DNAS: Decoupling Neural Architecture Search for High-Resolution Remote Sensing Image Semantic Segmentation

Yu Wang, Yansheng Li *, Wei Chen, Yunzhou Li and Bo Dang

School of Remote Sensing and Information Engineering, Wuhan University, Wuhan 430079, China
* Correspondence: yansheng.li@whu.edu.cn; Tel.: +86-027-6877-0771

Abstract: Deep learning methods, especially deep convolutional neural networks (DCNNs), have been widely used in high-resolution remote sensing image (HRSI) semantic segmentation. In literature, most successful DCNNs are artificially designed through a large number of experiments, which often consume lots of time and depend on rich domain knowledge. Recently, neural architecture search (NAS), as a direction for automatically designing network architectures, has achieved great success in different kinds of computer vision tasks. For HRSI semantic segmentation, NAS faces two major challenges: (1) The task’s high complexity degree, which is caused by the pixel-by-pixel prediction demand in semantic segmentation, leads to a rapid expansion of the search space; (2) HRSI semantic segmentation often needs to exploit long-range dependency (i.e., a large spatial context), which means the NAS technique requires a lot of display memory in the optimization process and can be tough to converge. With the aforementioned considerations in mind, we propose a new decoupling NAS (DNAS) framework to automatically design the network architecture for HRSI semantic segmentation. In DNAS, a hierarchical search space with three levels is recommended: path-level, connection-level, and cell-level. To adapt to this hierarchical search space, we devised a new decoupling search optimization strategy to decrease the memory occupation. More specifically, the search optimization strategy consists of three stages: (1) a light super-net (i.e., the specific search space) in the path-level space is trained to get the optimal path coding; (2) we endowed the optimal path with various cross-layer connections and it is trained to obtain the connection coding; (3) the super-net, which is initialized by path coding and connection coding, is populated with kinds of concrete cell operators and the optimal cell operators are finally determined. It is worth noting that the well-designed search space can cover various network candidates and the optimization process can be done efficiently. Extensive experiments on the publicly open GID and FU datasets showed that our DNAS outperformed the state-of-the-art methods, including artificial networks and NAS methods.

Keywords: decoupling neural architecture search (DNAS); high-resolution remote sensing image; semantic segmentation; deep learning

1. Introduction

With the advancement of remote sensing technology, large numbers of HRSIs are available [1,2]. To make full use of HRSI, HRSI semantic segmentation has received extensive attention from researchers. HRSI semantic segmentation aims to produce the pixel-level classification map from one HRSI, which is a fundamental and critical problem in the remote sensing image understanding domain [3]. It has important application value in remote sensing image interpretation tasks, such as building extraction [4,5], road extraction [6,7], water extraction [8,9] and land-use classification [10,11]. As HRSIs often
have rich structures and complex contextual features, HRSI semantic segmentation is still a very challenging research task.

In the past decade, deep learning-based methods have gained great success in many image tasks [12, 13]. For semantic segmentation, the mainstream methods are deep convolutional neural networks (DCNNs), which achieve excellent performance, due to their powerful representation capabilities. After Long et al. [14] proposed the milestone work: fully convolutional networks (FCNs), many superior DCNNs have been put forward, such as UNet [15, 16], DeepLab [17, 18], and HRNet [19, 20]. In addition, there are a variety of DCNN-based methods that dominate the field of HRSI semantic segmentation [10, 11, 21]. Unlike traditional methods [22–24], DCNNs can automatically extract rich features from images by fitting the model parameters with massive, labeled data [25]. Generally, the basic difference between these models based on deep learning is their model architecture. For instance, UNet [15] constructs a U-shape structure and concatenates the early features to corresponding up-sampled features. DeepLab v3+ [18] introduces the dilated convolution to expand the receptive field and fuses the front and back features. HRNet [19] stacks blocks layer by layer and maintains the high spatial resolution of features. Therefore, the issue of creating a better classification method becomes one of creating a more sophisticated DCNN model. Nevertheless, compared with designing features for traditional methods, the DCNN is a black-box optimization problem [14], which needs a high level of expert knowledge and involves expensive computation time. Meanwhile, the human being’s prior knowledge of the designed model may be task-dependent, which means we have to design various model architectures towards different segmentation tasks (water extraction [9], building extraction [4], land-use classification [26]). Naturally, automatic design of network architecture, using a data-driven method, would be a promising solution to address various applications in remote sensing.

Neural Architecture Search (NAS), the process of automatic architecture engineering, is, thus, a logical next step for semantic segmentation. NAS aims to obtain superior model architecture on a target dataset directly without human design [27]. The principal idea of NAS can be summarized in three steps: (1) the architecture search space; (2) the search strategy, which selects a sub-model for the following step; and (3) the performance estimation strategy, which receives the previous step’s model architecture and returns the model performance [28]. After several years of NAS development, the main methods can be divided into four types: (1) reinforcement learning (RL); (2) evolutionary algorithms (EA); (3) one-shot NAS; and (4) gradient-based NAS. Early methods based on RL [27, 29] or EA [30, 31] train candidate architectures from scratch and update controller (RL) or population (EA) with estimation results iteratively, which is extremely time-consuming (thousands of GPU days). To overcome this problem, a weight-sharing mechanism was introduced to later methods, which can be divided into one-shot NAS [32, 33] and gradient-based NAS [34, 35]. One-shot methods first define the super-net, instead of sampling candidate models, to prevent training every model discretely. Benefiting from the weight-sharing mechanism, these methods just need to train the super-net once, the parameters of which are shared with all subnets. Finally, the subnets are sampled with parameters and are directly evaluated on the validation dataset until a satisfactory model is found. Gradient-based methods use weight-sharing strategy as well, but unlike one-shot NAS, they relax the discrete search space into a continuous super-net with architecture parameters, which are optimized during the search [34]. Most existing NAS methods are designed for natural images. However, due to the large differences between remote sensing images and natural images, it is usually inappropriate to directly transfer these NAS methods to the remote sensing domain.

Most recently, NAS has been gradually applied to remote sensing image scene classification and has achieved excellent results [36–40]. However, there has been little research on the dense image prediction task, semantic segmentation. RSNet [39] and Auto-deeplab [35] follow the weight-sharing strategy and design different search spaces to obtain the segmentation model. Notably, though the weight-sharing strategy decreases the
search time significantly, the introduction of the super-net, which is huger than the general model, takes up more display memory. Therefore, for researchers who lack sufficient computing resources, the resolution of images input has no choice but to reduce, which is detrimental, in particular, for remote sensing image segmentation. Due to the very high resolution of remote sensing images, each cropped image has a smaller receptive field perspective and lacks global information. Meanwhile, more clipping also means more border effects. These all affect the performance of NAS and the final searched model. Moreover, semantic segmentation as a dense prediction task needs more complex model architecture. It puts forward higher requirements for the design of super-net.

To address the aforementioned problem, this paper proposes a novel NAS framework, as shown in Figure 1, which is able to take into account sufficient input image size, while having a larger search space. To achieve that, our approach adopted a decoupling strategy, which can be manifested in a super-net design and search method. For super-net design, we first abstracted DCNN architecture into three aspects: up and down sampling, skip connections, and operators. Then, inspired by the two-level (network-level, cell-level) hierarchy super-net in [35], we designed a three-level (path-level, connections-level, cell-level) super-net corresponding to the aspects mentioned above. What is more, due to using a decoupling strategy, our approach complicated the super-net progressively. For the search method, we followed the gradient-based methods [34] to train the super-net level by level. Benefiting from the continuous search space, the super-net in each level could be trained end-to-end, which means it is more flexible and it becomes easier to decode the superior subnet. After obtaining the subnet on each level, we used it to generate the next step of the super-net to make the whole decoupling search coherent. Though the search process is divided into three stages and the super-net needs to train and regenerate three times, the search is still efficient. Finally, we obtained a three-level NAS framework, which covers as much of the network architecture candidates as possible, and the memory consumption is also acceptable for HRSI. Massive experiments on the GID [26] and FU [41] datasets demonstrated that our proposed DNAS can outperform the state-of-the-art handcrafted DCNNs (e.g., PSPNet [42], HRNet [19], and MSFCN [11]) and some existing NAS methods [35,43]. We have released the source code for DNAS at https://github.com/faye0078/DNAS.

Figure 1. The framework of decoupling neural architecture search.
To summarize, our main contributions are as follows:

1. We propose a novel decoupling NAS (DNAS) framework for HRSI semantic segmentation, which can automatically design DCNNs in a data-driven manner. The effectiveness of the proposed framework was verified on the GID and FU datasets.

2. A three-level super-net design is first proposed in this work. Compared with the existing NAS methods, our hierarchical super-net can cover much more network architecture candidates under the same hardware.

3. A decoupling search strategy is designed for the three-level super-net, which can largely reduce the display memory consuming during training of the super-net. Therefore, our method can adapt to HRSIs and is able to search for a more suitable network architecture.

The rest of this paper is organized as follows. Related work is discussed in Section 2. Section 3 describes our proposed method. In Section 4, we show the evaluation experimental results. Finally, Section 5 concludes this paper.

2. Related Work

In the following, we first review the development of NAS methods in natural image processing. Then, we present a brief description of existing NAS methods for HRSI processing, including classification and semantic segmentation.

2.1. NAS Methods for Natural Image Processing

Over the past years, many NAS methods have been proposed to promote the process of automatic architecture engineering and have achieved excellent success in different image processing tasks, such as image classification [27,34] and semantic segmentation [35,43]. Early NAS works [27,44] mostly aimed at image classification and directly searched the whole network architecture based on RL or EA. All candidates were trained from scratch to validate the performance. Though these works achieved impressive results, their expensive computation overheads hindered their applications in common downstream tasks (e.g., semantic segmentation). To alleviate this situation, the researchers modified the NAS method in two aspects: search space and search strategy. For search space, researchers proposed restricted search spaces. NASNet [45] first introduced the cell-based search space and searched the single cell architecture, which constrained the size of the search space, so as to make the search process easier. After that, many works [35,43,46] followed the cell-based search space design. For search strategy, the weight-sharing mechanism was introduced to NAS, which meant the search process did not need to train each candidate from scratch, but only one super-net. Gradient-based NAS [34,47] and one-shot NAS [33,48] methods both adapt this mechanism. Benefitting from improvements in these two aspects, NAS methods have gradually been used for more complex semantic segmentation tasks. Due to semantic segmentation demands, both global contextual semantic information and local detailed features, led to researchers beginning to design more flexible search space, containing a multi-scale path. For example, Auto-deeplab [35] introduced a scale path into the search space for the first time, and designed a two-level hierarchical search space. DCNAS [49] designed a more complex super-net structure than Auto-deeplab, which contains cross-layer connections in the search space and adopts a lighter cell structure. In this work, we designed a more flexible search space to extract multi-scale features.

2.2. NAS Methods for HRSI Processing

Currently, most of the NAS methods for HRSI focus on image classification. In [36,38,40] the cell-based search space and gradient-based search strategy were followed. In [37] the NAS method based on EA was used, where the computational complexity and the performance error of the searched network are balanced by employing the multi-objective optimization method. Only a few works have targeted HRSI semantic segmentation. RSNet [39] follows the two-level hierarchical search space in Auto-deeplab, and
adapts the gradient-based search strategy. RSNet [50] is based on the one-shot NAS method and uses EA to search the optimal architecture. Notably, RSNet and RSBNet all search the backbone architecture, which is manually equipped with different recognition heads in the final retraining process.

Above all, the most similar works to ours are Auto-deeplab [35], DCNAS [49], and RSNet [39]. They all use the gradient-based NAS method and design a multi-scale search space for semantic segmentation. However, since these methods use huge super-net, which makes it difficult for the size of the input image to meet the needs of efficient convergence of the super-net, especially for HRSI, the NAS result may be adversely affected.

3. Methodology

In DNAS, three levels of decoupling NAS framework are proposed for HRSI semantic segmentation. We summarize the topology of the DCNN architecture as follows: scale selection path, cross-layer connection, and convolution operator. According to these three characteristics, we constructed a path-level search space, connection-level search space, and cell-level search space, respectively, which decouples the search space so that the entire search process is divided into three stages and forms a progressive search strategy. The overall pipeline of our method is illustrated in Figure 1. In the following subsections, we firstly explain the three-level search space in detail and, then, we expound on the gradient-based search method and decoding method for different levels of the super-net.

3.1. Decoupling Architecture Search Space

Inspired by the idea of a hierarchical search space [35], we divided the search space into three levels and constructed them decoupled. For the path-level, we used the path search space similar to [35], which can cover various options for scaling paths. For the connection-level, we added skip connections between cells to aggregate features from the preceding module and expand the search space. For the cell-level, we populated each cell with specific operators, which are prevalent in modern DCNNs.

3.1.1. Path-Level Search Space

The selection of various scale features is crucial for semantic segmentation, and a key aspect influencing the scale features is the network scale path. Therefore, to obtain an optimal scale path, we adopted the skeleton hyper-network in [35] as the path search space, as shown in Figure 2. Unlike [35], we only used a simple $3 \times 3$ convolution kernel to fill each cell, focusing the search process on scale path while reducing the display memory consumption of the super-net. Specifically, the super-net structure could be abstracted as a directed acyclic graph, and each cell was simply connected to the adjacent cells before and after through three sampling methods: down-sampling, up-sampling, and keeping. The input feature $F_i^s$ of each cell can be expressed as:

$$F_i^s = \beta_{i}^{s,s} \text{Conv} \left( F_{i-1}^s \right) + \beta_{i}^{s,s} \text{Conv}(F_i^s) + \beta_{i}^{s,s} \text{Conv}(F_{i+1}^s)$$

(1)

where $(s, i) \in C_1 = S \times L$ represents all cells in the search space, $S = 4, 8, 16, 32$ represents the scale space of the super-net, $L$ represents the length of the super-net, and $\text{Conv}$ represents the $3 \times 3$ convolution operation, $\beta_{i}^{s,s}$ represents the architecture weight of the path-level super-net, which is constrained by the softmax function in the process of super-net training as:

$$\beta_{i}^{s,s} = \frac{\text{exp} (\beta_{i-1}^{s,s})}{\sum_{s \in S} \text{exp} (\beta_{i}^{s,s})}$$

(2)

In the same way as [34], we used the gradient-based NAS method for the super-net. The architecture weight $\beta_{i}^{s,s}$ is optimized during super-net training and selected during super-net decoding, which is described in detail in Section 3.2. Since only a simple single
convolution kernel was used to fill each cell, the entire super-net is lighter than [35], and it is more capable of processing higher resolution and batch size images. Meanwhile, the lightweight cell structure makes the search focus on scale selection, which leads to a better scale path.

![Figure 2. Path-level Search Space.](image)

### 3.1.2. Connection-Level Search Space

After searching through the scale path, we fixed the searched path as P and added cross-layer connections with forward cells to it, as shown in Figure 3. In contrast to the scale path search space, the path of the backbone network was fixed, and the corresponding network architecture parameters were no longer updated. At the same time, we deleted lower-scale cells and introduced complex connections to the scale path P, which was regarded as a critical path. Specifically, we increased the cross-layer connections between the cells on P and the forward cells and retained high-scale cells, which further increased the complexity of the connection between cells. Each cell still used a single convolution operator. Finally, the cells were divided into normal cells and complex cells according to whether they were on the critical path P. For normal cells, in the same way as for the cell structure of the path-level search space, they were only connected with the adjacent cells before and after by up and down sampling. Let \( \gamma \) represent the network parameters, the input features of normal cells can be expressed as:

\[
F^s_i = \gamma^{s-s^2}_{i-1} \text{Conv} \left( F^s_{i-1} \right) + \gamma^{s-s^2}_{i-1} \text{Conv} \left( F^s_{i-1} \right) + \gamma^{2s-s^2}_{i-1} \text{Conv} \left( F^s_{i-1} \right)
\]  

(3)

For complex cells, due to the addition of cross-layer connections with previous cells, the input features are represented as:

\[
F^{s2}_{i_2} = \sum_{s \in S, i < i_2} \gamma^{s-s^2}_{i-i_2} \text{Conv} \left( F^s_i \right) + \beta^{s1-s^2}_{i_1-i_2} \text{Conv} \left( F^{s1}_{i_1} \right)
\]  

(4)

and the network parameters are constrained by the softmax function as:

\[
\gamma^{s1-s^2}_{i_1-i_2} = \frac{\exp \left( \gamma^{s1-s^2}_{i_1-i_2} \right)}{\sum_{s \in S, i < i_2} \exp \left( \gamma^{s-s^2}_{i-i_2} \right)}
\]  

(5)
where $s_y, l_y$ represents the scale of the output feature and the super-net length of the corresponding position, and $s_{y-1}, l_{y-1}$ represents the scale of the previous critical path and the supernet length of the corresponding position.

Using the gradient-based NAS method, $\gamma$ is also optimized in the process of supernet training and selected in the process of super-net decoding. We highlighted the role of cross-layer connections in the search process by fixing the network parameters $\beta$ of the critical path, while still populating cells with a simple convolution operator.

### 3.1.3. Cell-Level Search Space

In the cell-level search space, we defined the operators of each cell, as shown in Figure 4. Specifically, the simple single $3 \times 3$ convolution operation in the path and connection-level search space were replaced by a richer set of operators:

$$Conv(F^i_t) \rightarrow \sum_{op \in OP} (a^i_t)_{op} op(F^i_t)$$

The operators in $\text{OP}$ are popular in current deep network designs, as shown in Table 1. Compared to [35,39,49], we used a wider variety of operator operations, including different types of pooling operators, convolution operators of different types, kernel sizes, and dilation rates. Thanks to the decoupling search space design, although the number of operators increased, fewer computing resources would be consumed. In addition, since the cross-layer connections between cells have been considered in the connection search space, equality operations were not included in these operators.

### Table 1. Candidate operators in the cell-level search space.

<table>
<thead>
<tr>
<th>Type</th>
<th>Kernel Size</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pooling</td>
<td>$3 \times 3$</td>
<td>max pooling</td>
</tr>
<tr>
<td></td>
<td>$3 \times 3$</td>
<td>average pooling</td>
</tr>
<tr>
<td></td>
<td>$-$</td>
<td>global pooling</td>
</tr>
<tr>
<td>Convolutions</td>
<td>$1 \times 1$</td>
<td>convolution</td>
</tr>
<tr>
<td></td>
<td>$3 \times 3$</td>
<td>convolution</td>
</tr>
<tr>
<td></td>
<td>$3 \times 3$</td>
<td>atrous conv with rate 3</td>
</tr>
<tr>
<td></td>
<td>$3 \times 3$</td>
<td>atrous conv with rate 12</td>
</tr>
<tr>
<td></td>
<td>$3 \times 3$</td>
<td>depthwise-separable conv</td>
</tr>
<tr>
<td></td>
<td>$5 \times 5$</td>
<td>depthwise-separable conv</td>
</tr>
<tr>
<td></td>
<td>$7 \times 7$</td>
<td>depthwise-separable conv</td>
</tr>
</tbody>
</table>

In the cell-level search space, the topology between cells was fixed, and only the network architecture weight $\alpha$ of the operator was updated in the process of super-net optimization.
3.2. Optimization and Decoding of Differentiable Decoupling Search Spaces

The network architecture parameters \((\beta, \gamma, \alpha)\) were used in the construction of the search space of the above three-level search spaces. We directly trained the entire super-net, and decoded the search results through the network architecture parameters. Below, we elaborate on the training optimization of the three-level super-net and the corresponding decoding process.

3.2.1. Optimization of Decoupling Search Spaces

Inspired by [35], when conducting a differentiable search, we divided the training data into two parts, respectively optimizing the original weight parameters of the training model and the architecture parameters introduced when building the search space. The three-level search space contains network architecture parameters \((\beta, \gamma, \alpha)\), which represent scale paths, cross-layer connections, and operators, respectively. Let \(\xi_i (i = 1, 2, 3)\) represents \(\beta, \gamma, \alpha\), and \(\omega\) represents the weight parameters of the network model. The super-net optimization process can be expressed in the following way: (1) Divide the training data into training set A and training set B equally. (2) The network weight parameters \(\omega\) are updated by calculating \(\nabla L_{trainA}(\omega, \xi_i)\) through the training set A. (3) The architecture parameters \(\xi_i\) are updated by calculating \(\nabla L_{trainB}(\omega, \xi_i)\) through the training set B, where the function \(L\) is the cross entropy loss function. After completing the super-net training, the searched network structure can be obtained by different decoding strategies.

Since we designed a three-level search space, the training of the super-net needs to be performed three separate times. However, the total time of the three searches does not increase compared to [35,39] because the super-net in each search stage is sufficiently lightweight compared to the coupled methods.

3.2.2. Decoding of Decoupling Search Spaces

After completing the search phase of each search space, the optimized network structure parameters need to be decoded to construct the optimal network architecture obtained from the search. In our work, the network structure parameters of the three stages needed to be decoded separately. These decoding processes are described separately below.

(1) Scale structure parameter decoding: In this search space, since the structure parameter \(\beta\) of each cell satisfied Equation (2), the parameter could be regarded as the probability of each scale selection. Therefore, the goal of decoding was to find a path with the highest selection probability. We used the Viterbi algorithm [51] to solve the path with the highest probability. After obtaining the optimal path, \(P\), the construction of the connection search space needed to rely on the decoding result. The complex cell in Figure 2 is the optimal path obtained by decoding at this time.

(2) Connection structure parameters decoding: In order to ensure that the cross-layer connection was fully utilized, we used the decoding strategy in [49], and selected the connection with \(\gamma > 0\) for the \(\gamma\) before normalization in Equation (5). As mentioned above, the construction of the operator search space depended on the connection decoding and path decoding results.
Operator structure parameters decoding: Since more operator operations were used, compared with [35,39,49], we selected the largest three operations in the structure parameter $\alpha$ for each cell as the search results.

Finally, we constructed the searched segmentation network by concatenating the connection decoding results and operator decoding results. Then, we trained it on the full training set.

4. Experimental Results and Discussion

In this section, we first introduce the dataset used in the experiment. Then, we describe the detailed settings in the experiment and, finally, we give the experimental results and compare them with the current popular methods.

4.1. Datasets Description

To verify the effectiveness of the proposed method on high resolution remote sensing data, we conducted experiments on the Gaofen image dataset (GID) [26] and the FU dataset [41]. Some examples of the two datasets are shown in Figures 5 and 6.

The original GID dataset contains 150 HRSIs acquired by the Gaofen-2 satellite in more than 60 cities in China, and has the advantages of large coverage, wide distribution, and high spatial resolution. These images are widely distributed over a geographic area of more than 50,000 square kilometers. As shown in Figure 5, the GID dataset image contains 4 bands: red, green, blue, and near-infrared bands. The labels contain 5 types of objects: built-up, farmland, forest, meadow, and water. During the experiment, we cut the original image of size 6800 $\times$ 7200 and the corresponding label into blocks of size 512 $\times$ 512 and obtained a total of 31,500 high-score remote sensing images with labels. These blocks were randomly divided into a training set, a validation set, and a test set in a ratio of 6:2:2. When training the model and calculating the accuracy, we did not consider unlabeled black pixels.

The FU dataset contains 321 HRSIs from 16 cities in different regions of France, each with a size of 10,000 $\times$ 10,000. The FU dataset contains 5 bands: red, green, blue, slope, and aspect. The labels contain 12 ground object categories: built-up, infrastructure, mining land, artificial meadow, arable land, permanent land, pasture, forest, shrub land, bare land, wetlands, and water. Similarly, the original image and the corresponding labels were cut into blocks of 512 $\times$ 512 size. These blocks were randomly divided into a training set, a validation set, and a test set in a ratio of 6:2:2.

![Figure 5](image_url). The example image and corresponding label from the GID dataset.
Figure 6. FU dataset example image and corresponding classification label map.

4.2. Evaluation Metrics

In this paper, we calculated the overall accuracy (OA), the mean intersection over union (MIoU), and the frequency weighted intersection over union (FWIoU) to evaluate the performance of the water-body detection [52]. These evaluation indicators can be defined as Equations (7)–(9):

\[
MIoU = \frac{1}{n+1} \sum_{i=0}^{n} \frac{TP_i}{FN_i + FP_i + TP_i}
\]

(7)

\[
OA = \frac{TP + TN}{TP + FP + TN + FN}
\]

(8)

\[
FWIoU = \frac{1}{n+1} \sum_{i=0}^{n} \left( \frac{TP_i}{FN_i + FP_i + TP_i} \cdot \frac{TP_i + FN_i}{TP_i + FP_i + TN_i + FN_i} \right)
\]

(9)

where \(TP\), \(TN\), \(FP\) and \(FN\) are the number of true positives, true negatives, false positives, and false negatives, respectively, and \(n\) is the number of classes.

4.3. Experiment Settings

We set the length of the search space to \(l = 12, 14\) and set the number of operators to 3 when searching for operators. At the same time, the model started with a down-sampling module, which consisted of two convolution operations with stride 2, turning the spatial resolution of the feature to \(s = 4\), and increasing the number of channels to 40. Inside the super-net, the number of channels increased by the same multiple as the feature scale increased. For example, when \(s = 32\), the number of channels increases to 320. The model finally ended with an up-sampling module, which included two convolution operations and up-sampling operations with stride 1, and the dropout rates of the convolution operations were 0.5 and 0.1, respectively.

When performing the model search, the super-net was trained for a total of 60 epochs in the training set. We divided the training set into training sets A and B equally and adopted a random flipping data augmentation strategy during training. For the first 20 epochs, we only used training set A to train the model parameter \(\omega\), and for the last 40 epochs, we used training sets A and B to train the model parameter \(\omega\) and model structure parameter \(\xi_i(i = 1,2,3)\), respectively. Training set A used a stochastic gradient descent (SGD) optimizer with a momentum factor of 0.9 to update the model parameter. The learning rate decreased from 0.025 to 0.001, according to cosine annealing, which used a weight decay strategy with a coefficient of 0.0003. Training set B used an adaptive moment estimation (Adam) optimizer to update the structural parameter \(\xi\). The learning rate was
set to 0.003, and a weight decay strategy with a coefficient of 0.001 was used. For the searched model, the head and tail sampling modules were the same as above, and we replaced the super-net with the decoded model. During retraining, all training sets participated in training for 200 epochs using the same SGD optimizer and data augmentation strategy as during super-net training. The experiments were based on the pytorch framework and completed on a NVIDIA GeForce RTX 3090 GPU.

4.4. Comparison with the State-of-the-Art Methods

We conducted experiments on GID and FU datasets with the above experimental settings and compared our proposed method with some popular methods. The artificially designed models included PSPNet [42], DeeplabV3+ [18], HRNet [19], and MSFCN [11]. The backbone networks of PSPNet and DeeplabV3+ were resnet101. To ensure a fair comparison, these models were trained from scratch.

Tables 2 and 3 show that our method achieved the best MIoU on two datasets, and the visualization results of our method are shown in Figures 7 and 8. In detail, compared with the optimal comparison model MSFCN on the GID dataset, our method was 0.27% higher, and compared with the optimal comparison model Deeplabv3+ on the FU dataset, it was 1.09% higher. Figures 1 and 2 show the visualization results of our method on the two datasets, which was better than that of the comparison model. What is more, compared with artificially designed networks, DNAS was more flexible and robust. For example, MSFCN [11] achieved a sub-optimal MIoU score of 0.9112 on the GID dataset (0.9140 for DNAS), but the accuracy on the FU dataset was 0.4708, which was significantly lower than 0.5216 for DNAS. Deeplabv3+ [18] achieved a sub-optimal MIoU score on the FU dataset of 0.5107 (0.5216 for DNAS), but the MIoU on the GID dataset was 0.8752, which was significantly lower than 0.9140 for DNAS. This shows that the performance of artificially designed networks on different datasets is divergent, and it is often necessary to manually try various structures through a large number of experiments to obtain better prediction results. Our proposed search space can cover enough possible network architecture candidates, including comparison networks, such as Deeplabv3+, HRNet. Based on this search space, our method could search for network structures that outperform the above artificial networks. Therefore, by decoupling the NAS strategy, our method designs proprietary network structures for different datasets and achieves state-of-the-art results with stronger robustness.

In addition, we compared the proposed method with some existing NAS methods. In the experiments of Auto-deeplab, we used the officially provided super-net model and then completed the experiment according to the same settings. Due to the huge memory usage of the super-net, the batch size was only set to 2 during training. In the experiments of Fast-NAS, we used the officially provided code and searched 1000 discrete networks. Auto-deeplab uses a gradient-based NAS method, which is the same as DNAS, but the batch size of its input images during the search training process is too small, which makes it tough for the super-net to converge. Therefore, the network performance obtained by the search was not as good as DNAS. Fast-NAS uses the NAS method based on reinforcement learning. Since it only trains each selected sub-network for 6 epochs and employs a subset of datasets, the final retraining performance of the network was unstable. In DNAS, the search space is more complex than Auto-deeplab and Fast-NAS, and the super-net is trained for 60 epochs on the whole dataset, so the architecture parameters are well converged and can be decoded to a better network. Compared with the existing NAS methods, our proposed method was more suitable for HRSI.
<table>
<thead>
<tr>
<th>Model</th>
<th>MIoU</th>
<th>OA</th>
<th>FWIoU</th>
<th>Params</th>
<th>Memory</th>
<th>FLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSPNet [42]</td>
<td>0.5003</td>
<td>0.7181</td>
<td>0.5730</td>
<td>72.58 M</td>
<td>2820.78 M</td>
<td>279.64 G</td>
</tr>
<tr>
<td>Deeplabv3+ [18]</td>
<td>0.5107</td>
<td>0.7341</td>
<td>0.5896</td>
<td>59.35 M</td>
<td>1117.32 M</td>
<td>88.9 G</td>
</tr>
<tr>
<td>HRNet [4]</td>
<td>0.5034</td>
<td>0.7254</td>
<td>0.5782</td>
<td>65.85 M</td>
<td>1438.48 M</td>
<td>93.77 G</td>
</tr>
<tr>
<td>MSFCN [11]</td>
<td>0.4708</td>
<td>0.7090</td>
<td>0.5592</td>
<td>14.17 M</td>
<td>2769.02 M</td>
<td>197.62 G</td>
</tr>
<tr>
<td>Auto-deeplab [35]</td>
<td>0.4971</td>
<td>0.7268</td>
<td>0.5808</td>
<td>14.52 M</td>
<td>1035.15 M</td>
<td>18.92 G</td>
</tr>
<tr>
<td>Fast-NAS [43]</td>
<td>0.4586</td>
<td>0.7116</td>
<td>0.5606</td>
<td>2.82 M</td>
<td>792.26 M</td>
<td>4.2 G</td>
</tr>
<tr>
<td>Our DNAS (L = 12)</td>
<td>0.5155</td>
<td>0.7331</td>
<td>0.5870</td>
<td>6.57 M</td>
<td>1291.14 M</td>
<td>39.49 G</td>
</tr>
<tr>
<td>Our DNAS (L = 14)</td>
<td><strong>0.5216</strong></td>
<td><strong>0.7345</strong></td>
<td><strong>0.5912</strong></td>
<td>12.81 M</td>
<td>1360.04 M</td>
<td>46.97 G</td>
</tr>
</tbody>
</table>

Table 3. Comparison with the existing methods on the FU dataset.
Figure 7. Visible semantic segmentation of the GID dataset. (a) and (b) are raw images and ground truth, respectively. (c) and (d) are the results of PSPNet and the results of Deeplabv3+, respectively. The results of HRNet and the results of MSFCN are shown in (e) and (f), respectively. (g) presents the results of Auto-deeplab, and (h) presents the results of Fast-NAS. The results of our proposed DNAS are displayed in (i).
Figure 8. Visible semantic segmentation examples on the GID dataset. (a) and (b) are raw images and ground truth, respectively. (c) and (d) are the results of PSPNet and the results of Deeplabv3+, respectively. The results of HRNet and the results of MSFCN are shown in (e) and (f), respectively. (g) presents the results of Auto-deeplab, and (h) presents the results of Fast-NAS. The results of our proposed DNAS are displayed in (i).

4.5. Efficiency Analysis

Firstly, we analyzed the efficiency of the search process. In Table 4, we compare the NAS methods using the gradient descent strategy. Benefiting from the decoupling search space design, each level super-net was lightweight enough for training to converge. Path-cell super-net was not filled with complex cell structures. Connection-level and cell-level
super-net were constrained by the decoding results of the previous stage. Although our method needed to go through a three-stage search process, the search time did not increase. At the same time, our method could accommodate larger input image size and training batch size, making the hyper-network easier to converge, so that a more suitable network architecture could be searched.

Table 4. The searching efficiency comparison on the GID dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Image size</th>
<th>Batchsize</th>
<th>GPU Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto-deeplab [35]</td>
<td>512</td>
<td>2</td>
<td>2.5</td>
</tr>
<tr>
<td>RSNet [39]</td>
<td>256</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>Our DNAS</td>
<td>512</td>
<td>24, 16, 6</td>
<td>5.3</td>
</tr>
</tbody>
</table>

We also analyzed the efficiency of the network models obtained from the search, as shown in Tables 2 and 3. The input image size of all experiments was $512 \times 512$. Compared with the artificially designed network [18,19,42], the searched network has fewer parameters, memory usage, and computation. In [43] a network search method for real-time semantic segmentation tasks was designed, so the obtained network is the most lightweight, and the search for cross-layer connections is not introduced in [35]. So, the searched network structure was more lightweight than our proposed method.

4.6. Ablation Study

Finally, we conducted ablation experiments on the method on the GID dataset. This part verified the role of the three search spaces we designed and the effect of the size of the hyperparameter $l$. on the results, as shown in Tables 5 and 6, respectively. For the three-level search space, we trained the network obtained by each stage search on all training sets. In the three-level search, the architecture of the network was continuously optimized. The path-level determined the backbone structure of the network, the connection-level determined the cross-layer connections of the network, and the cell-level determined the specific operators of each cell. It can be seen from Table 5 that as the search continued, the performance of the network was getting better and better, which proved the effectiveness of our designed decoupling search space. On the other hand, for the search space length $l$, we set it to 12, 14, and 16 for experiments. The best performance was the super-net with $l = 14$. Intuitively, with the complexity of the search space, the obtained model should be better. However, from another perspective, the expansion of the search space made optimization difficult, so that the optimal network could not be effectively searched. In fact, during the experiment, the training batch size of the super-net was only 2 when $l = 16$, which was the same as the input of the other NAS methods above. This also proved that we needed a suitable search space that could adapt to image inputs with enough resolution and batch size, and which could converge effectively.

Table 5. The efficiency analysis of three-level search space on the GID dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Built-Up</th>
<th>Farmland</th>
<th>Forest</th>
<th>Meadow</th>
<th>Water</th>
<th>MIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours-Stage1</td>
<td>0.9583</td>
<td>0.9392</td>
<td>0.9509</td>
<td>0.5328</td>
<td>0.9430</td>
<td>0.8648</td>
</tr>
<tr>
<td>Ours-Stage2</td>
<td>0.9615</td>
<td>0.9450</td>
<td>0.9511</td>
<td>0.5786</td>
<td>0.9511</td>
<td>0.8775</td>
</tr>
<tr>
<td>Ours-Stage3</td>
<td>0.9685</td>
<td>0.9584</td>
<td>0.9694</td>
<td>0.7121</td>
<td>0.9616</td>
<td>0.9140</td>
</tr>
</tbody>
</table>

Table 6. The efficiency analysis of different $l$ on the GID dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Built-Up</th>
<th>Farmland</th>
<th>Forest</th>
<th>Meadow</th>
<th>Water</th>
<th>MIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l = 12$</td>
<td>0.9678</td>
<td>0.9550</td>
<td>0.9680</td>
<td>0.6206</td>
<td>0.9585</td>
<td>0.8940</td>
</tr>
<tr>
<td>$l = 14$</td>
<td>0.9685</td>
<td>0.9584</td>
<td>0.9694</td>
<td>0.7121</td>
<td>0.9616</td>
<td>0.9140</td>
</tr>
<tr>
<td>$l = 16$</td>
<td>0.9663</td>
<td>0.9499</td>
<td>0.9565</td>
<td>0.5543</td>
<td>0.9537</td>
<td>0.8762</td>
</tr>
</tbody>
</table>
5. Conclusions

For the remote sensing field, which has abundant image data, NAS becomes important, due to its characteristic data-driven automatic design of network architecture. To efficiently converge the NAS process, while enriching the search space, this paper proposed a novel DNA method for HRSI semantic segmentation. In the DNA, the search process is divided into three stages to automatically construct a semantic segmentation DCNN model with extremely high complexity. For the first time, we introduced the search of cross-layer connections, which increases the search space complexity and can extract richer image features. Meanwhile, through the method of decoupling search, the method ensures that the super-net can input enough training images in the search process, so that the super-net can effectively converge to get a more suitable search model.

The method was experimented on with two datasets to analyze its accuracy and efficiency, respectively. We compared our method with popular deep network methods and some NAS methods on the GID and FU datasets. The experiments demonstrated that our method outperformed the compared methods in accuracy. Compared with other NAS methods, our method makes it easier to search for excellent models, because it can accept enough input image. In future work, we may expand the existing operator search space (e.g., some remote sensing operators) and prepare to introduce a more efficient neural search method in the third stage of search.

Author Contributions: Conceptualization, Y.L. (Yansheng Li); methodology, Y.W., Y.L. (Yansheng Li); validation, Y.W., W.C.; writing—original draft preparation, Y.W., Y.L. (Yunzhou Li); writing—reviewing and editing, Y.L. (Yunzhou Li), B.D., W.C.; project administration, Y.L. (Yansheng Li); funding acquisition, Y.L. (Yansheng Li) All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Natural Science Foundation of China under grant 41971284; the State Key Program of the National Natural Science Foundation of China under grant 42030102; the Fundamental Research Funds for the Central Universities under grant 2042022kfl201; Zhizhuo Research Fund on Spatial-Temporal Artificial Intelligence under grant ZZJJ202210.

Data Availability Statement: The experiments are conducted on publicly open datasets. The download sites of the publicly open datasets can be found in the corresponding published papers; we do not repeat these sites here.

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

References


