Evolving Deep Convolutional Neural Networks for Hyperspectral Image Denoising

Yuqiao Liu$^{(a)}$, Yanan Sun$^{(a)}$, Bing Xue$^{(b)}$, and Mengjie Zhang$^{(b)}$
$^{(a)}$College of Computer Science, Sichuan University, Chengdu 610064, China
$^{(b)}$School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand
Emails: lyqguitar@gmail.com, ysun@scu.edu.cn, bing.xue@ecs.vuw.ac.nz, and mengjie.zhang@ecs.vuw.ac.nz

Abstract—**Hyperspectral images (HSIs) are susceptible to various noise factors leading to the loss of information, and the noise restricts the subsequent HSIs object detection and classification tasks. In recent years, learning-based methods have demonstrated their superior strengths in denoising the HSIs. Unfortunately, most of the methods are manually designed based on the extensive expertise that is not necessarily available to the users interested. In this paper, we propose a novel algorithm to automatically build an optimal Convolutional Neural Network (CNN) to effectively denoise HSIs. Particularly, the proposed algorithm focuses on the architectures and the initialization of the connection weights of the CNN. The experiments of the proposed algorithm have been well-designed and compared against the state-of-the-art peer competitors, and the experimental results demonstrate the competitive performance of the proposed algorithm in terms of the different evaluation metrics, visual assessments, and the computational complexity.**

I. Introduction

Image denoising is one of the fundamental tasks of image processing. Different from the natural 2D image, the Hyperspectral image (HSI) has three dimensions to additionally display the spectral and spatial information. HSIs are widely used in urban planning, agriculture, and forestry [1], [2]. But in the harsh space environment, the multi-detector for generating the HSIs is susceptible, which consequently results in the HSIs having noise. In general, the noise in HSIs has many different types, such as the gaussian noise and the stripe noise. The corrupted hyperspectral data with the noise will affect the accuracy of the consequent work, for instance, the classification tasks [3]. Thus, the HSI denoising has been a hot topic in the past few years [4], [5]. Many algorithms have been proposed, such as the K-singular value decomposition (KSVD) [6] and the Tenser-SVD [7]. Generally, the HSI denoising methods are divided into three different categories as follows.

1) Filter-Based Methods: The core idea of the filter-based methods is to use the filtering operations with a variety of filters including the Fourier transform and the wavelet transform. Particularly, one of the hundreds of channels in an HSI can be regarded as a grayscale image. So, the traditional gray-level image denoising methods, for example, the block-matching 3-D filtering (BM3D) [8], can be adopted to every channel directly. The limitation of these filtering methods remains in their sensitiveness to the transform function, mainly due to the manually set parameters. In addition, ignoring the correlations across the spectral bands also leads to their relatively poor performance in practice.

2) Optimization-Based Methods: These methods work by adopting reasonable assumptions or the priors, such as the Total Variation (TV), the Non-local (Non-Local), the Sparse Representation (SR), and the Low-Rank (LR) models, etc, focusing on preserving the spatial and spectral characteristics. Because of the high-dimensional feature set and strong spectral correlations in HSIs, the LR regularization has been widely used in HSI denoising, owing to its effective ability of revealing the low-dimensional structure from the high-dimensional data. Due to the high spectral correlation between the adjacent bands and the high similarity in each band, Zhang *et al.* [9] proposed the Low-Rank Matrix Recovery (LRMR) for HSIs restoration by transforming a 3-D cube into a 2-D matrix. To better combine the spatial and spectral information, the tensor-based approaches have been presented recently [10], [11], which can achieve good results by running on the powerful computation platforms. In summary, although these methods can perform well in HSI denoising, the inadaptability to the mixed noise becomes a barrier to performance improvements.

3) Learning-Based Methods: In recent years, deep learning methods have been proposed and perform well in denoising the HSIs [12]. The Convolutional Neural Networks (CNNs) are the most representative method of the deep learning-based methods, and widely used in natural image denoising [13]–[15]. However, the architecture of CNNs needs to be redesigned as the problem changes. In practice, designing an optimal CNN for the problems at hand is not an easy work, since the best network architecture for a specific problem is unknown, i.e., the depth of the CNN, the number of different types of layers, and the parameters of each layer are are to determine. Meanwhile, the weights of the network which play a vital role in its performance [16] need to be retrained by a gradient-based algorithm for achieving the promising performance, which highly relies on its initialization [17].

To better explore the merits of CNNs in image denoising, and reduce the human expertise intervention during the architecture design and the weight initialization, in this paper, we propose a novel algorithm (denoted as Evolve-CNN) based on a Genetic Algorithm (GA) [18], [19] to design a good architecture and weight initialization for CNNs, to effectively and efficiently address the HSI denoising task. In summary, the contributions of the proposed Evolve-CNN are summarized as follows:

1) **A novel gene encoding strategy of GA is proposed to encode the individuals for the automated CNN**

architecture design. The encoded individual carries the information of the corresponding CNN architecture. A population of such individuals can evolve a better CNN for HSI denoising.

2) **Effective genetic operators are designed for the exploration and the exploitation search.** In GA, the fixed-length chromosome can execute genetic operators easily. However, only a variable-length encoding strategy can search the network architectures efficiently, but there is no accepted good method to play the role of the genetic operators. The proposed operators can perform well on the proposed encoding strategy.
3) **An improved slack binary tournament selection is proposed for choosing promising individuals for offspring generation.** Usually, huge computing resources are required for training CNNs. We regard the complexity as an important evaluation criterion in the environment selection. The improved method can help us search to find a high-performance CNN with lower complexity.

The remainder of this paper is organized as follows. The background of CNN is provided in Section II. Section III describes the proposed algorithm in detail. Section IV and V provide the experimental design and the experimental results. Finally, the conclusion is given in Section VI.

II. Background

In this section, the skeleton of CNNs is provided, which is the base work of the proposed algorithm. We will introduce the convolution layer, the Reflect Padding (RP) layer, and the Batch Normalization (BN) layer which constitute the skeleton of the CNNs.

A. Convolution Layer

The convolution layer plays a vital role in CNNs since the convolution operation can extract features from the images. To be specific, the operation provided as the following. First, given an input matrix with the size of $n \times n$, the filter travels from the top left to the bottom right of the input data to generate a value in the feature map with the convolutional operation. Second, the action is performed again after moving downward with the set stride, until reaching the bottom right of the matrix. Noting that the convolution padding type is an important parameter in CNNs because the type decides the output image size. After a convolution operation where the convolution kernel size is greater than one, the output size will become smaller. There are two padding types in convolution layer: the VALID type and the SAME type. The SAME type can get the same size of the output as that of the input by padding zeros in the input matrix. The VALID type will not pad any elements to the input matrix. Noting that, the VALID type will be used in the proposed algorithm because the SAME type convolutional operation cannot well address the boundary of the images.

B. Reflect Padding (RP)

RP is one of the most common image padding methods, which can restore input image to the original image. Specifically, it pads the input image using the reflection of the input data, i.e., if we want to add a new row at the top of the input image by RP, the row is obtained from the second row in the input image through vertical reflection and diagonal reflection. Fig. 1 is an example to explain how RP works. For example, the first and the last element in first green row is 0.7 by diagonal reflection. The middle elements are equal to those in second row in blue part by vertical reflection.

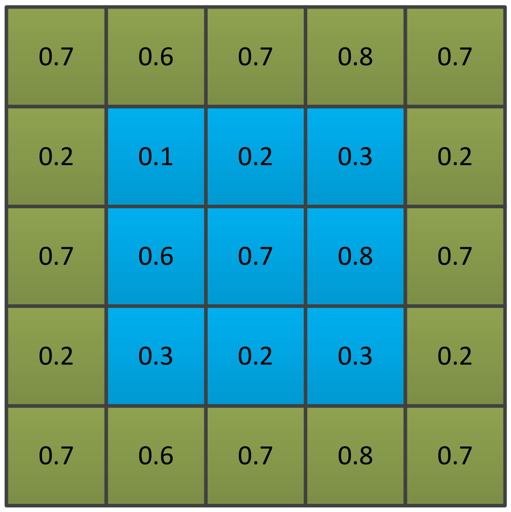

Fig. 1. An example to show RP. The blue part that represents the input image is given as a 3×3 matrix, and we use RP to add the green part to get a 5×5 matrix which represents the original image. The middle of the first row in green is the same as the second row in blue. The other three sides were obtained in the same way. Elements in the green corner are generated by diagonal reflection.

C. Batch Normalization (BN)

In HSI denoising, a CNN can work well if the BN layer can be jointly used [12]. Particularly, the BN layer allows a higher learning rate, significantly increases the speed of training, and avoids the gradient vanishing or divergence issue. The BN layer begins to work by performing BN when a batch of data is input to the CNN, where BN is achieved by calculating the mean and the standard deviation (std) of the input data [20].

III. The proposed algorithm

A. Algorithm Overview

Algorithm 1 shows the framework of the proposed Evolve-CNN method, where the contributions are highlighted in bold and italic. Firstly, the population is randomly initialized, and each individual in the population is generated randomly with the proposed gene encoding strategy (line 1). Then, the initialized population is evaluated for the parent selection (line 2). After that, the evolution begins to take effect until the stopping criterion is satisfied (lines 4-9). Finally, the expected CNN, built by decoding the selected best individual, is ready for the final training (line 10).

During the evolution, all of the initialized individuals' fitness are evaluated. To speed up the evolution, we let individuals be trained only one epoch on the training dataset, while the fitness evaluation is performed on the evaluation dataset. And then, two parent individuals are chosen by the improved slack binary operation from the population, and the offspring is generated with the proposed genetic operation (line 6). Subsection III-C will illustrate this operation in detail. After offspring generation and fitness evaluation (line 7), the environment selection using the elite selection mechanism starts to choose the next generation from the existing individuals and the newly generated offspring (line 8). Then the next generation continue the next round of evolution.

Algorithm 1: Framework of Evolve-CNN

```
1  $P_0 \leftarrow$ Randomly initialize the population with the proposed gene encoding strategy;
2  Evaluate fitness of $P_0$;
3  $t \leftarrow 0$;
4  while stopping criterion is not satisfied do
5      $t \leftarrow t + 1$;
6      $O_t \leftarrow$ Choose parent individual by the improved slack binary operation from $P_{t-1}$ to generate the offspring with the proposed genetic operation;
7      Evaluate fitness of $O_t$;
8      $P_t \leftarrow$ Environment selection from $P_{t-1} \cup O_t$;
9  end
10 Return $P_t$.
```

B. Gene Encoding Strategy

Generally, the optimal architecture of the CNN is difficult to determine without a prior knowledge. The architecture of the CNN, especially the depth of the CNN, plays a decisive role in the performance of CNNs [21]–[24]. In the traditional design of the CNN architectures, the depth is set based on domain expertise, which is not necessarily accurately assigned for every task at hand. In the proposed algorithm, this issue is addressed by the proposed variable-length gene encoding strategy that is able to automatically find the optimal depth without any expertise. In the proposed encoding strategy, the CNN is built by multiple sequential blocks, and each block consists of a convolution layer, a BN layer and a Rectified Linear Unit (ReLU). In addition, we also add a RP layer after the convolution layer whose kernel size unequal to one to avoid the image size diminishing. Considering that the output image is close to the real one, we set the last block as only a convolution layer whose kernel size and out feature map size are unchangeable.

As mentioned above, three different types of the layers, i.e., the convolution layer, the BN layer, and the RP layer exist in the architectures of Evolve-CNN. Because the RP layer can be regarded as the appurtenance of the convolution layer and the BN layer works well in the default setting, we use the information of the convolution layer to encode the architecture into one chromosome for the evolution. An example of two types of blocks and two chromosomes with different lengths from Evolve-CNN is illustrated in Fig. 2. Commonly, hundreds of thousands of weights may exist in a CNN, which cannot be all initialized explicitly. In the proposed gene encoding strategy, the mean and std are used to efficiently initialize the weights of CNN using the Gaussian initialization mechanism. In addition, the important parameters of the convolution layer, including the filter width, filter height, and the number of feature maps are all encoded into the chromosome.

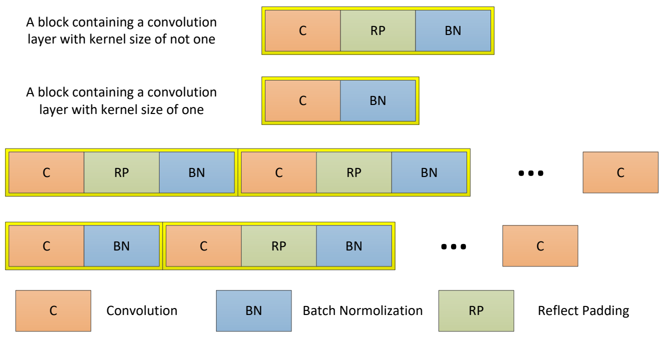
A block containing a convolution layer with kernel size of not one
C
RP
BN
A block containing a convolution layer with kernel size of one
C
BN
•••
C
Convolution
BN
Batch Normolization
RP
Reflect Padding

Fig. 2. An example to show two types of blocks and two chromosomes with different lengths.

Algorithm 2: Population Initialization

```
   Input: the population size $N$; the maximal number of blocks, $N_{max}$; the minimal number of blocks, $N_{min}$
   Output: Initialized population, $P_0$
1  $P_0 \leftarrow \emptyset$
2  while $|P_0| \leq N$ do
3      $head \leftarrow \emptyset$;
4      $r \leftarrow$ uniformly generate a random integer between $[N_{min}, N_{max}]$;
5      while $|head| < r$ do
6          $l \leftarrow$ generate a convolution layer with random settings;
7          $head \leftarrow head \cup l$;
8          $s \leftarrow$the kernel size of the convolution layer;
9          if $s \neq 1$ then
10             $l \leftarrow$ generate a reflect padding layer with $\frac{s-1}{2}$ size;
11             $head \leftarrow head \cup l$;
12         end
13         $l \leftarrow$ generate a batch-normalization layer with random settings;
14         $head \leftarrow head \cup l$;
15     end
16     $l \leftarrow$ generate a convolution layer with predefined number of feature maps and kernel size;
17     $head \leftarrow head \cup l$;
18     $P_0 \leftarrow P_0 \cup head$;
19 end
20 return $P_0$
```

Algorithm 2 shows the population initialization by using the proposed genetic encoding strategy, where $|\cdot|$ is a cardinality operator. Firstly, an empty population of size N is initialized (line 1). Secondly, individuals are created randomly to fill in the population (lines 2-19) until the population reached its predefined size of N. Finally, an initialized population P_0 is returned (line 20). The CNN architecture encoded by the individuals is divided into two parts, namely the head and the last convolution layer, according to whether there are fixed parameters or not, and the individual initialization is also divided into two parts. The first part is generating the head with the random settings (lines 3-15), and the second part is adding a partially randomly initialized convolution layer (lines 16-17).

Noting that line 4 sets the upper and the lower bounds of the depth because a shallow CNN may not perform well on the HSI denoising and a deep CNN will take up a lot of computing resources unnecessarily. In addition, because the output size of the CNN is determined, and the feature map size and the filter size of the last convolution layer are fixed accordingly, while the mean and std of filter elements are variable (line 16). In order to find a suitable RP layer, the kernel size of the convolution layer must be an odd number.

C. Offspring Generation

Because the Mean Squared Error (MSE) represents the pixel value difference between the denoised image and the original image, we use it as the fitness evaluation criterion for the task investigated in this work. Specifically, the MSE is calculated by Equation (1)

$$\text{MSE} = \frac{||D-O||^2}{M \times N} \tag{1}$$

where $||\cdot||$ denotes the L2 norm of a matrix, D and O denote the pixel values of the image after the denoising and the original image, respectively, M and N denote the length and width of the image. The smaller MSE is, the better denoising effect is, and the better fitness degree the individual is. In practice, there will be multiple CNN architectures resulting in almost the same MSE, while the one having the least number of parameters should be the promising one because fewer parameters of the CNN means the low complexity and potentially provides better generalization ability. Thus, the number of CNN parameters, indicating the complexity of the CNN, is also used as part of the individuals' fitness.

Just like traditional GAs and as mentioned in Section III-A, after the population initialization and the fitness evaluation for the original population, the offspring will be generated with the proposed genetic operations. Traditionally, the offspring are generated by performing the genetic operators, which usually consists of the crossover operator and the mutation operator. The steps of generating the offspring are shown below:

1) Select two parents by Algorithm 3 (Section III-D will illustrate this algorithm in detailed);
2) Perform mutation and crossover on the chosen parent individuals, and generate the offspring;
3) Store the offspring and reperform Step 1 until the number of offspring reaches the predefined size.

In the proposed algorithm, the mutation operation is divided into two steps. The first step allows the encoded variable information mentioned above to mutate in a given range by the Polynomial Mutation operator (PM) [25]. The second step is to randomly increase or decrease the depth of the CNN. This step may perform on each position of the CNN, where the position is randomly decided. After that, one particular mutation operation is randomly selected from: 1) adding a block which is randomly initialized, 2) removing a block with the exception of the last layer, and 3) keeping the depth unchanged. Then, the operation performs on the selected position. Each type of the operation has a 1/3 chance of being randomly selected.

In the proposed crossover operation, we use the Simulated Binary Crossover (SBX) [26] to perform the crossover owing to its promising performance on the real numbers, i.e., the encoded variable information in the proposed encoding strategy.

Fig. 3 illustrates the crossover process where C denotes the convolution layer, BN denotes the BN layer, and RP denotes the RP layer. In this example, the first chromosome has three blocks while the second chromosome has four. Initially, the blocks of each chromosome align based on the order as shown in Fig. 3.a. Secondly, the matched blocks perform the crossover operation. There are two different situations when doing the crossover operation: one is for the same structures when both blocks consist of C+RP+BN or C+BN. The other is for the different structures when one block is C+RP+BN and the other is C+BN. The first case perform the crossover operation by just swapping the encoded variable information, while for the second case, not only the encoded variable information, but also taking the RP to the side having no RP. It's worth noting that if the lengths of the chromosomes are not the same, the crossover operation is performed on the last layer of the shorter chromosome and the corresponding convolution layer of the longer one. In order to get the valid image size, the number of the feature maps will not participate this operation at the last layer. Finally, the offspring is generated as shown in Fig. 3.b. where bold and italics indicate that this layer has been modified.

D. Environmental Selection

In order to maintain a population with the promising convergence and diversity, the elite mechanism is used in the developed environment selection which is named as "Slack Binary Tournament Selection". Algorithm 3 shows the details of the design. Firstly, two individuals are selected randomly from the population (line 1). Secondly, the individual with the smaller MSE assigns to i_1, and the other one assigns to i_2 (lines 2-3). Thirdly, their MSE values and complexity assign to the corresponding parameters (lines 4-5). In the end, the proportion (line 6) and the difference (line 9) are compared with the given threshold to choose the better individual. Specifically, if a CNN's performance is much better than the other one's (lines 6-7), we choose the better one directly, when their performances are about the same, picking the one with less complexity

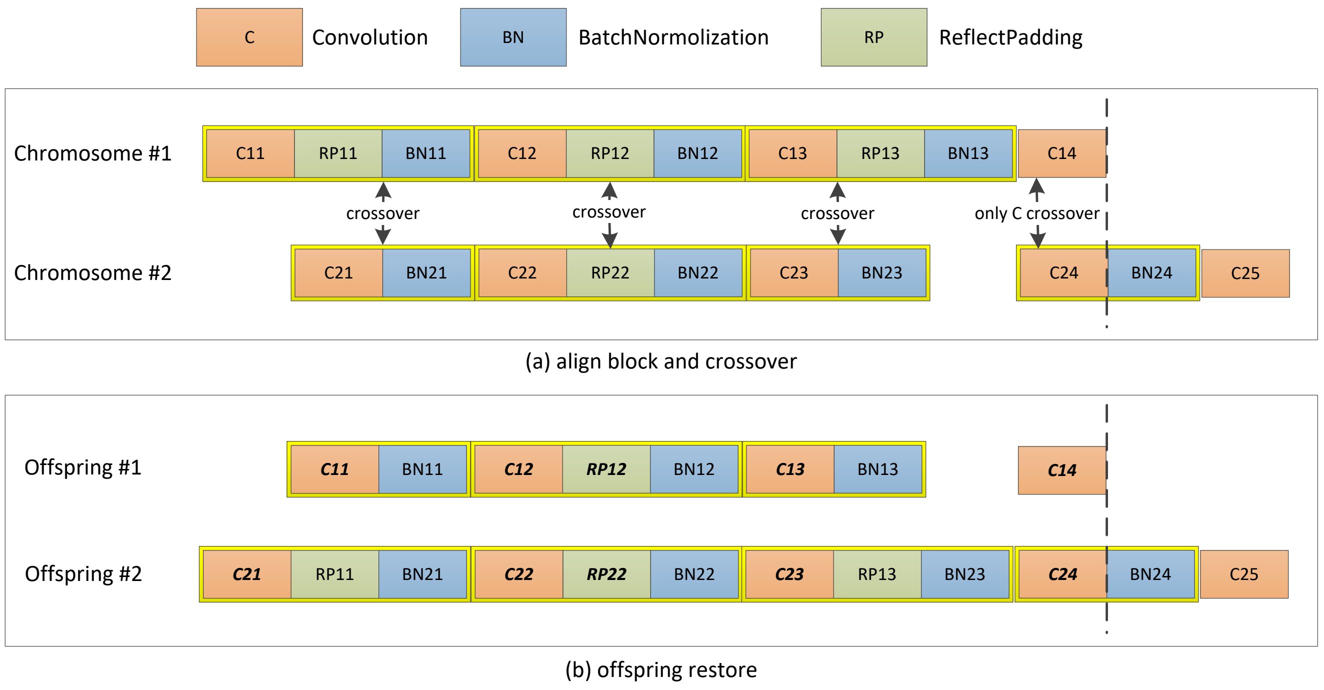

Fig. 3. A crossover example.

Algorithm 3: Slack Binary Tournament Selection

```
Input: MSE threshold, $\alpha$; complexity threshold, $\beta$; The population
Output: The selected individual
1  Randomly select two individuals from the population
2  $i_1 \leftarrow$ the individual with smaller MSE;
3  $i_2 \leftarrow$ the other individual;
4  $m_1, m_2 \leftarrow$ the MSE of $i_1, i_2$;
5  $c_1, c_2 \leftarrow$ the complexity of $i_1, i_2$;
6  if $\frac{m_2 - m_1}{m_1} > \alpha$ then
7      return $i_1$
8  else
9      if $c_1 - c_2 > \beta$ then
10         return $i_2$
11     else
12         return $i_1$
13     end
14 end
```

(lines 9-10). When the performance and the complexity are both about the same, the performance is preferred (lines 11-12).

Noting that we use a proportion instead of a simple difference as a threshold, because in the earlier stage of the evolution, the MSE value of all individuals is relatively large, but in the later stage of the evolution, most individuals have good denoising performance, i.e., the MSE value is generally small. So the proportion is better to maintain the consistency of selection operation. When the evolution process is finished, the best individual which has the smallest MSE is selected from the last generation to perform the final training of the CNN.

IV. EXPERIMENT DESIGN

In order to quantify the performance of the proposed Evolve-CNN algorithm, an experiment is designed to compare with the state-of-the-art peer competitors on the chosen benchmark dataset. In the following, the benchmark dataset and how to build the training, the evaluation, and the test dataset are introduced at first. Then, the peer competitors are listed. After that, the parameter settings, including the training parameters for the proposed algorithm, are detailed.

A. Benchmark Dataset

The Indian Pines dataset [27] which is widely used in HSI denoising [12], [28] is chosen as the benchmark dataset. The dataset is gathered by AVIRIS sensor over the Indian Pines test site in North-western Indiana. Particularly, the Indian Pines dataset contains two HSIs with different sizes. The first is with the size of 614×2678×220, and the second is with the size of 1848×614×220. In order to do a fair comparison, the Gaussian noise whose intensity level $\sigma = 0.778$ is added to the original images to generate the data used by the compared algorithms for the denoising.

Based on the conventions, the dataset has been divided into three parts for the experiments, i.e., the training dataset, the evaluation dataset, and the test dataset, which are randomly selected and have 17535, 4000 and 4838 images, respectively, i.e., account for 66.5%, 15.2%, and 18.3%, respectively. In order to obtain the sufficient images to train the CNN, the original clean images and generated noise images are cropped with the size of 30×30, with the sampling stride equaling to 10.

B. Peer Competitors

Multiple state-of-the-art HSI denoising methods are chosen as the peer competitors. Considering the proposed algorithm focusing on the CNN methods, we also choose the algorithm proposed by Chang [12], which recently reported its promising performance on HSI denoising, as one of the peer competitors in this experiment.

Overall, the chosen peer competitors are listed below: 1) the BM3D method [8]; 2) the low-rank tensor approximation (LRTA) method [29]; 3) the Total-variation-regularized low-rank matrix factorization (LRTV) method [30]; 4) the tensor dictionary learning method (TDL) [31]; 5) the block matching 4-D filtering method (BM4D) [32]; 6) the intrinsic tensor sparsity regularization method (ITSReg) [33]; 7) the LRMR method [9]; 8) the Chang's method (Artificial-CNN) [12].

Noting that, the optimal CNN selected by the proposal algorithm is chosen based on the evaluation dataset after it has been trained on the training dataset. When the evolutionary process is finished, the best performance is obtained by training it on both the training dataset and the evaluation dataset following the conventions of the deep learning community. The test dataset remains unseen during this process.

C. Parameter Settings

In the experiment, the parameter settings include the parameters of the evolution in the proposed algorithm and the parameters of the peer competitors.

All the parameter settings of the evolution are specified following the conventions of GA community [25], [26]. Considering both the performance and the complexity, the depth of CNN varies from 4 to 8. The filter size of convolution layer, equivalent to the filter width and filter height, are randomly chosen from $\{1, 3\}$. All layer's feature map sizes vary from 128 to 512 except the last convolution layer whose size is fixed. Preliminary experiment and experience specify the mean range from -0.8 to 0.8, and the std from 0 to 0.5. In addition, the population size and the total generation number are set to be 30 and 10, respectively. The distribution index of the SBX and PM are both set to 1 based on the conventions [25], [26], and their associated probabilities are specified as 0.9 and 0.2, respectively. In the environment selection, the elitism rate is specified as 20% based on the Pareto principle. As mentioned in Section III-C, the MSE and the complexity are used in the fitness function during the evolutionary stage.

In addition, all the parameters of peer methods are set by default because their default settings can give them the best performance.

D. Training Details

In this subsection, we give the training details in both the evolution process and the final train.

In the evolution stage, each individual needs to be trained at first for the fitness evaluation. We use the Adam method [34] as the optimizer for this stage, and the learning rate is set to 0.004 and exponential decay rate is set as the default setting. The training dataset will not be shuffled at evolution stage to ensure the consistency in training each individual.

For the final train, the Adam configured with default setting is used again as the optimizer of the CNN, respectively and the learning rate is initialized to 0.001. It is worth noting that, the training dataset and the evaluation dataset are both used to train the CNN, and these two datasets are shuffled in this stage.

The training of the Artificial-CNN uses the same settings as those of the proposed Evolve-CNN, in addition to its weight initialization method that uses the Xavier initializer [17]. We employed the Pytorch to implement both CNNs on a Ubuntu server with an NVIDIA 2080 Ti GPU card.

Noting that, the batch sizes of the proposed evolve-CNN method and the Artificial-CNN are set to be 100 for both the training and the test stage. Furthermore, we did not limit the training epoch but stop the training and test by investigating the performance does not increase.

V. Experimental Results and Analysis

To acquire an integrated comparison for all peer competitors and the proposed Evolve-CNN method, the quantitative evaluation indicators and a visual comparison are used to analyze the experimental results. We employ four Image Quality Measurements (IQMs) as the quantitative evaluation indicators, including the Mean Peak Signal to Noise Ratio (MPSNR), the Mean Structural SIMilarity index (MSSIM) [35], the Mean Feature Similarity index (MFSIM) [36], and the Mean Erreur Relative Globale Adimensionnelle de Synthèse (MERGA) [37]. MPSNR, MSSIM and MFSIM evaluate the similarity between the target image and the reference image based on the MSE value, the structural consistency and the perceptual consistency. The larger they are, the more similar the two images are, i.e., the better the method is. Different from the former three indicators, MERGA measures the fidelity, and the smaller MERGA is, the better the method is.

A. Overall Results

The average results of the four performance evaluation indicators are listed in Table I. Band 187 was chosen randomly for visual comparison that is shown in Fig. 4.

In Table I, the best performance of each evaluation indicator is marked in bold. As can be seen from Table I, the CNN-based methods have the absolute leading position among the compared algorithms. Specifically, the proposed Evolve-CNN provides the highest values in MPSNR, MSSIM, and MFSIM, and the lowest MERGA, and becomes the only method with MPSNR over 70. In Fig. 4, the BM3D could not deal with this noise well, furthermore, the BM4D produces over-smoothing in this result. Residual gaussian noise is clearly visible in LRTA and LRMR, and the left methods have high fidelity.

TABLE I
AVERAGE DENOISING PERFORMANCE COMPARISON EIGHT COMPETING METHODS WITH RESPECT TO FOUR PQIS OF THE INDIAN PINE UNDER GAUSSIAN NOISE $\sigma = 0.778$

Method / Measure	Nosiy	BM3D	LRTA	LRTV	TDL	BM4D	ITSReg	LRMR	Artificial-CNN	Evolve-CNN
MPSNR	50.311	62.881	64.939	64.852	67.248	65.122	66.979	65.115	69.317	**70.051**
MSSIM	0.98997	0.99905	0.99937	0.99954	0.99967	0.99951	0.99956	0.99958	0.99974	**0.99977**
MFSIM	0.96443	0.97201	0.99030	0.98072	0.98883	0.98107	0.98829	0.98599	0.99197	**0.99213**
MERGA	29.857	7.567	5.877	5.929	4.483	5.575	4.906	5.333	4.095	**3.976**

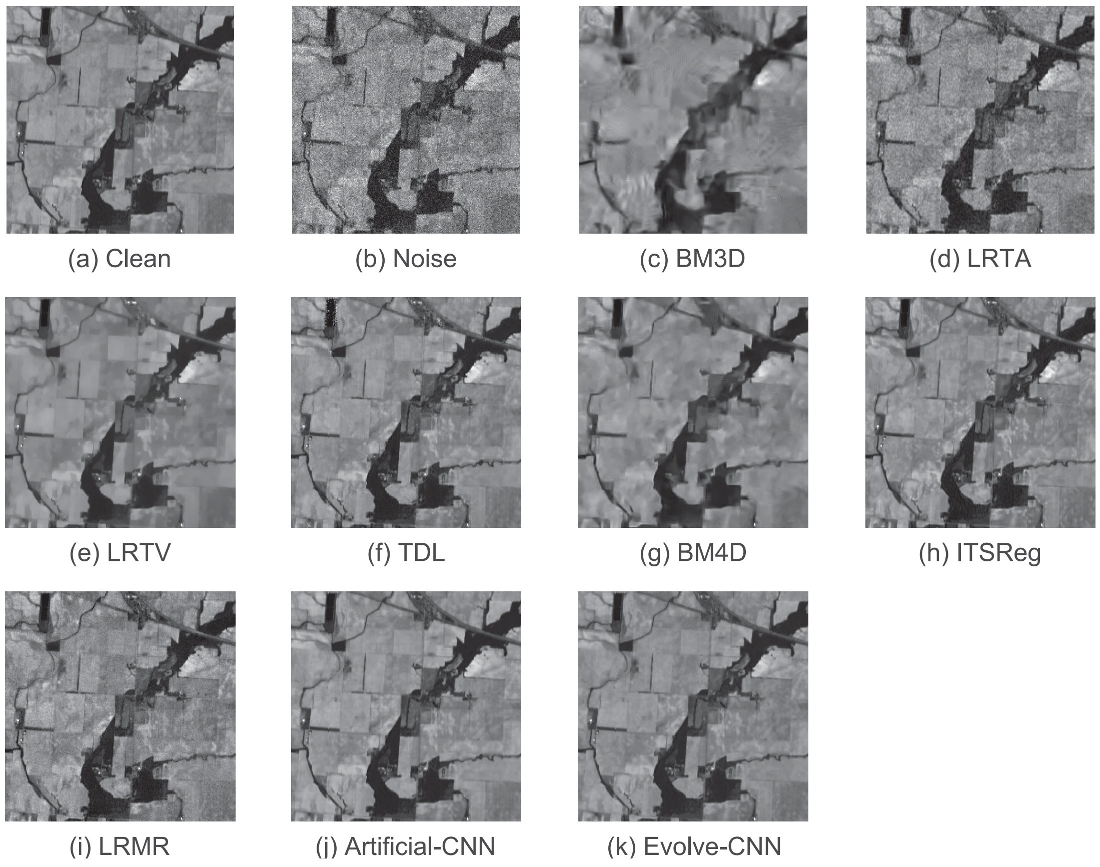

Fig. 4. Visual comparison in band 187. (a)Clean. (b)Noise. (c)BM3D. (d)LRTA. (e)LRTV. (f)TDL. (g)BM4D. (h)ITSReg. (i)LRMR. (j)Artificial-CNN. (k)Evolve-CNN.
.

B. Comparisons with Artificial-CNN

The CNN-based methods are compared separately to tell the effect from the automatic architecture selection mechanism. Table II shows the depth, the number of total parameters, the training time and the performance derived from evaluation indicators. We define one hour on one GPU service as a GPUh to measure the training time.

As can be seen from Table II, the Artificial-CNN method is much more complicated than the proposed Evolve-CNN method. Meanwhile, the training time of Artificial-CNN is three times more than that of Evolve-CNN. However, the shorter CNN perform better in the evaluation indicators, although both CNNs have excellent denoising effect. Furthermore, in terms of the IQMs results upon the four employed measures, the Evolve-CNN model is twice significantly better than Artificial-CNN, and twice significantly equal to Artificial-CNN.

VI. CONCLUSIONS

We have proposed an automatic method by using the genetic algorithm to design the CNNs for HIS denoising. Particularly, we have designed an improved genetic encoding strategy for encoding the CNN architectures and the weight initialization parameters, the corresponding genetic operators to effectively and efficiently find the optimal CNN architecture during the evolution process, and a new tournament selection to choose

TABLE II
COMPARISON WITH ARTIFICIAL CNN. THE SYMBOLS "+", "=," AND "-" DENOTE WHETHER THE IQMS RESULTS OF THE PROPOSED EVOLVE-CNN ARE BETTER THAN, EQUAL TO OR WORSE THAN THAT OF THE ARTIFICIAL-CNN WITH A SIGNIFICANT LEVEL 1%

	Evolve-CNN	Artificial-CNN
depth	5	16
total params	1,838,603	32,254,420
training time	15 GPUhs	54 GPUhs
evolutionary time	0.5 GPUh	0
+/=/-		2/2/0

the promising parent individuals for the evolution performance. In the experiment, we found the proposed method is much better than other methods in terms of the evaluation indicator values and the visual assessments. The automated CNN by the proposed algorithm has much fewer parameters than the state-of-the-art CNN peer competitors. Furthermore, the automated CNN by our proposed algorithm can achieve the performance of that designed by experts. In the future, we will devote to the research of fitness evaluation methods, and also investigate which new components contribute more to the performance.

REFERENCES

[1] L. Zhang, L. Zhang, D. Tao, and X. Huang, "On combining multiple features for hyperspectral remote sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 3, pp. 879–893, 2011.

[2] P. S. Thenkabail and J. G. Lyon, *Hyperspectral remote sensing of vegetation*. CRC press, 2016.

[3] J. Li, H. Zhang, Y. Huang, and L. Zhang, "Hyperspectral image classification by nonlocal joint collaborative representation with a locally adaptive dictionary," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 6, pp. 3707–3719, 2013.

[4] Z. Kong and X. Yang, "Color image and multispectral image denoising using block diagonal representation," *IEEE Transactions on Image Processing*, 2019.

[5] W. He, Q. Yao, C. Li, N. Yokoya, and Q. Zhao, "Non-local meets global: An integrated paradigm for hyperspectral denoising," *arXiv preprint arXiv:1812.04243*, 2018.

[6] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[7] Z. Zhang, G. Ely, S. Aeron, N. Hao, and M. Kilmer, "Novel methods for multilinear data completion and de-noising based on tensor-svd," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 3842–3849.

[8] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on image processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[9] H. Zhang, W. He, L. Zhang, H. Shen, and Q. Yuan, "Hyperspectral image restoration using low-rank matrix recovery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 8, pp. 4729–4743, 2013.

[10] H. Fan, C. Li, Y. Guo, G. Kuang, and J. Ma, "Spatial–spectral total variation regularized low-rank tensor decomposition for hyperspectral image denoising," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 10, pp. 6196–6213, 2018.

[11] Q. Xie, Q. Zhao, D. Meng, and Z. Xu, "Kronecker-basis-representation based tensor sparsity and its applications to tensor recovery," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 8, pp. 1888–1902, 2017.

[12] Y. Chang, L. Yan, H. Fang, S. Zhong, and W. Liao, "Hsi-denet: Hyperspectral image restoration via convolutional neural network," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 2, pp. 667–682, 2018.

[13] V. Jain and S. Seung, "Natural image denoising with convolutional networks," in *Advances in neural information processing systems*, 2009, pp. 769–776.

[14] S. Lefkimmiatis, "Non-local color image denoising with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3587–3596.

[15] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in neural information processing systems*, 2012, pp. 341–349.

[16] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Transactions on Evolutionary Computation*, 2019.

[17] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[18] D. Ashlock, *Evolutionary computation for modeling and optimization*. Springer Science & Business Media, 2006.

[19] T. Back, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[22] Y. Bengio and O. Delalleau, "On the expressive power of deep architectures," in *International Conference on Algorithmic Learning Theory*. Springer, 2011, pp. 18–36.

[23] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[24] O. Delalleau and Y. Bengio, "Shallow vs. deep sum-product networks," in *Advances in Neural Information Processing Systems*, 2011, pp. 666–674.

[25] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001, vol. 16.

[26] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex systems*, vol. 9, no. 2, pp. 115–148, 1995.

[27] N. Aviris, "Indiana's indian pines 1992 data set," 2012.

[28] Q. Zhang, Q. Yuan, J. Li, X. Liu, H. Shen, and L. Zhang, "Hybrid noise removal in hyperspectral imagery with a spatial-spectral gradient network," *IEEE Transactions on Geoscience and Remote Sensing*, 2019.

[29] N. Renard, S. Bourennane, and J. Blanc-Talon, "Denoising and dimensionality reduction using multilinear tools for hyperspectral images," *IEEE Geoscience and Remote Sensing Letters*, vol. 5, no. 2, pp. 138–142, 2008.

[30] W. He, H. Zhang, L. Zhang, and H. Shen, "Total-variation-regularized low-rank matrix factorization for hyperspectral image restoration," *IEEE transactions on geoscience and remote sensing*, vol. 54, no. 1, pp. 178–188, 2015.

[31] Y. Peng, D. Meng, Z. Xu, C. Gao, Y. Yang, and B. Zhang, "Decomposable nonlocal tensor dictionary learning for multispectral image denoising," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2949–2956.

[32] M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi, "Nonlocal transform-domain filter for volumetric data denoising and reconstruction," *IEEE transactions on image processing*, vol. 22, no. 1, pp. 119–133, 2012.

[33] Q. Xie, Q. Zhao, D. Meng, Z. Xu, S. Gu, W. Zuo, and L. Zhang, "Multispectral images denoising by intrinsic tensor sparsity regularization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1692–1700.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[36] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "Fsim: A feature similarity index for image quality assessment," *IEEE transactions on Image Processing*, vol. 20, no. 8, pp. 2378–2386, 2011.

[37] L. Wald, *Data fusion: definitions and architectures: fusion of images of different spatial resolutions*. Presses des MINES, 2002.