remote sensing

MDPI

Article

Continuous Particle Swarm Optimization-Based Deep Learning Architecture Search for Hyperspectral Image Classification

Xiaobo Liu [1,2,3,*], Chaochao Zhang [1,2], Zhihua Cai [3,4,5], Jianfeng Yang [1,2], Zhilang Zhou [1,2] and Xin Gong [1,2]

[1] School of Automation, China University of Geosciences (Wuhan), Wuhan 430074, China; cz@cug.edu.cn (C.Z.); yangjianfeng@cug.edu.cn (J.Y.); zlzhou1106@cug.edu.cn (Z.Z.); xgong@cug.edu.cn (X.G.)
[2] Hubei Key Laboratory of Advanced Control and Intelligent Automation for Complex Systems, China University of Geosciences (Wuhan), Wuhan 430074, China
[3] Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences (Wuhan), Wuhan 430078, China; zhcai@cug.edu.cn
[4] School of Computer Science, China University of Geosciences (Wuhan), Wuhan 430074, China
[5] Beibu Gulf Big Data Resources Utilisation Laboratory, Beibu Gulf University, Qinzhou 535011, China
[*] Correspondence: xbliu@cug.edu.cn

Abstract: Deep convolutional neural networks (CNNs) are widely used in hyperspectral image (HSI) classification. However, the most successful CNN architectures are handcrafted, which need professional knowledge and consume a very significant amount of time. To automatically design cell-based CNN architectures for HSI classification, we propose an efficient continuous evolutionary method, named CPSO-Net, which can dramatically accelerate optimal architecture generation by the optimization of weight-sharing parameters. First, a SuperNet with all candidate operations is maintained to share the parameters for all individuals and optimized by collecting the gradients of all individuals in the population. Second, a novel direct encoding strategy is devised to encode architectures into particles, which inherit the parameters from the SuperNet. Then, particle swarm optimization is used to search for the optimal deep architecture from the particle swarm. Furthermore, experiments with limited training samples based on four widely used biased and unbiased hyperspectral datasets showed that our proposed method achieves good performance comparable to the state-of-the-art HSI classification methods.

Keywords: convolutional neural network; hyperspectral image; particle swarm optimization; neural architecture search; SuperNet

Citation: Liu, X.; Zhang, C.; Cai, Z.; Yang, J.; Zhou, Z.; Gong, X. Continuous Particle Swarm Optimization-Based Deep Learning Architecture Search for Hyperspectral Image Classification. *Remote Sens.* **2021**, *13*, 1082. https://doi.org/10.3390/rs13061082

Academic Editor: Gwanggil Jeon

Received: 31 December 2020
Accepted: 27 February 2021
Published: 12 March 2021

1. Introduction

A hyperspectral image (HSI) has a high spectral and spatial resolution, which provides rich information about ground truths [1]. Hyperspectral sensors mounted on different space platforms, i.e., imaging spectrometers, are used to capture images in the ultraviolet, visible, near-infrared, and mid-infrared regions of the electromagnetic spectrum, where tens to hundreds of continuous and subdivided spectral bands simultaneously image the target area. The exploitation of HSI signatures is a powerful tool for observing the surface of the Earth. Therefore, there are a wide range of processing techniques to explore HSI data [2–4], and HSI classification is one of the most popular techniques for HSI data exploitation. HSI classification is widely applied in many fields, such as agricultural applications [5], forestry and environmental management [6], water and maritime resource management [7], and military and defense applications [8].

Compared to traditional HSI classification algorithms, convolutional neural networks (CNNs) can not only explore spectral information, but can also combine spatial information with spectral information [9]. Moreover, because of the share of parameters to tackle the curse of dimensionality [10] and the local receptive field to learn spatial information [11], CNNs show good performance in HSI band selection [12,13], feature extraction and classification [14]. There are many CNN models used as HSI classifiers, including the spectral CNN [15,16], the spatial CNN [17,18], and the spectral–spatial CNN [19,20]. With the advance of remote sensing technology, better classifiers are being developed to further improve the classification performance and the complexity of CNNs. On the one hand, some new types of CNNs have been introduced for HSI classification, such as the dense CNN using shortcut connections between layers [21] and the deep residual network with skip connections [22]. On the other hand, some other techniques have been combined with CNNs, such as the attention mechanism [23], sparse representation [24], multi-scale fusion [25], and discriminant embedding [26].

Although the aforementioned CNN models have achieved remarkable progress in HSI classification tasks in recent years, there are two major problems that should be considered when applying these CNN models for HSI classification. On the one hand, designing state-of-the-art successful CNN architectures requires substantial professional knowledge of human experts and consume a significant amount of time for repeated debugging. For example, ResNet [27] and DesNet [21] were carefully handcrafted by taking into consideration the manual domain knowledge. On the other hand, manually optimizing a CNN when applying it to different data sources may take days or even weeks and should always be fine-tuned; additionally, this process results in a heavy computational burden, and is not affordable for most researchers and developers [28].

To overcome the problems of handcrafting CNN models, neural architecture search (NAS) [29] is an import method to automatically design neural architectures. Since Zoph et al. [30] successfully discovered a neural network architecture that can achieve comparable performance to handcrafted CNNs, there has been a growing interest in designing a robust and well-performing neural architecture from the predefined search space by NAS. There are many different search strategies that can be used to design a proper CNN architecture, including evolutionary algorithms (EAs), reinforcement learning (RL), and gradient-based methods. Recently, Chen et al. [31] proposed the automatic design of CNNs for HSI classification, based on the gradient descent method, and the automatic CNNs achieved better performance compared to some state-of-the-art deep learning methods. However, the method based on gradient descent easily converges to the local optimal, and it cannot be used when there is a vanishing gradient problem or an exploding gradient problem. EAs search globally, and population-based optimization affords EAs a very good global search ability. To address the NAS problem, a global search technique is required, which has always been a topic of interest for discovering the neural architecture by the EA search strategy. Junior et al. [32] found good chain CNN architectures based on the particle swarm optimization (PSO) search strategy using nine datasets, which can only search simple chain-CNNs. Real et al. [33] proposed AmoebaNet-A and found comparable cell-based neural architectures to some state-of-the-art ImageNet models using the CIFAR-10 dataset, showing that they consumed a significant time without weight-sharing. An alternative architecture optimization method is neuroevolution, which is inspired by the evolution of natural brains. Stanley et al. [34] proposed a classic neuroevolutionary approach, NEAT, Neural Networks through Augmenting Topologieswhich can modify weights and add nodes and connections in neural networks. HyperNEAT [35,36] is an improvement of NEAT, which uses an indirect encoding method named CPPNs [37] that can generate the patterns of weights in neural networks themselves to evolve into bigger network.

To automatically search for CNN architectures for HSI classification, we propose an NAS method named PSO-Net, which is based on PSO and consumes a significant amount of time to search for the optimal architecture. To further accelerate optimal architecture generation, we propose CPSO-Net with weight-sharing parameter optimization based on PSO-Net. We propose two cell-based CNN architecture search methods by PSO, capable of searching globally, compared to the gradient descent method. The main contributions of this paper are summarized as follows:

(1) Two methods based on PSO are explored to automatically design the CNN architecture for HSI classification.
(2) A novel encoding strategy is devised that can be used to encode architectures into arrays with the information of the connections and basic operations types between the nodes in computation cells.
(3) To improve the search efficiency, CPSO-Net maintains continuous SuperNet sharing parameters for all particles and optimizes by collecting the gradients of all individuals in the population.
(4) PSO-Net and CPSO-Net are tested on four biased and unbiased hyperspectral datasets with limited training samples, showing comparable performance to the state-of-the-art CNN classification methods.

The remainder of this paper is structured as follows. Detailed related work about neural architecture search and PSO is presented in Section 2. Section 3 introduces the proposed PSO-Net and CPSO-Net in detail. Section 4 summarizes and analyzes a series of experiments performed for HSI classification. Section 5 provides the main conclusions and perspectives of this work.

2. Related Work

In this section, we introduce two methods related to the method we proposed, namely neural architecture search and particle swarm optimization.

2.1. Neural Architecture Search

NAS aims to design a well-performing neural network architecture by selecting and combining basic operations and connections from a predefined search space.

There are three main parts in NAS, namely the search space, the search strategy, and the performance estimation strategy. The search space is a set of all possible architectures. The search strategy then automatically finds the optimal architecture from the search space. The estimation of performance aims to find the architectures with high predictive performance by training and validation based on data.

The search space should be predefined with the information of all basic operations and connection nodes. The search space mainly includes two types. The first is to directly construct the global search space of the entire neural network architecture, including the chain architecture search space [38] and the multi-branch architecture search space [30]; the second is to construct parts of the neural network architecture by repeating certain specific structures, such as the cell-based search space [39].

The search strategy contains gradient-based search and RL- and EA-based approaches. Gradient-based approaches [40–42] convert the discrete architecture space into a continuous space and find the optimal architecture using the gradient method in the continuous space. RL-based methods [30,39] set the state–action space as the NAS search space and set the policy function as the recurrent network to finish the search process. The validation accuracy of each individual is used as the fitness to evolve the next generation. In EA-based approaches [43,44], the population is composed of a group of candidate network structures, and individuals generate new network architectures through evolutionary algorithms. To improve the search efficiency of EA-based approaches, Yang et al. [45] maintained a SuperNet that is trained by the gradient descent method and shares parameters for all architectures.

The core of the estimation of performance is the weight parameter optimization step, which can be divided into two categories, namely individual and weight-sharing optimization. Individual weight parameter optimization often learns each network separately, typically incurring a heavy computational burden and taking a significant amount of time. Therefore, weight-sharing parameter optimization methods are used to accelerate training [43,46], which can be divided into two categories according to the search strategies, i.e., weight-sharing heuristic search strategies and weight-sharing differentiable search strategies. Weight-sharing heuristic search strategies [47,48] always train one SuperNet that includes all candidate operations first, and then focuses on each individual inherit parameter from the SuperNet. Rather than training all individuals, the SuperNet is optimized only once during one iteration, which dramatically reduces the computational complexity by separately learning each network. Weight-sharing differentiable search strategies [40] relax one architecture to one SuperNet, with the weights of all candidate operations being optimized by the gradient descent method.

2.2. Particle Swarm Optimization

PSO is a kind of evolutionary computation, which is derived from research on the predation behavior of birds [49]. The main idea of particle swarm optimization is to find the optimal solution through the cooperation and information-sharing among individuals from the swarm. In PSO, each candidate solution can be thought of as a bird in the solution space, which is called a particle. The optimal solution of itself in the past is the *pBest*, and the current global optimal solution in the swarm is the *gBest*. At every iteration, particles update themselves by tracking two extremes: *pBest* and *gBest*. Each particle is updated according to the following rule of PSO:

$$v_{i,j}(t+1) = w \times v_{i,j}(t) + c_p \times r_p \times (pBest_{i,j} - x_{i,j}(t)) + c_g \times r_g \times (gBest_j - x_{i,j}(t)) \quad (1)$$

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1) \quad (2)$$

where $v_{i,j}$ is the velocity of the i-th particle, $x_{i,j}$ is the current particle position of the i-th particle, w is a constant called momentum that controls how much the previous velocity affects the velocity at the current time step, c_p, and c_g are yjr constants defined beforehand, representing the weight of yjr particles approaching the *pBest* and *gBest*, and r_p and r_g are random numbers in [0, 1).

One of the positive characteristics of PSO is that it is capable of fast convergence [50] when compared to other evolutionary approaches, such as genetic algorithms. Due to the fact that searching CNNs takes a lot time, even on the most powerful computers, a faster converging characteristic can have a positive impact in terms of obtaining the optimal CNN architecture.

3. Proposed Method

In this section, we propose a cell-based CNN architecture search method by particle swarm optimization at first, named PSO-Net. To make the search stage more efficient and to reduce the time consumed, we developed another novel continuous PSO approach for searching neural architectures, namely CPSO-Net.

The two proposed methods are all based on the PSO architecture search step, including four procedures to search for the optimal CNN architecture: Construction of the search space, initialization of the architectures, weight optimization of the neural network constructed by the architectures, fitness evaluation of the individual particles, and particle update. CPSO-Net is an improvement based on PSO-Net, and the two methods have different procedures for the parameter optimization of architectures. The weight optimization of the former is the individual weight optimization of all particles, and the latter is weight-sharing by the inherited weights from the SuperNet.

Figure 1 shows the framework of the stages of PSO-Net. From the framework, one can see that network architectures are encoded into arrays. Then, the gradient descent method is used separately to obtain the weight parameters for all architectures, and PSO is used as the architecture search method to optimize the architectures by the fitness of all particles. Finally, the optimal architecture is found after reaching the number of iterations.

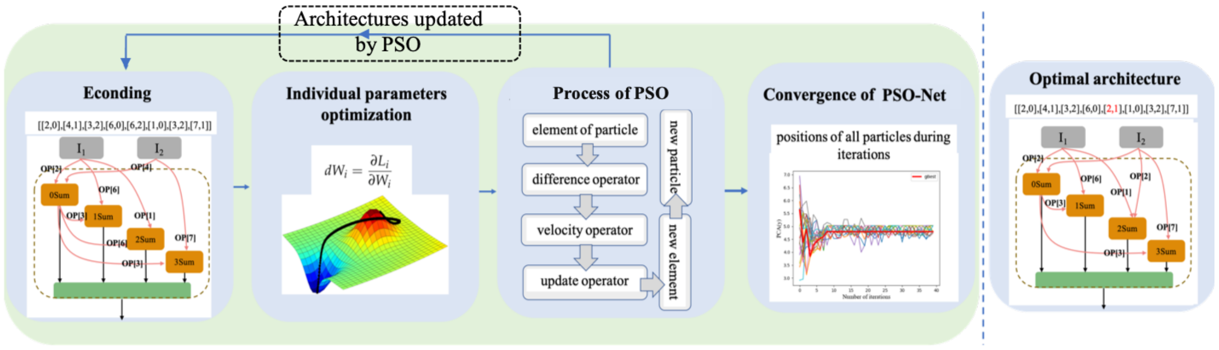


Figure 1. Framework of the stages of particle swarm optimization PSO-Net. The constructed network architectures from the search space are encoded into arrays, and populations are initialized. The gradient descent method is used to obtain the weight parameters for all architectures separately, and PSO is used to optimize the architectures and to search for the optimal architecture until reaching the number of iterations. The output is the optimal architecture.

Figure 2 shows the framework of the stages of CPSO-Net, which is proposed-based PSO-Net. The main difference of Figure 2 is the procedure of the parameter optimization of architectures. To accelerate the optimal architecture generation, a SuperNet is maintained to share the parameters for all particles and is optimized by collecting the gradients of all individuals in the population. Rather than training all the individuals of PSO-Net, the SuperNet of CPSO-Net is optimized only once during one iteration, which dramatically reduces the computational complexity by separately learning each network.

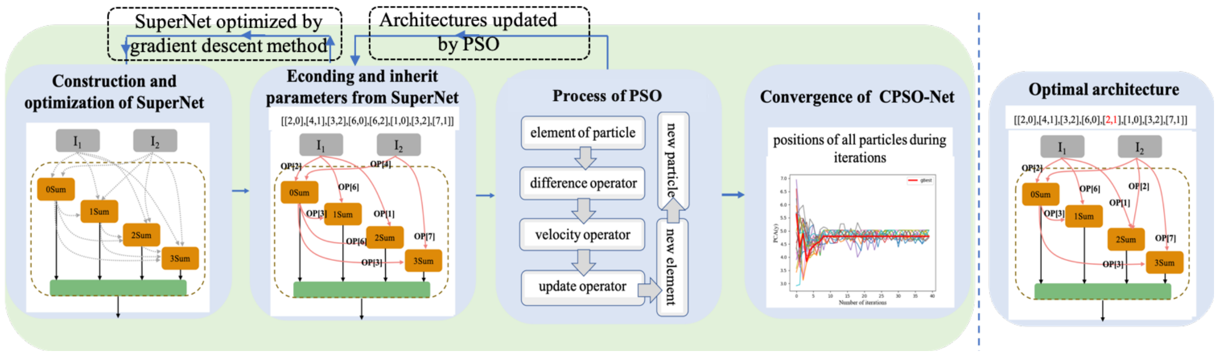

SuperNet optimized by gradient descent method
Architectures updated by PSO
Construction and optimization of SuperNet
Econding and inherit parameters from SuperNet
Process of PSO
element of particle
difference operator
velocity operator
update operator
new particle
new element
Convergence of CPSO-Net
positions of all particles during iterations
Optimal architecture

Figure 2. Framework of CPSO-Net. A continuous SuperNet that contains all candidate operations and can be trained by gradient descent is maintained first. Then, all architectures are encoded into an array, inheriting the parameters from the SuperNet, and search for the optimal architecture by PSO. Finally, the optimal architecture is found.

All the operations of PSO-Net and CPSO-Net are detailed in the following subsections.

3.1. Construction of the Search Space

The search space contains two parts, namely all candidate basic operations and connection nodes. On the one hand, there are three main basic operation types: Convolution operations, pooling operations, and nonlinear operations. All candidate basic operations are placed into the operations list named OP to prepare for the next encoding strategy. On the other hand, the cell-based search space is used to design CNN architectures in the algorithm we proposed. Each neural network architecture is built by stacking two types of k number computation cells, which includes normal cells and one reduction cell. Each cell consists of an ordered sequence with two input nodes, n intermediate nodes, and one output node. The input of the input nodes are the cell outputs in the previous two cells, and the output of the cell is a cascade by the n intermediate nodes, each of which is calculated by the sum of two computations. Additionally, each computation is based on a random one operation of the basic operations OP computing on a random one node of its previous nodes. Each computation is defined as an element, which is composed of the basic operation and the computed node. Therefore, there are 2*n* elements in each cell with *n* intermediate nodes. Figure 3 shows the general architecture stacking by the automatically designed computation cells by PSO-Net and CPSO-Net.

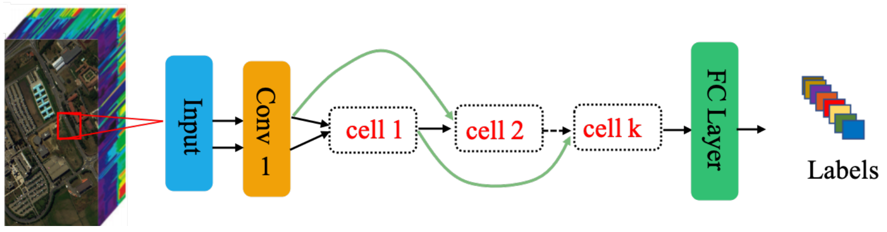
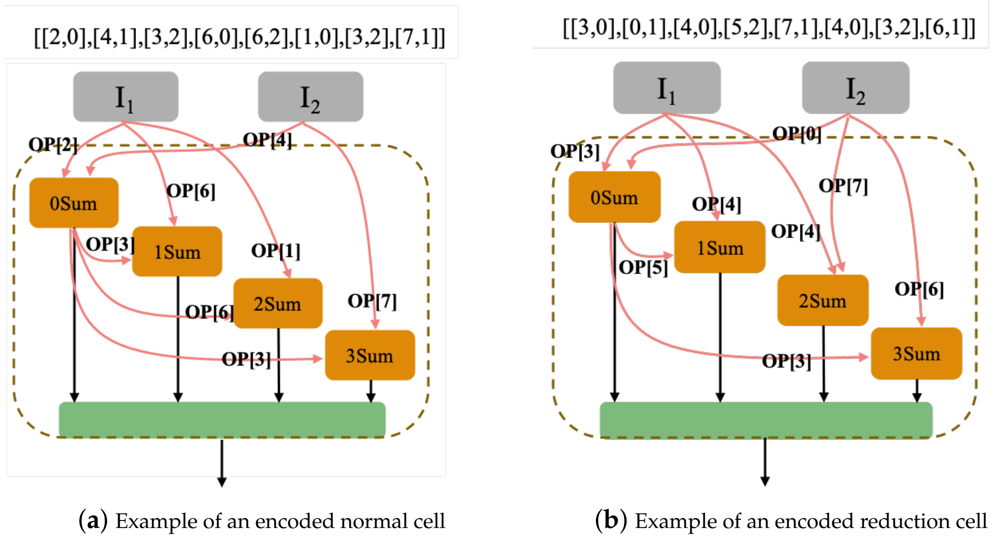


Figure 3. The general architecture stacking by the automatically designed computation cells by PSO-Net and CPSO-Net. The bottleneck convolution layer and the fully connected layer in Figure 3 are designed for data preprocessing and classification with human knowledge.

3.2. Initialization of the Swarm

Particle encoding is the core of the initialization of architectures. A novel encoding strategy was devised that can be used to encode architectures into arrays with the information of all of the basic operations and connections between the nodes in computation cells—where the first and second input node is numbered the 0th and 1th in all nodes, and the *n* intermediate nodes are numbered in cells successively following the number of two input nodes. First, an array with a shape of (2, 2*n*, 2) is built, which represents an architecture. The first dimension of the array represents two types of computation cells, namely normal cells and reduction cells. The second dimension of the array represents the 2*n* elements (defined in Section 3.1) in each cell with *n* intermediate nodes. The third dimension of the array is composed of the index of basic operations list *OP* and the computed node. Next, *P* particles of the swarm are initialized by the above encoding strategy. Figure 4 shows an example of an encoded architecture, including an encoded normal cell and a reduction cell with four intermediate nodes.



(**a**) Example of an encoded normal cell (**b**) Example of an encoded reduction cell

Figure 4. An example of an encoded architecture, including an encoded normal cell and a reduction cell with four intermediate nodes. There are eight elements in each cell with four intermediate nodes. (**a**) An example of an encoded normal cell. The 0th intermediate node, as well the 2th node in the cell, are calculated by the sum of two elements *OP* [2] computed based on the 0th node in the cell and *OP* [4] computed based on the 1th node in the cell, where the 0th and 1th nodes in the cell are the two input nodes. All the remaining intermediate nodes are encoded into arrays in the same way. (**b**) An example of an encoded reduction cell.

3.3. The Weight Optimization of the Neural Network Constructed by Architectures

The weight of the neural network constructed by architectures should be optimized, when receiving information of the connection and basic operation type between the nodes in computation cells.

3.3.1. Individual Parameter Optimization of PSO-Net

The parameter optimization of PSO-Net involves training individual particles by the gradient descent method. Each network N_i we searched can be represented by the particle P_i and a set of full precision parameters $(W_i, P_i), i \in \{1, \dots, P\}$, where P is the population size. The training dataset X is denoted as the input data, and $N_i(X)$ is the prediction of the i-th network N_i. The loss can be expressed as $L_i = \mathcal{H}(N_i(X), Y)$, where $\mathcal{H}$ is the criterion and Y is the target. The parameter W_i of N_i can be optimized by:

$$dW_i = \frac{\partial L_i}{\partial W_i} \tag{3}$$

3.3.2. Weight-Sharing Parameter Optimization of CPSO-Net

To further accelerate optimal network generation, we built a SuperNet $\mathcal{N}$ that includes all candidate operations and sharing parameters W for all particles. Each particle is regarded as a subnet and inherits the weight parameters from the fixed SuperNet, which can be optimized by the gradient descent method. Therefore, there is no need to train each particle from the beginning to the end, and the search time can therefore be greatly reduced. After the parameters have converged, the architecture could alternately be optimized by the PSO algorithm.

Sampling from the SuperNet $\mathcal{N}$, the different network N_i is a part of $\mathcal{N}$, and the parameter W_i of N_i can be inherited from W. Therefore, W_i can be represent as $W_i = W \odot P_i, i \in \{1, \dots, P\}$, where $\odot$ is the mask operation that retains the parameters of the corresponding nodes of particle P_i. Therefore, the gradient of W_i also can be calculated as:

$$dW_i = \frac{\partial L_i}{\partial W_i} = \frac{\partial L_i}{\partial W} \odot P_i \tag{4}$$

The SuperNet shares W for all different architectures, so the gradient of parameter W should be calculated by collecting the gradients of all individuals in the population:

$$dW = \frac{1}{P}\sum_{i=1}^{P} dW_i = \frac{1}{P}\sum_{i=1}^{P} \frac{\partial L_i}{\partial W} \odot P_i \tag{5}$$

Inspired by the mini-batch samples idea of stochastic gradient descent for updating parameters, we used mini-batch architectures to accumulate the gradients of all individuals for updating the shared weight W.

3.4. Fitness Evaluation

Fitness evaluation is performed using the HSI classification accuracy of the optimized neural network on the validation set, and aimed to estimate the performance of the architectures. Each particle is evaluated as a solution by the fitness evaluation function. Its optimal solution in the past is the $pBest$, and the current global optimal solution in the swarm is the $gBest$. By the fitness of PSO-Net and CPSO-Net, we can find the $pBest$ and $pBest$.

3.5. Particle Update

The difference and the velocity operator are devised based on PSO to update the particles. The particle P is updated element by element until all elements of the particle have been updated, so the particle update problem is transformed into the element update problem. Before being able to update an element, the difference between two elements with the same index in two particles needs to be measured, and the velocity of the element needs to be computed.

The difference $(E2 - E1)$ is calculated by comparing $E2$ and $E1$. If both elements are the same, the difference is "None", which means no difference. If the two elements are different, the result of $(E2 - E1)$ is $E2$.

The velocity of any given element of a particle P_E is based on the two differences: $(gBest\_E - P\_E)$ and $(pBest\_E - P\_E)$, where $gBest$, $pBest$, and P have the same index in a particle. Each element velocity is chosen from the two differences based on the decision factor c_g and a number r obtained at random from [0,1). If $r \geq c_g$, the algorithm selects from the difference $(gBest\_E - P\_E)$. Otherwise, the algorithm selects from the difference $(pBest\_E - P\_E)$.

Each element is updated according to its element velocity. If its element velocity is "None", the element remains the same, otherwise it should be displaced by its velocity. Particles are modified element by element according to the element velocity. Figure 5 shows the update of an element of a particle.

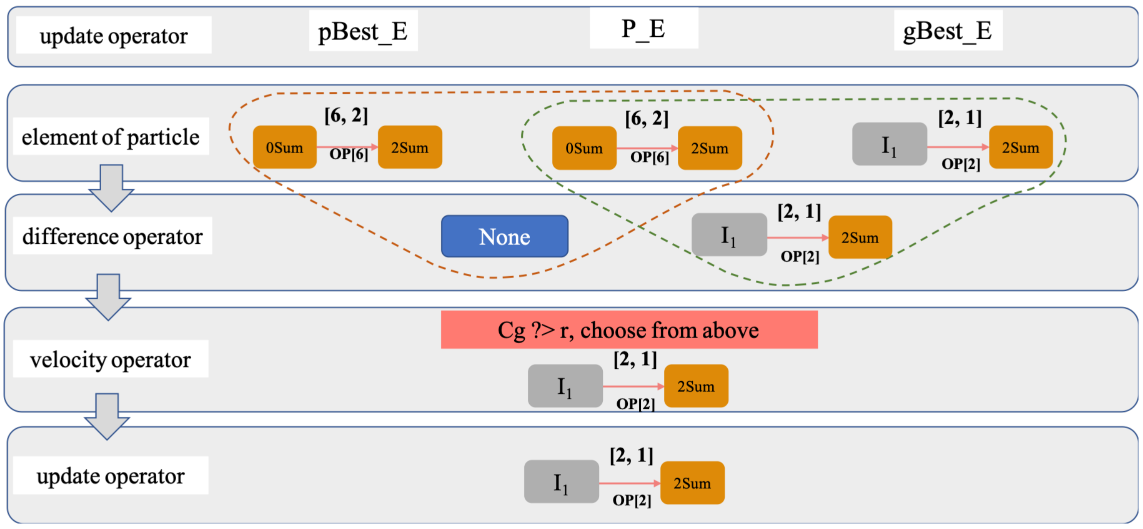


Figure 5. The update process for an element in a particle. This example shows an update of an element of the 2*nd* intermediate node. Before being able to update the element of a particle, the difference between $(gBest\_E - P\_E)$ or $(pBest\_E - P\_E)$ by the different, and the velocity of the element needs to be computed using the velocity operator.

The essence of the role of a particle update is that a particle update can be converted to an element update in the particle. We assumed that the elements of $pBest$ and $gBest$ can obtain good fitness, so that $pBest$ and $gBest$ can achieve good fitness. Therefore, we compared P to $pBest$ and $gBest$ element by element, and determined whether the element P needs to be replaced by $pBest$ or $gBest$. Figure 4 shows the update process of particles.

4. Experimental Results and Analysis

In this section, we explain the experimental setting of the method proposed and a comparative algorithm, followed by the empirical results, to confirm the effectiveness of the proposed method for automatically designing CNNs for HSI classification based on four HSI datasets with biased and unbiased data. The experimental conditions were as follows: A NVIDIA GTX1080 Ti GPU, a E5-2620 CPU, and a memory of 32 GB.

4.1. Datasets

In this section, we used four standard hyperspectral datasets to evaluate the methods we proposed. As shown in Table 1, the four datasets were Salinas Valley in California in the USA (Salinas), a mixed vegetation site over the Indian Pines test area in northwestern Indiana in the USA (Indian Pines), an urban site over the University of Pavia in Italy (Pavia), and the Kennedy Space Center (KSC) in Florida in the USA. The detailed training samples of each class are shown in Table 1.

Table 1. Number of samples in the Salinas Valley (Salinas), Indian Pines, University of Pavia (Pavia), and Kennedy Space Center (KSC) hyperspectral datasets.

Salinas Valley (Salinas)				**Indian Pines**			
No.	**Class**	**Samples**	**Training**	**No.**	**Class**	**Samples**	**Training**
1	Broccoli_green_weeds_1	2009	8	1	Alfalfa	46	1
2	Broccoli_green_weeds_2	3726	14	2	Corn–notill	1428	28
3	Fallow	1976	8	3	Corn–mintill	830	16
4	Fallow_rough_plow	1394	6	4	Corn	237	4
5	Fallow_smooth	2678	10	5	Grass–pasture	483	10
6	Stubble	3959	14	6	Grass–trees	730	14
7	Celery	3579	14	7	Grass–pasture–mowed	28	1
8	Grapes_untrained	11,271	40	8	Hay–windrowed	478	10
9	Soil_vineyard_develop	6203	22	9	Oats	20	1
10	Corn_senesced_green_weeds	3278	12	10	Soybean–notill	972	18
11	Lettuce_romaine_4wk	1068	4	11	Soybean–mintill	2455	47
12	Lettuce_romaine_5wk	1927	8	12	Soybean–clean	593	12
13	Lettuce_romaine_6wk	916	4	13	Wheat	205	4
14	Lettuce_romaine_7wk	1070	4	14	Woods	1265	24
15	Vineyard_untrained	7268	26	15	Buildings–grass–trees–drives	386	8
16	Vineyard_vertical_trellis	1807	6	16	Stone–steel–towers	93	2
	Total	43,980	200		Total	10,249	200
University of Pavia (Pavia)				**Kennedy Space Center (KSC)**			
No.	**Class**	**Samples**	**Training**	**No.**	**Class**	**Samples**	**Training**
1	Asphalt	6631	34	1	Scrub	761	24
2	Meadows	18,649	70	2	Willow swamp	243	10
3	Gravel	2099	12	3	Cp hammock	256	10
4	Trees	3064	14	4	Slash pine	252	10
5	Painted metal sheets	1345	6	5	Oak/broadleaf	261	10
6	Bare soil	5029	30	6	Hardwood	229	10
7	Bitumen	1330	14	7	Swamp	105	6
8	Self-blocking bricks	3682	16	8	Graminoid marsh	431	20
9	Shadows	947	4	9	Spartina marsh	520	20
				10	Cattail marsh	404	18
				11	Salt marsh	419	18
				12	Mud flats	503	20
				13	Water	927	24
	Total	42,776	200		Total	5211	200

The first dataset, Salinas, was collected by the Airborne Visible Infrared Imaging Spectrometer (AVIRIS) with 224-band over Salinas Valley, California, which is of a high spatial resolution (3.7 m/pixels). The available dataset consists of 204 bands with 512 $\times$ 217 pixels after discarding the 20 water absorption bands and the 16 land cover classes.

The second dataset, Indian Pines, was captured by the AVIRIS sensor with 220-band over the Indian Pines test area. The available dataset consists of 200 bands with 145 $\times$ 145 pixels after removing the water absorption bands. The ground reference map covers 16 classes of interest.

The third dataset, University of Pavia, was collected by the ROSIS-3 sensor with 115-band over the University of Pavia, which is of a high spatial resolution (1.3 m/pixels) to avoid mixed pixels. The available dataset consists of 103 bands with 610 × 340 pixel vectors after removing the noisy bands. The ground reference map covers nine classes of interest.

The fourth dataset, KSC, was captured by the airborne AVIRIS sensor over KSC, Florida, at an altitude of approximately 20 km with a spatial resolution of 18 m. The available dataset consists of 176 bands with 512 × 614 pixels after removing the water absorption and noisy bands. The ground reference map covers 13 classes of interest.

4.2. Comparative Experiment

To evaluate the performance of the two methods we proposed, some CNN classification methods, including handcrafted CNN models and automatically designed CNN models based on spatial–spectral information, were employed to achieve a comprehensive comparison with the PSO-Net and CPSO-Net methods. We chose several handcrafted CNN models that have reported results in the spectral–spatial information hyperspectral datasets to test the proposed algorithms. The CNN [51] was conducted, which is a sample CNN designed to tackle the problem of HSI spectral–spatial classification. The spectral–spatial residual network (SSRN) [22] and ResNet [27] are residual-based methods, which can achieve good classification performance. DesNet [21] using shortcut connections between layers was used for comparison. Moreover, an automatically designed CNN method, namely Auto-CNN, was introduced, which explores the automatic design of CNNs for HSI classification by the gradient descent and achieved good performance. To demonstrate the performance of the two methods we proposed, we made a comparison in the classification accuracy and the computational complexity with handcrafted and automatically designed CNN models. To further illustrate the search effectiveness of PSO-Net and CPSO-Net, we compared them with a peer, the automatically designed method Auto-CNN, in terms of time complexity.

4.3. Experimental Settings

In this section, we explain the detailed parameter setting for our experiments. For the automatically designed algorithms, including Auto-CNN, PSO-Net, and CPSO-Net, the search space should be predefined. We set the number of computation cells k to 3 and the intermediate nodes n to 4, and the basic operation list OP includes the following candidate basic operations: None; 3 × 3 max pooling, abbreviated as max_3 × 3; 3 × 3 average pooling, abbreviated as avg_3 × 3; skip connection, abbreviated as skip; 3 × 3 separable convolution, abbreviated as sep_3 × 3; 5 × 5 separable convolution, abbreviated as sep_5 × 5; 3× 3 dilated convolution, abbreviated as dil_3 × 3; 5 × 5 dilated convolution, abbreviated as dil_5 × 5. For HSI classification, the general architecture of automatically designed CNNs includes three parts. First, hyperspectral datasets as an input pass one 1 × 1 bottleneck convolution layers to set the number of HSI bands to 10 for the next computation cells. Then, two normal cells with padding and setting stride to 1, and reduction cells with padding and setting stride to 2 are stacked to complete the CNN. Finally, the final labels can be obtained by the fully connected layer.

For all the experiments in this work, we split the labeled samples of the HSI datasets into three subsets by sampling without replacement, namely the training, validation, and test sets. To ensure that all classes are included in the training, validation, and test dataset, we choose samples including all classes based on the proportion of every class in whole dataset and samples in each class are randomly selected with the same probability in this class. Based on the rule above, we chose 200 samples, including all classes as the training set and 600 samples as the validation set for each dataset. To demonstrate the stability of the two proposed methods, we conducted the experiments five times with random split datasets. Details of the training samples of each class of the four datasets are shown in Table 1. All the remaining labeled samples served as the test set to evaluate the capability of the network. The training set was used to train the network, and the validation set was used to evaluate the performance of the architectures found and to search for the optimal architecture for all the automatic design algorithms.

For biased datasets, we chose $7 \times 7 \times$ B neighborhoods of a pixel as input, where B is the number of HSI bands. For unbiased datasets, samples with 32×32 slides were not enough to divide into the training and validation datasets, so we chose 7×7 neighborhoods of a pixel, and none of the samples were overlapping. For all the CNN models designed by hand, i.e., 3-D CNN, SSRN, ResNet, DesNet, we set the training epochs to 100 epochs. The learning rate for the weights of all these models was 0.025. For 3-D Auto-CNN, we set the training epochs in the architecture search and test process both to 100 epochs. The learning rate for the weights in the search and test process was 0.025 and 0.05, respectively. For PSO-Net, we set the swarm size to 100, the number of iterations to 40, c_g to 0.5, the training epochs in the architecture search and test process both to 100 epochs, and the learning rate for the weights in the architecture search and test process to 0.025. For CPSO-Net, only the training epochs in the architecture search were different from above, inspired by the mini-batch architecture idea of stochastic gradient descent for updating parameters, with only one epoch achieving good performance. For the test process, the experimental settings for the different methods can be found in Table 2.

Table 2. Experimental settings for the different methods. CNN, convolutional neural network.

Algorithm	Architecture		CNN Training				PSO	
Parameters	**Cells**	**Nodes**	**Train_Samples**	**Valid_Samples**	**Epochs_Search**	**Epochs_Test**	**Pop_Size**	**Iterations**
Hand-CNNs	/	/	200	600	/	100	/	/
Auto-CNNs	3	4	200	600	/	100	/	/
PSO-Net	3	4	200	600	100	100	100	40
CPSO-Net	3	4	200	600	1	100	100	40

4.4. Results Analysis

In this section, we analyze the classification results and compare our algorithm with other methods in terms of classification accuracy, parameters, and time complexity on biased hyperspectral datasets to illustrate the effectiveness of our algorithms. We also demonstrate the optimal architectures searched by our methods, and analyze the convergence to illustrate the feasibility of the methods we proposed. Finally, we conduct experiments on unbiased hyperspectral datasets to further show our algorithms' advantages.

4.4.1. Classification Accuracy

Tables 3–6 show the detailed classification results on test datasets of the two methods we proposed and the other comparative algorithm. As shown in Tables 3–6, it is obvious that PSO-Net and CPSO-Net achieved a big improvement classification accuracy over the handcraft methods, including CNN, SSRN, ResNet, and DesNet. Therefore, we mainly compared the methods with Auto-CNN, which is an automatically designed method as well. For the University of Pavia dataset, CPSO-Net improved the OA, AA, and Kappa of Auto-CNN by 1.34%, 0.64%, and 0.0185, respectively, and PSO-Net exhibited the best OA, AA, and Kappa, with improvements of 1.78%, 2.31%, and 0.0221 over Auto-CNN. respectively. For the Salinas dataset, PSO-Net and CPSO-Net performed almost the same, and CPSO-Net exhibited the best OA, AA and Kappa, with improvements of 1.22%, 1.95% and 0.0221 over Auto-CNN, respectively. For the Indian Pines dataset, both PSO-Net and CPSO-Net achieved higher classification results than the other algorithms. CPSO-Net exhibited the best AA and Kappa, while PSO-Net exhibited the best OA, with improvements of 1.88%, 0.0171, and 1.37% over Auto-CNN, respectively. For the KSC dataset, similar results to the other datasets were obtained. Obviously, CPSO-Net dramatically accelerated the optimal network generation and achieved quality performance comparable to PSO-Net. For the Salinas, Indian Pines, and KSC datasets, CPSO-Net achieved even better classification performance than PSO-Net, with some improvement over PSO-Net in terms of AA, OA, and Kappa. There may be some significant differences in every class accuracy. For example, CPSO-Net achieved particularly improvement of 12.98% over Auto-CNN in the first-class classification on Salinas. In addition, Table 7 shows the detailed average rankings of the classification accuracy on four datasets through the minimize performance measure based Friedman test[52], which is a statistical test for the homogeneity of multiple samples. From Table 7, we can see that the proposed approaches get almost similar ranking and rank higher than other comparative algorithms in terms of OA, AA, and Kappa.

Table 3. Classification accuracy with bolding the best accuracy based on the biased Pavia dataset, run five times.

Method	CNN	SSRN	ResNet	DesNet	Auto-CNN	PSO-Net	CPSO-Net
OA (%)	91.79 ± 0.98	91.65 ± 2.64	88.67 ± 0.65	91.86 ± 0.23	92.58 ± 0.82	**94.36 ± 0.55**	93.96 ± 1.23
AA (%)	84.59 ± 2.97	89.00 ± 3.27	80.86 ± 1.12	84.86 ± 1.49	87.75 ± 1.23	**90.06 ± 1.02**	88.39 ± 2.11
K×100	89.42 ± 1.58	89.16 ± 3.37	85.60 ± 1.34	89.63 ± 1.87	90.47 ± 1.30	**92.68 ± 1.03**	92.32 ± 1.07
Asphalt	92.11 ± 2.40	95.08 ± 2.46	91.90 ± 2.48	94.32 ± 1.16	95.16 ± 1.38	**95.68 ± 1.35**	95.31 ± 1.48
Meadows	98.88 ± 0.72	96.20 ± 4.20	97.16 ± 0.23	98.38 ± 0.84	98.17 ± 0.79	**99.05 ± 0.48**	98.45 ± 0.59
Gravel	59.99 ± 13.85	69.92 ± 18.36	79.43 ± 8.27	81.87 ± 9.48	82.65 ± 12.93	70.93 ± 14.39	**84.23 ± 13.21**
Trees	**93.35 ± 4.35**	90.84 ± 6.93	52.57 ± 12.76	49.15 ± 8.82	63.49 ± 14.82	76.46 ± 6.32	71.24 ± 5.92
Painted metal sheets	97.10 ± 4.30	99.25 ± 2.01	99.33 ± 0.35	**99.48 ± 0.38**	96.72 ± 2.49	96.45 ± 1.05	98.28 ± 3.16
Bare Soil	93.97 ± 3.24	89.34 ± 10.88	93.42 ± 2.49	96.26 ± 2.83	98.76 ± 0.38	98.84 ± 0.47	**99.54 ± 0.46**
Bitumen	54.23 ± 12.48	82.36 ± 14.01	87.16 ± 1.84	94.24 ± 2.85	95.82 ± 1.37	**97.95 ± 1.25**	97.43 ± 1.85
Self-blocking bricks	88.77 ± 5.64	79.67 ± 18.68	95.42 ± 1.56	97.80 ± 0.37	96.42 ± 2.48	97.22 ± 1.37	**98.42 ± 0.76**
Shadows	82.95 ± 21.60	**98.34 ± 2.11**	31.55 ± 10.26	52.23 ± 11.72	62.74 ± 25.64	77.02 ± 8.29	52.61 ± 16.47
Parameters	141 K	216.5 K	25.6 M	7.3 M	92.4 K	85.7 K	37.7 K

Table 4. Classification accuracy with bolding the best accuracy based on the biased Salinas dataset, run five times.

Method	CNN	SSRN	ResNet	DesNet	Auto-CNN	PSO-Net	CPSO-Net
OA (%)	90.68 ± 1.28	91.16 ± 1.64	91.57 ± 0.74	92.97 ± 0.82	95.42 ± 1.48	96.18 ± 1.08	**96.64 ± 1.48**
AA (%)	88.18 ± 1.25	93.59 ± 2.35	92.80 ± 1.02	94.94 ± 1.14	95.37 ± 2.05	96.52 ± 1.24	**97.32 ± 1.24**
K×100	89.56 ± 1.45	90.16 ± 1.82	90.51 ± 0.95	92.11 ± 0.89	94.09 ± 1.85	95.84 ± 1.57	**96.23 ± 1.15**
Brocoli_green_weeds_1	81.76 ± 6.68	99.59 ± 0.72	98.73 ± 0.36	95.44 ± 3.58	81.69 ± 8.29	84.80 ± 8.35	**94.67 ± 5.33**
Brocoli_green_weeds_2	94.88 ± 6.07	**99.54 ± 1.35**	74.36 ± 0.44	89.69 ± 1.45	97.52 ±0.28	97.53 ± 0.48	96.95 ± 1.21
Fallow	91.35 ± 4.62	82.03 ± 20.05	**100.00 ± 0.00**	95.11 ± 2.13	92.36 ± 1.40	94.22 ± 1.37	97.68 ± 1.04
Fallow_rough_plow	88.31 ± 7.49	98.61 ± 1.57	95.42 ± 2.59	**100.00 ± 0.00**	93.28 ± 1.27	96.45 ± 1.21	96.82 ± 1.60
Fallow_smooth	95.60 ± 4.40	97.13 ± 3.98	96.76 ± 0.33	97.31 ± 0.42	97.36 ± 0.58	**97.62 ± 0.70**	96.93 ± 1.46
Stubble	99.74 ± 0.63	99.81 ± 0.49	99.36 ± 0.63	99.37 ± 0.56	99.78 ± 0.17	99.82 ±0.05	**99.87 ± 0.12**
Celery	97.42 ± 2.03	99.62 ± 0.58	99.45 ± 0.35	99.48 ± 0.27	99.47 ± 0.12	**99.49 ± 0.59**	98.66 ± 0.49
Grapes_untrained	88.48 ± 3.13	81.13 ± 8.86	96.76 ± 1.42	97.16 ± 0.67	95.26 ± 1.21	96.12 ± 2.48	**98.03 ± 3.37**
Soil_vineyard_develop	98.96 ± 1.70	98.96 ± 2.30	97.39 ± 1.35	99.37 ± 0.54	99.36 ± 1.10	**99.38 ± 0.34**	95.42 ± 3.91
Corn_senesced_green_weeds	95.20 ± 5.57	91.58 ± 3.72	93.82 ± 4.48	99.21.00 ± 0.93	99.81 ± 0.25	99.22 ± 0.64	**99.86 ± 0.52**
Lettuce_romaine_4wk	77.33 ± 8.43	83.38 ± 28.67	98.02 ± 0.83	99.86 ± 0.43	98.85 ± 0.12	**99.92 ± 0.13**	95.89 ± 1.14
Lettuce_romaine_5wk	89.66 ± 9.04	98.72 ± 3.02	96.12 ± 0.65	99.45 ± 0.57	92.63 ± 0.64	99.82 ± 1.08	**99.92 ± 1.89**
Lettuce_romaine_6wk	87.92 ± 9.43	95.30 ± 5.09	78.30 ± 6.34	78.81 ± 6.87	95.42 ± 0.72	**96.61 ± 1.46**	95.42 ± 1.79
Lettuce_romaine_7wk	90.53 ± 8.71	96.95 ± 3.15	97.02 ± 0.85	96.50 ± 1.45	93.78 ± 1.03	94.77 ± 1.12	**97.44 ± 3.35**
Vineyard_untrained	81.78 ± 11.24	79.72 ± 7.47	71.30 ± 10.29	74.63 ± 8.27	95.62 ± 2.34	95.89 ± 1.01	**96.55 ± 1.56**
Vineyard_vertical_trellis	61.12 ± 14.38	95.37 ± 2.84	91.96 ± 2.48	**97.65 ± 3.81**	93.79 ± 4.96	92.70 ± 5.23	97.12 ± 6.67
Parameters	193 K	370.3 K	27.5 M	7.6 M	101.8 K	61.6 K	47.1 K

To sum up from the classification accuracy analysis above, we can draw two conclusions. On the one hand, the PSO-Net and CPSO-Net algorithms made significant improvements over the state-of-the-art CNN models designed by hand, including CNN, SSRN, ResNet, and DesNet. Additionally, the proposed approaches are able to find better architectures than Auto-CNN optimized by the gradient descent. On the other hand, the two methods proposed herein achieved almost the same classification performance, while CPSO-Net dramatically reduced the time computation.

Table 5. Classification accuracy with bolding the best accuracy based on the biased Indian Pines dataset, run five times.

Method	CNN	SSRN	ResNet	DesNet	Auto-CNN	PSO-Net	CPSO-Net
OA (%)	85.29 ± 1.45	80.66 ± 2.91	81.61 ± 1.23	87.77 ± 1.17	88.12 ± 1.42	**89.52 ± 0.98**	89.32 ± 1.27
AA (%)	69.94 ± 4.83	65.35 ± 6.85	66.71 ± 5.64	72.13 ±4.76	73.05 ±6.29	74.25 ± 4.04	**74.42 ± 7.26**
K×100	82.70 ± 1.75	77.83 ± 3.37	78.02 ± 1.84	85.34 ± 1.49	85.87 ± 1.94	87.52 ± 1.59	**87.58 ± 1.53**
Alfalfa	35.95 ± 35.52	12.46 ± 22.28	25.39 ± 46.34	**46.72 ± 41.47**	36.63 ± 42.27	38.63 ± 45.42	38.32 ± 39.62
Corn–notill	82.49 ± 14.94	79.05 ± 10.59	87.37 ± 4.26	**96.21 ± 5.36**	86.02 ± 4.93	89.90 ± 2.21	89.95 ± 3.83
Corn–mintill	64.88 ± 14.94	73.92 ± 16.97	62.38 ± 18.94	62.86 ± 22.63	**79.63 ± 19.54**	79.07 ± 26.98	67.46 ± 17.62
Corn	45.67 ± 20.51	38.96 ± 21.69	51.37 ± 26.43	49.06 ± 20.52	54.19 ± 38.16	**72.49 ± 18.64**	68.27 ± 39.48
Grass–pasture	58.91 ± 16.72	75.46 ± 16.00	67.39 ± 20.23	58.33 ± 23.84	74.63 ± 18.92	**74.82 ± 24.85**	68.63 ± 29.94
Grass–trees	89.89 ± 8.48	94.84 ± 5.20	87.57 ± 5.42	94.41 ± 4.26	**94.89 ± 4.37**	89.63 ± 3.62	86.62 ± 7.76
Grass–pasture–mowed	48.26 ± 42.07	33.70 ± 43.84	18.54 ± 43.36	34.83 ± 44.32	56.38 ±45.29	69.73 ± 43.74	**73.84 ± 56.16**
Hay–windrowed	92.54 ± 7.28	99.53 ± 0.69	89.54 ± 4.73	92.41 ± 2.62	98.05 ± 0.75	**98.34 ± 1.37**	96.35 ± 3.64
Oats	24.29 ± 31.89	8.48 ± 13.13	35.84 ± 35.74	**52.63 ± 31.41**	13.86 ± 26.95	19.92 ± 43.87	21.92 ± 43.82
Soybean–notill	86.89 ± 4.65	75.52 ± 8.84	80.13 ± 2.65	86.35 ± 2.73	91.64 ± 3.23	90.42 ± 3.25	**94.87 ± 2.78**
Soybean–mintill	92.72 ± 1.57	82.76 ± 6.90	85.75 ± 5.83	90.61 ± 2.84	92.76 ± 5.54	87.50 ± 7.53	**94.36 ± 1.35**
Soybean–clean	68.58 ± 16.91	68.04 ± 19.14	71.36 ± 22.75	**85.32 ± 19.31**	72.27 ± 21.93	71.89 ± 19.21	73.05 ± 16.95
Wheat	92.99 ± 13.36	**95.41 ± 8.34**	86.73 ± 2.56	71.39 ± 3.75	94.02 ± 5.85	83.21 ± 4.86	82.83 ± 8.42
Woods	98.89 ± 1.28	96.38 ± 3.15	93.79 ± 2.65	95.36 ± 2.79	97.46 ± 1.38	**97.62 ± 1.74**	95.49 ± 1.35
Buildings–grass–trees	**75.44 ± 13.80**	60.64 ± 19.60	64.52 ± 32.28	71.16 ± 21.84	53.72 ± 31.57	54.37 ± 27.46	68.42 ± 32.49
Stone–steel–towers	57.62 ± 26.55	50.40 ± 42.30	59.72 ± 32.48	66.39 ± 27.48	**72.68 ± 27.54**	71.42 ± 16.29	70.34 ± 29.67
parameters	191.4 K	364.2 K	27.4 M	7.6 M	38.4 K	53.6 K	55.1 K

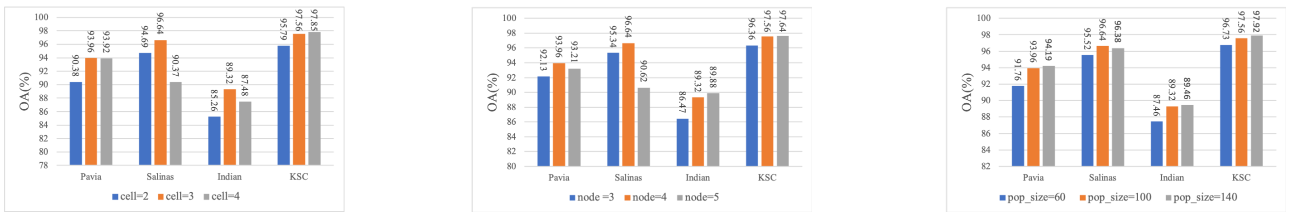
Table 6. Classification accuracy with bolding the best accuracy based on the biased KSC dataset, fun five times.

Method	CNN	SSRN	ResNet	DesNet	Auto-CNN	PSO-Net	CPSO-Net
OA (%)	95.98 ± 0.83	96.77 ± 0.83	95.76 ± 0.72	95.92 ± 1.52	96.04 ± 0.90	97.43 ± 1.13	**97.56 ± 1.69**
AA (%)	94.07 ± 1.39	94.70 ± 0.86	95.12 ± 1.58	94.24 ± 1.46	94.01 ± 1.19	96.21 ± 0.61	**96.41 ± 1.24**
K×100	95.53 ± 0.95	96.41 ± 0.93	95.28 ± 0.63	95.46 ± 0.73	95.64 ± 1.10	97.07 ± 0.77	**97.28 ± 1.15**
Scrub	**99.72 ± 0.29**	97.72 ± 2.16	99.16 ± 1.04	97.89 ± 1.78	94.42 ± 1.54	93.28 ± 2.31	96.22 ± 137
Willow swamp	82.18 ± 9.58	90.35 ± 9.95	**95.40 ± 3.63**	83.12 ± 5.02	83.50 ± 5.26	86.37 ± 6.18	88.64 ± 7.31
Cp hammock	92.80 ± 7.60	95.88 ± 2.52	93.42 ± 2.57	90.53 ± 1.48	89.84 ± 6.45	90.91 ± 5.35	**94.53 ± 6.34**
Slash pine	78.38 ± 7.19	91.68 ± 5.97	95.23 ± 3.29	84.94 ± 3.35	95.03 ± 3.91	**96.18 ± 4.67**	94.84 ± 5.39
Oak/broadleaf	91.57 ± 7.51	81.88 ± 9.97	95.76 ± 4.27	83.86 ± 1.58	85.27 ± 2.57	**96.38 ± 1.39**	87.58 ± 2.58
Hardwood	88.02 ± 8.54	91.55 ± 8.02	97.38 ± 2.56	97.96 ± 2.56	88.47 ± 4.84	96.67 ± 2.12	**97.55 ± 1.79**
Swamp	96.64 ± 7.55	89.91 ± 10.76	89.27 ± 5.73	100.00 ± 0.00	98.67 ± 1.46	98.00±1.23	**100.00 ± 0.00**
Graminoid marsh	95.81 ± 5.42	96.96 ± 4.31	84.37 ± 8.26	98.12 ± 2.49	97.95 ± 2.62	**98.31 ± 1.08**	96.00 ± 4.57
Spartina marsh	98.46 ± 3.19	88.86 ± 0.70	96.23 ± 2.14	99.58 ± 1.12	100.00 ± 0.00	98.23 ± 1.77	**100.00 ± 0.00**
Cattail marsh	**100.00 ± 0.00**	98.16 ± 3.77	94.96 ± 1.50	93.37 ± 1.48	92.45 ± 1.23	99.65 ± 0.68	98.56 ± 2.49
Salt marsh	100.00 ± 0.00	98.28 ± 2.83	96.25 ± 1.77	96.50 ± 1.94	100.00 ± 0.00	**100.00 ± 0.00**	100.00 ± 0.00
Mud flats	99.33 ± 1.07	99.97 ± 1.63	99.13 ± 0.92	99.25 ± 1.84	98.69 ± 1.09	99.36 ± 0.69	**99.44 ± 0.56**
Water	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
Parameters	178.9 K	327.2 K	26.7 M	7.5 M	100.4 K	69.4 K	60.1 K

Table 7. Average rankings of Friedman test on four datasets in terms of OA, AA and Kappa, bold the best results.

Method	CNN	SSRN	ResNet	DesNet	Auto-CNN	PSO-Net	CPSO-Net
Ranking (OA)	2.5	2.5	1.75	3.5	4.75	**6.5**	**6.5**
Ranking (OA)	2	3.5	2.5	3.5	3.75	6.25	**6.5**
Ranking (Kappa)	2.5	1.75	1.75	4	5	**6.5**	**6.5**

In addiction, taking CPSO-Net as example, we made a sensitivity hyperparameter analysis of the cell numbers with an additional two and four cells, intermediate nodes with an additional three and five nodes, and a population size with an additional 60 and 140. From Figure 6a,b, it can be seen that fewer number of cells or intermediate nodes achieved a lower OA, but the models with a higher number of cells or intermediate nodes did not necessarily lead to better performance. The main reason is that the models with simple architectures may not be able to extract deep features, and complex models may cause an overfitting problem with limited training samples. From Figure 6c, we can see that a small population may make architectures fall into the local optimal, but a large population will not greatly improve the classification performance when spending much more time.

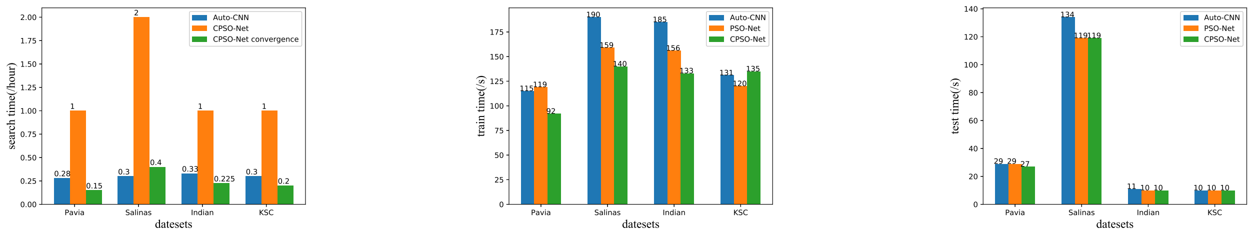


(**a**) Cell (**b**) Node (**c**) Population size

Figure 6. Classification results of different models searched by CPSO-Net with different numbers of cells, intermediate nodes, and population sizes based on four biased hyperspectral image (HSI) datasets.

4.4.2. Complexity Analysis

To analyze the complexity of the method we proposed, the results of several trainable parameters compared to both human-designed and automatically designed CNNs are shown in the bottom row of Tables 3–6, and the time consumed compared to the peer automatically designed method are shown in Figure 7. From Tables 3–6, it is obvious that the models searched by PSO-Net and CPSO-Net achieved better results with the state-of-the-art CNNs designed by human experts while having significantly less parameters, which means less time and computational complexity to train the CNN models. There are two main reasons for the above results. First, a CNN model with a high number of trainable parameters may cause an overfitting problem with limited training samples. Second, automatically designed methods can search CNNs more suitably for specific HSI datasets. For the automatically designed methods Auto-CNN, PSO-Net, and CPSO-Net, even though they had the same search space, CPSO-Net and PSO-Net performed better than Auto-CNN. However, there was no consistent pattern between the number of parameters of the models searched by the three automatically designed methods.



(**a**) Search time (**b**) Train time (**c**) Test time

Figure 7. Time consumed in the different stages of the different automatically designed CNNs methods based on four biased HSI datasets.

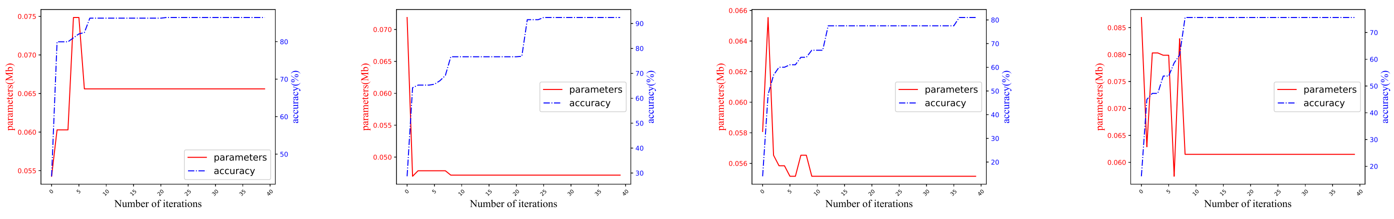
To further compare the search effectiveness of these automatically designed methods, Figure 7 shows the time consumed in the different stages of the peer automatically designed CNN methods based on four biased HSI datasets. PSO-Net consumed approximately 15 h to converge, which is a significant time compared to the other two automatically designed methods. Therefore, Figure 7a only shows the search time of Auto-CNN, CPSO-Net, and CPSO-Net convergence. From Tables 3–6 and Figure 7a, we can see that CPSO-Net and PSO-Net achieved better performance with a big improvement for Auto-CNN, but CPSO-Net dramatically reduced the search time by the weight-sharing parameters from the SuperNet compared to PSO-Net. The time of the total iteration of CPSO-Net consumed more time than Auto-CNN, while it converged at six, eight, nine, and eight iterations (the detailed convergence analysis can be seen is Section 4.4.3) based on Pavia, Salinas, Indian Pines, and KSC, respectively. CPSO-Net only consumed approximately 15 min to converge, which is almost close to and even less than Auto-CNN. From Figure 7b, we can see that the training time of the CNNs searched by CPSO-Net was less than that of PSO-Net and Auto-CNN based on the Pavia, Salinas, and Indian Pines datasets. From Figure 7c, the three automatic-designed method, i.e., Auto-CNN, PSO-Net, and CPSO-Net, consumed almost the same time to test all the data of the CNNs searched.

To sum up, from Figure 7, we can see that PSO-Net consumed a significant amount of time to converge, while CPSO-Net dramatically accelerated the search generation. The time CPSO-Net consumed to converge was almost same as and even less than Auto-CNN.

4.4.3. Convergence Analysis

To dramatically accelerate optimal network generation, CPSO-Net shared the weights of SuperNet for all particles based on PSO-Net and achieved quality performance comparable to PSO-Net. Therefore, we mainly analyze the convergence of CPSO-Net in this section. The convergence of the architectures was different from the other solutions, and it is not enough only to consider the fitness of particles. The fitness was determined by the HSI classification accuracy of the optimized neural network, which was composed of the architectures and weight parameters. However, the number of parameters of the architectures and the position of the particles were only related to architecture. Therefore, the architecture convergence analysis of CPSO-Net took the number of parameters of the architectures and the position of all the particles as important criteria.

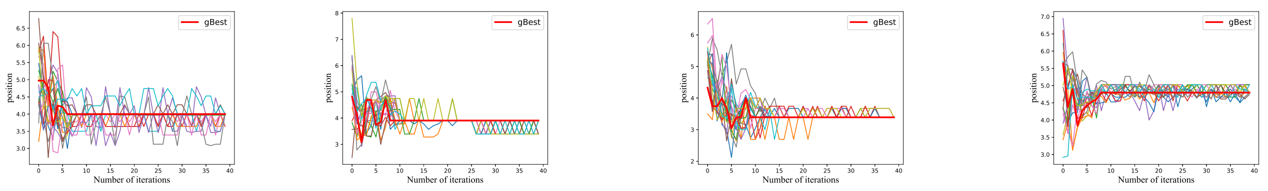
The number of parameters of the architectures were closely related to the operations in said architectures. With the alter of the operations in the architectures, the number of parameters changed accordingly, representing the architectures converging when the number of parameters remained the same during the generations. Figure 8 shows the accuracy and number of parameters of *gBest* during iterations based on four datasets by CPSO-Net. From the number of parameters of *gBest*, we can see that the architectures converged at six, eight, nine, and eight iterations based on the Pavia, Salinas, Indian Pines, and KSC datasets, respectively. The *gBest* accuracy of the validation dataset still increased after the convergence of architectures. The main reason is that the parameters of the architectures inherited from the SuperNet continued to be optimized, while SuperNet was trained until reaching the maximum iterations.



(**a**) Pavia (**b**) Slinas (**c**) Indian Pines (**d**) KSC

Figure 8. The accuracy and parameters of *gBest* during the iterations based on four biased datasets by CPSO-Net.

We further visualized the position of 20 particles selected randomly and *gBest* during the iterations based on four datasets by CPSO-Net in Figure 9. The position of the particles was projected from the array (encoding from the architectures) into one-dimensional space using principal component analysis (PCA). As we can see, all particles divided globally at initialization, gradually tended toward focusing on global exploration with the iteration, toward the best particle position found thus far, and the position of *gBest* converged at the same iteration as the number of parameters of *gBest*, as shown in Figure 8.

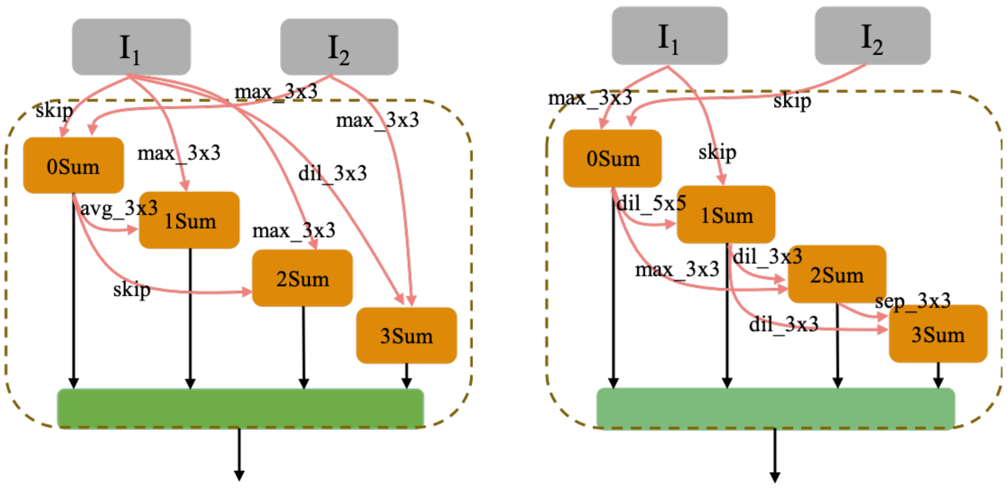


(**a**) Pavia (**b**) Slinas (**c**) Indian Pines (**d**) KSC

Figure 9. Positions of 20 particles and *gBest* during the iterations based on four biased HSI datasets by CPSO-Net.

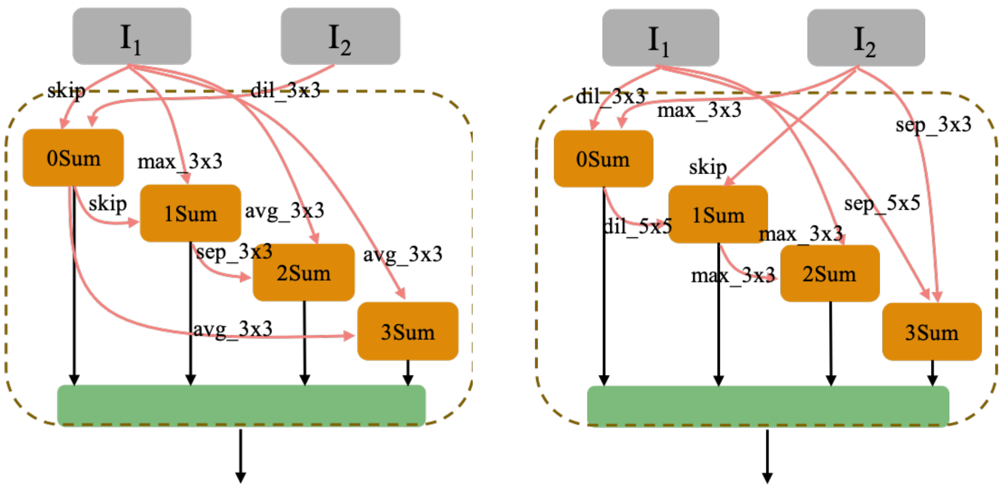
4.4.4. Optimal Architecture

To find the optimal architecture is the main purpose of the method we proposed, the optimal architecture can be searched at the end of the algorithms. After searching, the optimal architecture can be used to classify all the data. Herein, we took PSO-Net based on biased the University of Pavia dataset and CPSO-Net based on the biased Salinas dataset as examples. The architectures of the optimal CNN architectures found by the proposed PSO-Net and CPSO-Net are shown in Figures 10 and 11. Compared to the handcraft CNNs, PSO-Net and CPSO-Net designed the optimal architectures in a predefined search space without professional knowledge of human experts and without spending significant time on repeated debugging. The search space is constructed by incorporating typical properties of successful architectures designed by hand. Therefore, the optimal architectures searched includes some typical properties of handcraft CNNs. For example, inspired by the ResNet, the optimal architectures in Figures 10 and 11 introduced skip connections.



(**a**) Normal cell (**b**) Reduction cell

Figure 10. Optimal architecture of PSO-Net for the biased Pavia dataset.



(**a**) Normal cell (**b**) Reduction cell

Figure 11. Optimal architecture of CPSO-Net for the biased Salinas dataset.

4.4.5. Classification Maps

To clearly represent the classification results, we plotted the whole image classification maps of all the models. The classification maps obtained by the different models based on four biased datasets are shown in Figures 12–15. We evaluated the classification accuracies from the noise scatter in the visual perspective. From the results, we can see that CPSO-Net achieved less scatter in the class with the largest area compared to the other models, which means that CPSO-Net can obtain more precise classification accuracy in this class. From the resulting images, we figured out how the different classification methods affected the classification results. Obviously, the automatically designed CNNs had less noise than the handcrafted CNN model, which demonstrates the feasibility of designing CNN models automatically. It is obvious that the CNN search by the methods we proposed had less scatter than the automatically design method Auto-CNN for HSI classification, which further demonstrates the effectiveness of PSO-Net and CPSO-Net.

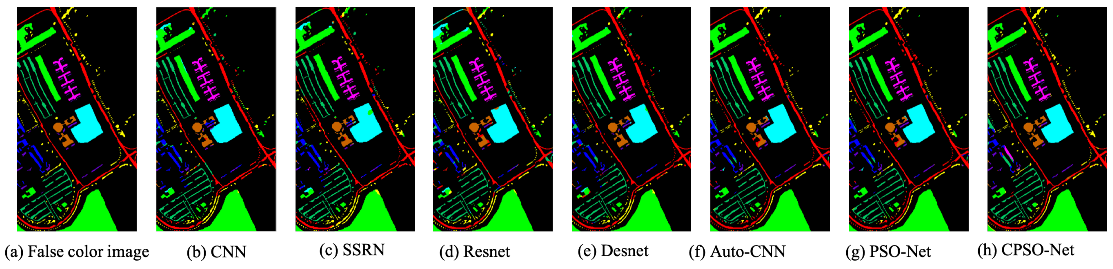


Figure 12. Classification Maps on biased Pavia dataset.

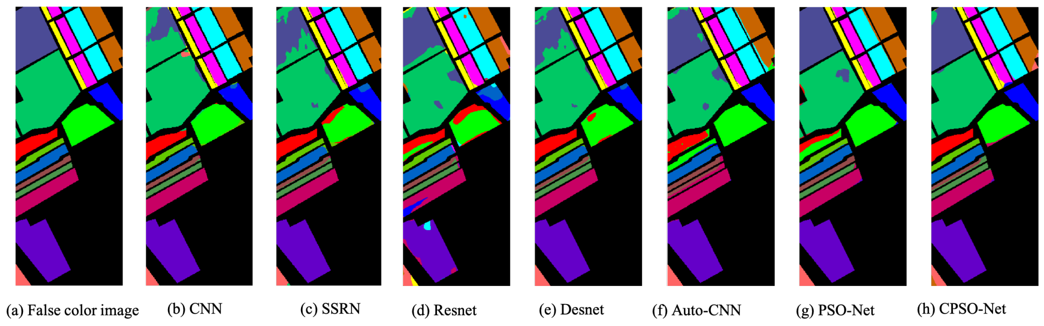


Figure 13. Classification maps based on the biased Salinas dataset.

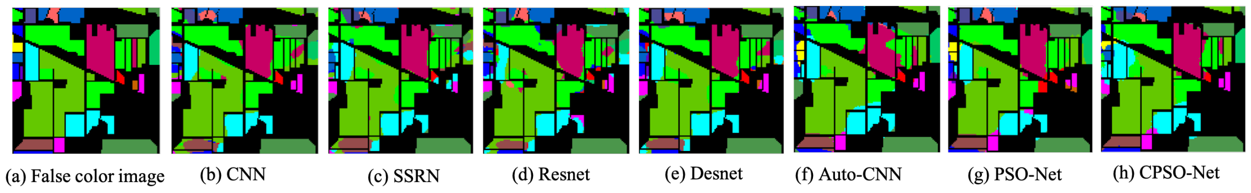


Figure 14. Classification maps based on the biased Indian Pines dataset.

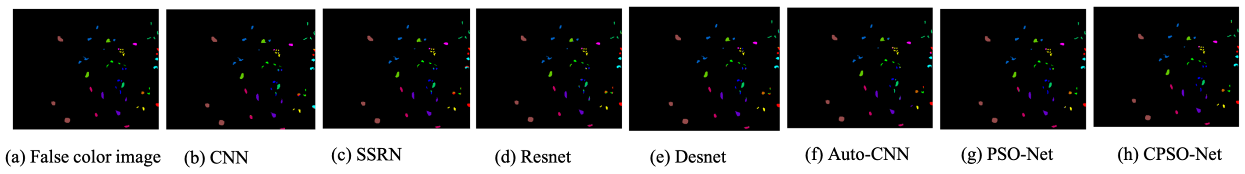


Figure 15. Classification maps based on the biased KSC dataset.

4.5. Unbiased HSI Classification

There may be pixel overlapping and training-test information leakage problems on biased datasets which lead to over-optimistic results for spectral-spatial methods. Therefore, to further demonstrate the effectiveness of PSO-Net and CPSO-Net, we conducted some more experiments on the unbiased hyperspectral datasets and tested the two methods in comparison to CNN, SSRN, ResNet, DesNet, and Auto-CNN based on the four biased HSI datasets of Pavia, Salinas, Indian Pines, and KSC. There are some methods to split training and test datasets without pixels overlapping and training-test information leakage, such as patch-based [53] and set-to-sets distance based [54] methods.

For the unbiased datasets, samples with 32 × 32 slides were not enough for dividing into the training and test datasets, so we chose 7 × 7 neighborhoods of a pixel according to the patch-based [53] method, and none of the samples were overlapping. When the All other experimental settings were the same with the aforementioned biased datasets.

From Tables 8–11, it is obvious to see that PSO-Net and CPSO-Net performed much better than the other models, and the improvements of PSO-Net and CPSO-Net were significant. In particular, the performance of CPSO-Net was better and consumed much less time than PSO-Net. However, the results of different unbiased spatial-based HSI classification methods were all lower than the performance of the biased methods.

Table 8. Classification accuracy with bolding the best accuracy based on the unbiased Pavia dataset, run five times.

Method	CNN	SSRN	ResNet	DesNet	Auto-CNN	PSO-Net	CPSO-Net
OA (%)	83.12 ± 2.69	82.46 ± 2.58	79.93 ± 0.93	82.99 ± 1.32	86.05 ± 1.43	86.70 ± 1.26	**87.13 ± 0.76**
AA (%)	76.35 ± 2.46	78.63 ± 3.59	73.82 ± 1.57	76.25 ± 1.46	79.50 ± 1.78	81.46 ± 1.34	**81.98 ± 1.45**
K×100	78.42 ± 2.83	78.462 ± 3.48	73.35 ± 1.86	77.33 ± 1.83	81.44 ± 1.47	81.94 ± 1.39	**82.92 ± 1.14**
Asphalt	79.12 ± 2.69	83.45 ± 2.34	84.12 ± 2.58	86.71 ± 1.78	**93.99 ± 1.96**	91.31 ± 1.67	94.14 ± 2.79
Meadows	85.24 ± 1.53	93.75 ± 2.73	94.35 ± 2.21	95.50 ± 1.93	95.00 ± 1.36	**95.74 ±1.38**	95.70 ± 1.39
Gravel	53.14 ± 16.58	59.84 ± 18.43	**80.37 ± 8.35**	71.29 ± 9.68	51.91 ± 17.43	50.17 ± 15.72	50.63 ± 13.23
Trees	83.57 ± 5.18	82.69 ± 6.54	78.45 ± 11.35	85.92 ± 8.69	87.17 ± 5.82	88.60 ± 5.35	**89.37 ± 5.24**
Painted metal sheets	86.95 ± 5.51	89.95 ± 2.41	74.38 ± 3.15	71.76 ± 5.28	94.81 ± 3.78	**99.58 ± 1.05**	97.34 ± 3.57
Bare soil	62.32 ± 3.58	69.78 ± 10.73	52.50 ± 2.33	52.52 ± 2.56	66.44 ± 3.23	66.10 ± 4.59	**67.31 ± 5.35**
Bitumen	63.74 ± 16.72	62.32 ± 14.42	61.77 ± 1.97	70.78 ± 2.59	57.68 ± 1.37	**71.26 ± 1.64**	65.80 ± 1.46
Self-blocking bricks	75.63 ± 7.44	71.85 ± 18.51	59.82 ± 1.34	65.39 ± 1.68	72.10 ± 2.31	71.73 ± 1.75	**80.07 ± 2.73**
Shadows	97.49 ± 19.27	98.84 ± 2.24	78.62 ± 11.24	86.38 ± 11.72	96.45 ± 15.73	**98.73 ± 7.94**	97.51 ± 8.47

Table 9. Classification accuracy with bolding the best accuracy based on the unbiased Salinas dataset, run five times.

Method	CNN	SSRN	ResNet	DesNet	Auto-CNN	PSO-Net	CPSO-Net
OA (%)	80.36 ± 1.69	81.21 ± 1.58	83.13 ± 0.97	81.99 ± 1.46	84.41 ± 1.53	88.83 ± 1.36	**89.75 ± 1.72**
AA (%)	82.86 ± 1.57	84.62 ± 2.58	80.37 ± 1.48	81.58 ± 1.39	87.21 ± 1.83	88.90 ± 1.62	**90.53 ± 1.43**
K×100	78.62 ± 1.72	80.37 ± 1.49	81.20 ± 1.24	79.91 ± 1.34	82.69 ± 1.67	87.54 ± 1.38	**88.58 ± 1.84**
Brocoli_green_weeds_1	92.46 ± 3.49	73.63 ± 4.7	58.93 ± 3.57	38.13 ± 19.45	60.22 ± 8.29	86.19 ± 8.84	**97.50 ± 7.39**
Brocoli_green_weeds_2	92.73 ± 6.73	91.52 ± 1.58	93.03 ± 1.29	96.65 ± 1.38	98.66 ± 1.52	**99.55 ± 1.35**	99.36 ± 1.20
Fallow	86.63 ± 4.68	82.84 ± 20.58	47.17 ± 10.34	79.07 ± 2.47	**87.06 ± 1.39**	76.72 ± 1.67	84.81 ± 1.38
Fallow_rough_plow	92.52 ± 7.57	91.37 ± 1.82	93.26 ± 2.93	92.61 ± 1.58	97.39 ± 1.97	96.65 ± 1.29	**99.60 ± 1.56**
Fallow_smooth	87.43 ±4.83	92.59 ± 3.68	94.28 ± 1.23	93.59 ± 1.48	**94.58 ± 1.73**	**90.78 ± 3.72**	92.22 ± 2.39
Stubble	93.52 ± 0.38	93.69 ± 0.60	97.85 ± 1.46	96.84 ± 1.44	99.34 ± 1.26	98.85 ± 1.37	**99.37 ± 1.05**
Celery	93.34 ± 2.85	94.77 ± 0.55	97.70 ± 1.26	97.52 ± 1.27	**98.87 ± 1.24**	99.07 ± 0.95	99.82 ± 0.73
Grapes_untrained	70.42 ± 3.58	73.37 ± 8.34	78.54 ± 1.96	75.71 ± 1.84	65.87 ± 5.38	**86.60 ± 2.59**	84.70 ± 4.45
Soil_vineyard_develop	95.34 ± 1.73	94.74 ± 2.37	95.10 ± 1.68	96.53 ± 1.72	97.01 ± 1.37	**98.64 ± 1.46**	97.33 ± 3.80
Corn_senesced_green_weeds	72.74 ± 5.79	74.64 ± 3.75	89.46 ± 2.57	87.41 ± 1.25	88.36 ± 3.21	**91.19 ± 2.52**	89.42 ± 1.26
Lettuce_romaine_4wk	68.63 ± 8.70	84.73 ± 28.82	38.58 ± 2.67	34.33 ± 8.39	71.34 ± 1.42	63.45 ± 3.73	**77.94 ± 1.58**
Lettuce_romaine_5wk	82.57 ± 9.52	88.62 ± 3.47	98.37 ± 1.36	98.89 ± 1.67	97.03 ± 1.45	**97.50 ± 1.28**	88.96 ± 6.96
Lettuce_romaine_6wk	84.97 ± 9.23	95.30 ± 5.68	98.53 ± 6.34	98.57 ± 1.52	88.77 ± 2.71	84.97 ± 3.49	**88.95 ± 2.02**
Lettuce_romaine_7wk	82.47 ± 8.78	83.89 ± 3.29	70.77 ± 1.46	79.33 ± 1.52	**96.68 ± 1.14**	93.24 ± 1.38	79.60 ± 2.13
Vineyard_untrained	54.69 ± 11.44	57.72 ± 7.45	72.91 ± 10.58	72.76 ± 2.62	74.32 ± 2.37	68.77 ± 1.48	**74.45 ± 1.36**
Vineyard_vertical_trellis	74.27 ± 14.83	79.57 ± 2.85	61.53 ± 2.59	67.44 ± 4.19	79.86 ± 4.85	90.24 ± 5.94	**94.45 ± 6.48**

Table 10. Classification accuracy with bolding the best accuracy based on the unbiased Indian Pines dataset, run five times.

Method	CNN	SSRN	ResNet	DesNet	Auto-CNN	PSO-Net	CPSO-Net
OA (%)	58.11 ± 1.94	61.83 ± 1.62	58.46 ± 1.23	58.59 ± 1.46	68.23 ± 1.73	**70.61 ± 0.73**	69.89 ± 1.48
AA (%)	45.23 ± 6.93	49.24 ± 6.84	40.57 ± 6.64	47.28 ± 7.37	54.63 ± 6.52	56.47 ± 5.73	**60.80 ± 7.42**
K×100	52.14 ± 1.46	55.25 ± 3.47	51.35 ± 1.84	52.77 ± 2.74	63.55 ± 1.42	66.90 ± 1.27	**66.79 ± 1.73**
Alfalfa	13.55 ± 35.45	12.78 ± 22.61	13.95 ± 42.35	20.23 ± 43.28	14.78 ± 49.57	21.74 ± 43.63	**24.78 ± 39.35**
Corn–notill	32.47 ± 14.73	35.84 ± 10.48	34.70 ± 3.73	24.93 ± 17.53	53.29 ± 4.93	50.83 ± 5.52	**65.20 ± 3.53**
Corn–mintill	40.54 ± 14.48	42.53 ± 16.68	26.06 ± 29.25	34.55± 13.63	40.40 ± 29.52	49.75 ± 26.42	**51.59 ± 17.24**
Corn	5.41 ± 20.53	30.96 ± 21.41	15.13 ± 12.25	37.39 ± 29.53	34.46 ± 38.74	**53.08 ± 24.65**	40.17 ± 39.51
Grass–pasture	55.74 ± 16.53	57.96 ± 16.38	31.89 ± 32.46	65.19 ± 17.45	76.42 ± 18.27	68.62 ± 23.53	**81.86 ± 29.96**

Table 10. *Cont.*

Method	CNN	SSRN	ResNet	DesNet	Auto-CNN	PSO-Net	CPSO-Net
Grass-trees	88.52 ± 8.92	82.94 ± 5.79	95.0 6± 3.43	97.19 ± 1.37	90.04 ± 4.74	**92.88 ± 3.74**	92.77 ± 7.82
Grass–pasture–mowed	0.00 ± 0.00	8.92 ± 43.92	0.00 ± 0.00	0.00 ± 0.00	8.58 ± 45.93	2.84 ± 13.36	**16.42 ± 56.51**
Hay–windrowed	89.92 ± 7.71	93.74 ± 1.79	86.06 ± 1.85	96.06 ± 1.94	**97.88 ± 1.79**	95.49 ± 1.52	90.04 ± 3.84
Oats	0.00 ± 0.00	1.68 ± 13.68	0.00 ± 0.00	0.00 ± 0.00	8.80 ± 36.95	0.00 ± 0.00	**22.00 ± 23.2**
Soybean–notill	49.62 ± 4.48	57.05 ± 8.64	25.35 ± 4.75	36.95 ± 7.43	**75.41 ± 3.67**	64.65 ± 4.52	65.33 ± 2.83
Soybean–mintill	69.62 ± 1.70	69.84 ± 6.68	78.70 ± 3.85	66.44 ± 9.42	75.32 ± 5.52	**81.96 ± 7.94**	66.91 ± 1.37
Soybean–clean	31.31 ± 16.49	33.94 ± 19.54	24.47 ± 25.31	46.13 ± 13.35	38.20 ± 21.64	37.73 ± 19.47	**40.45 ± 16.93**
Wheat	78.51 ± 13.72	78.81 ± 8.89	54.03 ± 4.49	69.40 ± 7.42	42.54 ± 5.52	75.41 ± 4.63	**83.32 ± 8.62**
Woods	88.52 ± 1.69	85.06 ± 3.19	87.56 ± 2.57	85.54 ± 1.82	87.88 ± 1.31	**95.96 ± 1.42**	92.76 ± 1.73
Buildings–grass–trees	23.42 ± 13.62	30.85 ± 19.28	30.13 ± 23.17	31.78 ± 27.48	48.92 ± 31.62	41.54 ± 27.96	**54.71 ± 27.63**
Stone–steel–towers	55.41 ± 26.37	64.96 ± 42.35	46.03 ± 23.48	42.70 ± 21.47	73.76 ± 27.26	71.12 ± 16.93	**94.62 ± 22.36**

Table 11. Classification accuracy with bolding the best accuracy based on the unbiased KSC dataset, run five times.

Method	CNN	SSRN	ResNet	DesNet	Auto-CNN	PSO-Net	CPSO-Net
OA (%)	68.46 ± 1.56	69.52 ± 1.72	68.19 ± 1.83	73.57 ± 1.74	74.41 ± 1.37	82.50 ± 1.47	**83.77 ± 1.74**
AA (%)	65.46 ± 1.46	68.35 ± 1.47	59.59 ± 1.94	64.10 ± 1.64	66.41 ± 1.76	74.98 ± 1.85	**76.55 ± 1.87**
K×100	64.31 ± 1.68	68.61 ± 1.41	64.01 ± 1.84	70.65 ± 1.79	71.46 ± 1.86	80.49 ± 1.28	**81.96 ± 1.85**
Scrub	91.72 ± 2.31	93.72 ± 2.57	69.69 ± 1.64	79.61 ± 1.94	91.11 ± 1.84	**98.40 ± 1.49**	96.54 ± 2.97
Willow swamp	68.18 ± 9.35	71.28 ± 9.51	37.73 ± 3.74	33.76 ± 5.64	40.41 ± 5.75	76.63 ± 7.96	**80.25 ± 6.24**
Cp hammock	70.80 ± 7.04	72.94 ± 2.79	73.86 ± 2.92	73.44 ± 2.53	65.69 ± 6.74	**74.04 ± 6.87**	73.71 ± 5.29
Slash pine	23.24 ± 7.52	26.90 ± 5.47	20.88 ± 10.38	32.26 ± 12.45	27.62 ± 11.63	38.02 ± 15.37	**56.42 ± 4.73**
Oak/broadleaf	43.84 ± 7.15	46.67 ± 4.73	43.69 ± 4.90	36.85 ± 24.74	63.81 ± 2.47	**42.86 ± 8.95**	39.38 ± 11.33
Hardwood	22.29 ± 5.49	21.72 ± 6.68	20.63 ± 7.92	23.50 ± 5.63	52.40 ± 4.62	**63.24 ± 1.36**	52.21 ± 2.73
Swamp	62.69 ± 7.69	69.27 ± 5.36	61.30 ± 3.17	68.36 ± 2.65	70.29 ± 1.35	**76.38 ± 3.52**	62.67 ± 1.48
Graminoid marsh	75.48 ± 5.73	76.40 ± 4.38	75.56 ± 8.25	78.14 ± 1.45	45.86 ± 12.65	56.60 ± 6.42	**83.26 ± 1.34**
Spartina marsh	59.91 ± 3.29	64.92 ± 1.52	58.60 ± 2.47	66.45 ± 1.86	95.77 ± 1.42	**98.38 ± 1.69**	93.42 ± 1.83
Cattail marsh	76.15 ± 2.83	76.31 ± 3.27	71.47 ± 1.82	78.01 ± 1.89	41.59 ± 7.73	70.89 ± 4.84	**84.00 ± 2.63**
Salt marsh	81.16 ± 2.37	86.27 ± 2.63	82.62 ± 1.94	87.41 ± 1.28	97.42 ± 1.45	96.43 ± 1.95	**98.52 ± 1.85**
Mud flats	80.82 ± 1.48	82.82 ± 1.47	58.75 ± 1.82	80.73 ± 1.38	72.73 ± 1.09	**84.21 ± 1.76**	83.82 ± 3.43
Water	94.72 ± 3.52	99.38 ± 1.52	99.89 ± 1.23	94.78.00 ± 2.53	98.64 ± 1.24	**98.66 ± 1.96**	90.95 ± 1.74

5. Conclusions

In this paper, we proposed two novel evolutionary-based search methods, PSO-Net and CPSO-Net, which use particle swarm optimization as the search strategy to search for the optimal architecture. For PSO-Net, the gradient descent method was used to obtain the weight parameters for all architectures separately, and PSO was used to optimize the architectures and search for the optimal architecture until reaching the maximum number of iterations. To accelerate the optimal network generation, CPSO-Net made an improvement based on the PSO-Net parameter optimization of architectures. For CPSO-Net, we maintained a SuperNet with all candidate operations and shared the weight parameters for all individuals. There was no need to train each particle from the beginning to the end, so the search time was reduced greatly. Our results showed that PSO-Net and CPSO-Net can achieve better classification accuracies compared to the state-of-the-art algorithms of the automatic design of CNNs for HSI classification, and CPSO-Net can obtain almost the same results with much less time consumed than PSO-Net.

For future works, we will consider multiple complementary objectives, such as the number of parameters and the classification accuracy to achieve the Pareto optimal front. To further demonstrate the effectiveness of the methods we proposed herein, we will test these methods on the same common image classification datasets.

Author Contributions: Conceptualization, X.L. and C.Z.; methodology, X.L.; software, X.L. and C.Z.; validation, X.L., C.Z., and Z.C.; formal analysis, X.L.; investigation, C.Z.; resources, X.L. and Z.C.; data curation, C.Z.; writing—original draft preparation, C.Z.; writing—review and editing, X.L., Z.C., J.Y., Z.Z., and X.G.; visualization, C.Z.; supervision, X.L.; project administration, X.L. and Z.C.; funding acquisition, X.L. All authors read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Nature Science Foundation of China (grant numbers 61973285, 62076226, 61873249, and 61773355) and the Open Research Project of the Hubei Key Laboratory of Intelligent Geo-Information Processing (grant number KLIGIP-2019A04).

Acknowledgments: The authors would like to thank M. Gra$\overline{n}$a, M.A. Veganzons, and B. Ayerdi for the hyperspectral image datasets used in this paper, and Y. Chen for the Auto-CNN source codes.

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

References

1. Camps-Valls, G.; Tuia, D.; Bruzzone, L.; Benediktsson, J.A. Advances in hyperspectral image classification: Earth monitoring with statistical learning methods. *IEEE Signal Process. Mag.* **2014**, *31*, 45–54. [CrossRef]
2. Yi, C.; Zhao, Y.Q.; Chan, J.C.W. Hyperspectral image super-resolution based on spatial and spectral correlation fusion. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 4165–4177. [CrossRef]
3. Kang, X.; Zhang, X.; Li, S.; Li, K.; Li, J.; Benediktsson, J.A. Hyperspectral anomaly detection with attribute and edge-preserving filters. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 5600–5611. [CrossRef]
4. Haut, J.M.; Bernabé, S.; Paoletti, M.E.; Fernandez-Beltran, R.; Plaza, A.; Plaza, J. LowChigh-power consumption architectures for deep-learning models applied to hyperspectral image classification. *IEEE Geosci. Remote Sens. Lett.* **2018**, *16*, 776–780. [CrossRef]
5. Teke, M.; Deveci, H.S.; Haliloglu, O.; Gürbüz, S.Z.; Sakarya, U. A short survey of hyperspectral remote sensing applications in agriculture. In Proceedings of the 2013 6th International Conference on Recent Advances in Space Technologies (RAST), Istanbul, Turkey, 16 August 2013; pp. 171–176.
6. Shang, X.; Chisholm, L.A. Classification of australian native forest species using hyperspectral remote sensing and machine-learning classification algorithms. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2014**, *7*, 2481–2489. [CrossRef]
7. Han, Y.; Li, J.; Zhang, Y.; Hong, Z.; Wang, J. Sea ice detection based on an improved similarity measurement method using hyperspectral data. *Sensors* **2017**, *17*, 1124.
8. El-Sharkawy, Y.H.; Elbasuney, S. Hyperspectral imaging: Anew prospective for remote recognition of explosive materials. *Remote Sens. Appl. Soc. Environ.* **2019**, *13*, 31–38. [CrossRef]
9. Li, Y.; Zhang, H.; Shen, Q. Spectral-spatial classification of hyperspectral imagery with 3d convolutional neural network. *Remote Sens.* **2017**, *9*, 67. [CrossRef]
10. Luo, Y.; Zou, J.; Yao, C.; Zhao, X.; Li, T.; Bai, G. Hsi-cnn: A novel convolution neural network for hyperspectral image. In Proceedings of the International Conference on Audio, Language and Image Processing (ICALIP), Shanghai, China, 16–17 July 2018; pp. 464–469.
11. Santara, A.; Mani, K.; Hatwar, P.; Singh, A.; Garg, A.; Padia, K.; Mitra, P. BASS Net: Band-adaptive spectral-spatial feature learning neural network for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 5293–5301. [CrossRef]
12. Cai, Y.; Liu, X.; Cai, Z. BS-Nets: An end-to-end framework for band selection of hyperspectral image. *IEEE Trans. Geosci. Remote Sens.* **2020**, *58*, 1969–1984. [CrossRef]
13. Cai, Y.; Zhang, Z.; Cai, Z.; Liu, X.; Jiang, X.; Yan, Q. Graph convolutional subspace clustering: A robust subspace clustering framework for hyperspectral image. *IEEE Trans. Geosci. Remote Sens.* **2020**. [CrossRef]
14. Feng, J.; Wang, L.; Yu, H.; Jiao, L.; Zhang, X. Divide-and-conquer dual-architecture convolutional neural network for classification of hyperspectral images. *Remote Sens.* **2019**, *11*, 484. [CrossRef]
15. Charmisha, K.; Sowmya, V.; Soman, K. Dimensionally reduced features for hyperspectral image classification using deep learning. In Proceedings of the International Conference on Communications and Cyber Physical Engineering (ICCCE), Hyderabad, India, 24–25 January 2018; pp. 171–179.
16. Du, J.; Li, Z. A hyperspectral target detection framework with subtraction pixel pair features. *IEEE Access.* **2018**, *6*, 45562–45577. [CrossRef]
17. Jia, P.; Zhang, M.; Yu, W.; Shen, F.; Shen, Y. Convolutional neural network based classification for hyperspectral data. In Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Beijing, China, 10–15 July 2016; pp. 5075–5078.
18. Haut, J.; Paoletti, M.; Plaza, J.; Plaza, A. Hyperspectral image classification using random occlusion data augmentation. *IEEE Geosci. Remote Sens. Lett.* **2019**, *16*, 1751–1755. [CrossRef]
19. Mei, S.; Ji, J.; Geng, Y.; Zhang, Z.; Li, X.; Du, Q. Unsupervised spatial-spectral feature learning by 3d convolutional autoencoder for hyperspectral classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 6808–6820. [CrossRef]
20. 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]
21. Paoletti, M.; Haut, J.; Plaza, J.; Plaza, A. Deep&Dense convolutional neural network for hyperspectral image classification. *Remote Sens.* **2018**, *10*, 1454.
22. 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.* **2018**, *56*, 847–858. [CrossRef]
23. Fang, B.; Li, Y.; Zhang, H.; Chan, J. Hyperspectral images classification based on dense convolutional networks with spectral-wise attention mechanism. *Remote Sens.* **2019**, *11*, 159. [CrossRef]

24. 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]
25. Safari, K.; Prasad, S.; Labate, D. A multiscale deep learning approach for high-resolution hyperspectral image classification. *IEEE Geosci. Remote Sens Lett.* **2021**, *18*, 167–171. [CrossRef]
26. Shao, 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.
27. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
28. He, X.; Zhao, K.; Chu, X. AutoML: A survey of the state-of-the-art. *Knowl. Based Syst.* **2021**, *212*, 1–27. [CrossRef]
29. Elsken, T.; Metzen, J.H.; Hutter, F. Neural architecture search: A survey. *J. Mach. Learn. Res.* **2019**, *20*, 1–21.
30. Zoph, B.; Le, Q.V. Neural architecture search with reinforcement learning. *arXiv* **2017**, arXiv:1611.01578.
31. Chen, Y.; Zhu, K.; Zhu, L.; He, X.; Ghamisi, G.; Benediktsson, J.A. Automatic design of convolutional neural network for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 7048–7066. [CrossRef]
32. Junior , F.; Erivaldo, F.; Yen,G. Particle swarm optimization of deep neural networks architectures for image classification. *Swarm Evol. Comput.* **2019**, *49*, 62–74. [CrossRef]
33. Real, E.; Aggarwal, A.; Huang, Y.; Le, Q.V. Regularized evolution for image classifier architecture search. In Proceedings of the Conference on Artificial Intelligence (AAAI), Honolulu, HI, USA, 27 January–1 February 2019; pp. 4780–4789.
34. Stanley, K.O.; Miikkulainen, R. Evolving neural networks through augmenting topologies. *IEEE Trans. Evol. Comput.* **2002,** *10*, 99–127. [CrossRef]
35. Gauci, J.; Stanley, K.O. Autonomous evolution of topographic regularities in artificial neural networks. *Neural Comput.* **2010**, *22*, 1860–1898. [CrossRef] [PubMed]
36. Stanley, K.O.; D'Ambrosio, D.B.; Gauci, J. A hypercube-based indirect encoding for evolving large-scale neural networks. *Artif. Life* **2009,** *15*, 185–212. [CrossRef]
37. Stanley, K.O. Compositional pattern producing networks: A novel abstraction of development. *Genet. Program Evol. Mach.* **2007**, *8*, 131–162. [CrossRef]
38. Baker, B.; Gupta, O.; Naik, N.; Raskar, R. Designing Neural Network Architectures Using Reinforcement Learning. *arXiv* **2017**, arXiv:1611.02167.
39. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 8697–8710.
40. Liu, H.; Simonyan, K.; Yang, Y. Darts: Differentiable architecture search. *arXiv* **2019**, arXiv:1806.09055.
41. Xie, S.; Zheng, H.; Liu, C.; Lin, C. Snas: Stochastic neural architecture search. *arXiv* **2019**, arXiv:1812.09926.
42. Wu, B.; Dai, X.; Zhang, P.; Wang, Y.; Sun, F.; Wu, Y.; Tian, Y.; Vajda, P.; Jia, Y.; Keutzer, K. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 10734–10742.
43. Pham, H.; Guan, M.Y.; Zoph, B.; Le, Q.V.; Dean, J.Efficient neural architecture search via parameter sharing. In Proceedings of the Internation Conference on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018; pp. 6522–6531.
44. Xie, L.; Yuille, A. Genetic cnn. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 1388–1397.
45. Yang, Z.; Wang, Y.; Chen, X.; Shi, B.; Xu, C.; Xu, C.; Tian, Q.; Xu, C. CARS: Continuous evolution for efficient neural architecture search. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 1829–1838.
46. Elsken, T.; Metzen, J.H.; Hutter, F. Efficient multi-objective neural architecture search via lamarckian evolution. *arXiv* **2019**, arXiv:1804.09081.
47. Brock, A.; Lim, T.; Ritchie, J.M.; Weston, N. Smash: One-shot model architecture search through hypernetworks. *arXiv* **2017**, arXiv:1708.05344.
48. Saxena, S.; Verbeek, J. Convolutional neural fabrics. In Proceedings of the Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 4060–4068.
49. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the 6th International Symposium on Micro Machine and Human Science (MHS), Nagoya, Japan, 4–6 October 1995; pp. 39–43.
50. Sahu, A.; Panigrahi, S.K.; Pattnaik, S. Fast convergence particle swarm optimization for functions optimization. *Procedia Technol.* **2012**, *4*, 319–324. [CrossRef]
51. 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]
52. Friedman, M. A comparison of alternative tests of significance for the problem of m rankings. *Ann. Math. Stat.* **1940**, *30*, 86–92. [CrossRef]

53. Nalepa, J.; Myller, M.; Kawulok, M. Validating hyperspectral image segmentation. *IEEE Geosci. Remote Sens. Lett.* **2019**, *16*, 1264–1268. [CrossRef]
54. Cao, X.; Ren, M.; Zhao, J.; Lu, H.; Jiao, L. Non-overlapping classification of hyperspectral imagery based on set-to-sets distance. *Neurocomputing* **2020**, *378*, 422–434. [CrossRef]