remote sensing

MDPI

Article

Deep Kernel Extreme-Learning Machine for the Spectral–Spatial Classification of Hyperspectral Imagery

Jiaojiao Li [1,†], Bobo Xi [1,†], Qian Du [2,†], Rui Song [1,*], Yunsong Li [1,*] and Guangbo Ren [3,†]

1 The State Key Laboratory of Integrated Service Networks, School of Telecommunications Engineering, Xidian University, Xi'an 710200, China; jjli@xidian.edu.cn (J.L.); xibobo1301@foxmail.com(B.X.)
2 The Department of Electronic and Computer Engineering, Mississippi State University, Starkville, MS 39762, USA; du@ece.msstate.edu
3 The First Institute of Oceanography, State Oceanic Administration, Qingdao 266000, China; renguangbo@fio.org.cn
* Correspondence: ruiScientific@gmail.com (R.S.); ysli@mail.xidian.edu.cn (Y.L.); Tel.: +86-186-8189-3919 (R.S.); +86-158-2926-2265 (Y.L.)
† These authors contributed equally to this work.

Received: 12 Novemver 2018; Accepted: 12 December 2018; Published: 14 December 2018

Abstract: Extreme-learning machines (ELM) have attracted significant attention in hyperspectral image classification due to their extremely fast and simple training structure. However, their shallow architecture may not be capable of further improving classification accuracy. Recently, deep-learning-based algorithms have focused on deep feature extraction. In this paper, a deep neural network-based kernel extreme-learning machine (KELM) is proposed. Furthermore, an excellent spatial guided filter with first-principal component (GFFPC) is also proposed for spatial feature enhancement. Consequently, a new classification framework derived from the deep KELM network and GFFPC is presented to generate deep spectral and spatial features. Experimental results demonstrate that the proposed framework outperforms some state-of-the-art algorithms with very low cost, which can be used for real-time processes.

Keywords: hyperspectral classification; deep layer; kernel-based ELM; spectral and spatial features

1. Introduction

Hyperspectral imagery in remote sensing with hundreds of narrow spectral bands is used in many applications, such as global environment monitoring, land cover change detection, natural disaster assessment and medical diagnosis [1–4] etc. Classification is a significant information acquisition technique for hyperspectral imagery, which focuses on distinguishing physical objects and classifying each pixel into a unique label. With the development of machine learning, most machine learning algorithms based on statistical learning theory are employed in the information processing field. There are many traditional classification algorithms, such as k-nearest neighbors (KNN) [5], Bayesian models [6], random forests [7], etc. One of the most important and famous classifiers for hyperspectral image classification is the kernel-based support vector machine (KSVM), which can also be considered to be a neural network [8,9]. It provides superior classification performance by learning an optimal decision hyperplane to best separate different classes in a high-dimensional feature space through non-linear mapping. Some popular kernels include polynomial function and Gaussian radial basis function (RBF).

Recently, deep neural networks (DNNs) have been highlighted in the literature, which can learn high-level features hierarchically [10–13]. DNNs have demonstrated their potential in classification;

in particular, they have motivated successful applications of deep models in hyperspectral image classification, which outperform other traditional classification methods. The typical deep-learning architectures include stacked autoencoders (SAEs) [14], deep belief networks (DBNs) [15] and convolutional neural networks (CNNs) [16,17]. Chen et al. first employed SAE depth model to extract features of hyperspectral imagery and classify these features via logical regression [18]. Then, Chen et al. used DBN as a classifier to distinguish each pixel [19]. In addition, Li and Du proposed a hyperspectral classification model which combines an optimized DBN with a texture feature enhancement model, achieving superior classification accuracy [20]. In particular, due to its local receptive fields, CNNs play a dominant role for processing the visual-based issues. Hu et al. employed a CNN to classify hyperspectral images directly in spectral domain [21]. However, without enough training samples, the traditional CNN faces an over-fitting problem. Li et al. proposed a fully CNN for feature enhancement and obtained outstanding hyperspectral accuracy [22]. Nevertheless, due to the high computation cost and space complexity, the aforementioned algorithms are very time-consuming. Real-time application is the predominant trend in future hyperspectral processing, and most of the aforementioned algorithms may not meet the requirements in fast data processing and analysis.

Extreme-learning machine (ELM) as a very fast and effective machine learning algorithm with a single hidden-layer feed-forward neural network was proposed by Huang in [23]. The parameters between its input and hidden layers are simply random variables. The only parameters to be trained are output weights, which can be easily estimated by a smallest norm least-squares solution. Compared with the traditional gradient-based back propagation (BP) learning, ELM is computationally much more efficient than the SVM and BP neural networks. Therefore, plentiful works about ELM have already done in hyperspectral classification [24–27] and achieved acceptable contributions. However, with the randomly generated weights and bias of ELM, it leads to different results even with the same hidden nodes. The kernel-based ELM (KELM) [28] was proposed to overcome this problem, which employs a kernel function to replace the hidden layer of the ELM. In [29,30], KELM has been used for hyperspectral image classification and obtained appreciate results. However, the feature representation ability is limited of the shallow networks. Therefore, multilayer solutions are imperative. Inspired by the multilayer perception (MLP) theory, Huang extended ELM to a multilayer ELM (ML-ELM) through using ELM-based autoencoder (ELM-AE) for feature representation and extraction [31]. From the perspective of deep learning, ELM-AE is stacked by deep layers can further extract deep robust and abstract features. In [32], the ML-ELM and KELM were combined for handling the EEG classification, where the former acted as a feature extractor and the latter as a classifier. The architecture of the method is complex and too much hyperparameters need to be determined. Additionally, several studies [33–36] have focused on integrating spatial and spectral information in hyperspectral imagery to assist classification. Multi-features are extracted and employed for classification. For instance, Kang et al. used a guided filter to process the pixel-wise classification map in each class by using the first-principal component (PC) or the first three PCs to capture major spatial features [37].

In this paper, we firstly investigate the ML-ELM for its suitability and effectiveness for hyperspectral classification. Then, to acquire desirable performance, we promote a deep layer-based kernel ELM (DKELM) algorithm to extract the deep and robust features of hyperspectral imagery. In addition, spatial information through different filters is added to further enhance the classification accuracy. The main contributions of this paper are summarized below:

1. ML-ELM is investigated and applied firstly in hyperspectral classification.
2. A classification framework is proposed for hyperspectral classification which combines DKELM and a novel guided filtering first-principal component (GFFPC) spatial filter.
3. The DKELM model remains simple, because randomly generated parameters are not necessary but only kernel parameters need to be tuned in each layer. Furthermore, compared to the ML-ELM, the numbers of nodes for each hidden layer are not required to be set due to the kernel tricks.

4. The proposed framework can achieve superior performance with very fast training speed, which is beneficial for real-time application.

This paper is organized as follows. Section 2 briefly introduces the ELM, KELM and ML-ELM (for convenience, we use MELM to replace ML-ELM in the following sections). Section 3 proposes our new framework to address the hyperspectral classification problem. Section 4 depicts the datasets and parameters tuning. Experimental results are presented and discussed in Section 5. The conclusions are drawn in Section 6.

2. Related Works

2.1. ELM and KELM

The ELM is a single hidden-layer feed-forward neural network (SLFN) as depicted in Figure 1. Please note that the hidden layer is non-linear because of the use of a non-linear activation function. However, the output layer is linear without an activation function. It contains three layers: input layer, hidden layer, and output layer.

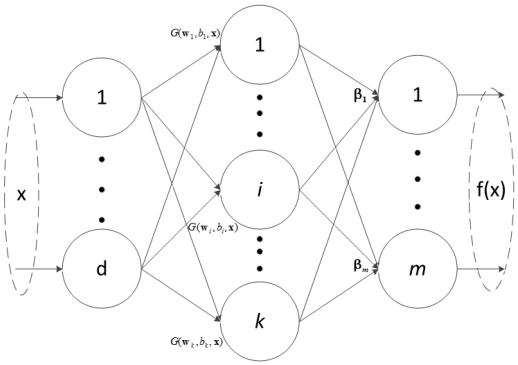

Figure 1. The structure of ELM.

Let $\mathbf{x}$ represent a training sample and $f(\mathbf{x})$ be the output of the neural network. The SLFN with k hidden nodes can be represented by the following equation:

$$f_{\mathbf{ELM}}(\mathbf{x}) = \mathbf{B}^T \bullet G(\mathbf{w}, b, \mathbf{x}), \tag{1}$$

where $G(\mathbf{w}, b, \mathbf{x})$ denotes the hidden-layer activation function, $\mathbf{w}$ is the input weight matrix connecting the input layer to the hidden layer, b means the bias weight of the hidden layer, and $\mathbf{B} = [\boldsymbol{\beta}_1 \boldsymbol{\beta}_2 ... \boldsymbol{\beta}_m]$ is the weight between the hidden layer and output layer. For an ELM with n training samples, d input neurons (i.e., the number of bands), k hidden neurons, and m output neurons (i.e., m classes), Equation (1) becomes

$$\mathbf{t}_i = \mathbf{B}^T \bullet g\left(\langle \mathbf{w}_j, \mathbf{x}_i \rangle + b_j\right), i = 1, 2, \cdots n, \tag{2}$$

where $\mathbf{t}_i$ is the m-dimensional desired output vector for the i-th training sample $\mathbf{x}_i$, the d-dimensional $\mathbf{w}_j$ represents the j-th weight vector from the input layer to the j-th hidden neuron, and b_j is the bias of the j-th hidden neuron. Here, $\langle \mathbf{w}_j, \mathbf{x}_i \rangle$ denotes the inner product of $\mathbf{w}_j$ and $\mathbf{x}_i$. The sigmoid function g is used as the activation function, so the output of the j-th hidden neuron is

$$g\left(\langle \mathbf{w}_j, \mathbf{x}_i \rangle + b_j\right) = 1 / \left(1 + \exp\left(-\frac{\mathbf{w}_j^T \mathbf{x}_i + b_j}{2\epsilon^2}\right)\right), \tag{3}$$

where $\exp(\cdot)$ denotes the exponent arithmetic, and ϵ^2 is the steepness parameter.

In matrix form, model (2) can be rearranged as

$$\mathbf{HB} = \mathbf{T}, \tag{4}$$

where $\mathbf{T} \in R^{n\times m}$ is the target output, $\mathbf{B} \in R^{k\times m}$. $\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_n) \end{bmatrix}$ is referred to as hidden-layer output matrix of ELM with the size of (n, k), which can also be expressed as follows:

$$\mathbf{H} = g\left(\mathbf{W}\cdot\mathbf{X} + \mathbf{b}\right) = \begin{bmatrix} g(<\mathbf{w}_1, \mathbf{x}_1> + b_1) & \cdots & g(<\mathbf{w}_k, \mathbf{x}_1> + b_k) \\ \vdots & \cdots & \vdots \\ g(<\mathbf{w}_1, \mathbf{x}_n> + b_1) & \cdots & g(<\mathbf{w}_k, \mathbf{x}_n> + b_k) \end{bmatrix}_{n\times k}. \tag{5}$$

Then, $\mathbf{B}$ can be estimated by a smallest norm least-squares solution:

$$\mathbf{B} = \mathbf{H}^{\dagger}\mathbf{T} = \mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T}, \tag{6}$$

where C is a regularization parameter. The ELM model can be represented as

$$f_{\mathrm{ELM}}(\mathbf{x}) = \mathrm{h}(\mathbf{x})\mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T}, \tag{7}$$

ELM can be extended to kernel-based ELM (KELM) via using kernel trick. Let

$$\mathbf{\Omega} = \mathbf{H}\mathbf{H}^T, \tag{8}$$

in which

$$\mathbf{\Omega}_{i,j} = \mathrm{k}(\mathbf{x}_i, \mathbf{x}_j), \tag{9}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ represent the i-th and j-th training sample, respectively. Then, replacing $\mathbf{H}\mathbf{H}^T$ by $\mathbf{\Omega}$, the representation of KELM can be written as

$$f_{\mathrm{KELM}}(\mathbf{x}) = \mathrm{h}(\mathbf{x})\mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{\Omega}\right)^{-1}\mathbf{T}, \tag{10}$$

where $f_{\mathrm{KELM}}(\mathbf{x})$ represents the output of the KELM model, and $\mathrm{h}(\mathbf{x})\mathbf{H}^T = \begin{bmatrix} \mathrm{k}(\mathbf{x}, \mathbf{x}_1) \\ \\ \mathrm{k}(\mathbf{x}, \mathbf{x}_n) \end{bmatrix}$.

Obviously, different from ELM, the most important characteristic of KELM is that the number of hidden nodes is not desired to be set and there are no random feature mappings. Furthermore, the computing time is reduced compared with ELM due to the kernel trick used.

2.2. Multilayer Extreme-Learning Machines (MELM)

Figure 2 depicts the detailed structure of ELM autoencoder(ELM-AE). ELM-AE represents features based on singular values. MELM is a multilayer neural network through stacking multiple ELM-AEs.

Let $\mathbf{X}^{(i)} = [\mathbf{x}_1^{(i)}, \cdots, \mathbf{x}_n^{(i)}]$, where $\mathbf{x}_k^{(i)}$ is the i-th data representation for input $\mathbf{x}_k$, k=1 to n. Suppose $\mathbf{\Lambda}^{(i)} = [\boldsymbol{\lambda}_1^{(i)}, \cdots, \boldsymbol{\lambda}_n^{(i)}]$ is the i-th transformation matrix, where $\boldsymbol{\lambda}_k^{(i)}$ is the transformation vector used for representation learning with respect to $\mathbf{x}_k^{(i)}$. According to Equation (4), $\mathbf{B}$ is replaced by $\mathbf{\Lambda}^{(i)}$ and $\mathbf{T}$ is replaced by $\mathbf{X}^{(i)}$ respectively in MELM.

$$\mathbf{H}^{(i)}\mathbf{\Lambda}^{(i)} = \mathbf{X}^{(i)}, \tag{11}$$

where $\mathbf{H}^{(i)}$ is the output matrix of the i-th hidden layer with respect to $\mathbf{X}^{(i)}$, and $\mathbf{\Lambda}^{(i)}$ can be solved by

$$\mathbf{\Lambda}^{(i)} = (\mathbf{H}^{(i)})^T(\frac{\mathbf{I}}{C} + \mathbf{H}^{(i)}(\mathbf{H}^{(i)})^T)^{-1}\mathbf{X}^{(i)}. \tag{12}$$

Then

$$\mathbf{X}^* = \mathrm{g}(\mathbf{X}^{(i)}(\mathbf{\Lambda}^{(i)})^T), \tag{13}$$

where $\mathbf{X}^*$ is the last representation of $\mathbf{X}^{(1)}$. $\mathbf{X}^*$ is used as hidden-layer output to calculate the output weight $\boldsymbol{\beta}^*$ and $\boldsymbol{\beta}^*$ can be computed as

$$\boldsymbol{\beta}^* = (\mathbf{X}^*)^\dagger\mathbf{T} = (\mathbf{X}^*)^T(\frac{\mathbf{I}}{C} + \mathbf{X}^*(\mathbf{X}^*)^T)^{-1}\mathbf{T}. \tag{14}$$

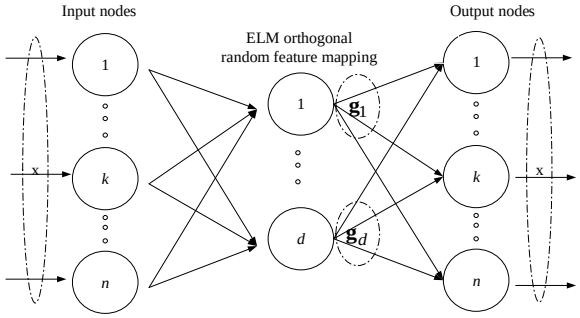

Figure 2. The structure of ELM-AE: both the input and output are $\mathbf{x}$, and ELM-AE has the same solution as the original ELM. $\mathbf{g}_d$ is the d-th hidden node for input $\mathbf{x}$.

3. The Proposed Framework for Hyperspectral Classification

In this section, we propose a new framework for hyperspectral classification which combines the hyperspectral spatial features with the Deep-layer-based KELM. The details of the proposed framework are discussed in this section. The main procedure of our proposed framework is briefly depicted in Figure 3. From Figure 3, we can see that the major three parts are as follows: data normalization, spatial feature enhancement and DKELM classification. The following sections introduce these three procedures in detail.

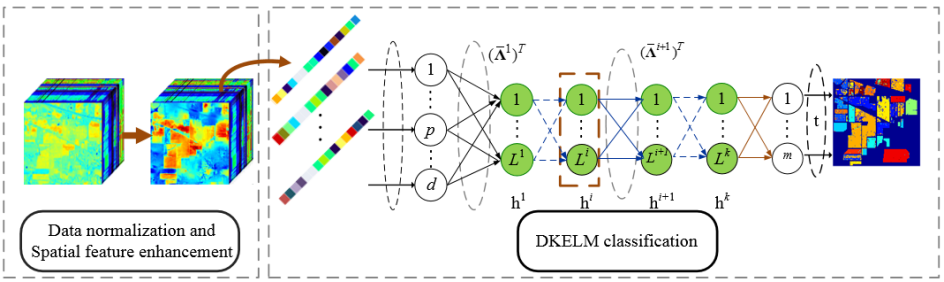

Figure 3. The main procedure of our proposed framework. $\bar{\mathbf{\Lambda}}^{(1)}$ is the first layer's transformation matrix of DKELM got from the KELM-AE of the input data. $\bar{\mathbf{\Lambda}}^{(i+1)}$ refers to the transformation matrix of $(i+1)$-*th* hidden layer $\mathbf{h}_{i+1}$ of DKELM, which obtained by KELM-AE of the i-th hidden layer $\mathbf{h}_i$.

3.1. Data Normalization

Let $\mathbf{X} \in R^{N*L}$ be a hyperspectral data, where N denotes the number of samples and L is the number of bands. Data normalization is the pre-procedure to make each sample standardization. For each sample:

$$\hat{\mathbf{x}}_i = \frac{\mathbf{x}_i - \mu_i}{\delta_i}, \tag{15}$$

where $\hat{\mathbf{x}}_i$ is the normalized sample, $\mathbf{x}_i \in \mathbf{X}, i \in \{1, \cdots, N\}$, and μ_i and δ_i denote the mean and variance of the samples, respectively. After this process, the data has zero mean and unit variance.

3.2. Spatial Features Enhancement

In our proposed framework, we use the Gaussian filters to extract spatial information; furthermore, a spatial feature enhancement method combined with guided filter (GF) and principal component analysis (PCA) is presented to enhance spatial information. Here, we introduce the GFFPC in detail.

The GF was proposed by He in 2012 [38], which can be used as an edge-preserving filter such as bilateral filter and it performs better near edges with fast time. As a local linear model, GF assumes that output image o is a linear transformation of guidance image g in a window $\boldsymbol{W}_k$ of size $\boldsymbol{\omega} * \boldsymbol{\omega}$ centered at the pixel k, where $\boldsymbol{\omega}$ = (2r+1):

$$o_i = a_k g_i + b_k, \forall i \in \boldsymbol{W_k}, \tag{16}$$

where o_i is the i-th pixel of the output image o and g_i is the i-th pixel of guidance image g, respectively. It is obvious that this model ensures $\nabla o = a \nabla g$, which means the output o will have an edge only if g has an edge. To calculate the coefficients a_k and b_k, a minimum energy function is applied as follows:

$$E\left(a_k, b_k\right) = \left(\left(a_k g_i + b_k - f_i\right)^2 + \boldsymbol{\alpha} a_k^2\right), \tag{17}$$

where f_i is the i-th pixel of the input image, and $\boldsymbol{\alpha}$ is a regularization parameter penalizing large a_k, which can affect the blurring for the GF. According to the energy function, it is expected that the output image o ought to be as close as possible to the input image f, while preserving the texture information of guidance image g through the linear model.

The solution to (16) can be addressed by linear regression as follows (Draper, 1981):

$$a_k = \frac{\frac{1}{|W|}\sum_{i \in \mathbf{W}_k} g_i f_i - \mu_k \bar{f}_k}{v_k^2 + \alpha}, \tag{18}$$

$$b_k = \bar{f}_k - a_k \mu_k, \tag{19}$$

where μ_k and v_k^2 are the mean and variance of guidance image g within the local window $\boldsymbol{W}_k$, respectively. $|W|$ is the number of pixels in $\boldsymbol{W}_k$. In addition, $\bar{f}_k = \sum_{k \in \mathbf{W}_k} f_k$ is the mean value of f in the window.

The structure of GFFPC is shown in Figure 4. We can see that the original hyperspectral image is processed by PCA method firstly, then we get the reconstructed dataset consisting of a new set of independent bands named PCs. After that, the GF is performed on each band of the original dataset, where the first PC of the reconstructed dataset is used as the gray-scale guidance image with the most information of the hyperspectral image including spatial features. The filtering output is the novel hyperspectral data cube with more distinctive edges and texture features, that can help further hyperspectral classification.

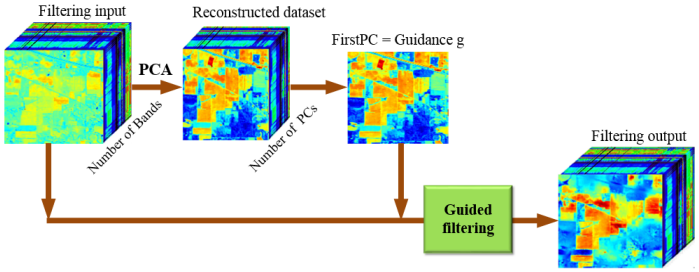

Figure 4. The structure of GFFPC.

3.3. DKELM Classification

DKELM consists of several KELM-auto-encoders (AEs) in deep layer. Thus, we firstly present a brief description of the KELM-AE.

3.3.1. KELM-AE

Figure 5 demonstrates the structure of KELM-AE which is very similar to ELM-AE except the kernel representation. The kernel operation in Figure 5 can be represented as

$$\mathbf{\Omega} = \mathrm{k}(\tilde{\mathbf{x}}, \mathbf{x}_j) = <\phi(\tilde{\mathbf{x}}), \phi(\mathbf{x}_j)>, \tag{20}$$

$\tilde{\mathbf{x}}$ is referred to as the testing samples, $\mathbf{x}_j$ denotes the j-th training sample, and ϕ is the mapping function to the reproducing kernel Hilbert space(RKHS). From Figure 5, the input matrix $\mathbf{X}^{(i)}$ is mapped into a kernel matrix $\mathbf{\Omega}^{(i)}$ through kernel function $\mathrm{k}^{(k)}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-||\mathbf{x}_i - \mathbf{x}_j||/2\sigma_k^2)$. In our proposed DKELM, we use the RBF kernel function with parameter σ_k.

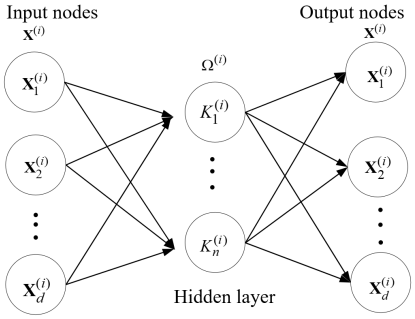

Figure 5. The structure of KELM-AE.

Then $\bar{\mathbf{\Lambda}}^{(i)}$ is employed to represent the i-th transformation matrix in KELM-AE, which is similar to ELM-AE in (10)

$$\mathbf{\Omega}^{(i)}\bar{\mathbf{\Lambda}}^{(i)} = \mathbf{X}^{(i)}, \tag{21}$$

and then $\bar{\mathbf{\Lambda}}^{(i)}$ is calculated via

$$\bar{\mathbf{\Lambda}}^{(i)} = (\frac{\mathbf{I}}{\mathrm{C}} + \mathbf{\Omega}^{(i)})^{-1}\mathbf{X}^{(i)}, \tag{22}$$

The data can be represented through the final data transformation procedure using:

$$\mathbf{X}^{(i+1)} = \mathrm{g}(\mathbf{X}^{(i)}(\bar{\mathbf{\Lambda}}^{(i)})^T), \tag{23}$$

where g is still an activation function. The hidden-layer activation functions can be either linear or non-linear. In our proposed DKELM, we use non-linear activation functions. Because deep distinct and abundant features can be learned and captured through the data representation via non-linear activation functions used between each KELM-AE. The combination of dense and sparse representations has more powerful interpretation than only linear learning. Compared to ELM-AE, we can find that the number of hidden nodes is not necessary to be set in advance because of the kernel trick used in each hidden layer. The pseudocode of KELM-AE is depicted in Algorithm 1.

3.3.2. DKELM

DKELM can obtain the universal approximation due to two separate learning procedures contained as same as in H-ELM [31]. Each pair of $\bar{\mathbf{\Lambda}}^{(i)}$ and $\mathbf{X}^{(i)}$ (in the i-th KELM-AE) can be computed via Equations (22) and (23), respectively. At last, the final data representation $\mathbf{X}_{\text{final}}^{*}$ is calculated, and then $\mathbf{X}_{\text{final}}^{*}$ is used as the training input to train a KELM classification model as:

$$\mathbf{\Omega}^{*}_{\text{final}}\boldsymbol{\beta} = \mathbf{T}, \tag{24}$$

where $\mathbf{\Omega}^{*}_{\text{final}}$ is obtained from $\mathbf{X}^{*}_{\text{final}}$, then the output weight $\boldsymbol{\beta}$ can be calculated via

$$\boldsymbol{\beta} = (\frac{\mathbf{I}}{\mathrm{C}} + \mathbf{\Omega}^{*}_{\text{final}})^{-1}\mathbf{T}. \tag{25}$$

The procedure of DKELM is depicted in Algorithm 2, including the training and testing phases.

Algorithm 1 The pseudocode of KELM-AE.

Input: Input matrix $\mathbf{X}^{(i)}$, regularization parameter C_i, kernel parameter σ_i, activation function g_i.
Output: Transformation matrix $\bar{\mathbf{\Lambda}}^{(i)}$, new data representation $\mathbf{X}^{(i+1)}$.
Step1: Calculate the kernel matrix $\mathbf{\Omega}_{k,j}{}^{(i)} \leftarrow K(\mathbf{x}_k, \mathbf{x}_j, \sigma_i)$, where $\mathbf{x}_k$ and $\mathbf{x}_j$ are referred to as the k-th and j-th training sample, respectively.
Step2: Calculate the output weight $\bar{\mathbf{\Lambda}}^{(i)} \leftarrow \left(\frac{\mathbf{I}}{C_i} + \mathbf{\Omega}^{(i)}\right)^{-1}\mathbf{X}^{(i)T}$.
Step3: Calculate the new data representation $\mathbf{X}^{(i+1)} \leftarrow \mathrm{g}_i(\bar{\mathbf{\Lambda}}^{(i)}\mathbf{X}^{(i)})^T$.
Return: $\mathbf{X}^{(i+1)}$, $\bar{\mathbf{\Lambda}}^{(i)}$.

Algorithm 2 The pseudocode of DKELM.

- **Training Phase**

Input: Input matrix $\mathbf{X}^{(i)}$, output matrix $\mathbf{T}$, regularization C_i for each hidden layer, kernel parameter σ_i for each hidden layer, activation function g_i for each hidden layer, and the number of layers N.
Output: Transformation matrix $\bar{\mathbf{\Lambda}}^{(i)}$ for each hidden layer, the final representation of the training samples $\mathbf{X}^N$ and final output weight $\boldsymbol{\beta}$ of the output layer.
Step1:
for $i = 1 : N-1$ **do**:
Calculate $\mathbf{X}^{(i+1)}, \bar{\mathbf{\Lambda}}^{(i)} \leftarrow \mathrm{KELM-AE}(\mathbf{X}^{(i)}, C_i, \sigma_i, g_i)$
end.
Step2: $\mathbf{X}^N \leftarrow \mathbf{X}^{(i+1)}$
Step3: Calculate $\mathbf{\Omega}^{\text{final}}_{k,j} \leftarrow K(\mathbf{x}^N_k, \mathbf{x}^{(N)}_j, \sigma_N)$, where $\mathbf{x}^N_k$ and $\mathbf{x}^{(N)}_j$ are the final representation of the training samples $\mathbf{x}_k$ and $\mathbf{x}_j$ respectively.
Step4: Calculate the final output weight $\boldsymbol{\beta}=\left(\frac{\mathbf{I}}{C_i} + \mathbf{\Omega}^{\text{final}}\right)^{-1}\mathbf{T}^T$.
Return $\bar{\mathbf{\Lambda}}^{(1)}$–$\bar{\mathbf{\Lambda}}^{(N-1)}$, $\mathbf{X}^N$, $\boldsymbol{\beta}$.

- **Testing Phase**

Input: Input matrix of testing samples $\mathbf{TX}^{(1)}$, output of the training phase $\bar{\mathbf{\Lambda}}^{(1)}$–$\bar{\mathbf{\Lambda}}^{(N-1)}$, $\mathbf{X}^N$, $\boldsymbol{\beta}$.
Output: Output matrix of the testing samples $\mathbf{TY}$.
Step1:
for $i = 1 : N-1$ **do**:
Calculate the hidden representation $\mathbf{TX}^{(l+1)} \leftarrow g_l(\Lambda^{(l)}\mathbf{TX}^{(l)T})$
end.
Step2: $\mathbf{TX}^N \leftarrow \mathbf{TX}^{(l+1)}$
Step3: Calculate the kernel matrix $\mathbf{\Omega}^{(N)}_{k,j} \leftarrow K(\mathbf{x}^N_k, \mathbf{x}^{(N)}_{TX\,j}, \sigma_N)$, where $\mathbf{x}^N_k$ and $\mathbf{x}^{(N)}_{TX\,j}$ are the final representation of the training samples $\mathbf{x}_k$ and the final representation of the testing sample $\mathbf{x}_j$, respectively.
Step4: Calculate the final output of the DKELM $\mathbf{TY} = (\mathbf{\Omega}^{(N)})^T\boldsymbol{\beta}$.

MELM employs the pseudoinverse to calculate the transformation matrix in each layer. Compared to MELM, exact inverse is used to calculate $\bar{\mathbf{\Lambda}}^{(i)}$ via invertible kernel matrix in the KELM-AEs of

DKELM. Therefore, a theoretically perfect reconstruction of $\mathbf{X}^{(i)}$ is resulted, which will reduce the error accumulation of the AEs in a certain degree. Consequently, DKELM can learn a better data representation and make for better generalization.

4. Experiments

In this section, we design a series of experiments to evaluate our proposed hyperspectral framework combining spatial filters with DKELM. As the comparison algorithms, ELM, KELM, KSVM and CNN are used in our experiments. Besides, all these algorithms are combined with GFFPC spatial feature enhancement method for further comparison purpose. The evaluation criteria employed are overall accuracy (OA) and kappa coefficient. Three classic hyperspectral benchmark datasets and one self-photographed hyperspectral dataset, noted as Indian Pines, University of Pavia, Salinas and Glycine ussuriensis, are used. In this paper, except CNN, all experiments are performed using MATLAB R2017b on a computer with 3.2 GHz CPU and 8.0 GB RAM. CNN algorithm is performed on NVIDIA Tesla K80 GPU.

4.1. Hyperspectral Datasets

The Indian Pines dataset was gathered by Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) sensor in North-western Indiana. It contains 220 spectral bands from 0.4–2.45 μm region with spatial resolution of 20 m. This image has 145 × 145 pixels with 200 bands after removing 20 noisy and water absorption bands. The Indian Pines scene includes two-third agriculture, and one-third forest. The groundtruth of this image is introduced in Figure 6. Sixteen classes are contained and their numbers of labeled samples are tabulated in Table 1. In Indian Pines, 10% to 50% of labeled data of each class is selected randomly as training samples, and the remaining is testing samples.

Figure 6. The groundtruth of Indian Pines.

Table 1. Labeled samples in Indian Pines dataset.

No.	Classes	Samples
1	ALFALFA	46
2	CORN-NOTILL	1428
3	CORN-MIN	830
4	CORN	237
5	GRASS/PASTURE	483
6	GRASS/TREES	730
7	GRASS/PASTURE-MOWED	28
8	HAY-WINDOWED	478
9	OATS	20
10	SPYBEAN-NOTILL	972
11	SOYBEAN-MIN	2455
12	SOYBEAN-CLEAN	593
13	WHEATS	205
14	WOODS	1265
15	BUILDING-GRASS-TREE-DRIVES	386
16	STONE-STEEL TOWERS	93
	TOTAL	10,249

The second dataset we used is about the University of Pavia, an urban scene acquired by the Reflective Optics Spectrometer (ROSIS) sensor. The ROSIS sensor can generate 115 spectral bands over 0.43 to 0.86 µm. This image scene has 610 × 340 pixels, and each pixel has 103 bands after noisy band removal. The geometric resolution is 1.3 m per pixel. The Nine groundtruth classes with the number of labeled samples are tabulated in Table 2. Figure 7 demonstrates the groundtruth of the Pavia University using as the referenced image. In Pavia University, 5% to 25% of labeled data of each class used as training samples to evaluate the proposed framework.

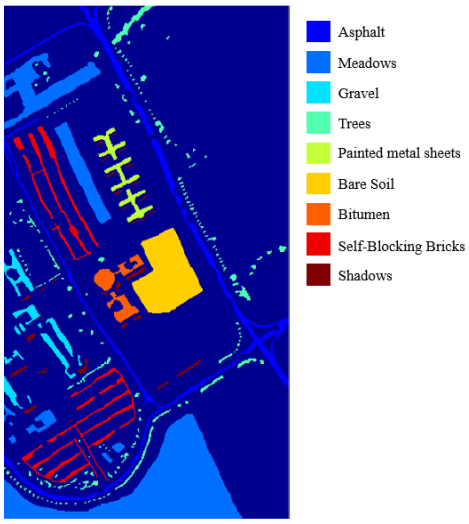

Figure 7. The groundtruth of University of Pavia.

Table 2. Labeled samples in University of Pavia.

No.	Classes	Samples
1	ASPHALT	6631
2	MEADOWS	18,649
3	GRAVEL	2099
4	TREES	3064
5	METAL SHEETS	1345
6	BARE SOIL	5029
7	BITUMEN	1330
8	BRICKS	3682
9	SHADOWS	947
	TOTAL	42,776

The third dataset is named Salinas which was also collected by the 224-band AVIRIS sensor over Salinas Valley, California. This image scene comprises 512 × 217 pixels. It has 204 bands after removing noisy and water absorption bands. The groundtruth depicted in Figure 8 contains 16 classes and the detailed samples are showed in Table 3. In Salinas dataset experiments, 5% to 25% of labeled samples of each class are chosen randomly for training our proposed classification framework.

The last dataset is named Glycine ussuriensis dataset which was collected over the Yellow River Delta National Nature Reserve, Qingdao, China. The image is acquired by the Nano-hyperspec imaging system equipped on unmanned aerial vehicle. This image scene comprises 355 × 266 pixels with 270 bands after removing noisy and water absorption bands. The Glycine ussuriensis dataset contains 4 classes shown in Figure 9 and Table 4. All four categories are plants, specifically, Glycine ussuriensis is a small-seeded species, which is grown in hills, roadsides, or shrubs at 100–800 m above sea level. Unlike the datasets mentioned before, the classes in Glycine ussuriensis have no strict geographical separation. For instance, the samples in tarragon are surrounded by other samples. In the

experiments, 5% to 25% of labeled samples of each class are chosen randomly to testify our proposed classification framework.

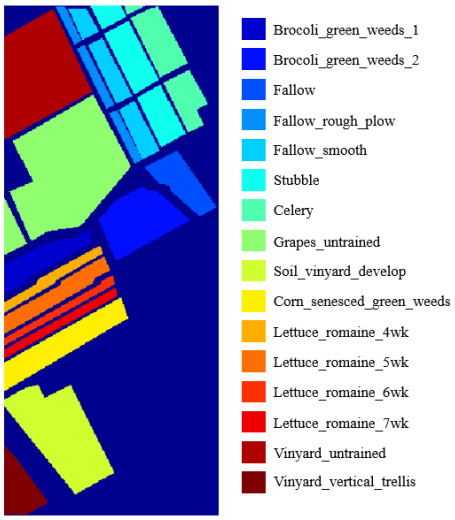

Figure 8. The groundtruth of Salinas Dataset.

Table 3. Labeled samples in Salinas Dataset.

No.	Classes	Samples
1	BROCOIL_GREEN_WEEDS_1	2009
2	BROCOIL_GREEN_WEEDS_2	3726
3	FALLOW	1976
4	FALLOW_ROUGH_PLOW	1394
5	FALLOW_SMOOTH	2678
6	STUBBLE	3959
7	CELERY	3579
8	GAPES_UNTRAINED	11,271
9	SOIL_VINYARD_DEVELOP	6203
10	CORN_SENNESCED_GREEN_WEEDS	3278
11	LETTUCE_ROMAINE_4WK	1068
12	LETTUCE_ROMAINE_5WK	1927
13	LETTUCE_ROMAINE_6WK	916
14	LETTUCE_ROMAINE_7WK	1070
15	VINYARD_UNTRAINED	7268
16	VINYARD_VERTICAL_TRELLIS	1807
	TOTAL	54,129

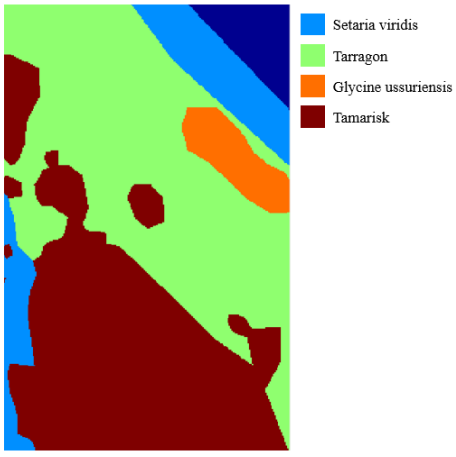

Figure 9. The groundtruth of Glycine Ussuriensis Dataset.

Table 4. Labeled samples in Glycine Ussuriensis Dataset.

No.	Classes	Samples
1	SETARIA VIRIDIS	8295
2	TARRAGON	33,985
3	GLYCINE USSURIENSIS	3446
4	TAMARISK	31,112
	TOTAL	76,838

4.2. Parameters Tuning and Setting

In our proposed framework, we need to tune several parameters. There are two user specified parameters are required for GFFPC, the size of local sliding window denoted ω and the regularization parameter denoted α, respectively, in which the latter determining the degree of the blurring for the GF. In the simulations, ω is chosen from [3, 5, 7, 9, 11], and α is selected in the range of [1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6]. Figure 10a shows the classification accuracies of DKELM in the ω subspace, where the parameter α is prefixed. It can be seen that better performance is achieved when ω equals 7 for all the datasets. Then Figure 10b indicates the impact on the classification results of different α. The OA first increased, and then decreased as α decreasing. Thus, we set α to 1e-4 finally.

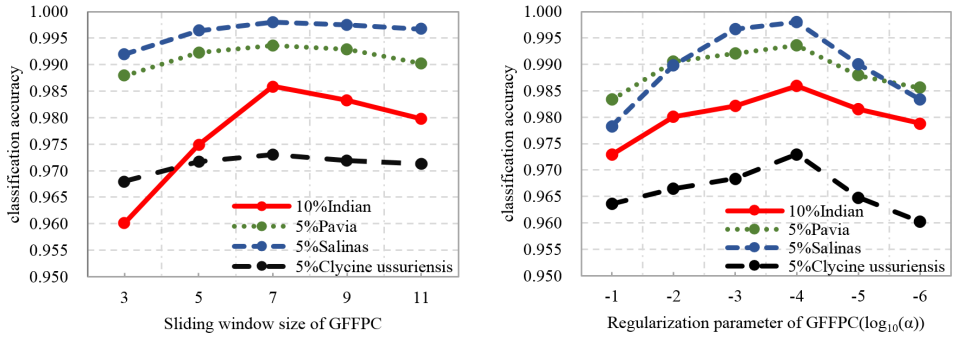

Figure 10. The impact on the classification results of different parameters of GFFPC (**a**) local sliding window size ω, (**b**) regularization parameter α.

Figure 11 shows the accuracies obtained with different numbers of kernel layers in DKELM. We can see that in the case of using three kernel layers, the classification performance of DKELM can achieve superior results, because, based on the characteristics of hyperspectral datasets, three kernel layers of the network can already extract sufficient refined and distinguished features for the classification task, and the over-fitting occurs when the network is too deep with limited training samples. Thus, in our proposed framework, we set the number of kernel layers to 3.

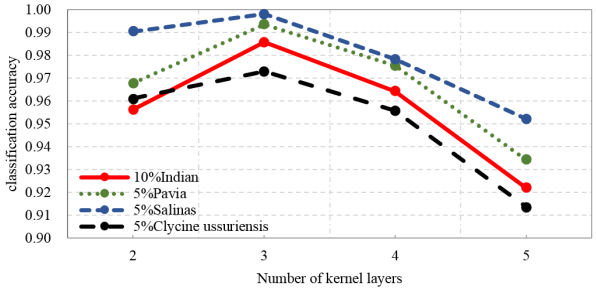

Figure 11. The accuracies obtained with different numbers of kernel layers in DKELM.

Kernel function we used in this paper is RBF. The activation functions employed to connect different KELM-AEs are sigmoid and ReLU function. Sigmoid function can constrain the output of each layer to the range from 0 to 1, which has been represented in Equation (3). The ReLU function [39] can be expressed as follows:

$$\mathrm{ReLU}(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{26}$$

where x is the input. The ReLU leads a sparse learning which can decrease the relationship among the parameters and eliminate the over-fitting problem. Therefore, the combination of sigmoid and ReLU functions can learn deep feature with different scales, which is beneficial to DKELM. Our proposed DKELM has three kernel layers, therefore three main parameters need to be tuned. σ_1, σ_2 and σ_3 are the parameters used in RBF kernel functions. To make full advantages of our framework, a gird search algorithm is employed in tuning σ_1, σ_2 and σ_3. Figure 12 depicts the detailed tuning procedures of the three parameters in Indian Pines, University of Pavia, Salinas, and Glycine ussuriensis datasets, respectively. The vertical coordinate axis represents σ_3, and the two horizontal coordinate axes express σ_1 and σ_2. The color bars in Figure 12 mean the classification accuracy obtained via the different sets of values. The black circle with best classification accuracy depicts the values we finally chosen. Here, we list the final chosen values employed in the four datasets: {4e2, 3.6e3, 5.2e7} , {2.9e2, 1e5, 6e6}, {4e2, 5e4, 8e6} and {4e2, 3.6e3, 6e7}.

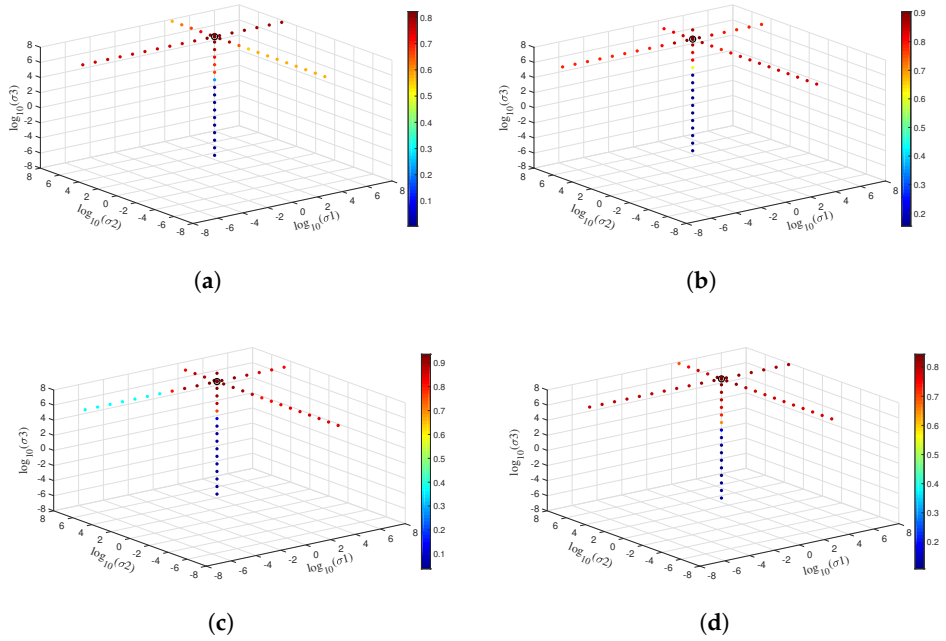

Figure 12. Parameters tuning of DKELM in (**a**) Indian pines, (**b**) University of Pavia, (**c**) Salinas, and (**d**) Glycine ussuriensis Datasets.

5. Experimental Results and Discussions

In this section, the proposed GFFPC and the novel classification model will be assessed, and the related results will be summarized and discussed at length. All the algorithms are repeated 10 runs with different locations of the training samples for each dataset, and the mean results with the corresponding standard deviation are reported.

5.1. Discussion on the Proposed GFFPC

In the experiments, we first investigate the impact of different spatial filters to MELM and DKELM. Figure 13 illustrates the OA of different spatial filters based on MELM and DKELM algorithms. Origin

denotes the performance obtained via the original MELM and DKELM. G_3 × 3 and G_5 × 5 mean using Gaussian filter with 3 × 3 and 5 × 5 size of windows respectively before employing MELM and DKELM. The GFFPC represents using GFFPC based on MELM and DKELM.

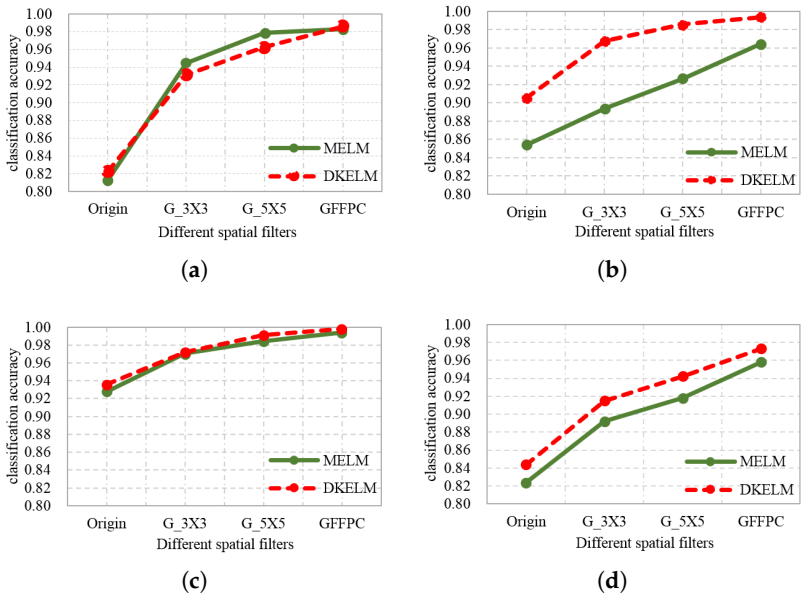

Figure 13. The performance of different spatial filters combined with MELM and DKELM in (**a**) Indian pines, (**b**) University of Pavia, (**c**) Salinas, and (**d**) Glycine ussuriensis Datasets.

From Figure 13, We can see that the performance with spatial filters become better than without spatial filters. Furthermore, the GFFPC obtains superior performance to Gaussian filters. Consequently, in our subsequent experiments, the GFFPC filter is our recommended spatial filter for strongly enhancing spatial features.

5.2. Discussion on the Classification Results

Table 5 tabulates the performance achieved by different classification algorithms and the combinations with GFFPC spatial filter through using 10% of labeled samples as training samples in Indian Pines. Although the OA of DKELM is slightly worse than the performance of CNN, our proposed framework DKELM-GFFPC outperforms other classification algorithms. In addition, quite apart from that, our proposed GFFPC can enhance the classification performance by a wide range. For instance, the OA of ELM-GFFPC is increased by 16.23% and the OA of CNN-GFFPC has been improved from 83.19% to 95.57%. In particular, the performance of MELM and DKLEM are enhanced by 17.02% and 16.38%, respectively. For class 1, 9 and 16 of Indian Pines, the accuracies increase from 83.33%, 66.67% and 98.33% to 100%, 94.44% and 100% when DKELM-GFFPC is used instead of DKELM. This phenomenon indicates that our proposed framework can beneficial to the performance of several small-size classes.

Table 6 also lists the classification performance of the comparison algorithms and the proposed algorithm in University of Pavia dataset through using 5% labeled samples as training samples. From the accuracies exhibited in Table 6, the OA of DKELM is improved by 13.95% via using GFFPC spatial filter. The performance of other algorithms is also enhanced in different degrees from 2.6% to 11.29%. It also can be seen that after combining with GFFPC, the classification performance is enhanced greatly. In particular, the DKELM-GFFPC achieves the best performance. Furthermore, the accuracies of small-size classes such as classes 5, 7 and 9 are improved through using GFFPC, implying that GFFPC can keep more distinctive features of small-size classes.

Table 5. Classification Accuracy(%) of the comparison and proposed algorithms in Indian Pines.

NO.	ELM	KELM	KSVM	CNN	MELM	DKELM	ELM-GFFPC	KELM-GFFPC	KSVM-GFFPC	CNN-GFFPC	MELM-GFFPC	DKELM-GFFPC
1	92.00 ± 2.24	100.00 ± 0.00	73.53 ± 1.25	75.00 ± 1.21	88.24 ± 4.67	83.33 ± 0.66	100.00 ± 0.00	100.00 ± 0.00	90.70 ± 0.76	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
2	70.79 ± 1.80	77.40 ± 2.24	78.37 ± 0.48	77.05 ± 0.77	71.46 ± 0.94	78.20 ± 3.65	84.48 ± 0.92	91.21 ± 0.85	95.30 ± 0.87	93.76 ± 0.41	96.48 ± 0.15	97.60 ± 1.22
3	69.13 ± 3.61	77.38 ± 1.14	74.34 ± 1.38	80.13 ± 0.56	70.75 ± 2.53	74.42 ± 1.52	90.15 ± 1.02	96.08 ± 0.57	94.24 ± 1.24	97.31 ± 0.96	95.75 ± 0.37	98.91 ± 0.88
4	61.58 ± 2.75	69.14 ± 3.57	59.49 ± 1.55	68.16 ± 2.66	74.44 ± 2.48	67.39 ± 2.07	93.67 ± 0.91	83.40 ± 1.17	83.68 ± 1.19	83.61 ± 0.34	97.70 ± 0.45	93.39 ± 1.38
5	84.91 ± 1.53	86.69 ± 0.97	86.40 ± 0.35	91.55 ± 0.83	85.56 ± 0.95	85.49 ± 1.31	94.35 ± 0.39	97.09 ± 0.55	96.26 ± 0.94	97.72 ± 0.64	99.01 ± 0.09	99.28 ± 0.06
6	89.73 ± 0.82	86.60 ± 0.12	90.84 ± 0.12	92.99 ± 0.49	90.50 ± 0.58	89.54 ± 0.13	96.89 ± 1.21	98.20 ± 0.99	99.85 ± 0.75	98.64 ± 0.26	100.00 ± 0.00	99.85 ± 0.05
7	80.00 ± 10.48	80.00 ± 0.00	70.00 ± 3.55	95.45 ± 1.80	94.12 ± 5.18	85.00 ± 1.17	95.65 ± 0.95	74.19 ± 1.33	71.88 ± 1.79	64.71 ± 2.40	82.76 ± 1.92	77.42 ± 2.22
8	94.21 ± 1.02	87.68 ± 1.03	96.09 ± 0.02	96.32 ± 2.11	91.59 ± 0.82	92.59 ± 0.64	99.06 ± 0.06	99.07 ± 0.65	99.77 ± 0.03	99.30 ± 0.57	99.77 ± 0.23	99.31 ± 0.18
9	66.67 ± 0.65	66.67 ± 3.00	75.00 ± 3.10	41.67 ± 6.78	100.00 ± 0.00	66.67 ± 0.55	0.00 ± 0.00	100.00 ± 0.00	70.83 ± 2.13	91.67 ± 1.52	100.00 ± 0.00	94.44 ± 0.64
10	69.73 ± 0.51	77.27 ± 2.14	74.97 ± 2.07	76.21 ± 3.66	78.16 ± 1.53	76.88 ± 0.69	91.95 ± 1.21	94.45 ± 0.50	94.32 ± 0.45	92.60 ± 0.82	97.37 ± 0.44	97.42 ± 0.42
11	70.60 ± 2.21	73.71 ± 0.29	77.35 ± 1.67	81.21 ± 0.37	79.67 ± 1.03	78.73 ± 1.22	93.58 ± 0.78	94.90 ± 0.42	98.86 ± 0.86	95.74 ± 1.21	99.28 ± 0.40	99.23 ± 0.45
12	74.34 ± 0.42	81.58 ± 0.31	81.02 ± 0.16	80.39 ± 1.16	76.89 ± 1.14	84.68 ± 0.47	85.56 ± 2.55	92.49 ± 0.91	94.12 ± 1.03	88.85 ± 0.92	95.90 ± 0.36	97.19 ± 1.44
13	96.77 ± 2.21	92.86 ± 0.48	86.26 ± 1.41	92.71 ± 0.21	94.27 ± 1.38	93.78 ± 0.58	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
14	88.75 ± 0.60	82.69 ± 1.38	92.49 ± 1.12	93.48 ± 1.07	91.35 ± 0.36	89.65 ± 2.31	99.91 ± 0.08	99.91 ± 0.06	99.12 ± 0.12	99.91 ± 0.05	99.82 ± 0.09	99.91 ± 0.04
15	63.76 ± 2.68	84.76 ± 0.61	69.58 ± 2.05	68.28 ± 2.35	88.04 ± 1.25	86.14 ± 1.45	95.39 ± 1.55	97.06 ± 1.21	96.43 ± 0.11	96.76 ± 0.82	99.04 ± 0.31	98.53 ± 0.36
16	98.39 ± 1.33	98.61 ± 0.00	98.57 ± 0.35	100.00 ± 0.00	100.00 ± 0.00	98.33 ± 0.08	100.0 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
OA	76.70 ± 0.28	79.42 ± 0.88	81.06 ± 1.20	83.19 ± 0.85	81.24 ± 0.28	82.21 ± 0.56	92.93 ± 0.33	95.31 ± 0.39	96.64 ± 0.91	95.57 ± 0.49	98.26 ± 0.12	**98.59 ± 0.40**
Kappa	73.25 ± 0.35	76.26 ± 0.44	78.34 ± 0.60	80.77 ± 0.98	78.52 ± 0.33	79.59 ± 0.78	91.92 ± 0.21	94.64 ± 0.22	96.17 ± 1.64	94.94 ± 0.36	98.02 ± 0.13	**98.39 ± 0.29**

Table 6. Classification Accuracy(%) of the comparison and proposed algorithms in University of Pavia.

NO.	ELM	KELM	KSVM	CNN	MELM	DKELM	ELM-GFFPC	KELM-GFFPC	KSVM-GFFPC	CNN-GFFPC	MELM-GFFPC	DKELM-GFFPC
1	89.22 ± 0.52	92.62 ± 1.02	93.87 ± 0.18	94.88 ± 0.71	90.69 ± 0.40	91.17 ± 0.29	89.03 ± 0.48	90.07 ± 0.87	95.47 ± 1.21	92.47 ± 0.50	94.74 ± 2.18	99.65 ± 0.34
2	81.88 ± 1.33	82.22 ± 0.19	95.97 ± 0.19	97.35 ± 1.09	85.54 ± 0.22	93.38 ± 0.59	92.69 ± 0.58	95.48 ± 0.49	99.48 ± 0.46	99.22 ± 0.76	97.94 ± 0.11	99.99 ± 0.01
3	75.20 ± 1.80	80.94 ± 0.10	84.90 ± 0.81	86.59 ± 2.55	78.31 ± 1.09	79.99 ± 2.08	98.23 ± 0.20	98.99 ± 0.09	98.94 ± 0.13	93.64 ± 0.90	95.45 ± 0.46	98.45 ± 0.50
4	83.84 ± 0.22	85.21 ± 0.14	97.84 ± 0.19	92.61 ± 0.15	88.05 ± 1.11	94.77 ± 1.32	85.28 ± 0.53	87.60 ± 1.01	94.07 ± 0.38	92.45 ± 0.60	88.01 ± 0.55	96.12 ± 0.74
5	99.45 ± 0.32	99.53 ± 0.08	98.99 ± 0.39	99.07 ± 0.06	99.84 ± 0.08	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
6	84.90 ± 7.79	92.49 ± 0.14	92.18 ± 1.14	93.00 ± 1.33	87.17 ± 0.29	90.38 ± 1.37	96.55 ± 0.92	98.38 ± 0.95	99.50 ± 0.08	98.88 ± 0.99	95.62 ± 2.12	99.71 ± 0.17
7	77.35 ± 0.12	86.61 ± 0.16	83.13 ± 2.48	87.45 ± 2.46	82.80 ± 0.47	91.47 ± 1.14	97.92 ± 0.88	99.92 ± 0.03	100.00 ± 0.00	95.82 ± 0.14	99.44 ± 0.09	99.84 ± 0.08
8	64.07 ± 0.13	71.14 ± 2.60	84.71 ± 0.87	83.26 ± 0.46	72.17 ± 0.22	74.51 ± 3.96	92.31 ± 1.01	96.02 ± 0.81	93.97 ± 1.43	92.20 ± 1.61	97.21 ± 1.59	98.23 ± 0.89
9	91.58 ± 0.67	91.29 ± 0.19	100.00 ± 0.00	100.00 ± 0.00	93.93 ± 2.00	91.42 ± 0.97	100.00 ± 0.00	100.00 ± 0.00	99.89 ± 0.07	99.89 ± 0.44	98.30 ± 0.33	98.33 ± 0.50
OA	81.52 ± 0.90	84.02 ± 0.21	93.59 ± 0.63	94.12 ± 0.52	85.41 ± 0.19	90.51 ± 1.21	92.81 ± 0.69	95.03 ± 0.26	98.01 ± 0.28	96.72 ± 0.60	96.40 ± 0.21	**99.36 ± 0.29**
Kappa	74.66 ± 1.36	78.06 ± 2.30	91.47 ± 0.14	92.21 ± 0.43	80.14 ± 0.28	87.31 ± 0.88	90.13 ± 0.59	93.35 ± 0.14	97.36 ± 0.31	95.65 ± 0.43	95.20 ± 0.15	**99.15 ± 0.45**

Table 7 demonstrates the performance obtained via different classification algorithms in Salinas dataset through adopting 5% of labeled samples as training samples. Compared with other algorithms, the proposed DKELM-GFFPC is still a predominant one. In addition, the classification performance of ELM, KELM, KSVM, CNN, MELM and DKELM is increased by 5.86%, 7.81%, 6.21%, 6.43%, 6.61% and 6.22%, respectively. Furthermore, classes 11, 13 and 14 are small classes with a few training samples, the performances of which are enhanced greatly through using the GFFPC spatial filter.

The classification accuracies achieved through the Glycine ussuriensis dataset are tabulated in Table 8. From Table 8, the DKELM-GFFPC obtains best performance than other classification frameworks. GFFPC still has the positive influence on enhancing the classification performance. For instance, the OAs of MELM and CNN are improved by 13.41% and 12.48% through adding GFFPC spatial filter. Moreover, the classification accuracy of the class Glycine ussuriensis with the least labeled samples is enhanced greatly through our proposed spatial filter and classification framework.

From Tables 5–8, we can see that CNN, MELM and DKELM can work better than other traditional classifiers. Therefore, to further testify our proposed framework, we compare DKELM-GFFPC with CNN-GFFPC and MELM-GFFPC when using different percentage of training samples in Figure 14. With the increasing training samples, the classification performance is increasing gradually. Obviously, DKELM-GFFPC outperforms other two classification frameworks regardless the number of training samples. One of the most significant advantages of the proposed classification framework is the very fast training procedure. Thus, the average training times of different algorithms are compared and demonstrated in Table 9. ELM-based methods are faster than KSVM and CNN. The training time of DKELM and KELM depend on the number of training data. Meanwhile, ELM and MELM rely on the number of hidden neurons. The more the hidden neurons, the more the training time since the higher model generalization is provided subject to the data complexity. CNN can achieve superior classification performance. Nevertheless, it spends more training time which is nearly 135 to 411 times than that of DKELM in different experimental datasets. Therefore, DKELM is the most appealing one with the best performance and the least training time.

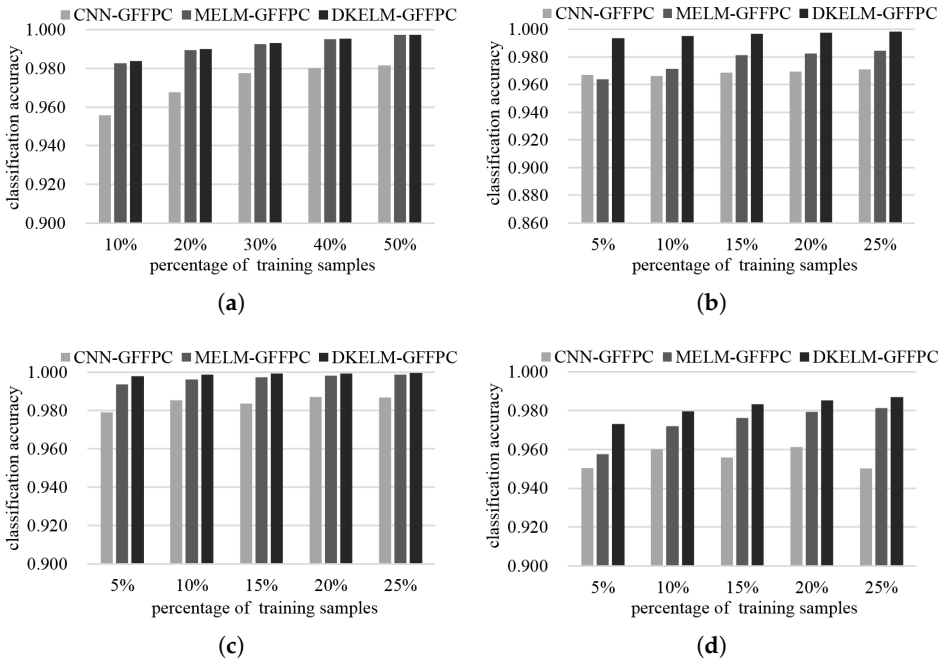

Figure 14. The performance of CNN, MELM and DKELM combined with GFFPC through using different size of training sample in (**a**) Indian pines, (**b**) University of Pavia, (**c**) Salinas, and (**d**) Glycine ussuriensis Dataset.

Table 7. Classification Accuracy(%) of the comparison and proposed algorithms in Salinas Dataset.

NO.	ELM	KELM	KSVM	CNN	MELM	DKELM	ELM-GFFPC	KELM-GFFPC	KSVM-GFFPC	CNN-GFFPC	MELM-GFFPC	DKELM-GFFPC
1	99.84 ± 0.16	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.78 ± 0.22	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
2	99.10 ± 0.64	98.25 ± 0.12	99.07 ± 0.05	99.24 ± 0.16	99.18 ± 0.08	99.49 ± 0.11	99.69 ± 0.18	99.97 ± 0.23	99.89 ± 0.07	99.69 ± 0.27	100.00 ± 0.00	100.00 ± 0.00
3	97.53 ± 1.01	98.81 ± 0.26	95.79 ± 0.33	94.56 ± 1.52	98.58 ± 0.98	97.55 ± 0.08	97.66 ± 1.05	99.79 ± 0.65	97.61 ± 1.18	97.81 ± 0.36	100.00 ± 0.00	100.00 ± 0.00
4	99.47 ± 0.30	99.54 ± 0.13	99.02 ± 0.21	97.20 ± 0.38	99.53 ± 0.09	99.54 ± 0.13	98.21 ± 0.91	98.73 ± 0.78	98.63 ± 0.51	98.14 ± 1.84	99.08 ± 0.08	99.02 ± 0.71
5	93.46 ± 1.04	96.86 ± 1.12	99.12 ± 0.20	99.35 ± 0.07	98.74 ± 0.55	98.78 ± 0.64	99.64 ± 0.33	99.76 ± 0.23	98.81 ± 0.59	99.72 ± 0.23	98.94 ± 0.14	99.45 ± 0.40
6	99.87 ± 0.03	99.95 ± 0.02	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.97 ± 0.01	99.95 ± 0.05	100.00 ± 0.00	99.87 ± 0.11	100.00 ± 0.00	99.97 ± 0.01
7	98.52 ± 1.09	99.41 ± 0.36	99.21 ± 0.35	99.94 ± 0.05	99.88 ± 0.09	99.59 ± 0.07	99.76 ± 0.18	99.88 ± 0.09	98.42 ± 0.41	99.79 ± 0.15	99.94 ± 0.02	99.97 ± 0.01
8	74.68 ± 1.45	75.78 ± 3.27	80.28 ± 1.26	76.98 ± 2.65	80.73 ± 0.58	82.42 ± 0.87	88.02 ± 1.23	98.76 ± 0.15	98.55 ± 0.62	97.67 ± 1.05	98.97 ± 0.37	99.91 ± 0.03
9	98.30 ± 0.28	98.68 ± 0.45	99.05 ± 0.45	98.05 ± 0.51	98.92 ± 0.07	98.92 ± 0.25	99.81 ± 0.12	99.95 ± 0.03	99.51 ± 0.36	99.49 ± 0.42	99.97 ± 0.02	99.85 ± 0.11
10	92.73 ± 0.05	95.85 ± 1.22	98.34 ± 0.62	94.87 ± 2.61	98.58 ± 0.36	98.42 ± 0.28	98.79 ± 0.52	99.17 ± 0.22	99.15 ± 0.80	98.67 ± 0.66	99.32 ± 0.58	99.20 ± 0.50
11	96.16 ± 0.07	97.33 ± 0.68	98.24 ± 0.06	92.69 ± 1.21	99.80 ± 0.19	98.83 ± 0.30	99.61 ± 0.07	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
12	92.99 ± 1.63	96.37 ± 0.78	98.12 ± 0.23	96.47 ± 0.97	99.29 ± 0.11	98.92 ± 1.30	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.78 ± 0.07	100.00 ± 0.00	100.00 ± 0.00
13	92.07 ± 0.42	92.21 ± 1.35	95.64 ± 1.22	93.44 ± 0.22	97.71 ± 0.22	99.07 ± 0.55	97.26 ± 1.28	99.53 ± 0.11	99.88 ± 0.09	97.59 ± 1.89	100.00 ± 0.00	100.00 ± 0.00
14	95.92 ± 0.68	96.73 ± 1.66	96.63 ± 0.55	94.37 ± 1.55	96.82 ± 0.44	96.87 ± 0.18	96.99 ± 0.81	97.16 ± 1.86	96.05 ± 0.69	97.01 ± 0.63	96.89 ± 1.59	98.51 ± 0.59
15	79.67 ± 0.38	82.36 ± 3.28	83.39 ± 0.26	85.64 ± 1.86	80.32 ± 1.06	84.17 ± 2.36	91.24 ± 1.39	95.85 ± 2.23	98.13 ± 0.97	91.67 ± 2.78	98.41 ± 0.14	99.80 ± 0.11
16	99.71 ± 0.18	99.82 ± 0.18	97.53 ± 0.78	98.66 ± 0.94	99.94 ± 0.24	99.77 ± 0.02	99.94 ± 0.02	99.94 ± 0.05	98.27 ± 0.39	99.94 ± 0.04	100.00 ± 0.00	99.94 ± 0.02
OA	90.01 ± 0.46	91.17 ± 1.37	92.69 ± 0.59	91.48 ± 1.21	92.77 ± 0.28	93.58 ± 0.78	95.87 ± 0.25	98.98 ± 0.58	98.90 ± 0.54	97.91 ± 0.87	99.38 ± 0.25	**99.80 ± 0.20**
Kappa	88.84 ± 0.52	90.13 ± 0.11	91.84 ± 0.45	90.48 ± 0.98	91.93 ± 0.31	92.83 ± 0.55	95.40 ± 0.19	98.87 ± 0.50	98.78 ± 0.34	97.68 ± 0.76	99.31 ± 0.31	**99.78 ± 0.22**

Table 8. Classification Accuracy(%) of the comparison and proposed algorithms in Glycine ussuriensis Dataset.

NO.	ELM	KELM	KSVM	CNN	MELM	DKELM	ELM-GFFPC	KELM-GFFPC	KSVM-GFFPC	CNN-GFFPC	MELM-GFFPC	DKELM-GFFPC
1	80.73 ± 1.81	80.01 ± 0.27	79.59 ± 0.95	80.65 ± 0.79	79.96 ± 2.67	83.30 ± 0.70	92.33 ± 0.69	92.74 ± 0.76	94.62 ± 0.70	93.69 ± 1.19	94.17 ± 0.75	96.34 ± 0.55
2	75.94 ± 2.90	77.35 ± 1.54	84.33 ± 1.48	82.58 ± 0.95	81.18 ± 1.75	82.64 ± 0.31	92.74 ± 0.31	95.02 ± 0.79	94.36 ± 0.75	94.75 ± 0.49	96.24 ± 0.25	97.80 ± 0.14
3	64.73 ± 1.90	84.26 ± 0.95	75.48 ± 0.80	77.37 ± 0.66	86.22 ± 2.74	86.65 ± 0.28	91.54 ± 0.95	92.06 ± 0.19	95.54 ± 1.28	97.39 ± 0.59	92.13 ± 0.51	96.76 ± 0.15
4	77.56 ± 2.12	85.26 ± 3.57	85.23 ± 1.14	83.55 ± 1.35	84.04 ± 1.39	86.52 ± 0.46	93.27 ± 0.30	95.77 ± 0.49	95.45 ± 1.68	95.48 ± 0.43	96.11 ± 0.70	97.09 ± 0.26
OA	76.50 ± 0.63	80.79 ± 0.95	83.78 ± 0.42	82.56 ± 0.84	82.36 ± 0.65	84.38 ± 0.71	92.85 ± 0.44	94.94 ± 0.45	95.38 ± 0.66	95.04 ± 0.59	95.77 ± 0.89	**97.30 ± 0.44**
Kappa	62.16 ± 0.97	69.16 ± 0.96	74.18 ± 0.91	72.09 ± 0.93	71.55 ± 0.71	74.85 ± 0.28	88.60 ± 0.38	91.93 ± 0.65	91.81 ± 1.16	94.94 ± 0.22	93.26 ± 0.96	**95.70 ± 0.25**

Table 9. Training time (In second) of different algorithms based on four datasets.

	ELM	KELM	KSVM	CNN	MELM	DKELM
10%Indian	2.59	0.07	406.80	246.7	5.18	0.60
5%Pavia	1.42	0.31	446.91	847.06	5.09	2.85
5%Salinas	1.37	0.62	2094.14	679.35	3.10	5.02
5%Glycine ussuriensis	6.91	1.51	4927.52	966.41	7.67	13.52

To illustrate the merits of our proposed classification framework and the spatial filter from the perspective of visualization, Figures 15–18 demonstrate the classification maps. Clearly, compared with the groundtruth shown in Figures 6–9, the classification maps obtained by our proposed framework are the smoothest and clearest. Besides, the classification maps achieved with GFFPC spatial filter are more distinct than without evidently. In particular, the border pixels and the boundaries of different classes in DKELM-GFFPC are more distinct. Compared with other classification methods, our proposed framework is better because of less salt-an-pepper noise contained in the classification maps.

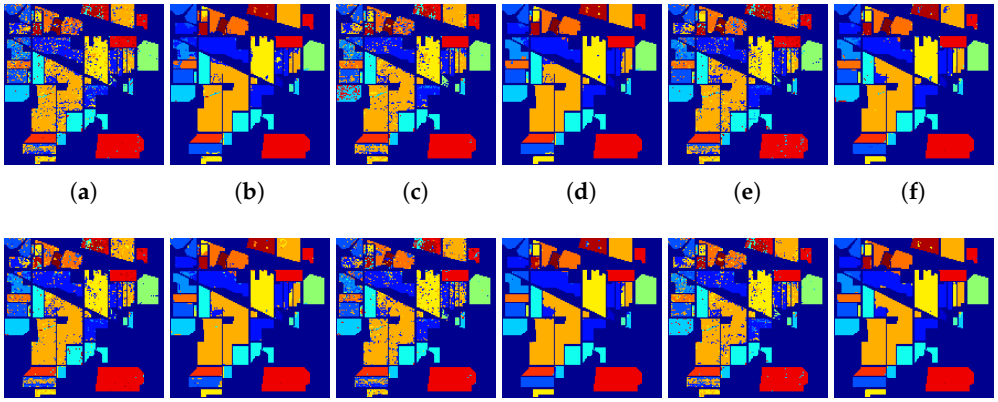

Figure 15. Classification maps of the (**a**) ELM, (**b**) ELM + GFFPC, (**c**) KELM, (**d**) KELM + GFFPC, (**e**) KSVM, (**f**) KSVM + GFFPC, (**g**) CNN, (**h**) CNN + GFFPC, (**i**) MELM, (**j**) MELM + GFFPC, and the proposed (**k**) DKELM, and (**l**) DKELM + GFFPC for the Indian Pines.

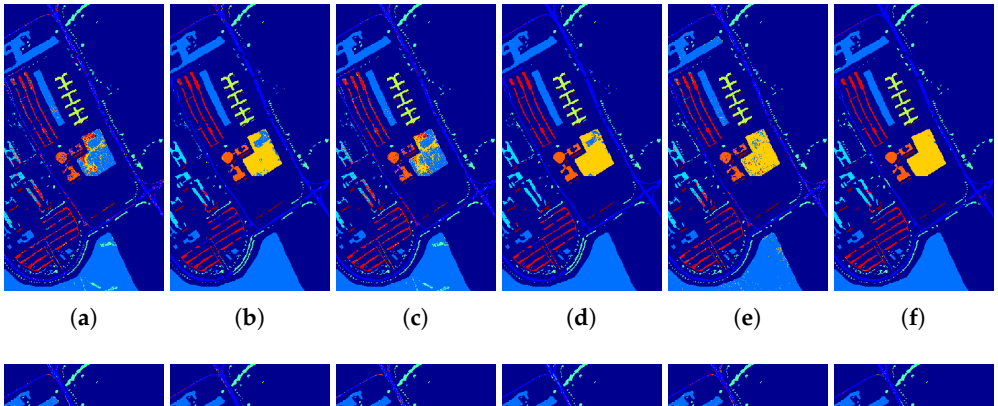

Figure 16. Classification maps of the (**a**) ELM, (**b**) ELM + GFFPC, (**c**) KELM, (**d**) KELM + GFFPC, (**e**) KSVM, (**f**) KSVM + GFFPC, (**g**) CNN, (**h**) CNN + GFFPC, (i) MELM, (**j**) MELM + GFFPC, and the proposed (**k**) DKELM, and (**l**) DKELM + GFFPC for the University of Pavia.

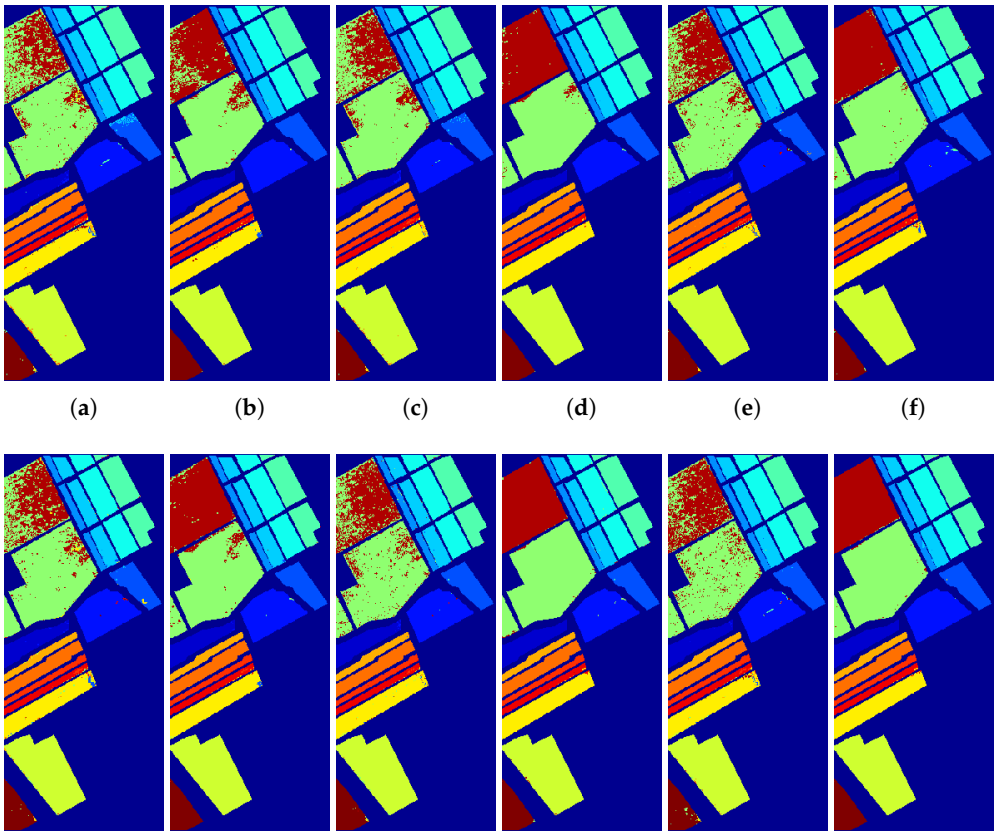

Figure 17. Classification maps of the (**a**) ELM, (**b**) ELM + GFFPC, (**c**) KELM, (**d**) KELM + GFFPC, (**e**) KSVM, (**f**) KSVM + GFFPC, (**g**) CNN, (**h**) CNN + GFFPC, (**i**) MELM, (**j**) MELM + GFFPC, and the proposed (**k**) DKELM, and (**l**) DKELM + GFFPC for the Salinas dataset

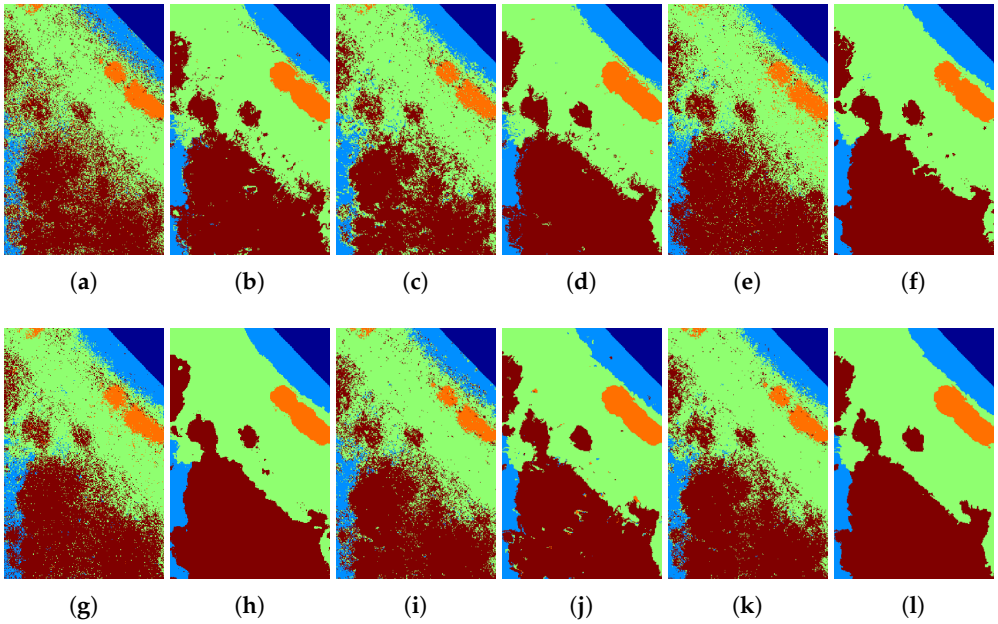

Figure 18. Classification maps of the (**a**) ELM, (**b**) ELM + GFFPC, (**c**) KELM, (**d**) KELM + GFFPC, (**e**) KSVM, (**f**) KSVM + GFFPC, (**g**) CNN, (**h**) CNN + GFFPC, (**i**) MELM, (**j**) MELM + GFFPC, and the proposed (**k**) DKELM, and (**l**) DKELM + GFFPC for the Glycine ussuriensis dataset.

6. Conclusions

In this work, the MELM algorithm is investigated and firstly applied to hyperspectral classification. Then, a DKELM-GFFPC framework is proposed consisting of the GFFPC for enhancing spatial features and the DKELM, a kernel version of MELM. Experimental results demonstrate that it can outperform other traditional algorithms, especially, DKELM-GFFPC can improve the accuracy of those classes with small-size samples in a different degree for each dataset. Moreover, compared with Gaussian filter, the proposed GFFPC can play an important role in enhancing hyperspectral classification performance. Finally, our proposed classification framework takes much lowest computation cost to achieve the highest classification accuracy. Based on the above-mentioned advantages, we believe that the proposed hyperspectral classification framework based on the novel DKELM and GFFPC is more suitable to process hyperspectral data in practical applications with low cost of computing, furthermore, in real-time application.

Author Contributions: J.L. and Q.D. conceived and designed the study; B.X. performed the experiments; G.R. and R. S. shared part of the experiment data; J.L. and Y.L. analyzed the data; J.L. and B.X. wrote the paper. Y.L., Q.D. and R. S. reviewed and edited the manuscript. All authors read and approved the manuscript.

Acknowledgments: This work was partially supported by the Fundamental Research Funds for the Central Universities JB170109, General Financial Grant from the China Postdoctoral Science Foundation (no. 2017M623124) and Special Financial Grant from the China Postdoctoral Science Foundation (no. 2018T111019). It was also partially supported by the National Nature Science Foundation of China (no. 61571345, 91538101, 61501346 and 61502367) and the 111 project (B08038).

Conflicts of Interest: The authors declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work and this paper was not published before, the founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Shan, J.; Zhao, J.; Liu, L.; Zhang, Y.; Wang, X.; Wu, F. A novel way to rapidly monitor microplastics in soil by hyperspectral imaging technology and chemometrics. *Environ. Pollut.* **2018**, *238*, 121–129. doi:10.1016/j.envpol.2018.03.026. [CrossRef]
2. Liu, K.; Su, H.; Li, X. Estimating High-Resolution Urban Surface Temperature Using a Hyperspectral Thermal Mixing (HTM) Approach. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2016**, *9*, 804–815. doi:10.1109/JSTARS.2015.2459375. [CrossRef]
3. Haboudane, D.; Tremblay, N.; Miller, J.R.; Vigneault, P. Remote Estimation of Crop Chlorophyll Content Using Spectral Indices Derived From Hyperspectral Data. *IEEE Trans. Geosci. Remote Sens.* **2008**, *46*, 423–437. doi:10.1109/TGRS.2007.904836. [CrossRef]
4. Pike, R.; Lu, G.; Wang, D.; Chen, Z.G.; Fei, B. A Minimum Spanning Forest-Based Method for Noninvasive Cancer Detection With Hyperspectral Imaging. *IEEE Trans. Biomed. Eng.* **2016**, *63*, 653–663. doi:10.1109/TBME.2015.2468578. [CrossRef]
5. Li, W.; Du, Q.; Zhang, F.; Hu, W. Collaborative-Representation-Based Nearest Neighbor Classifier for Hyperspectral Imagery. *IEEE Geosci. Remote Sens. Lett.* **2015**, *12*, 389–393. doi:10.1109/LGRS.2014.2343956. [CrossRef]
6. Rankin, B.M.; Meola, J.; Eismann, M.T. Spectral Radiance Modeling and Bayesian Model Averaging for Longwave Infrared Hyperspectral Imagery and Subpixel Target Identification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 6726–6735. doi:10.1109/TGRS.2017.2731955. [CrossRef]
7. Zhang, Y.; Cao, G.; Li, X.; Wang, B. Cascaded Random Forest for Hyperspectral Image Classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2018**, *11*, 1082–1094. doi:10.1109/JSTARS.2018.2809781. [CrossRef]
8. Melgani, F.; Bruzzone, L. Support vector machines for classification of hyperspectral remote-sensing images. *IEEE Trans. Geosci. Remote Sens.* **2004**, *42*, 1778–1790. doi:10.1109/IGARSS.2002.1025088. [CrossRef]
9. Camps-Valls, G.; Bruzzone, L. Kernel-based methods for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2005**, *43*, 1351–1362. doi:10.1109/TGRS.2005.846154. [CrossRef]

10. Chen, Y.; Lin, Z.; Zhao, X.; Wang, G.; Gu, Y. Deep Learning-Based Classification of Hyperspectral Data. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2014**, *7*, 2094–2107. doi:10.1109/JSTARS.2014.2329330. [CrossRef]
11. Li, B.; Dai, Y.; He, M. Monocular depth estimation with hierarchical fusion of dilated CNNs and soft-weighted-sum inference. *Pattern Recognit.* **2018**, *83*, 328–339. doi:10.1016/j.patcog.2018.05.029. [CrossRef]
12. Makantasis, K.; Doulamis, A.D.; Doulamis, N.D.; Nikitakis, A. Tensor-Based Classification Models for Hyperspectral Data Analysis. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 6884–6898. doi:10.1109/TGRS.2018.2845450. [CrossRef]
13. Li, B.; Shen, C.; Dai, Y.; van den Hengel, A.; He, M. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA , 7–12 June 2015 ; pp. 1119–1127. doi:10.1109/CVPR.2015.7298715. [CrossRef]
14. Özdemir, A.O.B.; Gedik, B.E.; Çetin, C.Y.Y. Hyperspectral classification using stacked autoencoders with deep learning. In Proceedings of the 2014 6th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), Lausanne, Switzerland, 24–27 June 2014; pp. 1–4. doi:10.1109/WHISPERS.2014.8077532. [CrossRef]
15. Zhong, P.; Gong, Z.; Li, S.; Schönlieb, C. Learning to Diversify Deep Belief Networks for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 3516–3530. doi:10.1109/TGRS.2017.2675902. [CrossRef]
16. Lee, H.; Kwon, H. Going Deeper With Contextual CNN for Hyperspectral Image Classification. *IEEE Trans. Image Process.* **2017**, *26*, 4843–4855. doi:10.1109/TIP.2017.2725580. [CrossRef]
17. Chen, Y.; Jiang, H.; Li, C.; Jia, X.; Ghamisi, P. Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 6232–6251. doi:10.1109/TGRS.2016.2584107. [CrossRef]
18. Lin, Z.; Chen, Y.; Zhao, X.; Wang, G. Spectral-spatial classification of hyperspectral image using autoencoders. In Proceedings of the 2013 9th International Conference on Information, Communications Signal Processing, Tainan, Taiwan, 10–13 December 2013; pp. 1–5. doi:10.1109/ICICS.2013.6782778. [CrossRef]
19. Chen, Y.; Zhao, X.; Jia, X. Spectral–Spatial Classification of Hyperspectral Data Based on Deep Belief Network. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 2381–2392. doi:10.1109/JSTARS.2015.2388577. [CrossRef]
20. Li, J.; Xi, B.; Li, Y.; Du, Q.; Wang, K. Hyperspectral Classification Based on Texture Feature Enhancement and Deep Belief Networksk. *Remote Sens.* **2018**, *10*, 396. doi:10.3390/rs10030396. [CrossRef]
21. Hu, W.; Huang, Y.; Wei, L.; Zhang, F.; Li, H. Deep Convolutional Neural Networks for Hyperspectral Image Classification. *J. Sens.* **2015**, *2015*, 1–12. doi:10.1155/2015/258619. [CrossRef]
22. Li, J.; Zhao, X.; Li, Y.; Du, Q.; Xi, B.; Hu, J. Classification of Hyperspectral Imagery Using a New Fully Convolutional Neural Network. *IEEE Geosci. Remote Sens. Lett.* **2018**, *15*, 292–296. doi:10.1109/LGRS.2017.2786272. [CrossRef]
23. Huang, G.; Zhu, Q.; Siew, C.K. Extreme learning machine: Theory and applications. *Neurocomputing* **2006**, *70*, 489–501. doi:10.1016/j.neucom.2005.12.126. [CrossRef]
24. Li, J.; Du, Q.; Li, W.; Li, Y. Optimizing extreme learning machine for hyperspectral image classification. *J. Appl. Remote Sens.* **2015**, *9*, 097296. doi:10.1117/1.JRS.9.097296. [CrossRef]
25. Jiang, M.; Cao, F.; Lu, Y. Extreme Learning Machine With Enhanced Composite Feature for Spectral-Spatial Hyperspectral Image Classification. *IEEE Access* **2018**, *6*, 22645–22654. doi:10.1109/ACCESS.2018.2825978. [CrossRef]
26. Li, W.; Chen, C.; Su, H.; Du, Q. Local Binary Patterns and Extreme Learning Machine for Hyperspectral Imagery Classification. *IEEE Trans. Geosci. Remote Sens.* **2015**, *53*, 3681–3693. doi:10.1109/TGRS.2014.2381602. [CrossRef]
27. Zhou, Y.; Peng, J.; Chen, C.L.P. Extreme Learning Machine With Composite Kernels for Hyperspectral Image Classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 2351–2360. doi:10.1109/JSTARS.2014.2359965. [CrossRef]

28. Huang, G.; Zhou, H.; Ding, X.; Zhang, R. Extreme Learning Machine for Regression and Multiclass Classification. *IEEE Trans. Syst. Man Cybern. Part B* **2012**, *42*, 513–529. doi:10.1109/TSMCB.2011.2168604. [CrossRef]
29. Pal, M.; Maxwell, A.E.; Warner, T.A. Kernel-based extreme learning machine for remote-sensing image classification. *Remote Sens. Lett.* **2013**, *4*, 853–862. doi:10.1080/2150704X.2013.805279. [CrossRef]
30. Chen, C.; Li, W.; Su, H.; Liu, K. Spectral-Spatial Classification of Hyperspectral Image Based on Kernel Extreme Learning Machine. *Remote Sens.* **2014**, *6*, 5795–5814. doi:10.3390/rs6065795. [CrossRef]
31. Tang, J.; Deng, C.; Huang, G. Extreme Learning Machine for Multilayer Perceptron. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *27*, 809–821. doi:10.1109/TNNLS.2015.2424995. [CrossRef]
32. Ding, S.; Zhang, N.; Xu, X.; Guo, L.; Zhang, J. Deep Extreme Learning Machine and Its Application in EEG Classification. *Math. Probl. Eng.* **2015**, *2015*, 1–11. doi:10.1155/2015/129021. [CrossRef]
33. Ma, X.; Wang, H.; Geng, J. Spectral–Spatial Classification of Hyperspectral Image Based on Deep Auto-Encoder. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2016**, *9*, 4073–4085. doi:10.1109/JSTARS.2016.2517204. [CrossRef]
34. Gao, W.; Peng, Y. Ideal Kernel-Based Multiple Kernel Learning for Spectral-Spatial Classification of Hyperspectral Image. *IEEE Geosci. Remote Sens. Lett.* **2017**, *14*, 1051–1055. doi:10.1109/LGRS.2017.2695534. [CrossRef]
35. Patra, S.; Bhardwaj, K.; Bruzzone, L. A Spectral-Spatial Multicriteria Active Learning Technique for Hyperspectral Image Classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2017**, *10*, 5213–5227. doi:10.1109/JSTARS.2017.2747600. [CrossRef]
36. Fauvel, M.; Tarabalka, Y.; Benediktsson, J.A.; Chanussot, J.; Tilton, J.C. Advances in Spectral-Spatial Classification of Hyperspectral Images. *Proc. IEEE* **2013**, *101*, 652–675. doi:10.1109/JPROC.2012.2197589. [CrossRef]
37. Kang, X.; Li, S.; Benediktsson, J.A. Spectral–Spatial Hyperspectral Image Classification With Edge-Preserving Filtering. *IEEE Trans. Geosci. Remote Sens.* **2014**, *52*, 2666–2677. doi:10.1109/TGRS.2013.2264508. [CrossRef]
38. He, K.; Sun, J.; Tang, X. Guided Image Filtering. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1397–1409. doi:10.1109/TPAMI.2012.213. [CrossRef]
39. Xu, B.; Wang, N.; Chen, T.; Li, M. Empirical Evaluation of Rectified Activations in Convolutional Network. *Comput. Sci.* **2015**, arXiv:1505.00853.