRetanh can be varied in different channels. When $\lambda_i = 0$, it turns to

$$f(h_i) = \max(0, f(h_i)) = \max(0, \tanh(h_i)). \tag{12}$$

When λ_i is a learnable parameter, we refer to (11) as a parametric rectified hyperbolic tangent function. Fig. 3 shows the shapes of tanh and PRetanh. Equation (11) is equivalent to

$$f(h_i) = \max(0, \tanh(h_i)) + \lambda_i \min(0, \tanh(h_i)). \tag{13}$$

In our method, the PRetanh parameter λ_i is adaptively learned jointly with the whole neural network model. We expect that end-to-end training can lead to more specialized activations. Note that extra parameters are introduced in PRetanh. The total number of extra parameters for each layer is equal to the number of channels, which is negligible when taking into account the number of weights of the whole network. Therefore, we anticipate no extra risk of overfitting with the same number of training samples. In addition, a channel-shared variant version of PRetanh can be considered

$$f(h_i) = \max(0, \tanh(h_i)) + \lambda \min(0, \tanh(h_i)) \tag{14}$$

where all channels of one layer share the same coefficient λ. In this case, only a single extra parameter is introduced for each layer.

Optimization: With respect to the training of PRetanh, we use the backpropagation algorithm and simultaneously optimize the parameters of PRetanh with the neural networks. Suppose we have an objective function L that we wish to minimize, and the update rule of parameter λ_i is derived by the chain rule

$$\frac{\partial L}{\partial \lambda_i} = \sum_{h_i} \frac{\partial L}{\partial f(h_i)} \frac{\partial f(h_i)}{\partial \lambda_i}. \tag{15}$$

The term $(\partial L)/(\partial f(h_i))$ is the gradient backpropagated from the deeper layer of PRetanh. The summation $\sum_{h_i}$ is applied in all positions of the feature maps. Specifically, the gradient of activation is given by

$$\frac{\partial f(h_i)}{\partial \lambda_i} = \begin{cases} 0, & \text{if } h_i > 0 \\ \tanh(h_i), & \text{if } h_i \leq 0. \end{cases} \tag{16}$$

Equation (16) can be rewritten as follows:

$$\frac{\partial f(h_i)}{\partial \lambda_i} = \min(0, \tanh(h_i)). \tag{17}$$

Moreover, for the channel-shared variant version, the gradient of λ is as follows:

$$\frac{\partial L}{\partial \lambda} = \sum_i \sum_{h_i} \frac{\partial L}{\partial f(h_i)} \frac{\partial f(h_i)}{\partial \lambda} \tag{18}$$

where $\sum_i$ sums over all channels of the layer.

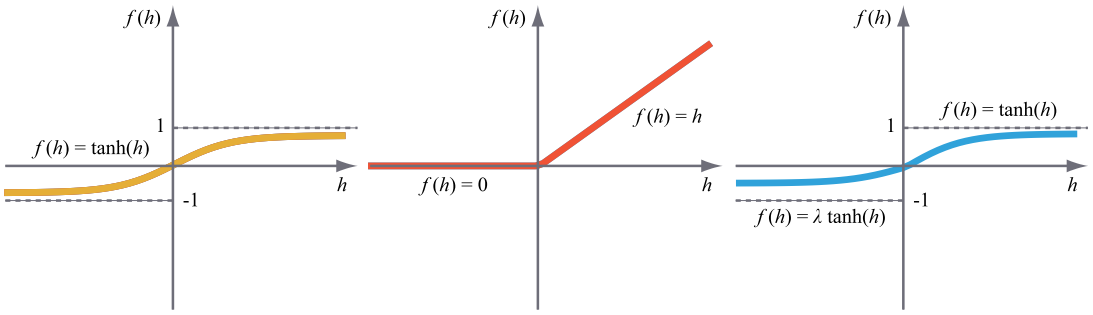


Fig. 3. (From left to right) tanh, ReLU, and PRetanh. For PRetanh, the coefficient λ of the negative part is not constant and is adaptively learned.

The momentum method is commonly used to help accelerate stochastic gradient descent in the relevant direction and dampens oscillations by adding a fraction γ of the update parameter. Here, we adopt the momentum method when updating parameter λ_i. The updating rule is

$$\Delta\lambda_i := \gamma\,\Delta\lambda_i + \eta\frac{\partial L}{\partial\lambda_i} \tag{19}$$

where η is the learning rate and γ is the momentum. Note that we do not use weight decay, i.e., ℓ_2 regularization, for updating λ_i, since a weight decay term tends to push the rectified parameters λ_i to zero.

Analysis: The two major advantages lie in obtaining adaptively sparse and bounded output. Sparsity arises when $\lambda_i = 0$ and $h_i \leq 0$. The more such units exist in a layer, the more sparse the resulting representations will be. The traditional tanh, in contrast, is always likely to generate some nonzero values, resulting in dense representations, while sparse representations seem to be better than dense representations. In addition, unlike the popular ReLU [25], which restricts the form of the negative part, we do not apply any constraints or regularization to it. As a result, the parameter λ_i that controls sparsity can be learned freely as the network trains. The other merit of PRetanh, the bounded output, is important from a practical perspective, because it means that the activations of the recurrent network will not blow up. The bounded output can reduce the probability of change in the distribution of internal nodes of deep networks to some extent, which allows fairly high learning rates to be used without the risk of divergence. In Section IV, it will be demonstrated that, compared with ReLU, using PRetanh as the activation function can effectively overcome the divergence of the recurrent network for hyperspectral image classification in the course of training.

Nevertheless, since PRetanh is affected by tanh, it likely moves many inputs into the saturated regime of the nonlinearity, and slows down the convergence. This effect is amplified as the recurrent network depth increases. In practice, the saturation problem and the resulting vanishing gradients are usually addressed by a carefully chosen initialization and the use of small learning rates. If, however, the distribution of inputs could be ensured to remain more stable as the training goes on, the optimizer would be less likely to fall into the saturation regime. In this paper, we combine a batch normalization technique with PRetanh to avoid the vanishing gradient problem.

For a layer with D dimensional output $\mathbf{h} = (h_1, h_2, \ldots, h_D)$, we normalize the ith channel as follows:

$$\hat{h}_i = \frac{h_i - \mathrm{E}[h_i]}{\sqrt{\mathrm{Var}[h_i]}} \tag{20}$$

where $\mathrm{E}[h_i] = \frac{1}{N}\sum_{j=1}^{N} h_i^{(j)}$ is the expectation, $\mathcal{H} = \{h_i^{(1)\ldots(N)}\}$ represents the set of values of h_i over a mini training batch, and $\mathrm{Var}[h_i]$ is the variance. We also need to scale and shift the normalized values; otherwise, just normalizing a layer would limit the layer in terms of what it can represent. Therefore, the normalized input h_i is transformed into

$$g(h_i) = \alpha_i\hat{h}_i + \beta_i \tag{21}$$

where α and β are parameters learned along with the original network parameters. Finally, batch normalization makes it possible to use PRetanh nonlinearities by preventing the network from getting stuck in the saturated modes.

B. Recurrent Unit

Recently, more and more empirical results have demonstrated that RNNs are not just powerful in theory [42], [44], [45] but can also be reliably learned in practice for processing long-term sequential data [36], [46], [47]. One interesting observation is that a few of these successes were obtained with the traditional RNN model. Rather, they used an RNN with sophisticated recurrent hidden units like LSTM, because such structures are capable of alleviating the vanishing gradient problem. However, available training samples for remote sensing image classification are often limited, forcing researchers to control the total number of trainable parameters of the network as much as possible. We, therefore, design a novel GRU with PReLU, which is able to deal with long-term sequential data like hyperspectral sequences and is more suitable for a small number of training samples, since it has fewer parameters than LSTM.

1) LSTM for Hyperspectral Image Classification: For a hyperspectral image classification task, given a hyperspectral pixel sequence $\mathbf{x} = (x^1, x^2, \ldots, x^K)$, a traditional RNN framework [34] calculates the hidden vector sequence $\mathbf{h} = (\mathbf{h}^1, \mathbf{h}^2, \ldots, \mathbf{h}^K)$ by iterating the following equation from $k = 1$ to K:

$$\mathbf{h}^k = \varphi(\mathbf{w}_{ih}x^k + \mathbf{W}_{hh}\mathbf{h}^{k-1} + \mathbf{b}_h) \tag{22}$$

where $\mathbf{w}_{ih}$ denotes the input-hidden weight vector, $\mathbf{W}_{hh}$ represents the context weight matrix of the hidden layer, $\mathbf{b}_h$ is the hidden bias vector, and $\varphi(\cdot)$ is the hidden layer activation function. Finally, the predicted label $\mathbf{y}$ can be computed as follows:

$$\mathbf{y} = \mathbf{W}_{oh}\mathbf{h}^K + \mathbf{b}_o \tag{23}$$

where $\mathbf{W}_{oh}$ is the output-hidden weight matrix and $\mathbf{b}_o$ is the bias vector of output layer.

In this paper, we want to use an RNN to characterize the spectral correlation and band-to-band variability when mapping between input pixel sequences and output labels. Unfortunately, for standard recurrent network architecture, the range of spectral contexts that can be accessed in practice is quite limited. The problem is that the influence of a given input on the hidden layer and, therefore, on the network output either decays or blows up exponentially as it cycles around the recurrent connections of the network. This effect is a common challenge in designing and training deep RNNs and is known as the vanishing gradient problem [48].

To process long-term sequences, which is crucial to the task, as hyperspectral imagery usually includes hundreds of spectral bands, LSTMs were proposed to address the vanishing gradient problem. LSTMs [42] introduce the gate concept and memory cell to help preserve the error that can be backpropagated through steps and layers. By maintaining a more constant error, they allow recurrent networks to continue to learn over many steps (over 1000) and thereby enable us to utilize a large range of spectral contexts, e.g., to link the first and last spectral bands remotely.

2) Gated Recurrent Unit With PReLU: However, LSTMs lead to more parameters, which need to be learned. And, as discussed earlier, the limited number of training samples drives a need to restrict the number of parameters, to avoid overfitting.

Therefore, a deep RNN with modified GRUs tailored for hyperspectral sequence analysis is proposed for hyperspectral image classification. GRUs [44], [45] have fewer parameters than LSTMs, and can also effectively process a long-term spectral sequence. Moreover, PReLU is introduced to our modified GRUs, allowing us to use fairly high learning rates without the risk of divergence.

A GRU can cause a recurrent unit to adaptively capture the dependences of different spectral bands. Similar to the LSTM unit, the GRU has gate units that control the flow of information inside the unit without including separate memory cells.

The activation h_i^k of the ith GRU at spectral band k is computed by a linear interpolation between the proposal activation p_i^k and the activation of the previous spectral band h_i^{k-1}

$$h_i^k = u_i^k p_i^k + \left(1 - u_i^k\right) h_i^{k-1} \tag{24}$$

where u_i^k is an update gate that determines how much the unit updates its activation or content. The update gate u_i^k can be calculated as follows:

$$u_i^k = \sigma\left(\mathbf{w}_{ui}x^k + \mathbf{W}_{uh}\mathbf{h}^{k-1}\right)_i \tag{25}$$

where $\mathbf{w}_{ui}$ is the input-update weight vector and $\mathbf{W}_{uh}$ represents the update-hidden weight matrix.

Similarly to LSTM, the GRU takes a linear sum between the newly computed state and the present state. However, it lacks a mechanism to control what part of the state information will be exposed, rather exposing the whole state value at each spectral band.

The proposal activation p_i^k is computed using the value of the existing spectral band and the activation of the previous band, which reflects the updated information of the recurrent hidden state. It is calculated with PRetanh and batch normalization as follows:

$$p_i^k = f\left(g\left(\mathbf{w}_{pi}x^k + \mathbf{W}_{rh}\left(\mathbf{r}^k \odot \mathbf{h}^{k-1}\right)\right)\right)_i \tag{26}$$

where $\mathbf{r}^k$ is a set of reset gates, $\mathbf{w}_{pi}$ denotes the proposal-input weight vector, and $\mathbf{W}_{rh}$ represents the reset-hidden weight matrix. Moreover, $f(\cdot)$ and $g(\cdot)$ represent PRetanh and batch normalization, respectively. When the reset gate r_i^k is fully OFF, i.e., r_i^k is 0, it will completely discard the activation of the hidden layer at previous spectral bands h_i^{k-1} and only use the value of the existing spectral band x^k. When open (r_i^k close to 1), in contrast, the reset gate will partially keep the information of the previous step.

Let $\hat{p}_i^k = \mathbf{w}_{pi}x^k + \mathbf{W}_{rh}(\mathbf{r}^k \odot \mathbf{h}^{k-1})$. Equation (26) can then be transformed as

$$p_i^k = \max\left(0, \tanh\left(\alpha_i \frac{\hat{p}_i^k - \mathrm{E}\left[\hat{p}_i^k\right]}{\sqrt{\mathrm{Var}\left[\hat{p}_i^k\right]}} + \beta_i\right)\right) + \lambda_i \min\left(0, \tanh\left(\alpha_i \frac{\hat{p}_i^k - \mathrm{E}\left[\hat{p}_i^k\right]}{\sqrt{\mathrm{Var}\left[\hat{p}_i^k\right]}} + \beta_i\right)\right). \tag{27}$$

The reset gate r_i^k is computed similar to the update gate

$$r_i^k = \sigma\left(\mathbf{w}_{ri}x^k + \mathbf{W}_{rh}\mathbf{h}^{k-1}\right)_i \tag{28}$$

where $\mathbf{w}_{ri}$ and $\mathbf{W}_{rh}$ are the reset-input weight vector and the reset-hidden weight matrix, respectively.

Fig. 4 shows the graphic model of the GRU through time.

IV. Experimental Results and Discussion

A. Data Description

1) Pavia University: This data set is acquired by reflective optics system imaging spectrometer (ROSIS). The image is of 610 × 340 pixels covering the Engineering School at the University of Pavia, which was collected under the HySens project managed by the German Aerospace Agency (DLR). The ROSIS-03 sensor comprises 115 spectral channels ranging from 430 to 860 nm. In this data set, 12 noisy channels have been removed and the remaining 103 spectral channels are investigated in this paper. The spatial resolution is 1.3 m per pixel. The available training samples of this data set cover nine classes of interests. Table I provides information about different classes and their corresponding training and test samples.

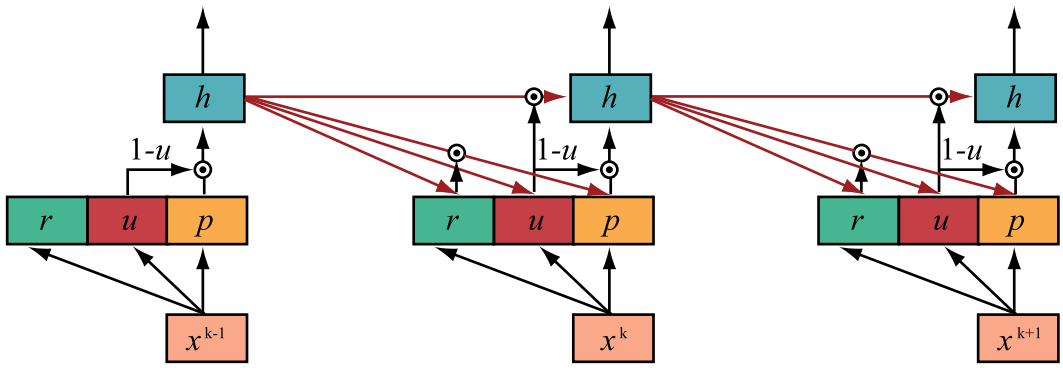


Fig. 4. Graphic model of a GRU through time. The reset and update gates are denoted by r and u, respectively, and p and h are the proposal activation and the final activation.

TABLE I
NUMBER OF TRAINING AND TEST SAMPLES USED IN THE PAVIA UNIVERSITY DATA SET

Class No.	Class Name	Training	Test
1	Asphalt	548	6631
2	Meadows	540	18649
3	Gravel	392	2099
4	Trees	524	3064
5	Metal sheets	265	1345
6	Bare Soil	532	5029
7	Bitumen	375	1330
8	Bricks	514	3682
9	Shadows	231	947
	TOTAL	3921	42776

TABLE II
NUMBER OF TRAINING AND TEST SAMPLES USED IN THE HOUSTON DATA SET

Class No.	Class Name	Training	Test
1	Grass Healthy	198	1053
2	Grass Stressed	190	1064
3	Grass Synthetic	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
12	Parking Lot 1	192	1041
13	Parking Lot 2	184	285
14	Tennis Court	181	247
15	Running Track	187	473
	TOTAL	2832	12197

2) Houston Data: The second data set was acquired over the University of Houston campus and its neighboring urban area. It was collected with an ITRES-CASI 1500 sensor on June 23, 2012 between 17:37:10 and 17:39:50 UTC. The average altitude of the sensor was about 1676 m, which results in 2.5-m spatial resolution data consisting of 349 by 1905 pixels. The hyperspectral imagery consists of 144 spectral bands ranging from 380 to 1050 nm and was processed (radiometric correction, attitude processing, GPS processing, geocorrection, and so on) to yield the final geocorrected image cube representing the sensor spectral radiance. Table II provides information about all 15 classes of this data set with their corresponding training and test samples.

TABLE III
NUMBER OF TRAINING AND TEST SAMPLES USED IN THE INDIAN PINES DATA SET

Class No.	Class Name	Training	Test
1	Alfalfa	50	1384
2	Corn-notill	50	784
3	Corn-min	50	184
4	Corn	50	447
5	Grass-pasture	50	697
6	Grass-trees	50	439
7	Grass-pasture-mowed	50	918
8	Hay-windrowed	50	2418
9	Oats	50	564
10	Soybean-notill	50	162
11	Soybean-mintill	50	1244
12	Soybean-clean	50	330
13	Wheat	50	45
14	Woods	15	39
15	Buildings-grass-trees	15	11
16	Stone-steel-towers	15	5
	TOTAL	695	9671

3) Indian Pines Data: The third data set was gathered by an airborne visible/infrared imaging spectrometer sensor over the Indian Pines agricultural site in Northwestern Indiana in June 1992, and presents 16 classes, mostly related to land covers. The data set consists of 145 by 145 pixels with a spatial resolution of 20 m per pixel and 10-nm spectral resolution over the range of 400–2500 nm. In this paper, we made use of 200 bands, after removing 20 bands affected by atmosphere absorption. The number of training and test samples is displayed in Table III.

B. General Information

To evaluate the performance of different models for hyperspectral image classification, we use the following evaluation criteria.

1) *Overall Accuracy (OA):* This index shows the number of hyperspectral pixels that are classified correctly, divided by the number of test samples.
2) *Average Accuracy (AA):* This measure is the average value of the classification accuracies of all classes.

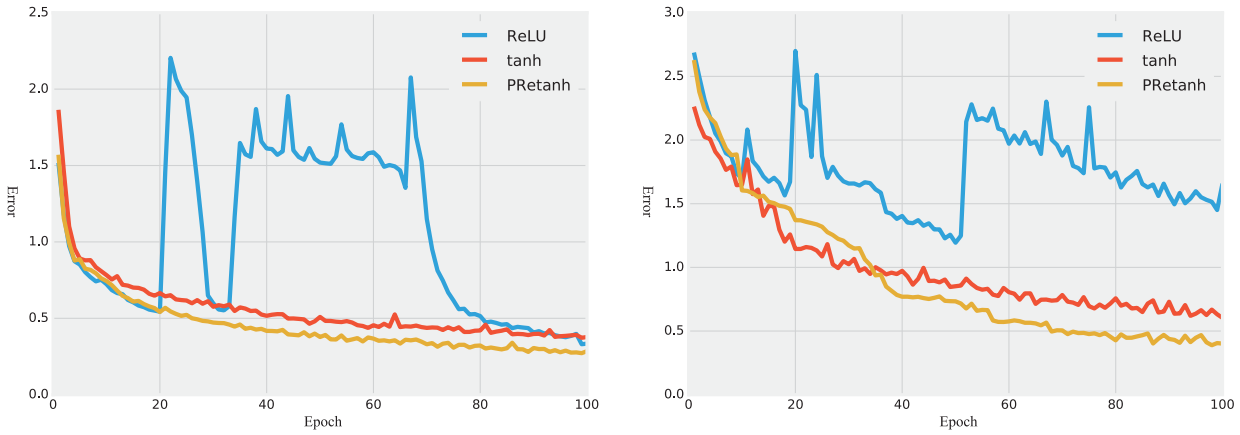


Fig. 5. Learning curves for a recurrent network with ReLU, one with tanh and one with the proposed PRetanh on the training samples of (a) Pavia University data set and (b) Houston data set. As shown in these figures, with PRetanh, we can make use of a fairly high learning rate, e.g., 1.0 instead of a relatively low default 0.002, to train the recurrent network for hyperspectral image classification without the risk of divergence. Meanwhile, it can be seen that the ReLU can cause the recurrent network to diverge when a fairly high learning rate is used. Here, we use the Adadelta optimization algorithm.

TABLE IV
CLASSIFICATION ACCURACIES OF DIFFERENT TECHNIQUES IN PERCENTAGES FOR PAVIA UNIVERSITY. THE BEST ACCURACY IN EACH ROW IS SHOWN IN BOLD

Class No.	Class Name	RF-200	SVM-RBF	CNN	RNN-LSTM	RNN-GRU-tanh	RNN-GRU-PRetanh
1	Asphalt	80.85	80.80	83.73	77.45	78.42	**84.45**
2	Meadows	55.29	66.78	65.70	61.83	69.17	**85.24**
3	Gravel	52.93	**73.18**	67.03	64.60	47.83	54.31
4	Trees	**98.79**	95.17	94.03	97.98	97.16	95.17
5	Metal Sheets	99.26	99.55	99.41	99.18	97.84	**99.93**
6	Bare Soil	78.76	92.90	**96.30**	91.19	85.86	80.99
7	Bitumen	84.36	90.08	**93.83**	90.90	86.84	88.35
8	Bricks	91.58	91.20	93.56	92.29	**94.27**	88.62
9	Shadows	98.20	93.77	99.79	97.47	94.93	**99.89**
OA	-	71.37	78.82	80.51	77.99	80.70	**88.85**
AA	-	82.23	87.05	**88.15**	85.88	83.59	86.33
Kappa	-	0.6484	0.7358	0.7423	0.7028	0.7201	**0.8048**

3) *Kappa Coefficient:* This metric is a statistical measurement of agreement between the final classification map and the ground-truth map. It is the percentage agreement corrected by the level of agreement that could be expected due to chance alone. It is generally thought to be a more robust measure than a simple percent agreement calculation, since k takes into account the agreement occurring by chance [1].

If the number of samples for each category is identical, OA and AA are equal. However, the category distribution suffers from an imbalanced phenomenon in practice. Adopting OA alone is not precise, since rare categories are commonly ignored. Therefore, AA is also utilized to evaluate the performance of different classification models. Strong differences between the OA and AA may indicate that a specific class is incorrectly classified with a high proportion.

To validate the effectiveness of the proposed RNN-based classification framework, it is compared with the most widely used vector-based classification models, SVM and random forest. The SVM with an RBF kernel was implemented using the libsvm package.[2] Fivefold cross-validation is taken into account to tune the hyperplane parameters. Furthermore, in this paper, experiments using other popular activation functions (i.e., tanh and ReLU) and recurrent units (i.e., LSTM) are also carried out to verify the validity of the proposed network. To conduct a fair comparison, PRetanh/tanh/ReLU models are trained using the same total number of epochs, and the same network architecture is adopted. The learning rates are also switched after running the same number of epochs. The methods included in the comparison are summarized as follows.

1) *RF-200:* Random forest with 200 trees.
2) *SVM-RBF:* RBF kernel SVM with cross validation.
3) *CNN:* The architecture of the CNN is set as in [30], and contains an input layer, a convolution layer, a max-pooling layer, a fully connected layer, and an output layer. The number of convolutional kernels is 20 for all three data sets. The length of each convolution kernel and pooling size is 11 and 3, respectively. Furthermore, 100 hidden units are included in the fully connected layer.
4) *RNN-LSTM:* RNN with LSTM recurrent units. We follow the implementation of LSTM as used in [42].

[2]https://www.csie.ntu.edu.tw/~cjlin/libsvm/

TABLE V
CLASSIFICATION ACCURACIES OF TEST SAMPLES ON THE HOUSTON DATA SET. THE BEST ACCURACY IN EACH ROW IS SHOWN IN BOLD

Class No.	Class Name	RF-200	SVM-RBF	CNN	RNN-LSTM	RNN-GRU-tahn	RNN-GRU-PRetanh
1	Grass Healthy	**82.62**	81.96	81.20	81.86	82.43	82.53
2	Grass Stressed	83.36	80.55	**83.55**	81.20	82.42	83.36
3	Grass Synthetic	98.02	99.80	99.41	99.41	97.23	**100**
4	Tree	91.76	**92.23**	91.57	90.06	89.30	90.53
5	Soil	97.06	97.63	94.79	93.09	78.22	**97.82**
6	Water	**99.30**	95.10	95.10	96.50	95.10	93.01
7	Residential	75.37	**76.59**	63.53	73.41	70.43	75.37
8	Commercial	32.95	35.52	**42.64**	34.09	32.57	42.36
9	Road	67.14	70.44	58.17	62.61	70.25	**77.62**
10	Highway	43.73	**60.04**	41.80	39.96	43.24	57.63
11	Railway	70.11	76.57	75.71	60.44	69.07	**77.42**
12	Parking Lot 1	54.95	73.10	**84.15**	65.42	50.72	69.74
13	Parking Lot 2	59.65	**68.77**	40.00	58.95	58.25	66.32
14	Tennis Court	99.19	**100**	98.79	96.76	97.98	**100**
15	Running Track	97.67	**98.10**	97.89	88.37	96.83	95.98
OA	-	72.93	77.09	85.42	85.41	85.73	**89.85**
AA	-	76.86	80.43	76.55	74.81	74.27	**80.65**
Kappa	-	0.7091	0.7536	0.7200	0.6889	0.6785	**0.7606**

TABLE VI
ACCURACY COMPARISON FOR THE INDIAN PINES DATA SET. THE BEST ACCURACY IN EACH ROW IS SHOWN IN BOLD

Class No.	Class Name	RF-200	SVM-RBF	CNN	RNN-LSTM	RNN-GRU-tahn	RNN-GRU-PRetanh
1	Alfalfa	54.84	60.77	56.79	46.03	68.93	**70.59**
2	Corn-notill	58.42	**77.68**	52.17	61.73	40.94	70.28
3	Corn-min	82.61	79.35	85.33	**86.96**	78.80	81.52
4	Corn	85.91	**91.05**	87.92	87.02	87.92	90.16
5	Grass-pasture	80.49	84.36	85.22	86.66	87.52	**91.97**
6	Grass-trees	94.76	92.03	**97.49**	**97.49**	97.27	96.13
7	Grass-pasture-mowed	77.34	69.61	74.62	59.69	82.79	**84.75**
8	Hay-windrowed	59.43	59.31	**67.99**	64.89	50.58	59.64
9	Oats	63.48	79.61	58.87	60.46	79.43	**86.17**
10	Soybean-notill	95.06	97.53	98.77	98.77	98.77	**99.38**
11	Soybean-mintill	**88.26**	85.21	87.62	75.32	84.73	84.97
12	Soybean-clean	54.85	63.64	72.42	71.82	61.21	**77.58**
13	Wheat	97.78	**100**	93.33	91.11	88.89	95.56
14	Woods	58.97	**87.18**	71.79	79.49	79.49	84.62
15	Buildings-grass-trees	81.82	**90.91**	**90.91**	**90.91**	**90.91**	**90.91**
16	Stone-steel-towers	**100**	**100**	**100**	**100**	**100**	**100**
OA	-	69.79	72.78	84.18	80.52	85.71	**88.63**
AA	-	77.13	82.39	80.08	78.65	79.89	**85.26**
Kappa	-	0.6589	0.6931	0.6852	0.6372	0.6633	**0.7366**

5) *RNN-GRU-tanh:* RNN with GRUs that use tanh as the activation function.
6) *RNN-GRU-ReLU:* ReLU is adopted to activate the output of recurrent units.
7) *RNN-GRU-PRetanh:* Our final network uses the proposed PRetanh activation function for the hidden representation of GRUs.

To make the proposed approach fully comparable with other supervised classifiers, we used the standard sets of training and test samples for the data sets. For instance, we used the training and test sets of the 2013 GRSS Fusion Contest for the classification of the Houston data.

The RNN was trained with the Adadelta algorithm, and all the suggested default parameters except the learning rate were used for all of the following experiments. We made use of a fairly high learning rate of 1.0 instead of the relatively low default of 0.002 to train the network. The proposed network model uses a single recurrent layer that adopts our modified GRUs of size 64 with sigmoid gate activation and PRetanh activation functions for hidden representations. The output layer uses softmax activation and then outputs a one-hot vector for hyperspectral image classification. All weight matrices in our RNN and bias vectors are initialized with a uniform distribution, and the values of these weight matrices and bias vectors are initialized in the range $[-0.1,0.1]$. Then, all the weights can be updated during the training procedure. In both hyperspectral data sets, we randomly chose 10% of the training samples as the validation set. That is, during the training, we used 90% of the training samples to learn the parameters, including the weight matrices $\mathbf{W}$, bias vectors $\mathbf{b}$, the parameters α and β of batch normalization, and the coefficients λ of PRetanh, and used the remaining 10% of the training samples as validation to tune the superparameters, such as the number of recurrent units in the recurrent layer. All test samples were used to evaluate the final performance of the trained recurrent

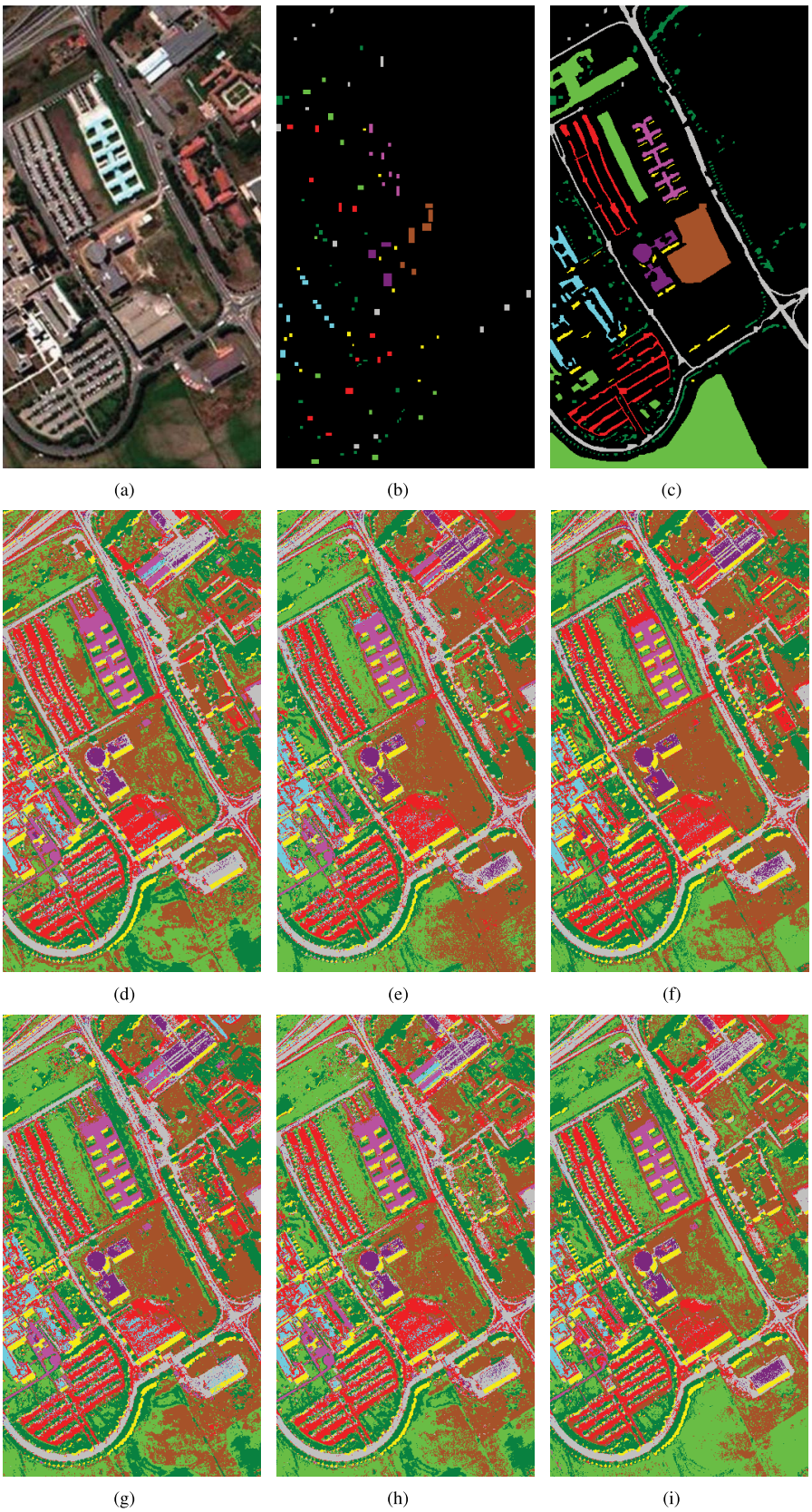


Fig. 6. Classification results obtained by different methods for the Pavia University scene. (a) Composite image of hyperspectral data. (b) Training data. (c) Ground-truth reference. (d) RF-200. (e) SVM-RBF. (f) CNN. (g) RNN-LSTM. (h) RNN-GRU-tanh. (i) RNN-GRU-PRetanh.

network. It is noteworthy that the Indian Pines data are a small and unbalanced data set, which is challenging for training a valid supervised recurrent network. To address this concern, we not only use a dropout with a probability of 0.5 on the output of recurrent layer but also utilize a dropout of 0.2 on the weight matrices of the network, which indicates the

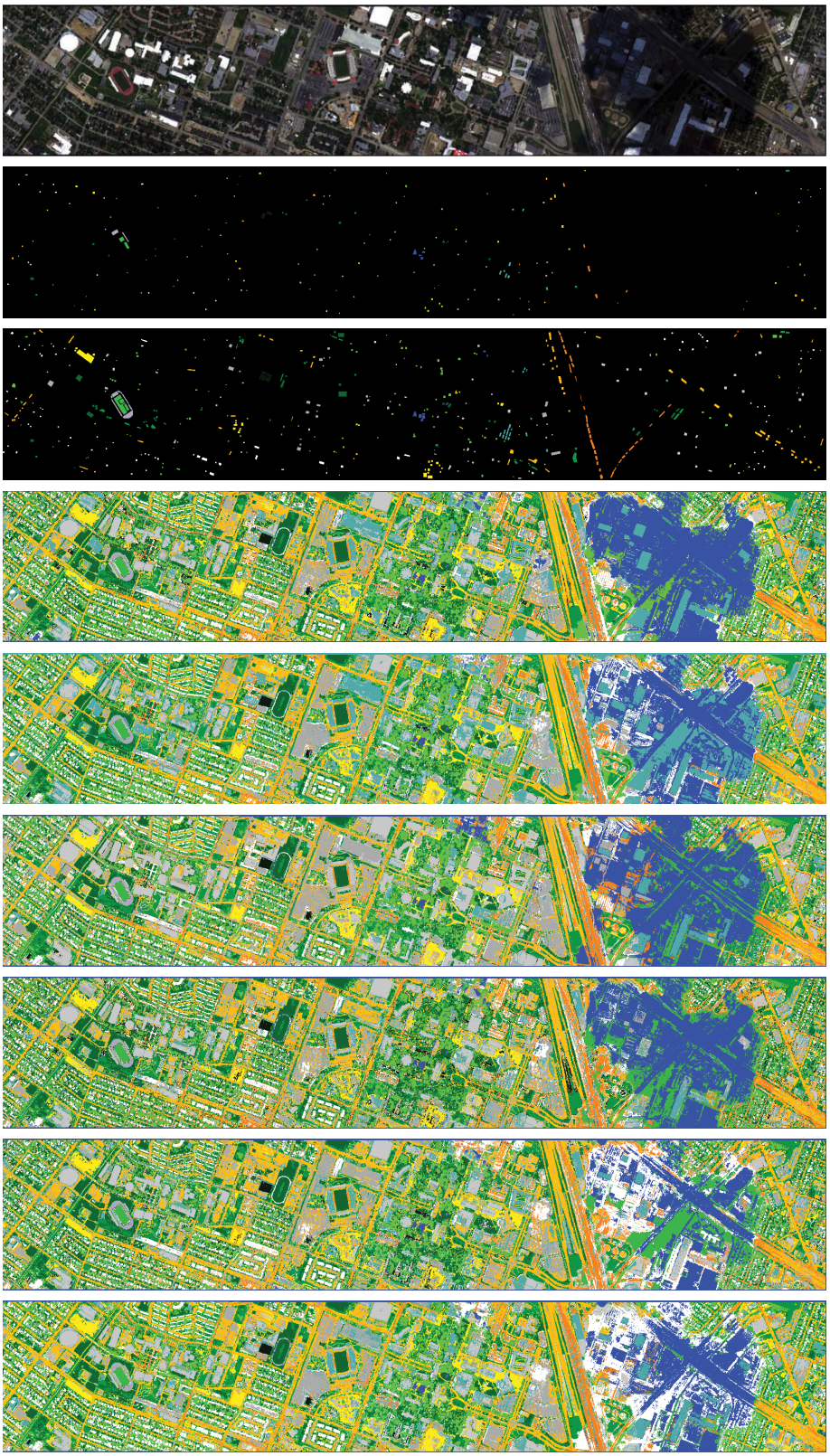


Fig. 7. Classification results obtained by different methods for the Houston scene. (From top to bottom) True-color composite of the hyperspectral data (wavelength R: 640.7 nm, G: 550.2 nm, and B: 459.6 nm), training data, ground-truth reference, RF-200, SVM-RBF, CNN, RNN-LSTM, RNN-GRU-tanh, and RNN-GRU-PRetanh.

fraction of the input units to drop for input gates and recurrent connections.

The experiments are organized into three parts. The first one aims primarily at analyzing the behavior of different activation functions, which involves tanh, ReLU, and the proposed PRetanh. The comparison of the LSTM unit and GRU is also discussed in this part. In the second experiment, the effectiveness of an RNN that is based on the sequential

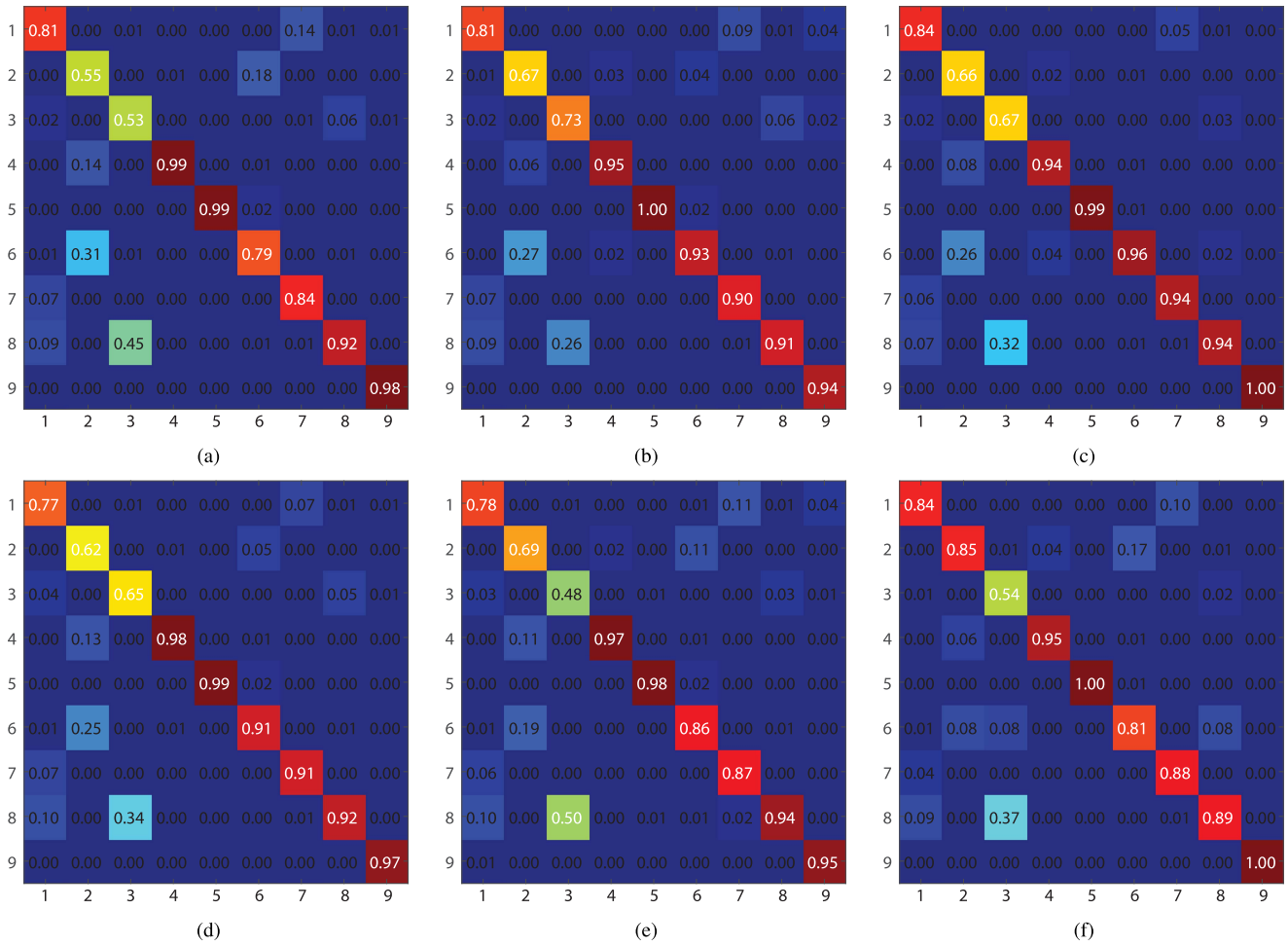


Fig. 8. Confusion matrix of different methods for the Pavia University data set. (a) RF-200. (b) SVM-RBF. (c) CNN. (d) RNN-LSTM. (e) RNN-GRU-tanh. (f) RNN-GRU-PRetanh.

perspective of a hyperspectral pixel is compared with the traditional vector-based models, such as random forest, SVM, and 1-D CNN. In the last part, we discuss processing time.

C. Analysis of the Network

1) Comparisons Between ReLU, tanh, and PRetanh: The activation function is a basic building block of a neural network, because it enables the network to detect nonlinear features in the data. Here, we investigate and compare the behaviors of three activation functions, ReLU, tanh, and PRetanh. Fig. 5 compares the convergence performance of RNN-GRU-ReLU, RNN-GRU-tanh, and RNN-GRU-PRetanh on both the Pavia University and Houston data. All activation functions can make the recurrent network converge except ReLU. Moreover, compared with tanh, the proposed PRetanh activation function starts reducing error earlier and finally reduces the loss to a lower value, which means that the network can converge to a better solution. In particular, PRetanh can obtain the error value of 0.272 on the Pavia University data set after 100 epochs, while the traditional tanh activation function can only achieve 0.334. As Fig. 5(a) shows, the RNN with ReLU as the activation function falls into divergence, which means that we cannot obtain a valid network. For the Houston data set, the recurrent network with the proposed PRetanh can quickly converge to the error of 0.401 after 100 iterations. In the same conditions, tanh can only yield 0.603. ReLU, however, cannot cause the recurrent network to converge. We also compare the classification accuracies of RNN-GRU-tanh and RNN-GRU-PRetanh. As shown in Tables IV–VI, compared with tanh, the network with the proposed PRetanh activation function increases the accuracy significantly by 8.15% of OA, 2.74% of AA, and 0.0847 of the Kappa coefficient, respectively, on the Pavia University data set. For the Houston data set, the accuracy increments on OA, AA, and Kappa coefficient are 4.12%, 6.38%, and 0.0821, respectively. On the Indian Pines data set, our network is able to achieve the accuracy increments of 2.92%, 5.37%, and 0.0733 for OA, AA, and the Kappa coefficient, respectively.

2) Comparison of Recurrent Unit Architecture: The most prominent trait shared between LSTM and GRU is that there is an additive loop of their update from $k-1$ to k, which is lacking in the conventional feedforward neural networks, such as CNNs. In contrast, compared with the traditional recurrent unit like (2), both LSTM and GRU keep the existing content and add the new content on top of it [see (7) and (24)]. These two units, however, have a number of differences as well. LSTM uses three gates and a cell, an input gate, an output

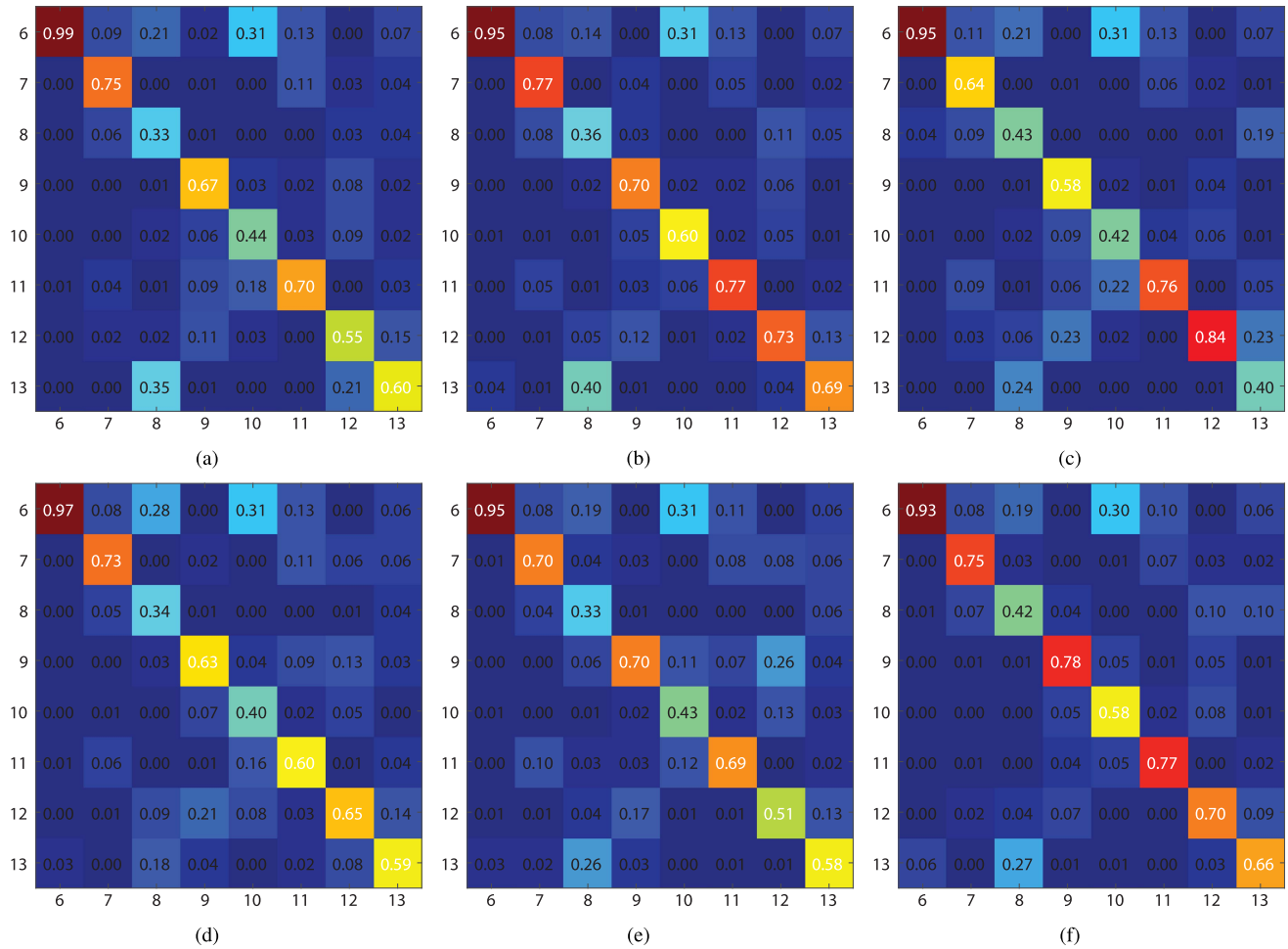


Fig. 9. Zoomed-in view confusion matrix of different methods for the Houston data set. (a) RF-200. (b) SVM-RBF. (c) CNN. (d) RNN-LSTM. (e) RNN-GRU-tanh. (f) RNN-GRU-PRetanh. To show the result more clearly, we show only class #6 to class #13, which are easily misclassified in the Houston data set.

TABLE VII
NUMBER OF TOTAL TRAINABLE PARAMETERS IN DIFFERENT RECURRENT LAYERS

Data set	LSTM	GRU-tanh	GRU-PRetanh
Pavia University (64 units)	16.50 K	12.38 K	12.57 K
Houston (128 units)	65.00 K	48.75 K	49.13 K
Indian Pines (128 units)	79.27 K	63.02 K	63.15 K

gate, a forget gate, and a memory cell, to control the exposure of memory content, while the GRU only employs two gates to control the information flow. In this way, the total number of parameters in the GRU is reduced by about 25%, which makes it the recurrent unit of choice in the recurrent layer for the hyperspectral image classification task. Table VII shows the number of total trainable parameters in different recurrent layers.

Tables IV–VI list the results obtained by our experiments. For all three data sets, the RNN-GRU-PRetanh outperforms the LSTM-based network (RNN-LSTM) on all indexes. Specifically, the RNN-GRU-PRetanh increases the accuracy significantly by 10.86% of OA, 0.45% of AA, and 0.1020 of Kappa, respectively, on the Pavia University data set; by 4.44% of OA, 5.84% of AA, and 0.0717 of the Kappa coefficient, respectively, on the Houston data set; and by 8.11% of OA, 6.61% of AA, and 0.0994 of the Kappa coefficient, respectively, on the Indian Pines data set.

D. Vector-Based Methods Versus Our Recurrent Network

The classification maps of the Pavia University data set obtained by the conventional vector-based models and our network are shown in Fig. 6, and the corresponding accuracy indexes are presented in Table IV. An analysis of the classification accuracies indicates that the SVM with RBF kernel (SVM-RBF) outperforms the random forest model, mainly because the kernel SVM generally handles nonlinear inputs more efficiently than the random forest model. It can be seen that the proposed recurrent network RNN-GRU-PRetanh outperforms the SVM-RBF and CNN in terms of OA and the Kappa coefficient. Compared with SVM-RBF and CNN, the proposed RNN-GRU-PRetanh increases the OA by 10.03% and 8.34%, respectively. Moreover, the proposed network achieves the best accuracies on some specific classes of the Pavia University data, such as asphalt, meadows, metal sheets, and shadows. For instance, the accuracy of the meadows category obtained by RNN-GRU-PRetanh reaches 85.24%,

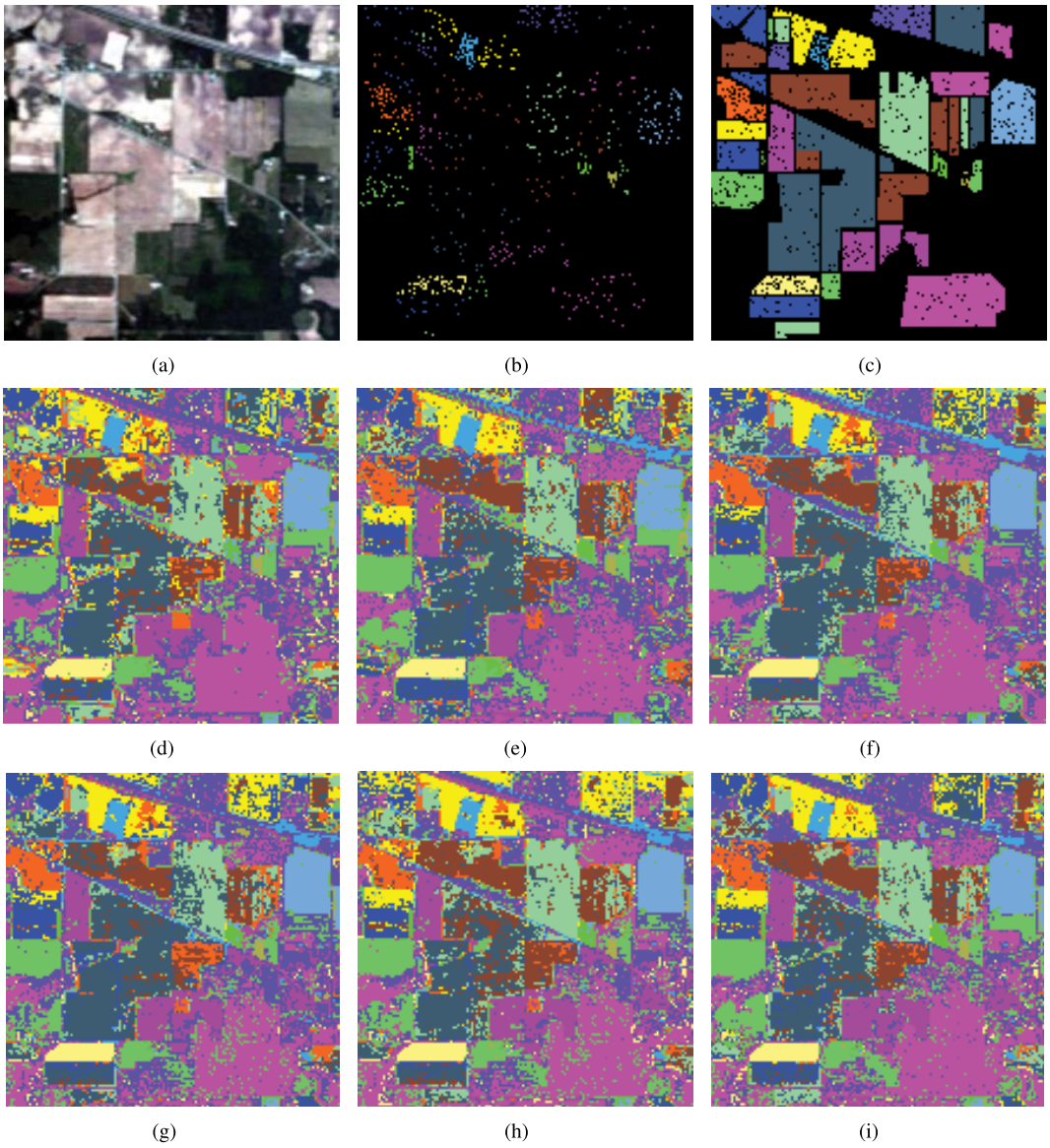


Fig. 10. Classification results obtained by different methods for the Indian Pines scene. (a) True-color composite (bands R: 26, G: 14, and B: 8). (b) Training data. (c) Ground-truth reference. (d) RF-200. (e) SVM-RBF. (f) CNN. (g) RNN-LSTM. (h) RNN-GRU-tanh. (i) RNN-GRU-PRetanh.

and the proposed network can achieve almost 100% on the shadows class.

Fig. 7 shows the classification maps on the Houston data set; the comparison of accuracies between the random forest, SVM-RBF, and RNN-GRU-PRetanh can be found in Table V. The proposed RNN-GRU-PRetanh achieves significantly better scores for OA, AA, and the Kappa coefficient compared with all other methods. Misclassification in this data set lies in similar objects, such as Road-Highway-Railway and Grass Healthy-Grass Stressed-Grass Synthetic. The proposed RNN-GRU-PRetanh achieves the best AA of 70.89% on Road-Highway-Railway, as well as the best AA of 88.63% on Grass Healthy-Grass Stressed-Grass Synthetic. Confusion matrices for the Pavia University data set and the Houston data set can be found in Figs. 8 and 9, respectively. Note that, for the Houston data set, because of the relatively large number of classes, only selected materials that have high misclassification rates are illustrated. In general, the proposed RNN-GRU-PRetanh also tends to show superior performance in distinguishing similar materials.

The classification maps and accuracy assessment for the Indian Pines data set are shown in Fig. 10 and Table VI. It can be seen that the proposed RNN-GRU-PRetanh yields substantially more accurate results than the other methods. Specifically, compared with SVM-RBF and CNN, the improvements in OA achieved by the proposed recurrent network are 15.85% and 4.45%, respectively, and the increments of AA obtained by RNN-GRU-PRetanh are 2.87% and 5.18%, respectively. Fig. 11 also shows that SVM-RBF and CNN are not very effective for discrimination between similar classes such as Grass-Pasture and Grass-Pasture-Mowed because of their similar spectral reflectance. The classification

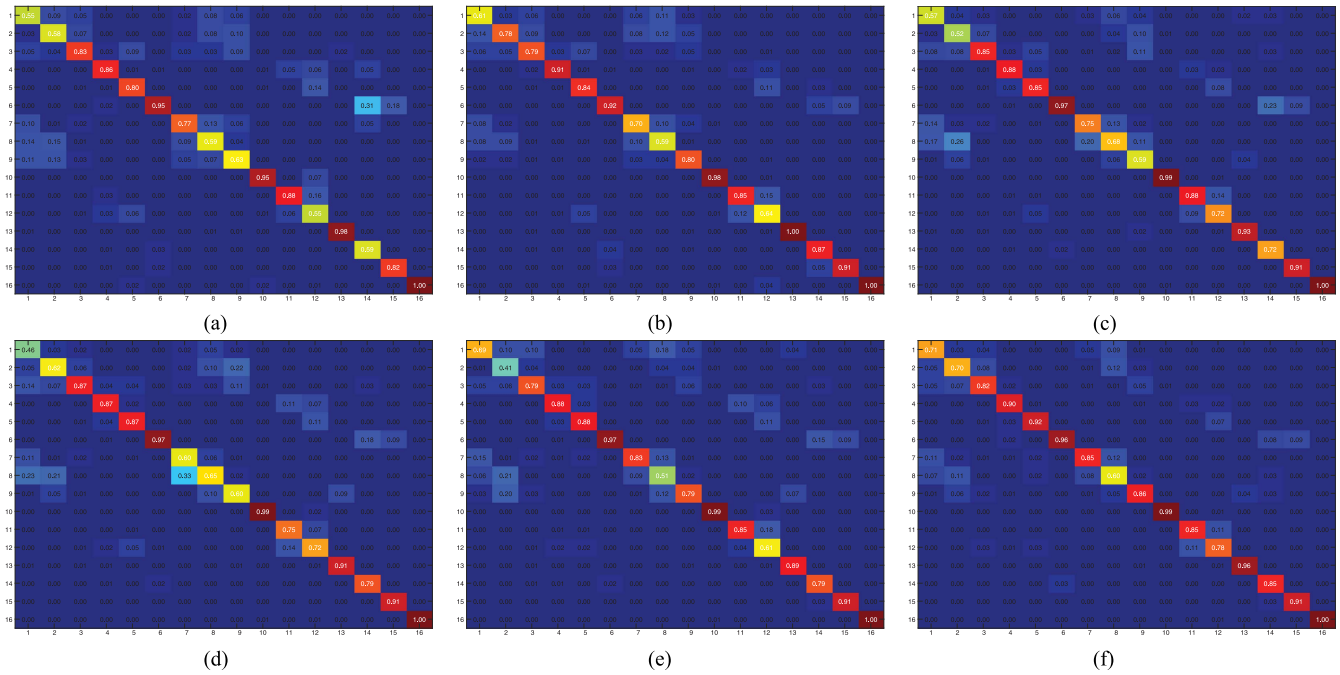


Fig. 11. Confusion matrix of different methods for the Indian Pines data set. (a) RF-200. (b) SVM-RBF. (c) CNN. (d) RNN-LSTM. (e) RNN-GRU-tanh. (f) RNN-GRU-PRetanh.

TABLE VIII
STATISTICS OF TRAINING TIME (min)

Data set	RF-200	SVM-RBF	CNN	Ours
Pavia University	1.2	17.0	33.3	77.4
Houston	0.9	15.6	39.3	88.8
Indian Pines	0.4	1.0	8.2	19.9

TABLE IX
STATISTICS OF TESTING EFFICIENCY (pixels/s)

Methods	RF-200	SVM-RBF	CNN	Ours
EFFICIENCY	2,042.43	2,034.96	9,427.27	8,396.76

of these similar land covers is improved with the proposed recurrent network.

E. Processing Time

Processing time of different methods is compared in this section. All the experiments are conducted on a personal computer equipped with an Intel Core I5 with 2.20 GHz. The training times of different approaches are shown in Table VIII. It is not surprising that deep neural network-based methods, including CNN and RNN, require a longer training time compared with other traditional vector-based classification models, such as random forest and SVM. Fortunately, such differences remain within one to two orders of magnitude. Between CNN and RNN, RNN requires more yet a tolerable training time, as it involves additional channel-by-channel updates. However, one advantage of deep neural networks is that they are fast in testing (see Table IX), which is very important in practice. Also, thanks to the rapid development of hardware technology, especially of GPU, deep neural networks' drawback of a long training time is becoming less and less decisive.

V. CONCLUSION

In this paper, we propose a novel RNN model for hyperspectral image classification, inspired by our observation that hyperspectral pixels can be regarded as sequential data. Specifically, we proposed a newly designed activation function PRetanh for hyperspectral data processing in RNN, providing an opportunity to use fairly high learning rates without the risk of getting stuck in the divergence. Furthermore, a modified GRU with PRetanh was developed to effectively analyze hyperspectral data. For hyperspectral image classification, our proposed recurrent network was shown to provide statistically higher accuracy than SVM-RBF and CNN. The proposed model considers the intrinsic sequential data structure of a hyperspectral pixel for the first time, representing a novel methodology for better understanding, modeling, and processing of hyperspectral data.

In the future, further experiments will be conducted to fully substantiate the features of deep RNN for hyperspectral image processing, providing more accurate analysis for remote sensing applications, such as transfer learning for remote sensing big data analysis and change detection. In addition, this paper only concentrates on modeling hyperspectral pixels in the spectral domain. An end-to-end convolutional-RNN will be considered for spatial-spectral hyperspectral image classification in the future. We believe that such a spatial-spectral network architecture can further improve classification accuracy.

ACKNOWLEDGMENT

The authors would like to thank the National Center for Airborne Laser Mapping for providing the Houston data set. The authors would also thank Prof. P. Gamba from the University of Pavia, Pavia, Italy, for providing the reflective optics system imaging spectrometer data and corresponding reference information.

REFERENCES

[1] J. A. Benediktsson and P. Ghamisi, *Spectral-Spatial Classification of Hyperspectral Remote Sensing Images*. Boston, MA, USA: Artech House, 2015.

[2] P. Ghamisi, M. D. Mura, and J. A. Benediktsson, "A survey on spectral–spatial classification techniques based on attribute profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 5, pp. 2335–2353, May 2015.

[3] Y. Gu, T. Liu, X. Jia, J. A. Benediktsson, and J. Chanussot, "Nonlinear multiple kernel learning with multiple-structure-element extended morphological profiles for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 6, pp. 3235–3247, Jun. 2016.

[4] J. Li, M. Khodadadzadeh, A. Plaza, X. Jia, and J. M. Bioucas-Dias, "A discontinuity preserving relaxation scheme for spectral–spatial hyperspectral image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 2, pp. 625–639, Feb. 2016.

[5] P. Ghamisi, J. A. Benediktsson, and J. R. Sveinsson, "Automatic spectral–spatial classification framework based on attribute profiles and supervised feature extraction," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 9, pp. 5771–5782, Sep. 2014.

[6] C. Wu, B. Du, and L. Zhang, "Slow feature analysis for change detection in multispectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 5, pp. 2858–2874, May 2014.

[7] A. Ertürk, M.-D. Iordache, and A. Plaza, "Sparse unmixing-based change detection for multitemporal hyperspectral images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 2, pp. 708–719, Feb. 2015.

[8] B. Demir, F. Bovolo, and L. Bruzzone, "Updating land-cover maps by classification of image time series: A novel change-detection-driven transfer learning approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 1, pp. 300–312, Jan. 2013.

[9] J. Meola, M. T. Eismann, R. L. Moses, and J. N. Ash, "Application of model-based change detection to airborne VNIR/SWIR hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 10, pp. 3693–3706, Oct. 2012.

[10] F. Hu, G.-S. Xia, J. Hu, and L. Zhang, "Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery," *Remote Sens.*, vol. 7, no. 11, pp. 14680–14707, 2015.

[11] Q. Zhu, Y. Zhong, B. Zhao, G.-S. Xia, and L. Zhang, "Bag-of-visual-words scene classifier with local and global features for high spatial resolution remote sensing imagery," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 6, pp. 747–751, Jun. 2016.

[12] X. Li, L. Mou, and X. Lu, "Scene parsing from an MAP perspective," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1876–1886, Sep. 2015.

[13] X. Lu, X. Li, and L. Mou, "Semi-supervised multitask learning for scene recognition," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1967–1976, Sep. 2015.

[14] L. G. Olmanson, P. L. Brezonik, and M. E. Bauer, "Airborne hyperspectral remote sensing to assess spatial distribution of water quality characteristics in large rivers: The Mississippi River and its tributaries in Minnesota," *Remote Sens. Environ.*, vol. 130, pp. 254–265, Mar. 2013.

[15] M. S. Moran, Y. Inoue, and E. M. Barnes, "Opportunities and limitations for image-based remote sensing in precision crop management," *Remote Sens. Environ.*, vol. 61, no. 3, pp. 319–346, Sep. 1997.

[16] S. Delalieux, B. Somers, B. Haest, T. Spanhove, J. V. Borre, and C. A. Mücher, "Heathland conservation status mapping through integration of hyperspectral mixture analysis and decision tree classifiers," *Remote Sens. Environ.*, vol. 126, pp. 222–231, Nov. 2012.

[17] J. Ham, Y. Chen, M. M. Crawford, and J. Ghosh, "Investigation of the random forest framework for classification of hyperspectral data," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 3, pp. 492–501, Mar. 2005.

[18] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, Aug. 1998.

[19] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 8, pp. 1778–1790, Aug. 2004.

[20] J. A. Gualtieri and S. Chettri, "Support vector machines for classification of hyperspectral data," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2000, pp. 813–815.

[21] J. Li, J. Bioucas-Dias, and A. Plaza, "Spectral–spatial classification of hyperspectral data using loopy belief propagation and active learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 2, pp. 844–856, Feb. 2013.

[22] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson, "Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 11, pp. 3804–3814, Nov. 2008.

[23] P. Ghamisi, "Spectral and spatial classification of hyperspectral data," Ph.D. dissertation, Dept. Faculty Elect. Comput. Eng., Univ. Iceland, Reykjavik, Iceland, 2015.

[24] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE Trans. Inf. Theory*, vol. 14, no. 1, pp. 55–63, Jan. 1968.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[26] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, "Recurrent network models for human dynamics," in *Proc. IEEE Int. Conf. Comput. Vis.*, Aug. 2015, pp. 4346–4354.

[27] Y. Yuan, L. Mou, and X. Lu, "Scene recognition by manifold regularized deep learning architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2222–2233, Oct. 2015.

[28] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, Jan. 2016.

[29] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, Aug. 2013.

[30] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *J. Sensors*, vol. 2015, pp. 258619-1–258619-12, Jan. 2015.

[31] K. Makantasis, K. Karantzalos, A. D. Doulamis, and N. D. Doulamis, "Deep supervised learning for hyperspectral data classification through convolutional neural networks," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2015, pp. 4959–4962.

[32] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 6232–6251, Oct. 2016.

[33] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2094–2107, Jun. 2014.

[34] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.

[35] P. Rodriguez, J. Wiles, and J. L. Elman, "A recurrent neural network that learns to count," *Connection Sci.*, vol. 11, no. 1, pp. 5–40, 1999.

[36] H. Lyu, H. Lu, and L. Mou, "Learning a transferable change rule from a recurrent neural network for land cover change detection," *Remote Sens.*, vol. 8, no. 6, p. 506, 2016.

[37] M. Sundermeyer, H. Ney, and R. Schlüter, "From feedforward to recurrent LSTM neural networks for language modeling," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 23, no. 3, pp. 517–529, Mar. 2015.

[38] D. Bahdanau, K. Cho, and Y. Bengio. (2014). *Neural Machine Translation by Jointly Learning to Align and Translate*. [Online]. Available: arXiv:1409.0473

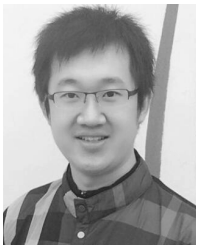
[39] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1764–1772.

[40] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] A. Graves. (2013). "Generating sequences with recurrent neural networks." [Online]. Available: https://arxiv.org/abs/1308.0850

[43] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Netw.*, vol. 18, nos. 5–6, pp. 602–610, Jul./Aug. 2005.
[44] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. (2014). "On the properties of neural machine translation: Encoder-decoder approaches." [Online]. Available: https://arxiv.org/abs/1409.1259
[45] Y. Gal and Z. Ghahramani. (2016). "A theoretically grounded application of dropout in recurrent neural networks." [Online]. Available: https://arxiv.org/abs/1512.05287
[46] J. Donahue *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 2625–2634.
[47] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic. (2016). "Generating images with recurrent adversarial networks." [Online]. Available: https://arxiv.org/abs/1602.05110
[48] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1310–1318.



Lichao Mou (S'16) received the bachelor's degree in automation from the Xi'an University of Posts and Telecommunications, Xi'an, China, in 2012, and the master's degree in signal and information processing from the University of Chinese Academy of Sciences, Beijing, China, in 2015. He is currently pursuing the Ph.D. degree with the German Aerospace Center, Wessling, Germany, and the Technical University of Munich, Munich, Germany.

In 2015, he was with the Computer Vision Group, Albert Ludwigs University of Freiburg, Freiburg im Breisgau, Germany. His research interests include remote sensing, computer vision, and machine learning, especially spaceborne remote sensing video analysis and deep networks with their applications in remote sensing.

Mr. Mou was a recipient of the first place award in the 2016 IEEE GRSS Data Fusion Contest.

Pedram Ghamisi (S'12–M'15) received the B.Sc. degree in civil (survey) engineering from Islamic Azad University South Tehran Branch, Tehran, Iran, the M.E. degree (Hons.) in remote sensing from the K. N. Toosi University of Technology, Tehran, in 2012, and the Ph.D. degree in electrical and computer engineering from the University of Iceland, Reykjavik, Iceland, in 2015.

He was a Post-Doctoral Research Fellow with the University of Iceland. Since 2015, he has been a Post-Doctoral Research Fellow with Signal Processing in Earth Observation, Technical University of Munich, Munich, Germany. He has been with GIScience and 3-D spatial data processing with the Institute of Geography, Heidelberg University, Heidelberg, Germany, since 2015. He has also been a Researcher with the Remote Sensing Technology Institute, German Aerospace Center, Wessling, Germany, on deep learning, since 2015. His research interests include machine learning, deep learning, hyperspectral image analysis, and multisensor data fusion.

Dr. Ghamisi received the Best Researcher Award for M.Sc. students from the K. N. Toosi University of Technology from 2010 to 2011. At the 2013 IEEE International Geoscience and Remote Sensing Symposium, Melbourne, he received the IEEE Mikio Takagi Prize for winning the Student Paper Competition at the conference among almost 70 people. He received the prestigious Alexander von Humboldt Fellowship in 2015. He was selected as a Talented International Researcher by the Iran's National Elites Foundation in 2016.

Xiao Xiang Zhu (S'10–M'12–SM'14) received the bachelor's degree in space engineering from the National University of Defense Technology, Changsha, China, in 2006, and the M.Sc., Dr.-Ing., and Habilitation degrees in signal processing from the Technical University of Munich (TUM), Munich, Germany, in 2008, 2011, and 2013, respectively.

Since 2011, she has been a Scientist with the Remote Sensing Technology Institute, German Aerospace Center, Wessling, Germany, where she is currently the Head of the Team Signal Analysis. Since 2013, she has also been a Helmholtz Young Investigator Group Leader, where she was also appointed as a TUM Junior Fellow. In 2015, she was appointed as the Professor for Signal Processing in Earth Observation with TUM. She was a Guest Scientist or a Visiting Professor with the Italian National Research Council, Naples, Italy, Fudan University, Shanghai, China, The University of Tokyo, Tokyo, Japan, and the University of California at Los Angeles, Los Angeles, CA, USA, in 2009, 2014, 2015, and 2016, respectively. Her research interests include advanced InSAR techniques, such as high dimensional tomographic SAR imaging and SqueeSAR; computer vision in remote sensing, including object reconstruction and multidimensional data visualization; and big data analysis in remote sensing, and modern signal processing, including innovative algorithms, such as sparse reconstruction, nonlocal means filter, robust estimation, and deep learning, with applications in the field of remote sensing, such as multi/hyperspectral image analysis.

Dr. Zhu is an Associate Editor of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING.