remote sensing

MDPI

Article

Fine-Grained Classification of Hyperspectral Imagery Based on Deep Learning

Yushi Chen [1,*], Lingbo Huang [1], Lin Zhu [1], Naoto Yokoya [2] and Xiuping Jia [3]

[1] School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin 150001, China; liantianbo@foxmail.com (L.H.); 17s105139@stu.hit.edu.cn (L.Z.)
[2] RIKEN Center for Advanced Intelligence Project, Tokyo 103-0027, Japan; naoto.yokoya@riken.jp
[3] School of Engineering and Information Technology, The University of New South Wales, Canberra, ACT 2600, Australia; x.jia@adfa.edu.au
* Correspondence: chenyushi@hit.edu.cn

Received: 16 October 2019; Accepted: 13 November 2019; Published: 18 November 2019

Abstract: Hyperspectral remote sensing obtains abundant spectral and spatial information of the observed object simultaneously. It is an opportunity to classify hyperspectral imagery (HSI) with a fine-grained manner. In this study, the fine-grained classification of HSI, which contains a large number of classes, is investigated. On one hand, traditional classification methods cannot handle fine-grained classification of HSI well; on the other hand, deep learning methods have shown their powerfulness in fine-grained classification. So, in this paper, deep learning is explored for HSI supervised and semi-supervised fine-grained classification. For supervised HSI fine-grained classification, densely connected convolutional neural network (DenseNet) is explored for accurate classification. Moreover, DenseNet is combined with pre-processing technique (i.e., principal component analysis or auto-encoder) or post-processing technique (i.e., conditional random field) to further improve classification performance. For semi-supervised HSI fine-grained classification, a generative adversarial network (GAN), which includes a discriminative CNN and a generative CNN, is carefully designed. The GAN fully uses the labeled and unlabeled samples to improve classification accuracy. The proposed methods were tested on the Indian Pines data set, which contains 33,3951 samples with 52 classes. The experimental results show that the deep learning-based methods provide great improvements compared with other traditional methods, which demonstrate that deep models have huge potential for HSI fine-grained classification.

Keywords: convolutional neural network (CNN); deep learning; generative adversarial network (GAN); hyperspectral imagery classification; semi-supervised classification

1. Introduction

Hyperspectral imaging obtains data of the observing target with spectral and spatial information simultaneously and has become a useful tool for a wide branch of users. Among the hyperspectral imagery (HSI) processing methods, classification is one of the core techniques, which tries to allocate a specific class to each pixel in the scene. HSI classification is widely-used including urban development, land change monitoring, scene interpretation, and resource management [1].

The data acquisition capability of remote sensing has been largely improved in the recent decades. In the context of hyperspectral remote sensing, varieties of instruments will be available for Earth observation. Those advanced technologies have increased different types of satellites images which have different resolutions in spectral and spatial dimensions. In general, it leads to difficulties and also opportunities for data processing techniques [2].

In common, the data shown in hyperspectral remote sensing have following features: abundant of object labels, a large size of pixels, and high dimensional features. To the best of our knowledge, most of hyperspectral datasets do not contain more than 20 classes individually. How to handle HSI classification with a large number of classes is a challenging task in real applications. In this study, we investigate the classification of Indian Pines dataset which contains 52 classes. As far as we can see, it is the only public dataset with more than 50 classes. Furthermore, for the Indian Pines dataset, there are fine-grained classes which contain more specific and detailed information compared with the traditional coarse definition of classes.

Among the HSI processing techniques, classification is one of the most vibrant topics. There are three types of HSI classification methods: supervised, unsupervised, and semi-supervised ones. Most of the existing HSI classifiers are based on supervised learning methods.

Due to the abundant spectral information of HSI, traditional methods have been focused on spectral classifiers including multinomial logistic regression [3], random forest [4], neural network [5], support vector machine [6,7], sparse representation [8], and deep learning [9–11].

HSI contains both spectral and spatial information. With the help of spatial information, the classification performance can be significantly improved. Therefore, spectral-spatial classifiers are the mainstream of supervised HSI classification [12]. Typical spectral-spatial classification techniques are based on morphological profiles [13,14], multiple kernels [15], and sparse representation [8]. Morphological profile and its extensions extract the spatial features of HSI, and support vector machines (SVMs) or random forests are followed to obtain the final classification map [16,17]. On the other hand, multiple kernel-based methods use different kernels to handle the heterogeneous spectral and spatial features in an efficient manner [18]. In sparse representation-based methods, spatial information is incorporated to formulate spectral-spatial classifiers [8].

The collection of labeled training samples is costly and time-consuming. Therefore, the number of training samples is usually limited in practice. On the other hand, there are many unlabeled samples in the dataset. The semi-supervised classification, which uses the labeled and unlabeled samples together is a promising way to solve the problem of limited training samples [19–21]. A transductive support vector machine has been introduced to classify remote sensing images [22]. The proposed transductive SVM significantly increased the classification accuracy compared to the standard SVM. In [23], the semi-supervised graph-based method, which combined composite kernels, has been developed for hyperspectral image classification. Although the number of publications of semi-supervised classification in the literature is smaller than that of supervised learning, it is very important in remote sensing applications.

The aforementioned supervised and semi-supervised methods do not classify HSI in a "deep" manner. Deep learning-based models have the advantages in feature extraction, which have shown their capability in many research areas computer vision [24], speech recognition [25], machine translation [26], and remote sensing [27,28].

Most of popular deep learning models, including stacked auto-encoder [29,30], deep belief network [12,31], convolutional neural network (CNN) [32–35], and recurrent neural network [36], have been explored for HSI classification. Among the aforementioned deep models, CNN-based methods are widely-used for HSI classification. Similar works for the purpose of extracting spectral-spatial features from pixel have been proposed using a deep CNN in [37]. Li et al. [38] leverage the CNN method to extract pixel-pairs features of HSI following by a majority voting strategy to predict the final classification result. Z. Zhong et al. proposed a 3D-deep network which receives 3D-blocks equipped with spatial and spectral information both from HSI and calculates 3D-convolution kernels for each pixel [39]. In [40], a light 3D-convolution was proposed for extracting the deep spectral-spatial-combined features and the proposed model was less likely to overfitting and easy to train. Furthermore, in [41], band selection was used to select informative and discriminative bands, and then the labeled and unlabeled samples were fed into a 3D-convolutional Auto-Encoder to get the encoded features for semi-supervised HSI classification. In the meanwhile, Romero et al. [34] raised an unsupervised deep CNN to grab

sparse features for the limitation of a small training set. In 2018, generative adversarial network (GAN) was used as a supervised classification method to obtain accurate classification of HSI [42], and in [43] a semi-supervised classification utilizing spectral features based on GAN was proposed.

Although deep learning-based methods have shown their capability in HSI processing, deep learning is still in the early stage for HSI classification. There are many specific classification problems, including few training samples and HSI with a great number of classes, to be solved by deep learning methods.

Fine-grained classification tries to classify data with small diversity of relatively large number of classes. Few methods have been proposed for HSI fine-grained classification. As far as we know, only in [44], SVM was used to classify HSI with a great number of classes. How to classify HSI with volume and variety is an urgent task to be tackled nowadays. Due to the advantages of deep learning, it is necessary to use deep learning methods for HSI fine-grained classification.

Furthermore, due to the difficulty and time-consuming to label samples, labeled training samples are usually limited. It is necessary to use the unlabeled samples, which can be used to improve the classification performance.

Moreover, the processing time is another important factor of practical applications. As we know, HSI classification with lots of training samples is time-consuming. In order to speed up the classification procedure, preprocessing of HSI, which reduces the computational complexity of classification, is usually needed. On one hand, these preprocessing techniques are traditionally performed through spectral dimensionality reduction algorithms (e.g., principal component analysis (PCA) [45] and Auto-Encoder (AE)). On the other hand, the deep learning methods which are the combinations of feature extraction and classification, can reduce the classification time via the feature extraction stage. In [30,46], the new spectral-spatial HSI classification methods based on the deep features extraction using stacked-auto-encoders (SAE) are proposed, which have achieved an effective performance on HSI classification.

Generally speaking, a further refinement process can produce an improved classification output. Considering that, some post-processing methods combining probabilistic graphical models such as MRF and conditional random field (CRF) with CNN have been explored in [47,48]. For example, Liu et al. [49] used CRF to improve the segmentation outputs by explicitly modeling the contextual information between regions. Furthermore, Chen et al. [50] proposed a fully connected Gaussian CRF model with respective unary potentials getting from a CNN instead of using a disconnected approach. And Zheng et al. [51] demonstrated a dense CRF with Gaussian pairwise potentials as a recurrent neural network to improve the low-resolution prediction by a traditional CNN.

In this study, the deep learning-based methods for hyperspectral supervised and semi-supervised fine-grained classification are investigated. With the help of deep learning models, the proposed methods achieve significant improvements in terms of classification accuracy. Besides, compared with traditional methods like SVM-based methods, the deep learning models can reduce the total running time (e.g., training and test time). In more details, the main contributions of this study are summarized as follows.

(1) The deep learning-based methods are explored for supervised and semi-supervised fine-grained classification of HSI for the first time.

(2) Densely connected convolutional neural network (DenseNet) is explored for supervised classification of HSI. Moreover, pre-processing (i.e., PCA and AE) and post-processing (i.e., CRF) techniques are combined with DenseNet to further improve the classification performance.

(3) A Semi-supervised deep model, semi-GAN, is proposed for semi-supervised classification of HSI. The Semi-GAN effectively utilizes the unlabeled samples to improve the classification performance.

(4) The proposed methods are tested on HSI under the condition of limited training samples, and the deep learning models obtain astounding classification performance.

The rest of the paper is organized as follows. Section 2 presents the densely connected CNN for HSI supervised fine-grained classification and Section 3 presents the GAN for HSI semi-supervised

fine-grained classification. The details of experimental results are reported in Section 4. In Section 5, the conclusions and discussions are presented.

2. Densely Connected CNN for HSI Supervised Fine-grained Classification

HSI usually covers a wide range of the observing scene, which means that the data contain complex data distribution and dozens of different classes at the same time. Without effective feature extraction, it is difficult to classify HSI accurately. Deep models, which use multiple processing layers to hierarchically extract the abstract and discriminant features of the inputs, have the potential to handle accurate classification of complex data. In this section, CNNs are explored for HSI classification.

2.1. Deep Learning and Convolutional Neural Network

In general, deep learning-based methods use multiple layers, which are composed by simple but nonlinear layers, to gradually learn semantically meaningful representation of the inputs. By accumulating enough nonlinear layers, complex functions can be learned by a deep model. The deep model starts with raw pixels and ends with abstract features, and the learned discriminant features suppressed the irrelevant variations. This procedure is extremely important for a classification task.

There are many different ways to implement the idea of deep learning and the mainstream implementations include stacked Auto-Encoder, deep belief network, deep CNN, and deep recurrent neural network. Among the popular deep models, CNN is the most widely-used method for image processing due to the advantages of local connections, shared weights, and pooling.

The convolutional operation with nonlinear transform is the core part of a CNN and it is formulated as follows:

$$\boldsymbol{x}_j^l = f\Big(\sum\nolimits_{i=1}^{M} \boldsymbol{x}_i^{l-1} * \boldsymbol{k}_{ij}^l + \boldsymbol{b}_j^l\Big), \tag{1}$$

where matrix $\boldsymbol{x}_i^{l-1}$ is the i-th feature map of the $(l-1)$-th layer, $\boldsymbol{x}_j^l$ is the j-th feature map of the l-th layer, and M is the total number of feature maps. $\boldsymbol{k}_{ij}^l$ is the convolution filter and $\boldsymbol{b}_j^l$ is the corresponding bias. $f(\cdot)$ is a nonlinear transform such as the rectified linear unit (ReLU) and $*$ denotes the convolution operation. All the parameters including weights and biases are determined by back-propagation learning.

The pooling operation merges the semantically similar features into one, which brings invariance to the feature extraction procedure. There are several pooling strategies and the most common pooling operation is max pooling.

By stacking convolution and pooling layers, deep CNN can be established. In the training of deep CNN, there are some problems such as gradient vanishing and weights initialization difficulty. Batch normalization (BN) [52] can stabilize the distributions of layer inputs, which is achieved by injecting additional BN layers with controlling the mean and variance of distributions. In [53], it has been proved that the effectiveness of batch normalization does not lie in reducing so-called internal covariate shift but lies in making the landscape of the corresponding optimization problem significantly more smooth. Let $\mathcal{B} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$ contains a mini-batch of inputs, then BN mechanism can be formulated as follows.

$$\boldsymbol{y}_i = BN_{\gamma,\beta}(\boldsymbol{x}_i) = \gamma \frac{\boldsymbol{x}_i - \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}_i}{\sqrt{\frac{1}{m}\sum_{i=1}^{m}\Big(\boldsymbol{x}_i - \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}_i\Big)^2 + \varepsilon}} + \beta. \tag{2}$$

The normalized result $\boldsymbol{y}_i$ is scaled and shifted by the learnable parameters γ and β. ε is a constant for numerical stability.

This implies that the gradients used in training are more predictive and well-behaved to cope with the gradient vanishing curse. BN is a practical tool in the training of a deep neural network and it usually speeds up the training procedure.

2.2. Densely Connected CNN for HSI Supervised Fine-grained Classification

In this subsection, the proposed DenseNet framework for HSI fine-grained classification is illustrated in Figure 1. From Figure 1, one can see that there are three parts: the data preparation, feature extraction, and classification. In data preparation part, Auto-Encoder is used to condense the information in the spectral domain, and then the neighbors of the pixel to be classified are selected as input. DenseNet, which is used for feature extraction, is the core part of the framework. At last, a softmax classifier is used to obtain the final classification result.

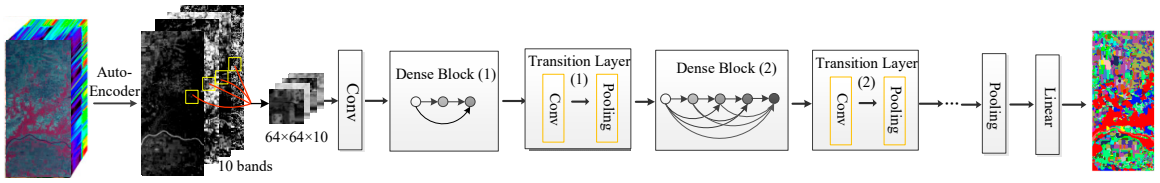


Figure 1. The framework of DenseNet for HSI supervised fine-grained classification.

Traditional CNNs stack the hidden layers to formulate a deep net. Simple stacking of layers leads to serious problems including vanishing gradient and inefficient feature propagation. Although there are some techniques including BN to alleviate the aforementioned problems, the classification performance of CNNs can be further improved by modifying its architecture. DenseNet is a relatively new type of CNN [54]. In DenseNet, each layer obtains additional inputs from all preceding layers. Hence, the l-th layer has l inputs obtained by l connections. This scheme introduces $L(L+1)/2$ connections in an L-layer network, while a traditional CNN of L layers only has L connections. Figure 2 shows the situation when $L = 4$. There are there composite functions, which are denoted by $H_l(\cdot)$, and one transition layer. From the figure, we can see that the total number of connections (colored lines) is 10.

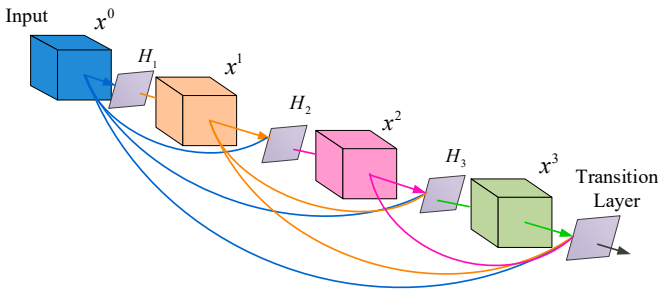


Figure 2. A four-layer dense block.

In dense connection, each layer is connected to all subsequent layers. Let $\boldsymbol{x}^l$ represents the feature maps of the l-th layer, and $\boldsymbol{x}^l$ is obtained through the combination of all previous layers:

$$\boldsymbol{x}^l = H_l\left(\left[\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^{l-1}\right]\right), \tag{3}$$

$\left[\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^{l-1}\right]$ represents the concatenation of previous feature maps produced in layers 0, 1, ... , $l-1$. $H_l(\cdot)$ is a composite function of operations: batch normalization, followed by an activation function (ReLU), and a convolution (Conv).

DenseNet can extract the discriminant features of similar classes, which are useful for fine-grained classification.

2.3. Dimensionality Reduction with DenseNet for HSI Fine-Grained Classification

HSI usually contains hundreds of spectral bands of the same scene, which can provide more abundant spectral information. With the increasing amount of bands, most of the traditional algorithms

dramatically suffer from the curse of dimensionality (i.e., Hughes phenomenon). In this study, two dimensionality reduction methods (i.e., whitening principal component analysis and Auto-Encoder) are combined with DenseNet for HSI fine-grained classification.

The whitening PCA, which is a modified PCA with the identity covariance matrix, is a common way of dimensionality reduction. The PCA can condense the data by reducing the dimensions to a suitable scale. In HSI dimensionality reduction, the whitening PCA is executed to extract the principal information on the spectral dimensions, and then the reduced image is regarded as the input of deep models. Due to PCA, the computational complexity is dramatically reduced, which alleviates the overfitting problem and improves the classification performance.

The Auto-Encoder is another way of dimensionality reduction. Auto-Encoder can non-linearly transform data into a latent space. When this latent space has lower dimension than the original one, this can be viewed as a form of non-linear dimensionality reduction. An Auto-Encoder typically consists of an encoder and a decoder to define the data reconstruction cost. The encoder mapping f adopts the feed-forward process of the neural network to get the embedded feature. However, the decoder mapping g aims to reconstruct the original input. The process can be formulated as:

$$z_i = f(e_i), \tag{4}$$

$$\widetilde{e_i} = g(z_i), \tag{5}$$

where e_i denotes the input pixel vector, $\widetilde{e_i}$ denotes the reconstructed vector, and z_i denotes the corresponding latent vector for classification. The difference between the original input vector and the reconstructed vector is reduced by minimizing the cost function:

$$C = \frac{1}{N}\sum_{i=1}^{N} \|e_i - \widetilde{e_i}\|_2^2, \tag{6}$$

where N is the numbers of pixels of an HSI.

The aforementioned whitening PCA or Auto-Encoder, which is used as a pre-processing technique, can be combined with DenseNet to build an end-to-end system to fulfill the HSI fine-grained classification task.

2.4. CRF with DenseNet for HSI Fine-grained Classification

Different from dimensionality reduction with DenseNet for HSI fine-grained classification, there is another way (post-processing with DenseNet) to improve classification performance. Therefore, in this study, conditional random field (CRF) is combined with DenseNet to further improve the classification accuracy of HSI.

In general, CRFs have been widely used in semantic segmentation based on an initial coarse pixel-level class label, which is predicted by the local interactions of pixels and edges [55,56]. The goal of CRFs is to make pixels in a local neighborhood having the same class label, especially they have been applied to smoothing noisy segmentation maps.

To overcome these limitations of short-range CRFs, we use the fully connected pairwise CRF proposed in [57] for its efficient computation, and ability to capture fine details based on long-range dependencies. In detail, we perform the CRF as a post-processing method on top of the convolutional network, which treats every pixel as a CRF node receiving unary potentials of the CNN and Auto-Encoder-DenseNet.

The fully connected CRF performs the energy function:

$$E(x) = \sum_i \theta_i(x_i) + \sum_{ij} \theta_{ij}\big(x_i, x_j\big), \tag{7}$$

where x is the label assignment for pixels. The unary potential $\theta_i(x_i) = -\log P(x_i)$, where $P(x_i)$ is the label predicted probability at pixel i as computed by convolution network. The pairwise potential uses a fully-connected graph and when we connect all pairs of image pixels, i, j, we get the energy function.

$$\theta_{ij}\left(x_i, x_j\right) = \mu\left(x_i, x_j\right)\left[w_1 \exp\left(-\frac{\|p_i - p_j\|^2}{2\sigma_\alpha^2} - \frac{\|C_i - C_j\|^2}{2\sigma_\beta^2}\right) + w_2 \exp\left(-\frac{\|p_i - p_j\|^2}{2\sigma_\gamma^2}\right)\right], \tag{8}$$

$$\mu\left(x_i, x_j\right) = \begin{cases} 1, & x_i \neq x_j \\ 0, & \text{others} \end{cases}. \tag{9}$$

In this function, one can see that it includes two Gaussian kernels, which stand for different feature spaces, the first kernel based on both pixel positions p and spectral band C, and the second kernel only depends on pixel positions. The scales of Gaussian kernels are decided according to the hyper parameters $\sigma_\alpha, \sigma_\beta$, and σ_γ. The Gaussian CRF potentials in the fully connected CRF model in [57] that we adopt can capture long-range dependencies and at the same time the model is amenable to fast mean field inference. The first kernel impels voxels in an area with similar positions and homogenous spectral band to equip with similar labels, and the second kernel only takes spatial proximity into consideration.

3. Generative Adversarial Networks for HSI Semi-Supervised Fine-grained Classification

The collection of labeled training samples is costly and time-consuming. In addition, there are tremendous unlabeled samples in the dataset. How to effectively combine the labeled and unlabeled samples is an urgent task in remote sensing processing. In this section, a GAN-based semi-supervised classification method is proposed for HSI fine-grained classification.

3.1. Generative Adversarial Network (GAN)

As a novel way to train a generative and discriminative model, GAN which was proposed by Goodfellow [58], has achieved successful development in many fields. Later, various GANs have been proposed like conditional GAN (cGAN) used for image generations [59], SRGAN for super-resolution [60], and image-to-image translation through CycleGAN [61] and DualGAN [62]. Other models were also developed for specific applications including video prediction [63], texture synthesis [64], and natural language processing [65].

Commonly GAN consists of two parts: the generative network G and the discriminative model D. The generator G can obtain the potential distribution of real data and generate a new similar data sample while the discriminator D is a binary classifier that can distinguish the real input samples from the fake samples.

Assuming that the input noise variable possesses a prior $p(z)$ and the real samples are equipped with data distribution $p(x)$. After accepting a random noise z as input, the generator can produce a mapping to data space $G(z)$, where G represents the function of the generative model. Similarly, we can define that D stands for the mapping function of the discriminative model.

In the optimized procedure, the aim of discriminator D is maximizing $log(D(x))$ which is the probability of assigning the correct labels to the correct sources, and the generator G tries to make the generated samples possess more similar distribution with real data, hence we can train the generator G to minimize $log(1 - D(G(z)))$. Therefore, the ultimate aim of the training procedure is to solve the *minimax* problem:

$$\min_G \max_D V(D, G) = E_{x \sim p(x)}[log(D(x))] + E_{z \sim p(z)}[log(1 - D(G(z)))], \tag{10}$$

where E is the expectation operator. However, the shallow multiply perceptrons are usually inferior to deep models in dealing with complex data. Considering that the deep learning-based methods have

achieved many novel implementations in variety of aspects, the deep networks (CNNs) are adopted to compose the model G and D in this paper [66].

3.2. Generative Adversarial Networks for HSI Semi-Supervised Fine-grained Classification

Although GAN has exhibited promising application in image synthesis [67] and many other aspects [61], the discriminative model D of traditional GAN can be used only in distinguishing the real samples from the generated samples, which is not suitable for the multiclass image classification. Recently, the concept of GAN has been extended to be a conditional model with semi-supervised methods where the labels of true training data were imported to the discriminator D.

To adapt GAN to the multiclass HSI classification issue, we need some additional information for both G and D. The introduced information are usually class labels to train the conditional GAN. In this study, the proposed Semi-GAN, whose discriminator D is modified to be a softmax classifier that can output multi-class labels probabilities, can be used for HSI classification. Besides, additional information from the training data with real class labels, the training data equipped with predicted labels are also introduced into the discriminator network. The main framework of Semi-GAN for HSI fine-grained classification is shown in Figure 3.

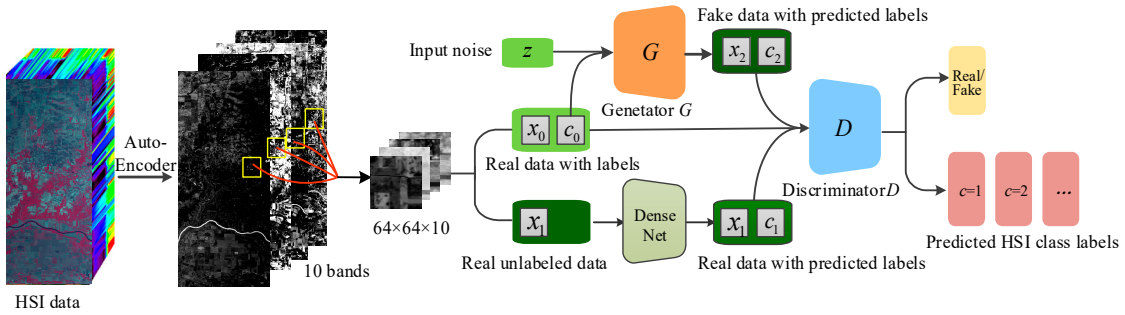


Figure 3. The framework of HSI semi-supervised fine-grained classification.

From Figure 3, one can see that the network can extract spectral and spatial features together. First of all, the HSI data are fed into an Auto-Encoder to obtain the dimensionality-reduced data. In the whole training process, the dimensionality-reduced real data are divided into two parts: one is composed of the labeled samples and the other is unlabeled part. The labeled samples are introduced into both model G and D, while the unlabeled samples are fed into the DenseNet to get the predicted the corresponding labels. The input of the discriminator D consists of real labeled training data, the fake data generated by the generator G and the real unlabeled training data with predicted labels, and then D will output the probability distribution $P(S|X) = D(X)$. Therefore, the ultimate aim of the discriminator D is to maximize the log-likelihood of the right source:

$$L = E[\log P(S = real|X_{real})] + E\Big[\log P\Big(S = fake\Big|X_{fake}\Big)\Big]. \tag{11}$$

Similarly, the aim of the G network is to minimize the log-likelihood of the right source.

In the network, one can see that the real training data are composed of two parts: one is the labeled real data and the other is unlabeled real data with labels predicted by trained DenseNet. The generator G also accepts two parts: the hyperspectral image class labels $c \sim p_c$ and the noise z, the output of G can be defined by $X_{fake} = G(z)$. The probability distribution of sources $P(S|X)$ and the probability distribution of class labels $P(C|X)$ are fed into the network D [68]. Considering the different sources and labels of data, the objective function can be divided into two parts: The log-likelihood of the right source of input data L_S and the log-likelihood of the right class labels L_c:

$$L_S = E[\log P(S = real|X_0)] + E[\log P(S = real|X_1)] + E\Big[\log P\Big(S = fake\Big|X_2\Big)\Big], \tag{12}$$

$$L_c = E[\log P(c = c_0|X_0)] + E[\log P(c = c_1|X_1)] + E[\log P(c = c_2|X_2)], \quad (13)$$

where X_0 and c_0 represent real labeled training samples and its true labels respectively. X_1 and c_1 stand for real unlabeled training samples and the predicted labels obtained by DenseNet, while the X_2 and c_2 signify the generated samples from model G and corresponding labels estimated by model D. In the whole training process, D is trained to maximize $L_S + L_c$, while G is optimized to maximize $L_c - L_S$.

4. Experimental Results

4.1. Data Description and Environmental Setup

In this study, Indian Pines was adopted to validate the proposed methods which contains 333,951 samples with 52 classes. It was a mixed vegetation site over the Indian Pines test area in North-western Indiana.

The dataset was collected by the Airborne Visible/Infrared Imaging Spectrometer over the Purdue University Agronomy farm northwest of West Lafayette and the surrounding area. In this experiment, we used the North-South scene due to the available North-South ground reference map. It was equipped with a size of 1403 × 614 and 220 spectral bands in the wavelength range of 0.4–2.5 µm. The false color image was shown in Figure 4a. In this experiment, fifty-two different land-cover classes (with more than 100 samples) are provided in the ground reference map, as shown in Figure 4b. In this study, the classification accuracy is mainly evaluated using the overall accuracy (OA), average accuracy (AA), and Kappa coefficient (K).

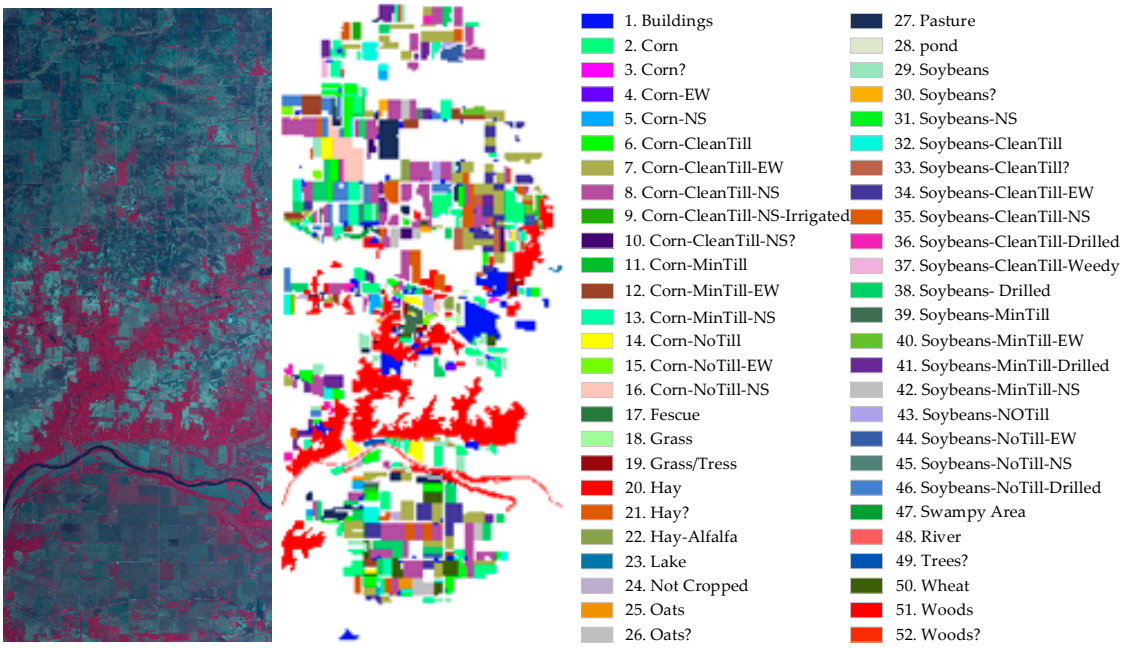


Figure 4. The Indian Pines dataset. (**a**) False-color composite image (Band 40, 25, 10); (**b**) ground reference map.

For the dataset, the labeled samples were divided into two subsets which contain the training and test samples, in the training process of supervised methods, 8000 training samples are used to learn the weights and bias of each neuron. In the test process, 20,000 test samples were used to estimate the performance of the trained network. In the semi-supervised methods, besides the labeled data, the unlabeled data are also used to improve the performance of trained network, 20,000 unlabeled samples are fed into the training process in this experiment. 20,000 samples were used to test the

classification performance. The training and test samples were randomly chosen among the whole samples. In order to obtain reliable results, all the experiments were run five times and the experimental results were given in the form of mean ± standard deviation. The number of training samples for the same class may be different in different running. Table 1 showed the distribution of the 52 classes and the number of training/test samples for each class in two runs (denoted by I or II).

Table 1. The distribution of the 52 classes.

No.	Color	Number of Samples	Number of Training Samples (I)	Number of Training Samples (II)	Number of Test Samples (I)	Number of Test Samples (II)
1		17,195	423	436	1080	1042
2		17,783	428	437	1043	1088
3		158	3	4	6	11
4		514	10	9	41	32
5		2356	53	68	130	139
6		12,404	319	300	677	732
7		26,486	638	612	1600	1556
8		39,678	985	947	2471	2400
9		800	12	16	47	47
10		1728	36	40	105	127
11		1049	29	31	54	63
12		5629	144	149	318	311
13		8862	204	205	550	541
14		4381	114	109	240	249
15		1206	36	43	77	80
16		5685	131	125	358	367
17		114	6	2	10	4
18		1147	29	27	56	78
19		2331	51	71	132	148
20		1128	30	25	53	61
21		2185	49	53	124	138
22		2258	52	51	144	140
23		224	5	7	8	13
24		1940	50	38	116	124
25		1742	42	53	103	106
26		335	7	10	14	18
27		10,386	273	210	634	657
28		102	4	2	6	8
29		9391	220	223	553	597
30		894	23	22	51	45
31		1110	23	26	74	75
32		5074	109	138	293	318
33		2726	45	61	159	166

Table 1. *Cont.*

No.	Color	Number of Samples	Number of Training Samples (I)	Number of Training Samples (II)	Number of Test Samples (I)	Number of Test Samples (II)
34		11,802	249	266	677	707
35		10,387	247	253	660	608
36		2242	64	40	115	126
37		543	20	6	28	23
38		15,118	339	382	904	885
39		2667	49	63	166	159
40		1832	51	54	122	107
41		8098	188	186	460	484
42		4953	128	140	281	295
43		2157	41	55	133	137
44		2533	39	56	120	158
45		929	26	28	49	55
46		8731	221	215	535	498
47		583	11	16	36	30
48		3110	92	69	190	194
49		580	12	14	39	36
50		4979	118	118	281	293
51		63,562	1519	1486	3861	3715
52		144	3	3	16	9
Total number		333,951	8000	8000	20,000	20,000

4.2. HSI Supervised Fine-Grained Classification

In this supervised experiment, the DenseNet was also compared with the traditional CNN used in HSI classification. The training and test samples were randomly chosen among the whole dataset and 8000 labeled samples for all classes were regarded as the training data in view of the large class numbers.

The details of basic DenseNet framework were shown in Table 2. In the Table 2, the DenseNet in the experiment had four dense blocks and three Transition Layers. Each Dense Block had the BN-ReLU-Conv (1×1)-BN-ReLU-Conv (3×3) version of $H_l(\cdot)$. The introduced 1×1 convolution before 3×3 convolution was used to reduce the number of input feature maps, and thus improved computational efficiency. The growth rate k in the experiment was set to 16 which meant that the number of input feature maps in next layer increased by 16 compared with the last layer. The numbers of $H_l(\cdot)$ in four dense blocks are 2, 4, 6, 8, respectively. For the convolutional layer in dense block, each side of the inputs was zero-padded by one pixel to keep the feature-map size fixed. Between two contiguous dense blocks, we used Transition Layer that contained 1×1 convolution followed by 2×2 average pooling to reduce the size of feature maps. At the end of the last dense block, a 7×7 global average pooling was executed, and then a softmax classifier was attached to get the predicted labels. The hyperspectral data after Auto-Encoder were preserved ten principal components in the Auto-Encoder-DenseNet. The classification results with different number of principal components were shown in Table 3. From Table 3, one can see that Auto-Encoder-DenseNet with ten principal components obtained best classification performance. Therefore, we preserved ten principal components in the Auto-Encoder-DenseNet. And similarly, the total channels were condensed to ten dimensions through PCA in the PCA-DenseNet for the comparison.

Table 2. The detailed framework of DenseNet.

Layers	Output Size	DenseNet
Convolution	56 × 56	9 × 9 conv, stride = 1
Dense Block (1)	56 × 56	$\begin{pmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{pmatrix}\times 2$
Transition Layer (1)	56 × 56 28 × 28	1 × 1 conv 2 × 2 average pool, stride = 2
Dense Block (2)	28 × 28	$\begin{pmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{pmatrix}\times 4$
Transition Layer (2)	28 × 28 14 × 14	1 × 1 conv 2 × 2 average pool, stride = 2
Dense Block (3)	14 × 14	$\begin{pmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{pmatrix}\times 6$
Transition Layer (3)	14 × 14 7 × 7	1 × 1 conv 2 × 2 average pool, stride = 2
Dense Block (4)	7 × 7	$\begin{pmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{pmatrix}\times 8$
Classification Layer	1 × 1	7 × 7 global average pool fully-connected, softmax

Table 3. Test accuracy of Auto-Encoder-DenseNet with different numbers of principal components.

Number of Principal Components	5	10	20
OA (%)	91.48 ± 0.42	92.35 ± 0.57	92.23 ± 0.53
AA (%)	86.54 ± 1.76	87.89 ± 2.07	87.44 ± 1.65
K × 100	89.61 ± 0.54	91.30 ± 0.66	91.12 ± 0.79

In the CNN-based methods and DenseNet-based methods, we used $64 \times 64 \times d$ where the d represented the number of spectral bands and $64 \times 64 \times 10$ neighbors of each pixel as the input 3D images without compressing and with compressing, respectively. For the RBF-SVM method, we used a "grid-search" method to find the most appropriate C and γ [69]. In this manner, pairs of (C, γ) were tried and the one with the best classification accuracy on the validation samples was picked. This method was convenient and straightforward. In this experiment, the ranges of C and γ were $\left[10^{-2}, 10^{-1}, \cdots 10^{5}\right]$ and $\left[10^{-4}, 10^{-3}, \cdots 10^{3}\right]$, respectively. Furthermore, in the RF-based methods, we preserved three principal components of hyperspectral data after AE stage, the $27 \times 27 \times 3$ neighbors of each pixel are regarded as the input 3D images. The input images are normalized into the range [−0.5–0.5]. The parameters of deep models were generally selected by trial-and-error. Table 4 showed the detailed parameter settings of deep models. The learning rate was chosen from {0.1, 0.01, 0.005, 0.002, 0.001, 0.0005, 0.0002, 0.0001}. The number of epochs was chosen from {100, 150, 200, 250, 300, 350}. The batch size was chosen from {50, 100, 200, 500, 1000, 5000}. We carefully trained and optimized the involved models for fair comparison.

Table 4. The detailed parameter settings of deep models.

Model	Learning Rate	Number of Epochs	Batch Size
Auto-Encoder	0.001	150	5000
CNN	0.001	150	200
DenseNet	0.001	150	200
Semi-GAN	0.0002	200	200

In this study, L_2 regularization was used as regularization techniques in DenseNet and Semi-GAN (mentioned later). L_2 regularization, which leads the value of weights tend to be smaller, is a common used technique to handle overfitting. In the experiments, the hyperparameter of weight decay was set to 0.0001. Furthermore, the global average pooling (GAP) were used in DenseNet to reduce number of the parameters for alleviating the problem of overfitting. The details about global average pooling can be found in the network structure.

The classification results obtained from different methods were shown in Table 5. Table 5 included the RF-based and original CNN-based methods to give a comprehensive comparison. To exploit spectral-spatial features, the extended morphological profiles with RF (EMP-RF), which is a popular method used in hyperspectral classification, was also performed. In the EMP-RF method, three principal components from HSIs were computed, and then the opening and closing operations were used to extract spatial information on the first three components. The shape structuring element (SE) was set as disk with an increasing size from 1 to 4. Therefore, 24 spatial features were generated. The extracted features were fed to Random Forest [70] to obtain the final classification results. We also used extended multi-attribute profiles (EMAPs), which was an extension of attribute profiles (APs) using different types of attributes [71,72]. For the EMAP-RF method [73], four morphological attributes types (area, diagonal, inertia, and standard deviation) were stacked together and computed for every connected component of a grayscale image. For every attribute, we set four thresholds and executed thinning or thickening operations according to the level between connected component and defined thresholds. For every band obtained from PCA, thus if N was the number of thresholds considered in the analysis, the AP was composed of 2N+1 images. We obtained 99 feature maps in the EMAPs due to the reserved three principal spectral bands and 16 thresholds in whole four attributes. In RF, 50 trees were chosen in the forest to train the samples and predict the labels of test data. In the CRF-based methods, we use the fully connected pairwise CRF proposed by [57] for its efficient computation, and the ability to capture fine details based on long-range dependencies. The CRF is regarded as a post-processing method on top of the convolution network, which treats every pixel as a CRF node receiving unary potentials of the CNN and Auto-Encoder-DenseNet. Furthermore, the original CNN was also used as a benchmark method.

Table 5 showed the classification results with different supervised methods and Table 6 showed the classification results with different preprocessing methods on the Indian Pines dataset. For further comparison, Figure 5 was introduced to illustrate the test accuracy of relevant methods.

From Figure 5, one can see that the OA, AA, and K of different classification methods on Indian Pines dataset were presented. The traditional methods (i.e., SVM, EMP-RF, and EMAP-RF) obtained relatively low classification accuracy compared with deep learning-based methods. For deep CNN-based methods, DenseNet obtained better classification performance compared with classical CNN. Moreover, the combination of DenseNet and pre-processing/post-processing obtained classification accuracy improvement compared with original DenseNet.

For example, OA, AA, and K of CNN were 85.47%, 78.29%, and 0.8286, respectively, improving these accuracies by 33.23%, 35.74%, and 0.3571% over RBF-SVM, respectively. And, DenseNet obtained better performance compared with CNN. On one hand, one can see that the preprocessing methods help improve the classification performance. For example, the Auto-Encoder-DenseNet obtained superior performance on OA, AA, and K, which outperformed the DenseNet by 1.51%, 5.47%, and 0.0103, respectively. In addition, one can see that Auto-Encoder-DenseNet obtained the best classification performance in terms of OA, which showed that the auto-encoder-based methods achieved better performance than PCA-based methods in terms of classification accuracy. On the other hand, the CRF combined with supervised methods achieved superior results compared with those without CRF. For example, the DenseNet-CRF obtained the highest scores in OA, AA, and K, which exceeded the DenseNet by 2.23%, 4.44%, and 0.0249, respectively. It showed that CRF could be used as a post-processing technique for further improving the classification performance of deep models.

Table 5. Test accuracy with different supervised methods on the Indian Pines dataset.

No	Color	SVM	EMP-RF	EMAP-RF	CNN	CNN-CRF	PCA-DenseNet	Auto-Encoder-DenseNet	DenseNet-CRF
1		52.61 ± 4.35	69.71 ± 3.24	60.09 ± 3.24	77.77 ± 2.01	85.54 ± 1.17	84.45 ± 0.49	84.93 ± 2.74	**90.92 ± 0.02**
2		32.68 ± 6.34	71.96 ± 5.69	66.84 ± 5.69	71.40 ± 4.56	85.08 ± 2.43	**94.59 ± 0.82**	89.44 ± 1.78	89.96 ± 0.26
3		18.142 ± 6.12	28.43 ± 0.38	23.05 ± 0.16	72.72 ± 0.04	80.22 ± 0.50	**87.22 ± 9.24**	81.11 ± 2.54	86.67 ± 0.15
4		17.00 ± 9.22	20.26 ± 5.12	49.05 ± 5.12	86.06 ± 5.35	72.05 ± 3.45	87.53 ± 3.54	76.86 ± 3.87	**91.47 ± 0.19**
5		18.91 ± 5.32	60.81 ± 5.76	67.48 ± 6.34	88.39 ± 1.93	83.61 ± 0.12	90.37 ± 1.94	**94.13 ± 4.33**	91.41 ± 0.05
6		28.30 ± 0.03	67.44 ± 4.35	78.24 ± 4.56	82.33 ± 0.45	84.90 ± 0.08	**93.08 ± 0.21**	90.26 ± 3.60	90.79 ± 0.07
7		33.86 ± 2.15	75.58 ± 0.34	72.65 ± 0.08	77.46 ± 2.55	80.06 ± 1.45	93.07 ± 0.25	**94.41 ± 3.05**	92.67 ± 0.43
8		34.25 ± 0.03	82.49 ± 1.04	85.05 ± 0.16	84.84 ± 1.45	84.87 ± 0.05	**95.82 ± 0.35**	94.68 ± 2.24	93.23 ± 0.21
9		9.56 ± 2.54	66.34 ± 3.02	59.15 ± 2.24	87.36 ± 4.45	97.45 ± 0.46	**100.00 ± 0.00**	**100.00 ± 0.00**	96.97 ± 0.19
10		16.10 ± 0.03	44.74 ± 8.43	68.45 ± 3.46	80.82 ± 5.34	90.90 ± 2.45	**96.37 ± 1.48**	96.18 ± 1.45	90.47 ± 0.03
11		19.77 ± 0.13	42.31 ± 1.16	76.37 ± 2.08	72.11 ± 0.17	**89.77 ± 2.56**	87.78 ± 2.46	81.26 ± 3.65	87.02 ± 0.09
12		27.30 ± 6.45	65.46 ± 5.02	76.15 ± 4.35	75.91 ± 3.45	89.94 ± 1.56	**95.05 ± 0.71**	92.90 ± 3.10	93.38 ± 0.16
13		37.87 ± 5.63	74.57 ± 4.53	81.86 ± 0.03	86.37 ± 3.56	91.46 ± 4.32	93.58 ± 0.91	**94.63 ± 2.70**	90.93 ± 0.15
14		36.31 ± 9.46	68.15 ± 3.46	81.58 ± 3.13	80.96 ± 0.08	86.49 ± 0.12	91.52 ± 0.24	88.32 ± 3.92	**92.03 ± 0.23**
15		35.84 ± 1.98	58.33 ± 3.45	66.70 ± 0.21	77.63 ± 2.46	89.23 ± 1.56	90.91 ± 0.45	96.36 ± 2.53	**96.42 ± 0.15**
16		57.14 ± 2.54	70.01 ± 0.42	85.65 ± 0.14	81.69 ± 0.43	83.57 ± 0.20	93.97 ± 0.96	96.97 ± 1.24	**94.54 ± 0.06**
17		39.25 ± 12.53	55.56 ± 9.43	43.25 ± 0.15	73.33 ± 9.30	65.42 ± 0.14	3.82 ± 5.25	76.61 ± 2.14	**91.37 ± 0.42**
18		23.19 ± 9.07	25.39 ± 7.45	34.78 ± 0.25	47.61 ± 0.15	78.72 ± 0.15	65.71 ± 5.81	69.27 ± 3.73	**80.51 ± 0.32**
19		52.65 ± 7.53	66.62 ± 6.26	78.95 ± 2.43	85.81 ± 2.45	85.87 ± 3.56	87.67 ± 1.55	91.17 ± 2.96	**91.28 ± 0.06**
20		48.63 ± 7.15	57.06 ± 4.56	55.58 ± 3.23	73.91 ± 3.45	90.47 ± 3.46	71.20 ± 3.96	**92.23 ± 1.24**	86.74 ± 0.04
21		63.14 ± 0.06	69.24 ± 0.14	82.85 ± 0.02	62.48 ± 0.21	79.62 ± 2.35	90.01 ± 1.87	**95.50 ± 2.37**	92.83 ± 0.12
22		68.35 ± 2.89	68.14 ± 1.35	83.25 ± 2.39	70.94 ± 0.98	79.36 ± 0.24	90.98 ± 1.55	**91.07 ± 3.10**	89.86 ± 0.14
23		61.51 ± 0.24	83.33 ± 0.16	72.57 ± 0.03	92.85 ± 0.01	89.18 ± 0.46	79.36 ± 3.19	28.34 ± 2.45	**100.00 ± 0.000**
24		39.94 ± 1.54	51.69 ± 0.68	71.34 ± 0.35	65.74 ± 0.15	**89.20 ± 0.23**	71.77 ± 3.64	79.79 ± 0.01	86.88 ± 0.16
25		44.63 ± 7.54	58.35 ± 5.45	63.02 ± 5.01	76.54 ± 2.45	82.74 ± 0.34	77.45 ± 7.22	69.13 ± 2.56	75.67 ± 0.24

Table 5. *Cont.*

No	Color	SVM	EMP-RF	EMAP-RF	CNN	CNN-CRF	PCA-DenseNet	Auto-Encoder-DenseNet	DenseNet-CRF
26		33.24 ± 0.05	14.14 ± 0.04	37.35 ± 0.13	35.05 ± 0.25	72.35 ± 0.16	58.40 ± 9.37	54.13 ± 2.72	**81.76 ± 0.32**
27		65.98 ± 4.33	76.01 ± 2.45	83.55 ± 3.09	90.36 ± 3.01	78.37 ± 0.26	91.13 ± 0.89	91.46 ± 1.50	**91.92 ± 0.04**
28		28.33 ± 0.14	25.22 ± 4.64	23.79 ± 13.54	20.49 ± 10.24	26.25 ± 0.19	70.83 ± 1.31	70.79 ± 3.18	**87.78 ± 0.42**
29		43.25 ± 3.46	69.28 ± 10.45	76.58 ± 0.19	73.01 ± 0.87	87.90 ± 1.56	89.48 ± 0.36	90.51 ± 1.07	**90.73 ± 0.13**
30		16.56 ± 2.06	62.32 ± 0.11	69.36 ± 0.04	46.69 ± 2.56	86.06 ± 0.12	90.44 ± 0.97	92.20 ± 0.13	**94.12 ± 0.04**
31		15.54 ± 6.74	70.62 ± 5.39	62.95 ± 6.43	89.41 ± 4.67	49.29 ± 1.45	84.97 ± 2.59	86.90 ± 1.08	**91.91 ± 0.20**
32		29.97 ± 7.46	57.49 ± 6.34	66.75 ± 4.67	79.16 ± 0.36	90.12 ± 2.45	90.47 ± 0.14	**92.92 ± 0.45**	90.82 ± 0.05
33		23.64 ± 6.42	68.59 ± 4.07	77.39 ± 1.56	**91.51 ± 4.57**	82.71 ± 0.25	90.52 ± 2.59	89.55 ± 2.85	88.65 ± 0.06
34		35.00 ± 4.56	74.15 ± 8.31	76.85 ± 6.46	82.73 ± 5.67	86.54 ± 0.43	**92.01 ± 0.14**	91.24 ± 1.27	90.90 ± 0.23
35		24.83 ± 5.03	68.78 ± 4.04	72.75 ± 6.23	87.37 ± 4.92	82.82 ± 0.51	**95.29 ± 0.52**	95.35 ± 0.15	91.35 ± 0.11
36		47.15 ± 0.05	53.33 ± 0.06	74.55 ± 0.14	91.26 ± 0.45	86.71 ± 1.56	94.15 ± 1.80	85.65 ± 0.07	**96.63 ± 0.39**
37		26.99 ± 5.43	54.77 ± 4.56	51.45 ± 6.43	81.42 ± 1.45	90.20 ± 0.32	92.47 ± 0.02	94.01 ± 0.04	**98.96 ± 0.06**
38		40.86 ± 6.42	72.75 ± 3.64	69.75 ± 2.34	74.94 ± 2.45	89.96 ± 0.12	90.70 ± 0.49	**92.95 ± 0.01**	86.10 ± 0.24
39		54.55 ± 4.56	68.54 ± 2.54	61.24 ± 2.45	88.53 ± 1.46	73.64 ± 1.57	90.18 ± 0.37	**91.61 ± 2.64**	89.91 ± 0.08
40		50.53 ± 2.03	70.01 ± 0.74	62.99 ± 3.45	86.42 ± 1.90	82.37 ± 0.31	94.70 ± 1.63	**97.26 ± 3.14**	85.82 ± 0.25
41		49.15 ± 6.64	71.46 ± 3.45	86.25 ± 0.13	89.72 ± 3.57	82.24 ± 0.34	**96.32 ± 0.94**	94.33 ± 0.13	90.99 ± 0.06
42		31.88 ± 4.43	65.17 ± 2.15	78.95 ± 1.45	84.82 ± 0.03	84.94 ± 0,56	**97.46 ± 0.37**	95.84 ± 3.25	91.01 ± 0.05
43		43.98 ± 9.46	65.55 ± 5.64	78.84 ± 0.09	85.49 ± 4.67	82.36 ± 0.45	86.02 ± 0.81	92.61 ± 2.14	**94.90 ± 0.12**
44		41.55 ± 3.56	82.38 ± 0.04	60.65 ± 0.06	85.12 ± 2.54	91.44 ± 0.10	88.62 ± 1.83	96.16 ± 2.06	**97.78 ± 0.10**
45		47.40 ± 3.64	59.18 ± 0.04	81.02 ± 3.42	72.66 ± 1.36	79.33 ± 0.42	69.66 ± 6.45	**88.13 ± 1.02**	78.03 ± 0.51
46		61.53 ± 0.04	71.59 ± 0.56	87.55 ± 3.46	89.08 ± 0.67	82.66 ± 0.32	92.61 ± 0.26	91.35 ± 0.45	**93.51 ± 0.05**
47		61.05 ± 1.97	40.18 ± 0.04	85.35 ± 0.32	86.04 ± 0.14	81.18 ± 0.24	**94.75 ± 1.51**	79.89 ± 4.88	91.30 ± 0.24
48		69.26 ± 4.56	99.49 ± 3.53	81.97 ± 0.36	95.06 ± 2.56	88.41 ± 0.42	96.40 ± 1.19	**98.26 ± 0.88**	95.08 ± 0.04
49		53.92 ± 0.14	79.82 ± 0.08	79.76 ± 0.57	78.37 ± 0.17	84.47 ± 3.46	79.32 ± 4.19	**100.00 ± 0.00**	95.99 ± 0.22
50		78.48 ± 4.56	82.81 ± 4.34	65.86 ± 2.44	92.16 ± 2.45	83.47 ± 2.45	90.40 ± 2.49	**93.02 ± 0.52**	91.70 ± 0.34
51		72.48 ± 4.64	96.95 ± 1.45	81.44 ± 0.97	96.26 ± 1.45	84.45 ± 0.46	**97.96 ± 0.19**	97.35 ± 0.21	96.58 ± 0.10
52		83.28 ± 1.45	61.71 ± 3.45	43.25 ± 0.68	77.50 ± 0.14	85.01 ± 0.21	**100.00 ± 0.00**	79.58 ± 3.43	91.67 ± 0.12
	OA (%)	52.24 ± 0.43	76.36 ± 0.24	77.95 ± 0.57	85.47 ± 0.28	86.08 ± 0.21	92.08 ± 0.87	92.35 ± 0.57	**93.07 ± 0.18**
	AA (%)	42.55 ± 0.16	64.79 ± 0.39	66.05 ± 0.12	78.29 ± 0.23	82.53 ± 0.86	87.01 ± 1.25	87.89 ± 2.07	**91.45 ± 0.31**
	K × 100	47.15 ± 0.20	72.24 ± 0.35	73.72 ± 0.66	82.86 ± 0.13	85.64 ± 0.27	90.26 ± 0.94	91.30 ± 0.66	**92.76 ± 0.19**

Table 6. Test accuracy with different preprocessing methods on the Indian Pines dataset.

NO.	Color	CNN	PCA-CNN	Auto-Encoder-CNN	DenseNet	PCA-DenseNet	DenseNet-1 × 1 Conv	Auto-Encoder-DenseNet
1		77.77 ± 2.01	77.69 ± 1.09	**86.18 ± 0.45**	83.32 ± 1.25	84.45 ± 0.49	82.45 ± 1.59	84.93 ± 2.74
2		71.40 ± 4.56	84.91 ± 0.42	77.87 ± 0.78	93.16 ± 0.43	**94.59 ± 0.82**	93.28 ± 1.26	89.44 ± 1.78
3		72.72 ± 0.04	72.40 ± 7.09	**94.56 ± 0.23**	68.58 ± 15.14	87.22 ± 9.24	66.59 ± 2.15	81.11 ± 2.54
4		86.06 ± 5.35	90.87 ± 5.19	80.57 ± 2.53	67.79 ± 5.82	**87.53 ± 3.54**	75.86 ± 3.19	76.86 ± 3.87
5		88.39 ± 1.93	94.02 ± 0.15	78.33 ± 0.66	88.62 ± 1.39	90.37 ± 1.94	81.19 ± 2.15	**94.13 ± 4.33**
6		82.33 ± 0.45	85.16 ± 0.60	85.61 ± 0.60	91.65 ± 2.05	**93.08 ± 0.21**	92.45 ± 0.43	90.26 ± 3.60
7		77.46 ± 2.55	89.57 ± 1.12	87.82 ± 0.16	93.63 ± 0.43	93.07 ± 0.25	93.79 ± 0.21	**94.41 ± 3.05**
8		84.84 ± 1.45	93.75 ± 0.66	94.47 ± 0.66	95.61 ± 0.42	95.82 ± 0.35	**95.69 ± 0.19**	94.68 ± 2.24
9		87.36 ± 4.45	**100.00 ± 0.00**	91.00 ± 0.12	96.98 ± 1.69	**100.00 ± 0.00**	96.46 ± 1.68	**100.00 ± 0.00**
10		80.82 ± 5.34	95.11 ± 2.25	93.11 ± 0.79	95.79 ± 0.89	**96.37 ± 1.48**	90.36 ± 0.09	96.18 ± 1.45
11		72.11 ± 0.17	75.95 ± 1.25	90.76 ± 1.25	**94.58 ± 0.80**	87.78 ± 2.46	88.04 ± 0.16	81.26 ± 3.65
12		75.91 ± 3.45	85.67 ± 1.54	88.00 ± 0.76	93.59 ± 0.72	**95.05 ± 0.71**	93.18 ± 0.15	92.90 ± 3.10
13		86.37 ± 3.56	94.41 ± 0.73	94.46 ± 0.11	**97.06 ± 0.32**	93.58 ± 0.91	92.43 ± 0.23	94.63 ± 2.70
14		80.96 ± 0.08	88.89 ± 1.56	91.69 ± 0.02	88.36 ± 1.50	91.52 ± 0.24	**94.73 ± 0.15**	88.32 ± 3.92
15		77.63 ± 2.46	95.19 ± 1.44	**98.58 ± 0.02**	85.33 ± 2.11	90.91 ± 0.45	96.47 ± 3.58	96.41 ± 2.53
16		81.69 ± 0.43	98.49 ± 0.38	**97.11 ± 0.69**	96.71 ± 0.68	93.97 ± 0.96	95.96 ± 0.42	96.97 ± 1.24
17		73.33 ± 9.30	60.02 ± 13.94	40.00 ± 13.34	32.75 ± 10.25	3.82 ± 5.25	**98.67 ± 0.32**	76.61 ± 2.14
18		47.61 ± 0.15	63.06 ± 2.51	64.76 ± 2.54	71.55 ± 6.27	65.71 ± 5.81	**75.26 ± 4.23**	69.27 ± 3.73
19		85.81 ± 2.45	93.22 ± 1.61	72.99 ± 2.03	83.52 ± 1.97	87.67 ± 1.55	**94.18 ± 2.41**	91.28 ± 2.96
20		73.91 ± 3.45	52.05 ± 3.62	71.06 ± 0.11	73.99 ± 3.65	71.20 ± 3.96	79.58 ± 1.12	**92.23 ± 1.24**
21		62.48 ± 0.21	93.11 ± 1.14	83.98 ± 0.22	71.94 ± 2.36	90.01 ± 1.87	90.14 ± 0.14	**95.50 ± 2.37**
22		70.94 ± 0.98	86.22 ± 1.86	83.45 ± 0.45	**93.12 ± 1.11**	90.98 ± 1.55	83.54 ± 2.59	91.07 ± 3.10
23		92.85 ± 0.01	77.19 ± 11.41	**100.00 ± 0.00**	96.29 ± 4.29	79.36 ± 3.19	90.91 ± 0.16	28.34 ± 2.45
24		65.74 ± 0.15	78.47 ± 4.27	69.81 ± 2.27	70.66 ± 2.83	71.77 ± 3.64	67.53 ± 0.24	**79.79 ± 0.01**
25		76.54 ± 2.45	75.59 ± 1.24	73.32 ± 1.89	70.16 ± 1.70	**77.45 ± 7.22**	71.69 ± 0.32	69.13 ± 2.56
26		35.05 ± 0.25	54.55 ± 5.86	**64.29 ± 4.77**	42.20 ± 7.19	58.40 ± 9.37	66.25 ± 0.04	54.13 ± 2.72
27		90.36 ± 3.01	93.01 ± 0.8	92.01 ± 0.17	**93.55 ± 0.98**	91.13 ± 0.89	93.59 ± 0.42	91.46 ± 1.50

Table 6. *Cont.*

NO.	Color	CNN	PCA-CNN	Auto-Encoder-CNN	DenseNet	PCA-DenseNet	DenseNet-1 × 1 Conv	Auto-Encoder-DenseNet
28		20.49 ± 10.24	20.42 ± 16.4	82.50 ± 1.67	58.45 ± 6.91	70.83 ± 1.31	13.25 ± 8.59	70.79 ± 3.18
29		73.01 ± 0.87	**83.14 ± 1.61**	84.35 ± 0.87	83.44 ± 1.18	89.48 ± 0.36	83.25 ± 0.13	90.51 ± 1.07
30		46.69 ± 2.56	75.73 ± 2.12	**92.82 ± 1.67**	86.65 ± 6.32	90.44 ± 0.97	71.25 ± 0.04	92.20 ± 0.13
31		89.41 ± 4.67	94.56 ± 1.60	74.56 ± 1.76	81.53 ± 4.05	84.97 ± 2.59	75.24 ± 0.20	**86.90 ± 1.08**
32		79.16 ± 0.36	87.03 ± 0.67	75.53 ± 0.67	**95.10 ± 0.73**	90.47 ± 0.14	83.02 ± 0.96	92.92 ± 0.45
33		91.51 ± 4.57	86.67 ± 1.16	89.88 ± 1.28	85.94 ± 4.12	**90.52 ± 2.59**	80.15 ± 0.06	89.55 ± 2.85
34		82.73 ± 5.67	85.51 ± 0.67	91.02 ± 0.67	85.60 ± 1.22	**92.01 ± 0.14**	87.25 ± 0.23	91.24 ± 1.27
35		87.37 ± 4.92	85.16 ± 1.20	79.61 ± 0.30	93.39 ± 0.28	95.29 ± 0.52	76.59 ± 0.11	**95.35 ± 0.15**
36		91.26 ± 0.45	87.39 ± 1.65	91.02 ± 1.45	91.42 ± 1.15	**94.15 ± 1.80**	91.03 ± 0.39	85.65 ± 0.07
37		81.42 ± 1.45	94.80 ± 4.57	97.58 ± 1.62	95.22 ± 3.51	92.47 ± 0.02	95.13 ± 0.06	**94.01 ± 0.04**
38		74.94 ± 2.45	86.04 ± 1.51	85.93 ± 0.30	88.88 ± 0.37	90.70 ± 0.49	85.69 ± 0.24	**92.95 ± 0.01**
39		88.53 ± 1.46	88.68 ± 1.32	91.07 ± 1.32	**92.27 ± 1.30**	90.18 ± 0.37	83.06 ± 0.08	91.61 ± 2.64
40		86.42 ± 1.90	84.69 ± 0.87	89.06 ± 0.34	92.10 ± 0.93	94.70 ± 1.63	72.28 ± 0.25	**97.26 ± 3.14**
41		89.72 ± 3.57	91.47 ± 0.48	89.86 ± 1.06	91.44 ± 0.31	**96.32 ± 0.94**	93.27 ± 1.04	94.33 ± 0.13
42		84.82 ± 0.03	90.37 ± 1.97	92.32 ± 0.09	95.59 ± 0.89	**97.46 ± 0.37**	95.27 ± 2.45	95.84 ± 3.25
43		85.49 ± 4.67	89.61 ± 2.24	87.67 ± 2.21	85.54 ± 1.72	86.02 ± 0.81	**94.39 ± 0.12**	92.61 ± 2.14
44		85.12 ± 2.54	97.45 ± 0.17	92.50 ± 0.46	**97.99 ± 0.09**	88.62 ± 1.83	95.52 ± 0.10	96.16 ± 2.06
45		72.66 ± 1.36	**89.25 ± 0.46**	86.44 ± 0.88	79.09 ± 2.45	69.66 ± 6.45	92.59 ± 0.51	88.13 ± 1.02
46		89.08 ± 0.67	91.75 ± 0.88	87.45 ± 0.49	92.50 ± 0.35	**92.61 ± 0.26**	92.10 ± 0.05	91.35 ± 0.45
47		86.04 ± 0.14	91.07 ± 0.74	94.48 ± 4.03	**96.34 ± 2.51**	94.75 ± 1.51	83.32 ± 0.24	79.89 ± 4.88
48		95.06 ± 2.56	98.08 ± 0.29	98.89 ± 0.05	**98.91 ± 0.48**	96.40 ± 1.19	94.09 ± 3.47	98.26 ± 0.88
49		78.37 ± 0.17	90.01 ± 1.45	92.00 ± 0.13	96.09 ± 2.94	79.32 ± 4.19	97.26 ± 0.22	**100.00 ± 0.00**
50		92.16 ± 2.45	89.11 ± 0.87	92.13 ± 0.10	87.92 ± 1.91	90.40 ± 2.49	92.14 ± 0.34	93.02 ± 0.52
51		96.26 ± 1.45	98.22 ± 0.14	98.22 ± 0.14	**98.62 ± 0.03**	97.96 ± 0.19	96.58 ± 1.47	97.35 ± 0.21
52		77.50 ± 0.14	75.68 ± 4.23	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	95.67 ± 1.89	79.58 ± 3.43
OA (%)		85.47 ± 0.28	87.95 ± 0.18	88.08 ± 0.29	90.84 ± 1.04	92.08 ± 0.87	91.17 ± 0.67	**92.35 ± 0.57**
AA (%)		78.29 ± 0.23	84.43 ± 0.41	86.13 ± 0.59	85.98 ± 2.38	87.01 ± 1.25	86.21 ± 1.14	**87.89 ± 2.07**
K × 100		82.86 ± 0.13	86.44 ± 0.39	86.62 ± 0.11	90.27 ± 1.42	90.26 ± 0.94	90.54 ± 0.24	**91.30 ± 0.66**
Train Time (min.)		457.25	40.12	143.20	785.64	188.16	544.63	284.71
Test Time (min.)		23.55	0.51	0.58	28.64	2.05	7.13	2.15

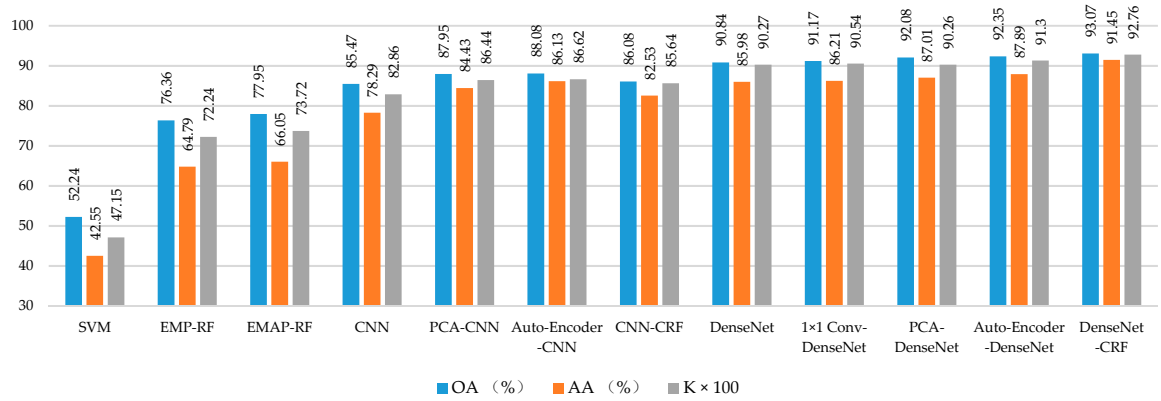


Figure 5. Test accuracy of different supervised methods on the Indian Pines dataset.

In general, a deeper network may have superior performance compared with a shallow network due to larger numbers of composition of non-linear operations. Whereas arbitrarily increases the depth of network cannot bring the benefit, it may deteriorate the generalization abilities of network and cause the overfitting phenomenon.

To evaluate the sensitivity of DenseNet over depth, we performed several experiments based on different network depths, the classification results, and cost time for five repeated experiments were shown in Table 7. In these experiments, the depth of DenseNet is controlled by the numbers of composite function $H_l(\cdot)$ in four dense blocks (i.e., (1, 2, 3, 4) and (2, 4, 6, 8)). The parameter (1, 2, 3, 4) means that the numbers of aforementioned $H_l(\cdot)$ in four dense blocks are 1, 2, 3, and 4, respectively. If we regard each $H_l(\cdot)$ as a composite layer, then it possesses 1 + 2 + 3 + 4 = 10 layers, and for the parameters (2, 4, 6, 8), it possesses 2 + 4 + 6 + 8 = 20 layers.

Table 7. Test accuracy of DenseNet over different depth in Indian Pines dataset.

	DenseNet (1, 2, 3, 4)	**DenseNet (2, 4, 6, 8)**	**DenseNet (3, 5, 7, 9)**
OA (%)	90.97 ± 0.40	92.08 ± 0.87	91.12 ± 0.51
AA (%)	85.63 ± 1.09	87.01 ± 1.25	86.21 ± 0.90
K × 100	89.75 ± 0.43	90.26 ± 0.94	89.13 ± 0.62
Train Time (min.)	117.74	188.16	224.86
Test Time (min.)	1.41	2.05	2.62

From the results, one can see that the classification accuracy firstly increased and then decreased with the growth of network depth increasingly. It demonstrated that adding the depth of DenseNet suitably could boost its superior ability. While too deep architecture may lead to the overfitting phenomenon. This can deteriorate the generalization abilities of network, which is a reason for descending of classification accuracy.

4.3. HSI Semi-Supervised Fine-grained Classification

From aforementioned methods, one can see that the supervised method usually requires a large number of labeled samples for training to learn its parameters. However, the labeled samples are commonly very limited for the real remote sensing application, due to the high labeling cost, the semi-supervised methods, which exploited both labeled and unlabeled samples, have been widely utilized to increase the accuracy and robustness of class predictions.

In this experiment, other semi-supervised classification methods including transductive SVM (TSVM) and Label Propagation were also executed to make a comprehensive comparison with the proposed Semi-GAN method. In TSVM, we used n-cross-validation method to execute model selection, and considering a multiclass problem defined by a set $C = \{C_1, C_2, \cdots C_N\}$ made up of N class labels,

originally the transductive process of TSVM was based on a structured architecture made up of binary classifiers [22], which was not proper for multiclass classification of unlabeled samples. In this experiment, a one-against-all multiclass strategy that involved a parallel architecture consisting of N different TSVMs, was adopted. The training and test data were chosen randomly among the whole dataset and with the assumption that there is at least one sample for each class. To assess the effectiveness of TSVM, the chosen test samples were regarded as unlabeled samples.

However, these samples have not been used for the model selection due to the assumption that the labels are unavailable. In the graph-based method like Label Propagation, we used a RBF kernel to construct a graph, and the clamping factor α was set to be 0.2, which represented that the 80 percent of original label distribution was always reserved and it changed the confidence of the distribution within 20 percent [23]. This method iterated on a modified version of the original graph and normalized the edge weights by computing the normalized graph Laplacian matrix, besides, it minimized a loss function that has regularization properties to make classification performance robust against noise.

In the proposed Semi-GAN, firstly, the real training samples from the dataset, which used 10 principal components through PCA were divided into the labeled part and unlabeled part, and the real labeled samples were introduced into both the models G and D, while the unlabeled samples were fed into the DenseNet to get the predicted corresponding labels. The size of input noise to the model G is $128 \times 1 \times 1$, and the model D converted the inputs to fake samples with the size of $64 \times 64 \times 10$. The data received by the discriminator D come from three sections: real labeled training data, the fake data generated by the generator G, and the real unlabeled training data with predicted labels. Besides, the label smoothing, a technique that replaced the 0 and 1 targets for a classifier with smoothed values with 0.2 and 0.8, was adopted to reduce the vulnerability of neural networks [74] in this paper. The experiment arrangement and details of the models G and D architectures were set like [42].

The classification results obtained from different semi-supervised approaches were listed in Table 8. From Table 8, one can see that the proposed Semi-GAN obtained the best performance compared with Label Propagation and TSVM. The OA, AA, and K of our approach were 94.02%, 90.11%, and 0.9276%, which are higher 34%, 38.91%, and 0.3359% than Label Propagation, respectively. Furthermore, the Label Propagation possessed a superior capacity than TSVM in coping with complex data. The OA, AA, and K of the Label Propagation were higher than those of TSVM by 5.53%, 9.18%, and 0.0848%, respectively.

Moreover, to explore the capacity of our Semi-GAN, several different experiments with progressively reduced training data are performed. Here, the number of labeled training samples fed into Semi-GAN to train the network, which decreased gradually, is represented by N. Moreover, N was set to 4000, 6000, and 8000 in this experiment, and the different classification accuracy results with different training data numbers were shown in Table 9. Besides, the number of unlabeled samples regarded as the training dataset of Semi-GAN is all set to 20,000 in these experiments.

From the results, one can see that as the number of labeled training samples decreased, the performance of Semi-GAN also deteriorated gradually. For example, the network with 8000 labeled samples and 20,000 unlabeled samples obtained the highest scores in OA, AA, and K, with which exceeded 6000 training samples by 1.49%, 1.97%, and 0.0101, respectively, in addition, they also outperformed 4000 training samples by 4.57%, 6.70%, and 0.05, respectively.

Table 8. Test accuracy with different semi-supervised methods on the Indian Pines dataset.

No.	Name	TSVM	Label Propagation	Semi-GAN
1	Buildings	54.48 ± 2.25	53.02 ± 2.32	**82.15 ± 1.45**
2	Corn	29.98 ± 0.43	39.96 ± 1.47	**86.92 ± 2.29**
3	Corn?	21.42 ± 8.56	22.02 ± 6.16	**91.67 ± 0.59**
4	Corn-EW	22.58 ± 6.82	44.67 ± 2.14	**100.00 ± 0.00**
5	Corn-NS	23.73 ± 2.39	33.33 ± 4.52	**95.24 ± 0.45**
6	Corn-CleanTill	25.58 ± 1.05	37.67 ± 2.59	**96.40 ± 0.03**
7	Corn-CleanTill-EW	44.21 ± 0.94	53.53 ± 3.62	**96.38 ± 0.57**
8	Corn-CleanTill-NS	66.54 ± 1.69	66.64 ± 0.02	**91.15 ± 1.43**
9	Corn-CleanTill-NS-Irrigated	6.12 ± 12.34	13.13 ± 9.54	**83.44 ± 2.46**
10	Corn-CleanTill-NS?	18.68 ± 1.34	26.24 ± 3.48	**84.97 ± 2.16**
11	Corn-MinTill	15.25 ± 0.13	32.36 ± 2.49	**100.00 ± 0.00**
12	Corn-MinTill-EW	27.65 ± 0.68	37.56 ± 3.49	**81.23 ± 0.13**
13	Corn-MinTill-NS	39.27 ± 0.72	49.34 ± 4.52	**90.54 ± 0.01**
14	Corn-NoTill	36.74 ± 1.97	48.21 ± 1.49	**95.73 ± 0.45**
15	Corn-NoTill-EW	37.36 ± 2.43	34.63 ± 3.21	**91.26 ± 0.98**
16	Corn-NoTill-NS	48.54 ± 0.68	63.52 ± 1.78	**83.33 ± 4.07**
17	Fescue	77.78 ± 5.62	79.00 ± 0.96	**90.43 ± 1.45**
18	Grass	20.69 ± 6.27	58.02 ± 4.78	**100.00 ± 0.00**
19	Grass/Tress	72.67 ± 1.97	74.67 ± 1.78	**90.30 ± 2.45**
20	Hay	46.38 ± 3.65	55.67 ± 0.79	**66.67 ± 0.15**
21	Hay?	69.23 ± 2.36	79.00 ± 6.23	**90.91 ± 0.63**
22	Hay-Alfalfa	77.27 ± 1.12	82.33 ± 0.25	**85.71 ± 0.57**
23	Lake	54.54 ± 3.29	63.14 ± 0.89	**100.00 ± 0.00**
24	NotCropped	38.18 ± 1.41	56.01 ± 0.21	**66.91 ± 0.34**
25	Oats	48.45 ± 4.29	43.94 ± 0.78	**91.91 ± 4.57**
26	Oats?	7.69 ± 2.83	4.34 ± 4.21	**57.62 ± 0.45**
27	Pasture	64.98 ± 1.07	75.00 ± 0.12	**92.26 ± 2.35**
28	pond	25.12 ± 7.21	40.14 ± 3.69	**67.54 ± 0.25**
29	Soybeans	40.25 ± 0.98	46.24 ± 0.36	**98.81 ± 0.42**
30	Soybeans?	20.23 ± 3.25	11.23 ± 4.56	**80.79 ± 0.94**
31	Soybeans-NS	19.64 ± 4.01	34.45 ± 7.52	**89.48 ± 2.42**
32	Soybeans-CleanTill	31.16 ± 2.73	37.54 ± 0.14	**95.16 ± 0.35**
33	Soybeans-CleanTill?	22.59 ± 1.22	32.45 ± 4.96	**80.95 ± 1.47**
34	Soybeans-CleanTill-EW	36.84 ± 0.85	44.33 ± 0.17	**92.85 ± 0.45**
35	Soybeans-CleanTill-NS	27.55 ± 1.15	27.24 ± 0.02	**94.36 ± 0.56**
36	Soybeans-CleanTill-Drilled	52.50 ± 3.50	46.00 ± 2.14	**84.48 ± 0.35**
37	Soybeans-CleanTill-Weedy	28.99 ± 0.98	23.67 ± 3.41	**93.33 ± 1.27**
38	Soybeans- Drilled	40.02 ± 1.30	52.32 ± 0.79	**98.49 ± 1.45**
39	Soybeans-MinTill	55.81 ± 0.93	65.00 ± 2.14	**89.04 ± 0.92**
40	Soybeans-MinTill-EW	53.84 ± 0.31	63.46 ± 3.78	**99.01 ± 0.17**
41	Soybeans-MinTill-Drilled	50.10 ± 0.89	50.36 ± 4.69	**91.26 ± 2.45**
42	Soybeans-MinTill-NS	31.10 ± 1.72	37.33 ± 0.05	**100.00 ± 0.00**
43	Soybeans-NOTill	49.62 ± 0.09	45.10 ± 0.03	**86.45 ± 4.14**
44	Soybeans-NoTill-EW	44.38 ± 2.45	47.33 ± 0.16	**100.00 ± 0.00**
45	Soybeans-NoTill-NS	16.20 ± 0.35	27.44 ± 0.79	**98.16 ± 0.25**
46	Soybeans-NoTill-Drilled	59.09 ± 2.51	70.00 ± 6.35	**95.48 ± 0.03**
47	Swampy Area	78.57 ± 0.47	**94.38 ± 1.45**	72.37 ± 0.11
48	River	98.94 ± 2.94	**99.37 ± 1.79**	88.12 ± 3.25
49	Trees?	48.57 ± 3.94	59.40 ± 0.17	**94.78 ± 2.45**
50	Wheat	78.87 ± 4.58	86.60 ± 4.23	**100.00 ± 0.00**
51	Woods	**92.03 ± 1.94**	90.23 ± 7.45	84.32 ± 0.01
52	Woods?	90.90 ± 1.59	**91.96 ± 6.32**	82.50 ± 3.45
	OA (%)	55.49 ± 0.87	60.02 ± 0.21	**94.02 ± 1.43**
	AA (%)	43.02 ± 1.04	51.20 ± 0.43	**90.11 ± 2.07**
	K × 100	50.38 ± 0.75	56.86 ± 0.24	**92.76 ± 1.03**

Table 9. Test accuracy with different training samples numbers on Indian Pines dataset.

Methods	N	4000	6000	8000
Semi-GAN	OA (%)	89.45 ± 3.02	92.53 ± 2.98	94.02 ± 2.43
	AA (%)	83.41 ± 3.87	88.14 ± 3.07	90.11 ± 2.57
	K × 100	87.76 ± 2.75	91.75 ± 2.09	92.76 ± 1.56

4.4. Limited Training Samples and Classification Maps

In this experiment, in order to make a comprehensive comparison between different supervised and semi-supervised methods, respectively, we calculated the results of OA when the number of training samples was changed. The results of supervised and semi-supervised methods were shown in Figures 6 and 7, respectively.

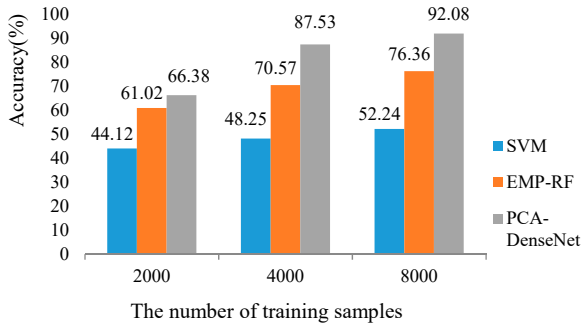


Figure 6. Test accuracy of different supervised methods with changed number of training samples.

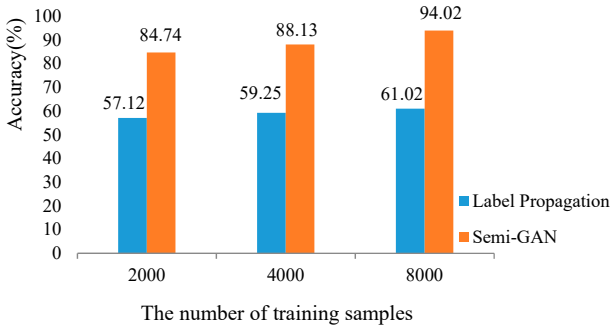
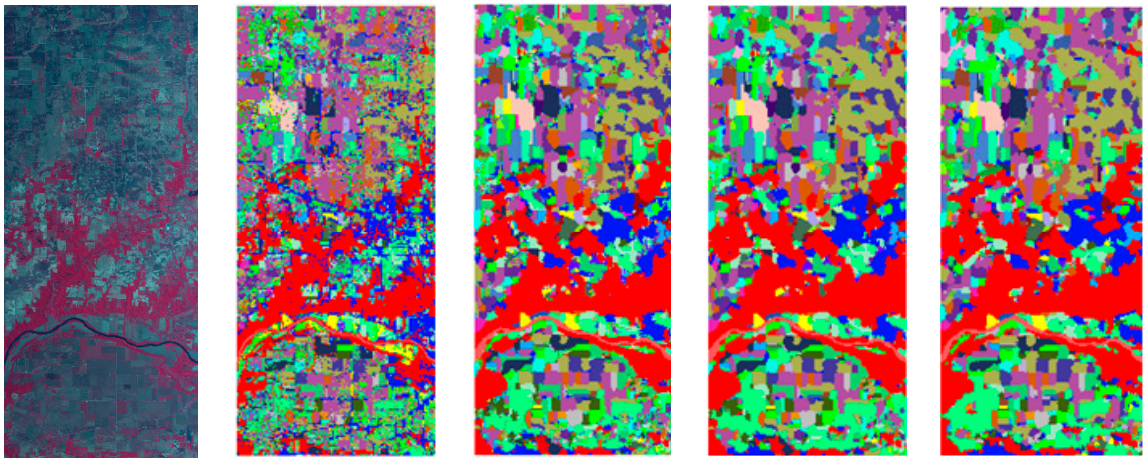


Figure 7. Test accuracy of different semi-supervised methods with changed number of training samples.

For supervised methods, we chose the SVM, EMP-RF, and PCA-DenseNet to report the different capacities in coping with complex data when the number of training samples was reduced. For semi-supervised methods, the Label Propagation and Semi-GAN were chosen as the contrastive methods. Take the two figures apart, we can see that the PCA-DenseNet and Semi-GAN always obtained the highest OA in three different conditions for both supervised and semi-supervised classification. Compared with the traditional approaches used in HSI classification, the results demonstrated that the deep learning methods can exploit huge capacities in coping with complex data. When combining the two figures together, the Semi-GAN showed the best performance in all supervised and semi-supervised methods. Furthermore, we can see that the reduction of Semi-GAN was lower than PCA-DenseNet when the number of available training samples decreased, which proved that the semi-supervised methods obtained a superior ability than supervised approaches in the limited training samples condition.

Moreover, we visually analyzed the classification results. The investigated methods include SVM, EMP-RF, PCA-DenseNet, and Semi-GAN. The classification maps for different approaches were shown in Figure 8. From the maps, one can see how the different methods affected the classification results.

The EMP-RF method had the lowest precise in the dataset (see Figure 8b), and compared with the traditional methods, the deep learning methods achieved a superior performance in classification, furthermore, we can see that our proposed Semi-GAN gave a more detailed classification map than PCA-DenseNet.



(**a**) (**b**) (**c**) (**d**) (**e**)

Figure 8. (**a**) False-color composite image of Indian Pines dataset; The classification maps using (**b**) EMP-RF; (**c**) PCA-DenseNet; (**d**) DenseNet-CRF; (**e**) Semi-GAN.

4.5. Consuming Time

In this study, the total running time for five repeated experiments of five methods, e.g., the CNN-based models and traditional SVM model, on this dataset were shown in Table 10. In the SVM method, we preserved three principal components of HSI after PCA, the $27 \times 27 \times 3$ neighbors of each pixel were regarded as the input 3D image. In the CNN-based methods, the size of each input image was $64 \times 64 \times d$, where the d represented the number of spectral bands in CNN and $64 \times 64 \times 10$ in Semi-GAN and PCA-DenseNet. All the experiments were run on a 3.2-GHz CPU with a GTX 1060 GPU card. The CNN-based methods were performed on PyTorch platform and the SVM method was performed on LibSVM library.

Table 10. Running time of five different methods.

Methods		Running Time (min.)
SVM	Training Test	2650.91 32.92
CNN	Training Test	457.25 23.55
Semi-GAN	Training Test	1053.42 0.81
PCA-DenseNet	Training Test	188.16 2.05
Auto-Encoder-DenseNet	Training Test	284.71 2.15

From Table 10, one can see that SVM method had the longest running time. When coping with complicated data with large volume, the consuming time of SVM-based method increased sharply along with the increasing numbers of training samples, which made SVM not suitable for the classification with lots of training samples.

The deep models reduced the total consuming time drastically and improved the classification performance at the same time when compared with SVM model. Besides, the additional preprocessing operations like PCA and Auto-Encoder reduced the running time greatly, which make deep learning methods more applicable for the HSI fine-grained classification with a great number of classes.

5. Conclusions

The fine-grained classification of HSI is a task to be solved nowadays. In this study, deep learning-based methods were investigated for HSI supervised and semi-supervised fine-grained classification for the first time. The obtained experimental results have shown that the proposed deep learning-based methods obtained superior performance in terms of classification accuracy.

For supervised fine-grained classification of HSI, densely connected CNN was proposed for accurate classification. The deep learning-based methods significantly outperformed the traditional spectral-spatial classifiers such as SVM, EMP-RF, and EMAP-RF in terms of classification accuracy. Moreover, the combination of DenseNet with pre-processing or post-processing technique was proposed to further improve classification accuracy.

For semi-supervised fine-grained classification of HSI, GAN was used to handle the labeled and unlabeled samples in the training stage. The proposed 3D Semi-GAN achieved better classification performance compared with traditional semi-supervised classifiers such as TSVM and Label Propagation.

The proposed deep learning models worked effectively with different numbers of training samples. The deep models exhibited good classification performance (e.g., OA was 88.23%) even under limited training samples (e.g., 4000 training samples were available, which meant there were only 77 training samples for each class on average in Auto-Encoder-DenseNet). The study demonstrates that deep learning has a huge potential for HSI fine-grained classification.

Author Contributions: Conceptualization, Y.C.; methodology, L.H., L.Z., and Y.C.; writing—original draft preparation, Y.C., L.H., L.Z., N.Y., and X.J.

Funding: This research was funded by the Natural Science Foundation of China under the Grant 61971164.

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

References

1. Fauvel, M.; Tarabalka, Y.; Benediktsson, J.A.; Chanussot, J.; Tilton, J.C. Advances in Spectral-Spatial Classification of Hyperspectral Images. *Proc. IEEE* **2012**, *101*, 652–675. [CrossRef]
2. Chang, C.-I. *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*; Kluwer Academic Publishers: New York, NY, USA, 2003; pp. 15–35.
3. Li, J.; Bioucas-Dias, J.M.; Plaza, A. Spectral–spatial hyperspectral image segmentation using subspace multinomial logistic regression and Markov random fields. *IEEE Trans. Geosci. Remote Sens.* **2011**, *50*, 809–823. [CrossRef]
4. Ham, J.; Chen, Y.; Crawford, M.M.; Ghosh, J. Investigation of the random forest framework for classification of hyperspectral data. *IEEE Trans. Geosci. Remote Sens.* **2005**, *43*, 492–501. [CrossRef]
5. Yang, H. A back-propagation neural network for mineralogical mapping from AVIRIS data. *Int. J. Remote Sens.* **1999**, *20*, 97–110. [CrossRef]
6. Gualtieri, J.A.; Cromp, R.F. Support vector machines for hyperspectral remote sensing classification. In Proceedings of the 27th AIPR Workshop: Advances in Computer-Assisted Recognition, Washington, DC, USA, 14–16 October 1998; pp. 221–232.
7. Melgani, F.; Bruzzone, L. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Trans. Geosci. Remote Sens.* **2004**, *42*, 1778–1790. [CrossRef]
8. Chen, Y.; Nasrabadi, N.M.; Tran, T.D. Hyperspectral image classification using dictionary-based sparse representation. *IEEE Trans. Geosci. Remote Sens.* **2011**, *49*, 3973–3985. [CrossRef]
9. Ghamisi, P.; Plaza, J.; Chen, Y.; Li, J.; Plaza, A.J. Advanced spectral classifiers for hyperspectral images: A review. *IEEE Geosci. Remote Sens. Mag.* **2017**, *5*, 8–32. [CrossRef]

10. Xu, X.; Li, W.; Ran, Q.; Du, Q.; Gao, L.; Zhang, B. Multisource remote sensing data classification based on convolutional neural network. *IEEE Trans. Geosci. Remote Sens.* **2017**, *56*, 937–949. [CrossRef]
11. Yang, X.; Ye, Y.; Li, X.; Lau, R.Y.; Zhang, X.; Huang, X. Hyperspectral image classification with deep learning models. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 5408–5423. [CrossRef]
12. 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. [CrossRef]
13. Palmason, J.A.; Benediktsson, J.A.; Sveinsson, J.R.; Chanussot, J. Classification of hyperspectral data from urban areas using morphological preprocessing and independent component analysis. In Proceedings of the 2005 IEEE International Geoscience and Remote Sensing Symposium, Seoul, Korea, 25–29 July 2005; pp. 176–179.
14. Pesaresi, M.; Benediktsson, J.A. A new approach for the morphological segmentation of high-resolution satellite imagery. *IEEE Trans. Geosci. Remote Sens.* **2001**, *39*, 309–320. [CrossRef]
15. Gu, Y.; Chanussot, J.; Jia, X.; Benediktsson, J.A. Multiple kernel learning for hyperspectral image classification: A review. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 6547–6565. [CrossRef]
16. Fauvel, M.; Benediktsson, J.A.; Chanussot, J.; Sveinsson, J.R. Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles. *IEEE Trans. Geosci. Remote Sens.* **2008**, *46*, 3804–3814. [CrossRef]
17. Song, B.; Li, J.; Dalla Mura, M.; Li, P.; Plaza, A.; Bioucas-Dias, J.M.; Benediktsson, J.A.; Chanussot, J. Remotely sensed image classification using sparse representations of morphological attribute profiles. *IEEE Trans. Geosci. Remote Sens.* **2013**, *52*, 5122–5136. [CrossRef]
18. Camps-Valls, G.; Gomez-Chova, L.; Muñoz-Marí, J.; Vila-Francés, J.; Calpe-Maravilla, J. Composite kernels for hyperspectral image classification. *IEEE Geosci. Remote Sens. Lett.* **2006**, *3*, 93–97. [CrossRef]
19. Baraldi, A.; Bruzzone, L.; Blonda, P. Quality assessment of classification and cluster maps without ground truth knowledge. *IEEE Trans. Geosci. Remote Sens.* **2005**, *43*, 857–873. [CrossRef]
20. Chi, M.; Bruzzone, L. A semilabeled-sample-driven bagging technique for ill-posed classification problems. *IEEE Geosci. Remote Sens. Lett.* **2005**, *2*, 69–73. [CrossRef]
21. Shahshahani, B.M.; Landgrebe, D.A. The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. *IEEE Trans. Geosci. Remote Sens.* **1994**, *32*, 1087–1095. [CrossRef]
22. Bruzzone, L.; Chi, M.; Marconcini, M. A novel transductive SVM for semisupervised classification of remote-sensing images. *IEEE Trans. Geosci. Remote Sens.* **2006**, *44*, 3363–3373. [CrossRef]
23. Camps-Valls, G.; Marsheva, T.V.B.; Zhou, D. Semi-supervised graph-based hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2007**, *45*, 3044–3054. [CrossRef]
24. Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
25. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.; Mohamed, A.-r.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Kingsbury, B. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Process. Mag.* **2012**, *29*, 82–97. [CrossRef]
26. Gao, J.; He, X.; Yih, W.-T.; Deng, L. Learning semantic representations for the phrase translation model. *arXiv* **2013**, arXiv:1312.0482.
27. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [CrossRef] [PubMed]
28. Zhang, L.; Zhang, L.; Du, B. Deep learning for remote sensing data: A technical tutorial on the state of the art. *IEEE Geosci. Remote Sens. Mag.* **2016**, *4*, 22–40. [CrossRef]
29. Chen, Y.; Lin, Z.; Zhao, X.; Wang, G.; Gu, Y. Deep learning-based classification of hyperspectral data. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2014**, *7*, 2094–2107. [CrossRef]
30. 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. [CrossRef]
31. Zhong, P.; Gong, Z.; Li, S.; Schönlieb, C.-B. Learning to diversify deep belief networks for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 3516–3530. [CrossRef]
32. Chen, Y.; Jiang, H.; Li, C.; Jia, X.; Ghamisi, P. Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 6232–6251. [CrossRef]

33. Hu, W.; Huang, Y.; Wei, L.; Zhang, F.; Li, H. Deep convolutional neural networks for hyperspectral image classification. *J. Sens.* **2015**, 1–12. [CrossRef]
34. Romero, A.; Gatta, C.; Camps-Valls, G. Unsupervised deep feature extraction for remote sensing image classification. *IEEE Trans. Geosci. Remote Sens.* **2015**, *54*, 1349–1362. [CrossRef]
35. Tao, Y.; Xu, M.; Lu, Z.; Zhong, Y. DenseNet-based depth-width double reinforced deep learning neural network for high-resolution remote sensing image per-pixel classification. *Remote Sens.* **2018**, *10*, 779. [CrossRef]
36. Mou, L.; Ghamisi, P.; Zhu, X.X. Deep recurrent neural networks for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 3639–3655. [CrossRef]
37. Zhao, W.; Du, S. Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 4544–4554. [CrossRef]
38. Li, W.; Wu, G.; Zhang, F.; Du, Q. Hyperspectral image classification using deep pixel-pair features. *IEEE Trans. Geosci. Remote Sens.* **2016**, *55*, 844–853. [CrossRef]
39. Zhong, Z.; Li, J.; Luo, Z.; Chapman, M. Spectral–spatial residual network for hyperspectral image classification: A 3-D deep learning framework. *IEEE Trans. Geosci. Remote Sens.* **2017**, *56*, 847–858. [CrossRef]
40. Li, Y.; Zhang, H.; Shen, Q. Spectral–spatial classification of hyperspectral imagery with 3D convolutional neural network. *Remote Sens.* **2017**, *9*, 67. [CrossRef]
41. Sellami, A.; Farah, M.; Farah, I.R.; Solaiman, B. Hyperspectral imagery classification based on semi-supervised 3-D deep neural network and adaptive band selection. *Expert Syst. Appl.* **2019**, *129*, 246–259. [CrossRef]
42. Zhu, L.; Chen, Y.; Ghamisi, P.; Benediktsson, J.A. Generative adversarial networks for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 5046–5063. [CrossRef]
43. Zhan, Y.; Hu, D.; Wang, Y.; Yu, X. Semisupervised hyperspectral image classification based on generative adversarial networks. *IEEE Geosci. Remote Sens. Lett.* **2017**, *15*, 212–216. [CrossRef]
44. Cavallaro, G.; Riedel, M.; Richerzhagen, M.; Benediktsson, J.A.; Plaza, A. On understanding big data impacts in remotely sensed image classification using support vector machine methods. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 4634–4646. [CrossRef]
45. Richards, J.A.; Richards, J. *Remote Sensing Digital Image Analysis*; Springer: Berlin, Germany, 1999; pp. 161–201.
46. Mughees, A.; Tao, L. Efficient deep auto-encoder learning for the classification of hyperspectral images. In Proceedings of the 2016 International Conference on Virtual Reality and Visualization (ICVRV), Hangzhou, China, 23–25 September 2016; pp. 44–51.
47. Chu, X.; Ouyang, W.; Wang, X. Crf-cnn: Modeling structured information in human pose estimation. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 316–324.
48. Kirillov, A.; Schlesinger, D.; Zheng, S.; Savchynskyy, B.; Torr, P.H.; Rother, C. Joint training of generic CNN-CRF models with stochastic optimization. In Proceedings of the Asian Conference on Computer Vision, Taipei, China, 20–24 November 2016; pp. 221–236.
49. Liu, F.; Lin, G.; Shen, C. CRF learning with CNN features for image segmentation. *Pattern Recognit.* **2015**, *48*, 2983–2992. [CrossRef]
50. Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv* **2014**, arXiv:1412.7062.
51. Zheng, S.; Jayasumana, S.; Romera-Paredes, B.; Vineet, V.; Su, Z.; Du, D.; Huang, C.; Torr, P.H. Conditional random fields as recurrent neural networks. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 13–16 December 2015; pp. 1529–1537.
52. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv* **2015**, arXiv:1502.03167.
53. Santurkar, S.; Tsipras, D.; Ilyas, A.; Madry, A. How does batch normalization help optimization? In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 2–8 December 2018; pp. 2483–2493.
54. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
55. Rother, C.; Kolmogorov, V.; Blake, A. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. Gr. (TOG)* **2004**, *23*, 309–314. [CrossRef]

56. Shotton, J.; Winn, J.; Rother, C.; Criminisi, A. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *Int. J. Comput. Vision* **2009**, *81*, 2–23. [CrossRef]
57. Krähenbühl, P.; Koltun, V. Efficient inference in fully connected crfs with gaussian edge potentials. In Proceedings of the Advances in Neural Information Processing Systems, Granada, Spain, 12–17 December 2011; pp. 109–117.
58. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2672–2680.
59. Mirza, M.; Osindero, S. Conditional generative adversarial nets. *arXiv* **2014**, arXiv:1411.1784.
60. Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z. Photo-realistic single image super-resolution using a generative adversarial network. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 4681–4690.
61. Zhu, J.-Y.; Park, T.; Isola, P.; Efros, A.A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2223–2232.
62. Yi, Z.; Zhang, H.; Tan, P.; Gong, M. Dualgan: Unsupervised dual learning for image-to-image translation. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2849–2857.
63. Mathieu, M.; Couprie, C.; LeCun, Y. Deep multi-scale video prediction beyond mean square error. *arXiv* **2015**, arXiv:1511.05440.
64. Li, C.; Wand, M. Precomputed real-time texture synthesis with markovian generative adversarial networks. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; pp. 702–716.
65. Yu, L.; Zhang, W.; Wang, J.; Yu, Y. Seqgan: Sequence generative adversarial nets with policy gradient. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
66. Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv* **2015**, arXiv:1511.06434.
67. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein gan. *arXiv* **2017**, arXiv:1701.07875.
68. Odena, A.; Olah, C.; Shlens, J. Conditional image synthesis with auxiliary classifier gans. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Volume 70, pp. 2642–2651.
69. Chang, C.-C.; Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* **2011**, *2*, 27. [CrossRef]
70. Breiman, L. RF/tools: A class of two-eyed algorithms. In Proceedings of the SIAM Workshop, San Francisco, CA, USA, 1–3 May 2003; pp. 1–56.
71. Dalla Mura, M.; Atli Benediktsson, J.; Waske, B.; Bruzzone, L. Extended profiles with morphological attribute filters for the analysis of hyperspectral data. *Int. J. Remote Sens.* **2010**, *31*, 5975–5991. [CrossRef]
72. Dalla Mura, M.; Benediktsson, J.A.; Waske, B.; Bruzzone, L. Morphological attribute profiles for the analysis of very high resolution images. *IEEE Trans. Geosci. Remote Sens.* **2010**, *48*, 3747–3762. [CrossRef]
73. Ghamisi, P.; Benediktsson, J.A.; Cavallaro, G.; Plaza, A. Automatic framework for spectral–spatial classification based on supervised feature extraction and morphological attribute profiles. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2014**, *7*, 2147–2160. [CrossRef]
74. Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X. Improved techniques for training gans. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 2234–2242.