Hierarchical Shared Architecture Search for Real-time Semantic Segmentation of Remote Sensing Images

Wenna Wang*, Lingyan Ran*, Hanlin Yin, Mingjun Sun, Xiuwei Zhang, Yanning Zhang, *Senior Member, IEEE*

***Abstract*—Real-time semantic segmentation of remote-sensing images demands a trade-off between speed and accuracy, which makes it challenging. Apart from manually designed networks, researchers seek to adopt neural architecture search (NAS) to discover a real-time semantic segmentation model with optimal performance automatically. Most existing NAS methods stack up no more than two types of searched cells, omitting the characteristics of resolution variation. This paper proposes the Hierarchical shared Architecture Search (HAS) method to automatically build a real-time semantic segmentation model for remote sensing images. Our model contains a lightweight backbone and a multi-scale feature fusion module. The lightweight backbone is carefully designed with low computational cost. The multi-scale feature fusion module is searched using the NAS method, where only the blocks from the same layer share identical cells. Extensive experiments reveal that our searched real-time semantic segmentation model of remote sensing images achieves the state-of-the-art trade-off between accuracy and speed. Specifically, on the LoveDA, Potsdam, and Vaihingen datasets, the searched network achieves 54.5% mIoU, 87.8% mIoU, and 84.1% mIoU, respectively, with an inference speed of 132.7 FPS. Besides, our searched network achieves 72.6% mIoU at 164.0 FPS on the CityScapes dataset and 72.3% mIoU at 186.4 FPS on the CamVid dataset.**

***Index Terms*—real-time semantic segmentation, neural network architecture search, feature aggregation module, hierarchical shared search strategy.**

I. Introduction

Semantic segmentation of remote sensing images holds significant importance for applications, such as comprehensive land use mapping [1] and urban change detection [2]. Most solutions consume massive computing costs, which brings challenges for moving platforms in reality that need an instant response, such as self-driving cars [3], real-time disaster monitoring [4], etc. Therefore, more and more researchers are paying attention to real-time semantic segmentation of remote sensing images to find a trade-off between speed and precision.

Many excellent works [5], [6], [7], [8], [9], [10], [11] have been proposed in real-time semantic segmentation, which can be transferred to remote sensing scene. Some researchers reduce the computation cost of the network by decreasing the number of channels or limiting the input size, such as ENet [12] and ESPNet [13]. Nowadays, most works usually adopt lightweight classification networks as their backbone and design an efficient feature aggregation part to improve the speed and remedy the accuracy decreasing simultaneously, such as SwiftNet [14], DFANet [15], and BiSeNet [16]. Their differences mainly rely upon feature aggregation modules since how to design an efficient feature aggregation model is the key to real-time semantic segmentation. Recently, there are also some real-time semantic segmentation methods that are directly designed for remote sensing images, such as ABCNet [17], DSANet [18], and UNetFormer [19]. Although these artificially designed models have made some progress, they have the following drawbacks: (1) Existing real-time semantic segmentation methods of remote sensing images are designed by human experts, and their performance is limited by expert experience, so the obtained architecture may not be optimal. (2) Existing methods adopt the same feature fusion structure for various feature maps during the feature fusion process, making the network unable to fully obtain context information at different scales. Therefore, we attempt to use neural architecture search (NAS) to automatically design a real-time semantic segmentation network for remote sensing images.

Recently, NAS has made remarkable achievements in the automatic design of efficient networks [20]. NAS aims to discover the optimal network by applying an efficient search strategy in a well-designed search space. At present, NAS has achieved remarkable progress in image classification [21], [22], [23], semantic segmentation [24], [25], etc. In this paper, we focus on exploring NAS to automatically design an efficient real-time semantic segmentation model for remote sensing images, to alleviate the problem of human-designed networks consuming too many resources and relying heavily on expert experience. Autodeeplab [26] is an earlier work introducing NAS to semantic segmentation. It searches the cell architecture and the connection operations between two nodes (identity, downsampling, and upsampling). GAS [27] explores graph convolution neural network to transfer structure information between cells and introduce latency constraints to improve the semantic segmentation speed. Previous methods follow the pipeline to search an optimized cell and stack copies of it with task demand. Although searching for one single fixed cell can improve the searching speed, it ignores the variations

Wenna Wang, Xiuwei Zhang, Mingjun Sun, Lingyan Ran, Hanlin Yin, and Yanning Zhang are with the Shaanxi Provincial Key Laboratory of Speech and Image Information Processing, the National Engineering Laboratory for Integrated Aero-Space-Ground-Ocean Big Data Application Technology, School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China.

This work was supported in part by the National Natural Science Foundation of China under Grants 61971356, U19B2037, and 62201467, in part by the Natural Science Foundation of Shaanxi province under Grants 2021KWZ-03 and 2022JQ-686, in part by the Natural Science Foundation of NingBo under Grants 2023J262. (Wenna Wang and Lingyan Ran contributed equally to this work) (Corresponding author: Hanlin Yin, Xiuwei Zhang)

of feature maps in the size and the channel numbers at various stages of the network, which may decrease the performance.

Therefore, we propose the hierarchical shared architecture search (HAS) method for real-time semantic segmentation of remote sensing images. Compared with existing methods, the proposed method has the following advantages: (1)Automatic design based on NAS. We introduce NAS to break through the limitations of human expertise by automatically designing the network structure, thus discovering a superior segmentation architecture. (2) Automatic design of different feature fusion structures (cells) for various resolutions. Our method automatically designs different cells for feature maps of various resolutions in our carefully designed search space. Compared to existing NAS algorithms that only search for a single cell structure, our method can better capture multi-scale semantic information. Combined with our carefully designed lightweight backbone, our method achieves improved semantic segmentation results while effectively balancing speed and accuracy. Experiments results show that our model achieves a state-of-the-art trade-off between speed and accuracy. The proposed method achieves 54.5%, 87.8% mIoU, and 84.1% mIoU on the LoveDA, Potsdam, and Vaihingen datasets respectively, with an inference speed of 132.7 FPS on a GeForce RTX 3090 with an image size of 1024×1024 pixels. Besides, the proposed model achieves 72.6% mIoU and 164.0 FPS on the CityScapes dataset and 72.3% mIoU and 186.4 FPS on the CamVid dataset.

The main contributions are as follows:

- We design a novel real-time semantic segmentation network for remote sensing images that consists of a well-designed lightweight backbone and a feature aggregation module, where the feature aggregation module is automatically designed by the NAS method.
- We propose a hierarchical shared search strategy with a novel search space, namely HAS. In this search space, different network layers have various cell architectures, which can better capture contextual information at different scales.
- We adopt the HAS strategy combined with a carefully designed lightweight backbone to automatically design the best network. Furthermore, to improve the search speed, we propose an operation selection strategy, which selects important operations to participate in network searches to enhance search efficiency. Experimental results show that the searched network achieves a state-of-the-art balance between speed and accuracy.

The rest parts are organized as follows. A detailed introduction to the relevant work is provided in Section II. Section III introduces the proposed network structure, search space, and search method. Section IV shows the specific searched cell architecture and introduces experimental results. The conclusion is given in Section V.

II. Related Work

In recent years, real-time semantic segmentation and NAS, have achieved impressive results in domains such as remote sensing scene perception and autonomous driving. This section provides a comprehensive review of relevant works of real-time semantic segmentation and NAS.

A. Semantic Segmentation

Semantic segmentation is a crucial task that aims to partition images into different semantic regions. Fully convolutional network (FCN) [28] is a common forerunner based on CNN, which can learn the hierarchical structure of features. Later on, various methods based on a type of encoder-decoder backbone network appear. SegNet [29] uses max pooling indexes technology to upsample features in the decoding stage. U-net [30] introduces skip-connection between the decoder layer and the encoder layer. DeepLab [31] presents an atrous spatial pyramid pooling (ASPP) module, which can capture multi-scale features of objects. PSPNet [32] proposes a pyramid pooling module to better integrate context information. DANet [33] enhances feature representation by introducing an attention module. HDNet[34] proposes a hybrid distance network to get the context information of each point.

B. Real-time Semantic Segmentation

Due to the real-time requirement of various application platforms with limited computing resources, such as real-time disaster monitoring, self-driving cars, and so on, an increasing number of researchers have turned their attention to real-time semantic segmentation. ENet [12] designs an encoder-decoder architecture, which is specifically created for tasks requiring low latency. ICNet [13] incorporates multi-resolution branches to effectively utilize high-resolution and low-resolution information, which can quickly achieve high-quality segmentation. BiseNet [9] uses two branches to get spatial information and context information respectively. Fast-SCNN [8] proposes a learning-to-downsample method that enables the two-branch network to share shallow feature maps. SwiftNet [14] employs lightweight upsampling modules to reconstruct images. BiseNetv2 [16] proposes a feature aggregation module based on bi-directional aggregation, which enables the information of two paths to be fully integrated. DSANet[35] reduces computation by employing channel split and shuffle strategies in the semantic encoding branch. SGCPNet [11] designs a context propagation component to reconstruct the lost spatial information and improve the model efficiency. SegNeXt [36] design a new network structure, which has the characteristics of a large receptive field, multi-scale, and adaptability.

C. Real-time Semantic Segmentation of Remote Sensing Images

Recently, in addition to RGB images, there has been significant attention given to real-time semantic segmentation for remote sensing images. Liu et al. [37] present a path-wise semantic segmentation method for remote sensing images. DDRNets [38] adopts a new contextual information extractor combined with multiple bilateral fusions to achieve a trade-off between speed and accuracy. ABCNet [17] introduces a novel feature aggregation module and an attention enhancement module based on Bisenet [9]. DSANet [18] proposes a simple

attention module to improve the inference speed for large images. UNetFormer [19] proposes a UNet-like Transformer to realize an efficient segmentation of remote sensing images.

D. Neural Architecture Search

NAS can automatically discover the best-performing neural network, which has attracted much attention because of its excellent performance. Early NAS methods are mainly based on reinforcement learning (RL) [21], [39], [40] and evolutionary algorithm (EA) [41], [22]. RL-based methods usually comprise a controller, which is used to sample a network structure. Then the sampled network is trained from scratch and returns a validation accuracy, which is used to update the parameters of the controller. EA-based methods are alternatives to RL-based methods, which evolve the network via mutation and recombination.

Although the above methods have made promising progress, they require huge computational resources. To address this issue, one-shot methods are proposed. One-shot methods train the parent network only once, and each sub-network inherits the weights of the parent network. NAO [42] embeds architecture into a latent space and performs optimization before decoding. ENAS [43] introduces a parameter-sharing method between sampled networks. DARTS [23] relaxes a discrete search space to be continuous, so gradient descent can be used for optimization. This dramatically reduces the resources needed for the search. Although DARTS [23] greatly reduces the search time, the introduction of architecture parameters inevitably leads to the problem of occupying too much memory. P-DARTS [44] increases the network depth and reduces the type of operations stage by stage, so as to reduce the occupation of memory. PC-DARTS [45] proposes a sampling mechanism based on channel, which lets only $1/K$ channel nodes train each time, and thus greatly reduces the memory consumption. In a similar vein of research, [46] and [47] progressively increase network complexity while training ranking networks in a parallel manner to reduce resource consumption.

Since DARTS [23] was introduced, many approaches have been carried out based on this method. DARTS greatly reduces the computing resources required by reinforcement learning or evolutionary algorithm and greatly promotes the development of NAS. Our work also adopts a differentiable NAS method and extends it to a more general hierarchical setting.

E. NAS for Semantic Segmentation

Due to the success of NAS in image classification, NAS has been introduced into other fields, such as semantic segmentation, object detection, etc. For semantic segmentation, DPC [48] constructs a novel search space that successfully extends NAS to dense image prediction tasks. AutoDeeplab [26] simultaneously searches cell architecture and resolution size to meet semantic segmentation requirements. Recently, CAS [49] proposes a searched multi-scale structure to make NAS more suitable for semantic segmentation. GAS [27] introduces a graph convolution neural network to transfer structure information between cells so that each cell structure is independent of each other. M-FasterSeg [50] designs multiple resolution branches for real-time semantic segmentation, which is searched by NAS.

Although above approaches have made considerable progress, their search methods still imitate the image classification work and adopt the way of sharing one or two same cells, which is not conducive to processing multi-scale images. So, we propose a hierarchical shared search method to discover the network structure which is more suitable for semantic segmentation.

III. METHOD

This section describes our network structure and the proposed search method. Firstly, the whole network structure is introduced in detail, including a backbone and an automatically designed multi-scale feature fusion module. Then, an automatically design method for feature fusion modules is described, namely the proposed hierarchical shared architecture search method, including the search space definition and the search strategy.

A. Network Structure

The whole network structure mainly includes a backbone and a feature aggregation module, which is shown in Fig. 1.

1) Backbone: The backbone network consists of five layers: Layer 1 - Layer 5. Each of these five layers $2\times$ downsamples its input feature maps. First, a 3×3 convolution operation is performed on the RGB input image of size (w, h, 3) to obtain a feature map of size (w, h, 32). Then, two conventional convolution layers (Layer 1 and Layer 2) are added for feature maps' downsampling and channel expansion. Each of these two layers contains two 3×3 convolutions. Meanwhile, the number of channels from Layer 1 to Layer 2 gradually increases from 32 to 128. Finally, the next three layers (Layer 3 - Layer 5) are used to extract high-level semantic information. Each layer is a gather and expansion block (GEblock), which is shown in Fig. 2. A GEblock includes three layers, one GElayer S1 and two GElayer S2. The GElayers adopt the bottleneck architecture used by ResNet [51]. Compared with GElayer S2, in GElayer S1, there is an additional 3×3 depth-wise convolution with a stride of 2 in the main path and the shortcut path, which downsamples the input feature map and scales up the receptive field. The number of channels for the last three layers remains at 128.

2) Automatically Designed Multi-scale Feature Fusion Module: There are a lot of ways to combine two kinds of feature responses, such as concatenation and element-wise summation. However, the feature maps output by the backbone have different sizes. Simple composition neglects the variety of both kinds of information, resulting in poor performance and difficulty in optimization. So, an automatically designed multi-scale feature fusion module is proposed to utilize multi-scale feature information thoroughly.

The proposed ADMFF module is obtained with HAS. As shown in Fig. 1 (a), the ADMFF module is made up of six searching cells. Specifically, this module contains three layers and each layer includes two same cells. There are three

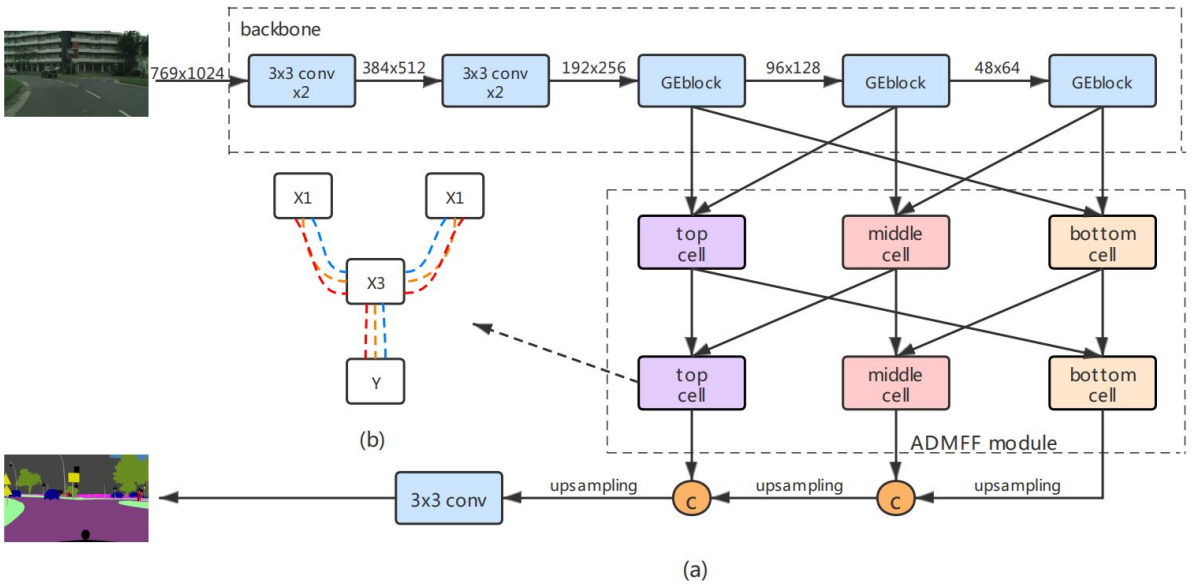

Fig. 1: The overall architecture. (a) the proposed network structure consists of a backbone and an ADMFF module. (b) the shared concrete structure of three types of cells in the ADMFF module.

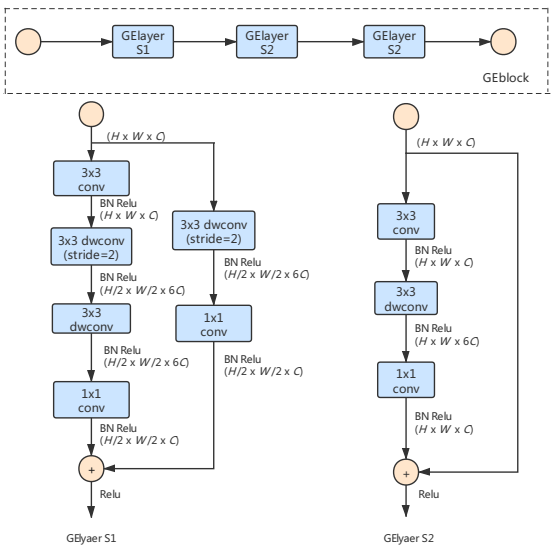

Fig. 2: The concrete structure of GEblock. It consists of three GElayers. GElayer S1 and S2 work as bottlenecks with similar structures. (*dwconv*: depth-wise convolution)

different cells, namely the top cell, the middle cell, and the bottom cell, respectively. The inputs of the feature aggregation module are the outputs of Layers 3, 4, and 5 of the backbone. The top cell and middle cell accept the output from their same layer and the deeper layer as their inputs. The inputs of the bottom cell are the output of the same layer and the top cell. Finally, we concatenate the outputs of the three-layer cells and get the last prediction result after a 3×3 convolution.

The automatic design method of the proposed feature aggregation module will be described in subsection III-B and subsection III-C. Following most NAS work, we describe the ADMFF module in terms of search space and search strategy.

B. Architecture Search Space

In this subsection, the search space built for searching the structure of the feature aggregation module is carefully introduced.

Most existing NAS approaches adopt the way of sharing one cell, so the network structure has many duplicate units. Although these methods can increase the efficiency of NAS, a single structure does not suit different feature maps very well. For semantic segmentation, different layers have to deal with feature maps of various resolutions, so it is better for each layer to design the corresponding structure according to diverse resolutions. Meanwhile, existing semantic segmentation networks have shown good performance in the feature extraction part, and recently researchers mainly devote their efforts to effectively fusing features of different layers. So, we also concentrate on the feature aggregation module and propose a hierarchical shared search space for the feature aggregation module.

As shown in Fig. 1 (a), the search space of the ADMFF module has three layers. Each layer corresponds to the input with different resolutions, and each layer has two cells with the same structure. In other words, there are three cells (top cell, middle cell, and bottom cell) that need to be searched in the search space.

The cell is represented as a directed acyclic graph (DAG), as shown in Fig. 1 (b). For speed concerns, the number of nodes is limited to 4 in the cell, containing two input nodes, an intermediate node, and an output node. The edge between nodes represents an operation o_i.

To reduce the dependence of search results on expert experience, we use a large search space O. There are three types of operations in O, i.e., non-learnable operations, standard

TABLE I: The whole search space consists of 14 operations. *Conv* is separable convolution, *dconv* represents dilated separable convolution.

Index	Operation	Index	Operation
1	skip connection	8	7 × 7 conv
2	3 × 3 max pooling	9	3 × 3 dconv (dilation=2)
3	5 × 5 max pooling	10	3 × 3 dconv (dilation=4)
4	3 × 3 avg pooling	11	3 × 3 dconv (dilation=8)
5	5 × 5 avg pooling	12	5 × 5 dconv (dilation=2)
6	3 × 3 conv	13	5 × 5 dconv (dilation=4)
7	5 × 5 conv	14	5 × 5 dconv (dilation=8)

convolutions, and dilated convolutions. Three types of non-learnable operations are considered, including avg pooling, max pooling, and skip connection. The kernel sizes for avg pooling and max pooling are 3 × 3 and 5 × 5, respectively. The standard convolutions with kernel sizes of 3, 5, and 7 are commonly utilized in CNNs. Dilated convolution can enlarge the receptive field without losing resolution and is extensively adopted in semantic segmentation. Therefore, we also adopt these operations in the operation set O. The dilated convolutions with the 2, 4, and 8 ratios are adopted. It is worth mentioning that we adopt the depth separable technique for both standard convolutions and dilated convolutions. Because depth-separable convolutions are more efficient than regular convolutions, we decompose the regular convolution into two steps, i.e., depth-wise convolution and point-wise convolution. So, in total, the operation set O consists of 14 operations, which is listed in Table I.

C. Hierarchical Shared Architecture Search Strategy (HAS)

In this subsection, we will describe the hierarchical shared search method, including continuous relaxation, optimization algorithm, and decoding.

1) Continuous Relaxation: Taking the top cell as an example, X_1 and X_2 denote two input nodes, Y represents the output node, and X_3 is the intermediate node. The input nodes X_1 and X_2 are transferred to the intermediate node X_3 after operation o. At node X_3, the two inputs are added and followed by a ReLU activation function. Finally, the output of X_3 is transferred to Y through operation o. So the whole cell can be represented by Eq. (1):

$$Y = Cell(X_1, X_2). \tag{1}$$

Each node in our cell is expressed as the sum of the outputs after k operations (k represents the number of operations). To make the search process differentiable, three vectors, α, β, γ, are introduced to make independent k operations continuous. Then, according to decoding rules, we determine which operation should be selected for the edge of the corresponding cell.

Within the cell, the transfer process of the precursor node to the next node can be expressed by Eq. (2):

$$\mathrm{y}_i = \sum_{o \in O} o(x_i) \tag{2}$$

where x_i represents the precursor nodes, y_i is the next node and o denotes one operation in operation set O.

Therefore, Eq. (1) can be expanded into Eq. (3):

$$Y = \sum_{o \in O} o(\sum_{o \in O} o(X_1) + \sum_{o \in O} o(X_2)) \tag{3}$$

To make the search space continuous, we introduce a $1 \times k$ vector for each edge, namely $\alpha_0, \alpha_1, \alpha_2$, which is used to measure the importance of each operation in operation set O. So, Eq. (3) can be expressed as:

$$Y = \sum_{o \in O} \alpha_2^o o(\sum_{o \in O} \alpha_0^o o(X_1) + \sum_{o \in O} \alpha_1^o o(X_2)), \tag{4}$$

where $\alpha = [\alpha_0, \alpha_1, \alpha_2]$ is the architecture parameter, and it is normalized by Eq. (5),

$$\alpha_i^o = \frac{exp(\bar{\alpha}_i^o)}{\sum_{o' \epsilon O} exp(\bar{\alpha}_i^{o'})} \tag{5}$$

where $i = 0, 1, 2$. α_i^o denotes the weight of the operation o on the ith edge. α_i^o satisfies $\sum_{o \epsilon O} \alpha_i^o = 1$.

The architecture vectors for middle cell β and bottom cell γ can also be done as that for top cell.

2) Optimization: Since the search space is differentiable, the optimization process can adopt cross optimization method [23]. The cross optimization combines the network training with the optimization of architecture parameters, which speeds up the network search speed. Because we introduce three different architecture parameters into the search strategy, we need to optimize the three parameters at the same time in the cross optimization, which is different from previous methods.

When the search space is continuous, the independent operations are linked by architecture parameters, which also need to be optimized. Since architecture parameters are involved in the calculation process of the neural network, they should be optimized at the same time. Therefore, after introducing the architecture parameters, the training process is split into two parts, one part optimizes the network parameters and the other part optimizes the architecture parameters.

Accordingly, the training set is divided into $trainA$ and $trainB$, which are used to learn the network parameters and architecture parameters, respectively. We calculate the loss on $trainA$ and $trainB$ and minimize them via the gradient descent method. The formula for this optimization process is:

- Updating ω by descending $\nabla_\omega L_{trainA}\ (\omega, \alpha, \beta, \gamma)$
- Updating architecture parameters α, β, γ by descending $\nabla_{\alpha,\beta,\gamma} L_{trainB}(\omega, \alpha, \beta, \gamma)$

where ω represents network weights. α, β, γ is the architecture parameters. L denotes the cross entropy loss.

In the optimization process, we follow the design of AutoDeepLab [26], which begins to optimize the architecture parameters α, β, and γ after training 20 epochs.

3) Decoding: When the network architecture search is completed, we need to decode the continuous search space to get the final network structure. Following the work of [23], for edge i, the final operation is derived by:

$$o_i = argmax_{o \epsilon O} \alpha_i^o \tag{6}$$

So, between two nodes, only the strongest operation is ultimately reserved. Similarly, the above processing is done for vectors β and γ. Finally, three structure fixed cells are generated through the above decoding, which are used to construct our feature aggregation module.

IV. Experiments

In this section, the datasets and the search process are presented in detail. Then, the operation set selection strategy is described. After that, the effectiveness of our hierarchical sharing strategy is verified. Finally, we show the semantic segmentation results of the proposed model on four datasets.

A. Dataset and Search Details

1) Dataset: To evaluate the effectiveness of our proposed method, experiments are conducted on the LoveDA [1], Potsdam [2], Vaihingen [3], CityScapes [52], and CamVid [53] datasets. LoveDA, Potsdam, and Vaihingen are three popular datasets used for semantic segmentation of remote sensing images. CityScapes is a large street view dataset, which is known for being a challenging dataset in semantic segmentation. CamVid is another widely used street view dataset for real-time semantic segmentation.

The LoveDA dataset is highly challenging. It comprises a total of 5987 fine-resolution optical remote sensing images(GSD 0.3 m), each with a size of 1024×1024 pixels. The dataset is divided into training, validation, and testing sets, containing 2522, 2669, and 1796 images, respectively.

The Potsdam dataset comprises 38 high-resolution images, each measuring 6000× 6000 pixels. Within the dataset, there are four multi-spectral bands (red, green, blue, and near-infrared), along with the Digital Surface Model (DSM) and Normalized Digital Surface Model (NDSM). To evaluate our models, we selected 23 images from the dataset for training and 14 images for model testing.

The Vaihingen dataset comprises 33 aerial images. Each image has dimensions of 2494×2064 pixels and consists of three bands corresponding to red, near-infrared, and green wavelengths. Additionally, the dataset includes DSM, which represents the height of all object surfaces within the images. In our experiments, 16 images are selected for training, while the remaining 17 images are used for testing.

The CityScapes collects a lot of stereoscopic video sequences, which cover 50 different cities' streets, consisting of 30 classes. There are 5000 pixel-level fine annotated images of size 1024 × 2048 and about 20000 coarsely annotated training images. Our experiments utilize 5000 fine pixel-level annotated images, with 1525 test images, 500 validation images, and 2975 training images.

The CamVid is a road scene set and contains 701 images with a size of 720 × 960 pixels. These images are divided into 233 test images, 101 validation images, and 367 training images.

[1]https://github.com/Junjue-Wang/LoveDA

[2]http://www2.isprs.org/commissions/comm3/wg4/2d-sem-labelvaihingen.html

[3]http://www2.isprs.org/commissions/comm3/wg4/2d-sem-labelpotsdam.html

TABLE II: Performance of different searched networks. Network-L, Network-P, and Network-C are searched on the LoveDA dataset, Potsdam dataset, and CityScapes dataset respectively. These three networks are retrained, and tested on the LoveDA dataset.

Network	mIoU(%)
Network-P	53.1
Network-C	54.2
Network-L	**54.5**

2) Search Details and Search Dataset: During the search process, the training images are sampled into two equal subsets: $trainA$ and $trainB$. One subset is used for optimizing the architecture parameters, while the other subset is used for optimizing the weight parameters of the network. The batch size is limited to 4 due to constraints in GPU memory. For architecture parameters α, β, and γ, we employ the Adam optimizer with a weight decay of 0.001 and a learning rate of 0.0003. For network weight ω, the SGD optimizer is adopted. The learning rate is 0.05, and a "poly" learning rate strategy is utilized. The search process is set to 200 epochs.

The search process is conducted on the LoveDA dataset. Since the LoveDA dataset is large, it can provide more samples and diversity for model searching, which helps to find a more robust model for real-time semantic segmentation of remote sensing images. The search process is also performed on the CityScapes [52] dataset and the Potsdam dataset. We tested the performance of different models which are searched on different datasets, and the results are shown in Table II. Experimental results show that Network-L can achieve better semantic segmentation accuracy on remote sensing images.

B. Operation Set Selection Strategy

To decrease the impact of human experience on search results, we introduce a large operation set containing 14 conventional operations. However, too many operations will reduce the search efficiency. Therefore, the entire network architecture search process is divided into two stages.

The first stage goes to search optimal network with the operation set mentioned above. To improve the experimental efficiency, we train the network 20 times with 30 epochs for each time and then count the weight of each operation each time. After 20 times of training, the average weight of each operation at the 30th epoch is obtained. Then, the ratio of the average weight of each operation to the total weight of all operations can be calculated. As shown in Fig. 3, we can see that the accumulated weight of the top 6 operations accounts for about 96 % of total operations, so they can be used to form the operation set for the next search stage.

The second stage is to search optimal network with these top 6 operations, including:

- 3 × 3 max pooling
- 3 × 3 avg pooling
- skip connection
- 3 × 3 conv

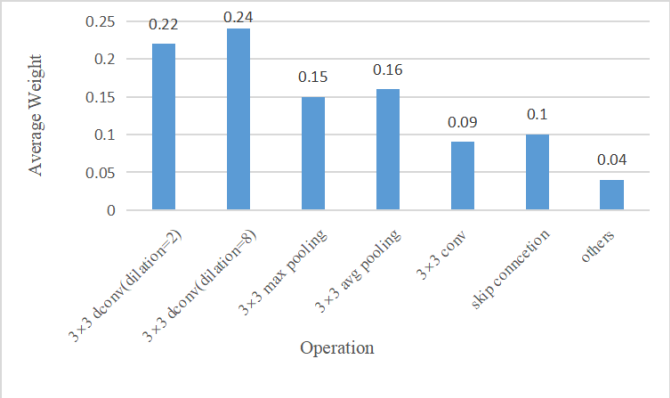

Fig. 3: The average weight of each operation after 20 experiments. *conv*: separable convolution, *dconv*: dilated separable convolution

TABLE III: The performance (mIoU(%)) of the HAS strategy and the traditional shared strategy on the LoveDA dataset and the CityScapes validation set.

Method	LoveDA	CityScapes
Traditional shared	51.6	71.3
HAS	**54.5**	**73.9**

- 3 × 3 dconv (dilation=2)
- 3 × 3 dconv (dilation=8)

We have a total of 14 candidate operations, so there are 14^9 structures in our search space (there are nine edges to be searched, and each edge has 14 choices). Thanks to the operation set selection strategy, the whole search process takes about 2 days on a GeForce GTX 1080 GPU. By comparison, GAS [27] takes about 6.7 days using a single TitanXP GPU.

C. Effectiveness of the HAS Strategy and the ADMFF Module

We propose the HAS method to search the cells that compose the feature fusion module, namely the ADMFF module. To illustrate the effectiveness of the HAS strategy and the ADMFF module, extensive ablation studies are conducted.

Firstly, for the HAS strategy, different search strategies are performed: a) stacking network by one shared cell (traditional shared); b) stacking network by independent cells between layers (HAS). These two search strategies are tested on the LoveDA dataset and the CityScapes dataset, and the semantic segmentation results are shown in Table III.

The results reported in Table III are the average values of three replicate experiments. As observed in Table III, HAS surpasses the traditional shared method on both the LoveDA and the CityScapes datasets. The network built using our searched cell structure achieves an impressive performance of 54.5% and 73.9% mIoU on the LoveDA dataset and the CityScapes validation set respectively.

Fig. 4 shows one of the best cell architectures searched by the proposed HAS method. As can be seen, different layers tend to choose various operations, which shows the effectiveness of our hierarchical shared idea. Furthermore, pooling operations are favored in the feature aggregation stage. As always, dilated convolutions play an important role in semantic segmentation. In the face of that feature maps have various resolutions, dilated convolution tends to choose diverse dilation rates.

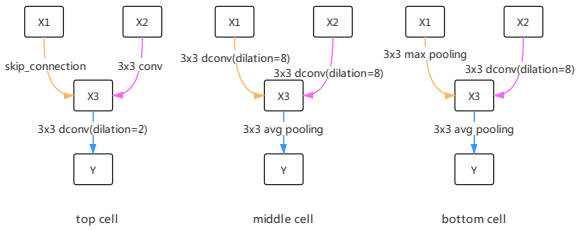

Fig. 4: One of the best cell architectures we have found through multiple searches. *conv*: separable convolution, *dconv*: dilated separable convolution.

TABLE IV: Experimental results on the LoveDA, Potsdam, and CityScapes validation set using different network backbones and the proposed ADMFF module. The mark + represents BiSeNet using ResNet18 as its backbone. DFANetA and DFANetB are two versions of DFANet with different numbers of channels.

Dataset	Method	w/o ADMFF	w ADMFF
LoveDA	BiSeNet[9]	47.0	**49.7**
	BiseNetv2 [16]	49.2	**50.9**
Potsdam	BiSeNet[9]	81.7	**83.9**
	BiseNetv2 [16]	82.3	**83.8**
CityScapes	BiSeNet[9]	69.0	**71.3**
	BiSeNet+[9]	74.8	**75.9**
	DFANetA[15]	71.3	**71.9**
	DFANetB[15]	67.1	**68.3**

Furthermore, to validate the effectiveness of the proposed feature aggregation module, namely the ADMFF module, we adopt the ADMFF module to replace the relevant part of some existing models. Specifically, the backbones of existing models and the proposed ADMFF module are adopted to form new networks. For the LoveDA and Potsdam datasets, BiseNet [9] and BiseNetv2 [16] are employed. For CityScapes dataset, BiseNet [9] and DFANet [15] are adopted. The results are shown in Table IV.

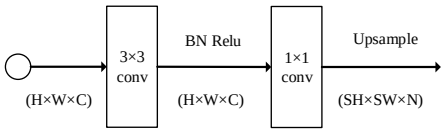

Fig. 5: The design of the segmentation head. H × W × C means the size of the feature maps, S represents the scale ratio of upsampling, and N denotes the final output channel number.

As shown in Table IV, the networks incorporating the backbone of existing methods and the proposed ADMFF module outperform the original network. It can prove the effectiveness of the ADMFF module.

TABLE V: Comparison results with other lightweight networks on the LoveDA dataset. Network-L is searched on the LoveDA dataset. The speed is measured using a single GeForce RTX 3090 GPU card with a 1024×1024 input.

Method	Background	Building	Road	Water.	Barren	Forest	Agriculture	mIoU	Speed
PSPNet [54]	44.4	52.1	53.5	76.5	9.7	44.1	57.9	48.3	52.2
DeepLabV3+ [55]	43.0	50.9	52.0	74.4	10.4	44.2	58.5	47.6	53.7
SemanticFPN [56]	42.9	51.5	53.4	74.7	11.2	44.6	58.7	48.2	52.7
FarSeg [57]	43.1	51.5	53.9	76.6	9.8	43.3	58.9	48.2	47.8
FactSeg [58]	42.6	53.6	52.8	76.9	16.2	42.9	57.5	48.9	46.7
BANet [59]	43.7	51.5	51.1	76.9	16.6	44.9	62.5	49.6	11.5
TransUNet [60]	43.0	56.1	53.7	78.0	9.3	44.9	56.9	48.9	13.4
Segmenter [61]	38.0	50.7	48.7	77.4	13.3	43.5	58.2	47.1	14.7
SwinUperNet [62]	43.3	54.3	54.3	78.7	14.9	45.3	59.6	50.0	19.5
UNetFormer [19]	44.7	58.8	54.9	79.6	20.1	46.0	62.5	52.4	115.3
PIDNet [63]	47.3	57.9	55.6	80.2	19.8	47.1	64.2	53.6	49.8
FastICENet [64]	44.5	59.1	54.2	80.5	18.2	48.0	61.7	52.3	92.2
Ours(Network-L)	**49.8**	**60.7**	**56.4**	**80.8**	**22.1**	**48.3**	**65.0**	**54.5**	**132.7**

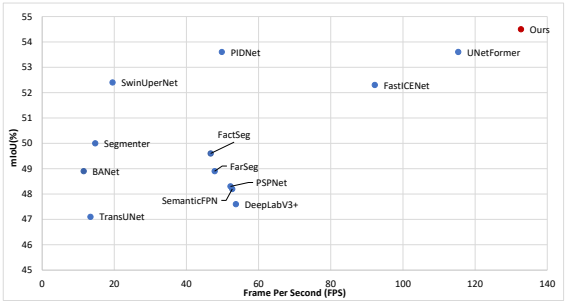

Fig. 6: The trade-off between accuracy and inference speed on the LoveDA dataset. The inference speed is estimated using PyTorch on a single RTX 3090.

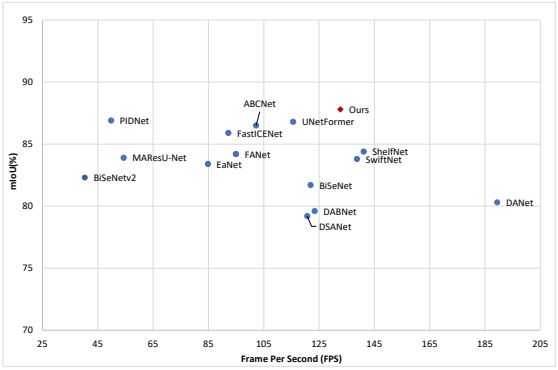

Fig. 7: The trade-off between accuracy and inference speed on the Potsdam dataset. The inference speed is estimated using PyTorch on a single RTX 3090.

D. Semantic Segmentation Results

To illustrate the final performance of the network we searched, we retrain and test the searched network on the LoveDA, Potsdam, and Vaihingen datasets. In the retraining stage, the training set is no longer split, and all images are adopted to optimize the network parameters until the model converges. For data augmentation, input images are randomly flipped horizontally, scaled, and cropped to a fixed size. Besides, to raise the training precision, we add the segmentation head illustrated as Fig. 5 to both the 3rd and 5th layers of the network backbone. The loss is computed by combining the outputs of these two segmentation heads with the network's final output. To further verify the effectiveness of the proposed method, we also conduct experiments on two street view image datasets: CityScapes and CamVid.

1) LoveDA: Experimental results are conducted on the LoveDA dataset. During training, the input images are cropped into 512×512 patches. The training process consists of 100 epochs. The speed is measured with a 1024×1024 input using the PyTorch programming framework on the same platform, which includes a single GeForce RTX 3090 GPU card, PyTorch 1.12, CUDA 11.3, cuDNN 8.3.2 and Linux Conda environment. No acceleration tools are used during the speed testing of the models. The comparison results with other lightweight models are shown in Table V and Fig. 6.

The proposed method obtains 132.7 FPS and 54.5% mIoU, achieving the state-of-the-art trade-off between performance and speed for real-time semantic segmentation of remote sensing images. Among previous works, PIDNet [63] is a promising model, and our method surpasses it by 0.9% mIoU.

2) Potsdam: Experimental results are conducted on the Potsdam dataset. The settings are the same as the LoveDA dataset during the training process. The comparison results with other lightweight models are shown in Table VI and Fig.7.

The proposed method obtains 132.7 FPS and 87.8% mIoU, achieving the state-of-the-art trade-off between performance and speed for real-time semantic segmentation of remote sensing images. Among previous works, UNetFormer [19] is a promising model, and our method surpasses it by 1% mIoU. Compared with DANet [33], ShelfNet [65], and SwiftNet [14], although our FPS is slightly lower, the proposed method has a greater advantage in accuracy.

3) Vaihingen: Experimental results are also conducted on the Vaihingen dataset, which is also widely used in remote sensing image segmentation tasks. The other details are the same as the LoveDA dataset. The comparison results with other lightweight networks are shown in Table VII and Fig. 8.

Our proposed method achieves the best trade-off between accuracy and speed. Compared with UNetFormer [19], our method has great advantages in both speed and accuracy. Compared with PSPNet [54], DANet [33], ShelfNet [65], and SwiftNet [14], our method has a slightly slower speed, but we have a great advantage in accuracy.

Extensive experiments demonstrate that the proposed method has a better trade-off between speed and accuracy for real-time semantic segmentation of remote sensing images. We

TABLE VI: Comparison results with other lightweight networks on the Potsdam dataset. The speed is measured using a single GeForce RTX 3090 GPU card with a 1024×1024 input.

Method	Imp. surf.	Building	Low veg.	Tree	Car	mIoU (%)	FPS
DABNet [7]	89.9	93.2	83.6	82.3	92.6	79.6	123.4
BiSeNet [9]	90.2	94.6	85.5	86.2	92.7	81.7	121.9
BiSeNetv2 [16]	91.3	94.3	85.0	85.2	94.1	82.3	40.3
DANet [33]	91.0	95.6	86.1	87.6	84.3	80.3	189.4
FANet [66]	92.0	96.1	86.0	87.8	94.5	84.2	94.9
EaNet [67]	92.0	95.7	84.3	85.7	95.1	83.4	84.8
ShelfNet [65]	92.5	95.8	86.6	87.1	94.6	84.4	141.1
DSANet [18]	83.0	89.5	71.9	74.3	77.3	79.2	120.7
MAResU-Net [68]	91.4	95.6	85.8	86.6	93.3	83.9	54.3
SwiftNet [14]	91.8	95.9	85.7	86.8	94.5	83.8	138.7
ABCNet [17]	93.5	96.9	87.9	89.1	95.8	86.5	102.2
UNetFormer [19]	93.6	**97.2**	87.7	88.9	**96.5**	86.8	115.6
PIDNet [63]	92.6	96.9	89.0	89.2	96.3	86.9	49.8
FastICENet [64]	93.0	96.8	88.2	88.9	96.0	85.9	92.2
Ours(Network-L)	**94.6**	96.2	**88.4**	**90.3**	95.8	**87.8**	132.7

TABLE VII: Comparison results with other lightweight networks on the Vaihingen dataset. The speed is measured using a single GeForce RTX 3090 GPU card with a 1024×1024 input.

Method	Imp. surf.	Building	Low veg.	Tree	Car	mIoU (%)	FPS
DABNet [7]	87.8	88.8	74.3	84.9	60.2	70.2	123.4
BiSeNet [9]	89.1	91.3	80.9	86.9	73.1	75.8	121.9
PSPNet [54]	89.0	93.2	81.5	87.7	43.9	68.6	145.3
DANet [33]	90.0	93.9	82.2	87.3	44.5	69.4	189.4
FANet [66]	90.7	93.8	82.6	88.6	71.6	75.6	94.9
EaNet [67]	91.7	94.5	83.1	89.2	80.0	78.7	84.8
ShelfNet [65]	91.8	94.6	83.8	89.3	77.9	78.3	141.1
MAResU-Net [68]	92.0	95.0	83.7	89.3	78.3	78.6	54.3
SwiftNet [14]	92.2	94.8	84.1	89.3	81.2	79.6	138.7
ABCNet [17]	92.7	95.2	84.5	89.7	85.3	81.3	102.2
DSANet [18]	79.5	86.0	63.9	73.6	58.4	72.3	120.7
UNetFormer [19]	92.7	95.3	84.9	90.6	88.5	82.7	115.6
PIDNet [63]	92.3	**95.5**	85.4	89.9	89.4	83.0	49.8
FastICENet [64]	92.5	94.8	85.1	88.0	87.6	81.5	92.2
Ours(Network-L)	**93.4**	94.5	**86.1**	**90.7**	89.6	**84.1**	132.7

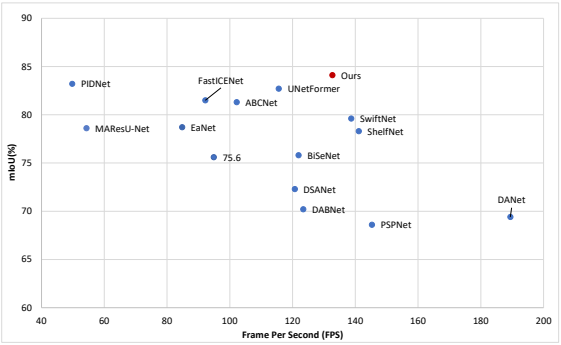

Fig. 8: The trade-off between accuracy and inference speed on the Vaihingen dataset. The inference speed is estimated using PyTorch on a single RTX 3090.

attribute this to three main reasons. Firstly, the feature fusion module, namely the ADMFF module obtained by HAS, is effective. Since we introduce NAS to automatically design the feature fusion module and the search process is conducted on a large-scale remote sensing dataset, HAS can find a superior architecture in the search space than the modules of artificial design, which is demonstrated in Table II - Table IV. Secondly, we make improvements on the basis of the traditional NAS methods, as shown in Table III, our hierarchical shared search method significantly improves the performance of the network. Thirdly, a lightweight backbone and a cell structure with only three operations are designed to improve the model's speed, which is shown in Table V - Table VII. So the proposed method achieves better performance than these baseline methods.

4) CityScapes: Experimental results are also conducted on two street view datasets, namely the CityScapes dataset and the CamVid dataset. In the CityScapes dataset, to ensure consistency with most comparison methods, the input images are resized to 512×1024 and 769×1537 pixels respectively. The SGD optimizer is applied to train the model with a momentum of 0.9. The learning rate is set the same as in the search stage. Besides, the model is trained for 150K iterations for the CityScapes dataset. We conduct our experiments using the PyTorch programming framework and test the speed of the models on the same platform, which includes a single GTX1080Ti GPU card, PyTorch 1.8, CUDA 10.1, cuDNN 7.6.3 and Linux Conda environment. No acceleration tools are used during the speed testing of the models.

The network searched by the proposed method is evaluated on the CityScapes validation and test set. As shown in Table VIII, for large-size models, one can see that using large-size models has significantly improved segmentation results. Nonetheless, these models exhibit high computational complexity, resulting in slower operation speeds, which are unsuitable for intelligent terminal hardware that demands high real-time capabilities. For medium-size models, compared with SwiftNet [14], although our model is lower than them in

TABLE VIII: Comparison results on the CityScapes validation and test set. "$\#FPS$" represents that the speed of segmentation models is remeasured on a single GTX 1080Ti without any acceleration tools. "+" denotes BiSeNet using ResNet18 as its backbone. "$*$" represents using both fine and coarse data for training.

	Architecture	Input size	Params(M)	mIoU(%)		GPU	FPS	$\#FPS$	Method
				val	test				
Large Size	PSPNet [54]	713×713	250.8	-	81.2	-	-	0.8	Manual
	SETR-PUP [69]	768×768	318.3	82.2	81.6	-	-	0.7	Manual
	BiSeNet+ [9]	768×1536	49.0	74.8	74.7	GTX 1080Ti	65.5	64.3	Manual
	DANet [33]	1024×1024	66.6	81.5	81.5	-	-	4.1	Manual
	CCNet [70]	1024×1024	66.5	81.3	81.4	-	-	4.5	Manual
	PIDNet [63]	1024×2048	36.9	80.9	80.6	RTX 3090	31.1	14.8	Manual
	Lawin Transformer [71]	1024×1024	-	-	84.4	Tesla V100	-	-	Manual
	SegFormer [72]	1024×2048	84.7	-	84.0	Tesla V100	2.5	-	Manual
Medium Size	DSANet[35]	512×1024	11.9	79.8	71.4	GTX 1080Ti	34.1	34.9	Manual
	FPANet-A [73]	512×1024	11.61	-	72.0	RTX 2080Ti	127	-	Manual
	BiSeNet [9]	768×1536	5.8	69.0	68.4	GTX 1080Ti	105.8	104.2	Manual
	SwiftNet [14]	1024×2048	11.8	75.4	75.5	GTX 1080Ti	39.9	39.5	Manual
	DFANet [15]	1024×1024	7.8	-	71.3	GTX TITAN X	100	94	Manual
	ICNet [13]	1024×2048	26.5	-	69.5	GTX TITAN X	30.3	28.7	Manual
Small Size	BiSeNetv2 [16]	512×1024	3.4	73.4	72.6	GTX 1080Ti	156	154.6	Manual
	DFFNet [74]	512×1024	1.9	-	71.0	GTX 1080Ti	62.5	63.0	Manual
	SGCPNet [11]	512×1024	0.61	-	69.5	GTX 1080Ti	178.5	177.7	Manual
	JPANet-G [75]	512×1024	3.49	-	71.62	GTX 1080Ti	109.9	111.0	Manual
	FBSNet[76]	512×1024	0.62	-	70.9	GTX 2080Ti	90	66.3	Manual
	ESPNet [6]	512×1024	0.36	-	60.3	GTX TITAN	112.9	-	Manual
	ENet [12]	512×1024	0.36	-	58.3	GTX TITAN X	76.9	-	Manual
	LBNet [77]	512×1024	0.76	-	69.6	-	70	-	Manual
	Fast-SCNN [8]	1024×2048	1.11	68.6	68.0	GTX TITAN XP	123.5	-	Manual
	RCNet [78]	1024×2048	1.96	72.49	-	Tesla V100	59.2	-	Manual
	CAS [49]	768×1536	2.2	71.6	70.5	GTX TITAN XP	108.0	-	NAS
	CAS* [49]	768×1536	2.2	72.5	71.3	GTX TITAN XP	108.0	-	NAS
	GAS [27]	769×1537	2.18	-	71.8	GTX TITAN XP	108.4	-	NAS
	More FasterSeg[50]	1024×2048	3.09	71.5	69.2	GTX TITAN XP	164.3	-	NAS
	Ours(Network-C)	512×1024	4.07	73.6	72.6	GTX 1080Ti	164.0	164.0	NAS
	Ours(Network-C)	769×1537	4.07	73.9	72.8	GTX 1080Ti	116.0	116.0	NAS

accuracy, it has a great advantage in speed. For small-size models, our method reports 164.0 FPS and 72.6% mIoU on the test set using only fine data and without any evaluation tricks, which achieves the state-of-the-art trade-off between the performance and the speed.

Compared with other NAS methods such as CAS [49], GAS [27], and More FasterSeg [50], our model achieves a significant improvement in both accuracy and speed. It is worth noting that More FasterSeg [50] uses TensorRT for acceleration, while our method does not employ any acceleration tools. To provide a more objective comparison with other methods, we also present the trade-off between speed and accuracy in Fig. 9. It can be observed that our method achieves a superior balance between speed and accuracy.

5) CamVid: The network searched on CityScapes is also transferred to CamVid directly to demonstrate the transferability of the searched model. The input size is 720 $\times$ 960 pixels and other settings are the same as above. The other details are the same as the Cityscapes dataset. The comparison results with other methods are shown in Table IX and Fig. 10.

Our model achieves 72.3% mIoU with 186.4 FPS, which demonstrates the superior performance of our model. Compared with SegNet [29], ENet [12], DFANet [15], ICNet [13], BiSeNet [9], DFFNet [74], RGPNet[79], FBSNet[76], CAS [49], and GAS [27], the proposed model achieves great advantages according to both speed and accuracy. Compared with BiSeNetV2 [16], the accuracy of the proposed model drops slightly, only 0.1%, but the proposed model achieves a higher speed. Although our method is slower than SGCPNet [11] and JPANet-G [75], the proposed method has a huge advantage in accuracy. Overall, our method achieves a better trade-off between speed and accuracy.

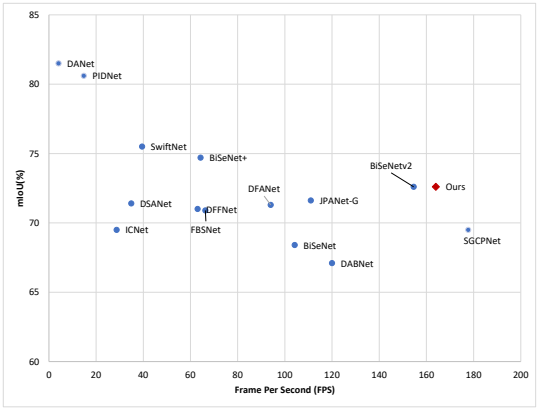

Fig. 9: The trade-off between accuracy and inference speed on the CityScapes dataset. The inference speed is measured using PyTorch on a single GTX 1080Ti.

E. Visualization

The visualization of the segmentation results of the proposed model on the Potsdam dataset and CityScapes validation set is shown in Fig. 11 and Fig. 12.

As shown in Fig. 11, the segmentation of our method is better than UNetFormer [19], which is the most promising

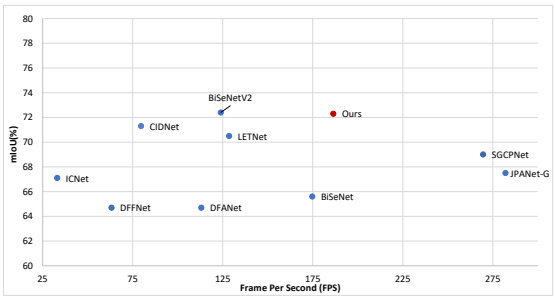

Fig. 10: The trade-off between accuracy and inference speed on the CamVid dataset. The inference speed is estimated using PyTorch on a single GTX 1080Ti.

TABLE IX: Results on the CamVid test set with a resolution 720 × 960. "$\#FPS$" represents that the speed of segmentation models is remeasured on a single GTX 1080Ti without any acceleration tools.

Architecture	mIoU(%)	GPU	FPS	$\#FPS$	Method
SegNet [29]	55.6	GTX TITAN	29.4	-	Manual
ENet [12]	51.3	GTX TITAN X	76.9	-	Manual
DFANet [15]	64.7	GTX TITAN X	120.0	113.1	Manual
ICNet [13]	67.1	GTX TITAN X	34.5	33.0	Manual
RGPNet[79]	69.2	NVIDIA TITAN V	68.2	-	Manual
BiSeNet [9]	65.6	GTX 1080Ti	175.0	174.7	Manual
BiSeNetV2 [16]	72.4	GTX 1080Ti	124.5	123.9	Manual
DFFNet [74]	64.7	GTX 1080Ti	62.5	63.2	Manual
SGCPNet [11]	69.0	GTX 1080Ti	278.4	269.6	Manual
JPANet-G [75]	67.5	GTX 1080Ti	294.0	282.0	Manual
CIDNet [80]	71.3	GTX 1080Ti	79.0	79.6	Manual
FPANet-A [73]	68.6	RTX 2080Ti	101.0	-	Manual
RELAXNet [81]	71.2	RTX 2080Ti	79.0	-	Manual
LETNet [82]	70.5	RTX 3090	200.0	128.5	Manual
CAS [49]	71.2	GTX TITAN XP	16.09	-	NAS
GAS [27]	71.9	GTX TITAN XP	153.1	-	NAS
Ours (Network-C)	**72.3**	GTX 1080Ti	186.4	186.4	NAS

network in previous work. For example, in the red box of the first row, trees are misclassified as background by UNetFormer, and in the red box of the second row, roads are misclassified as background by UNetFormer. As shown in Fig. 12, the segmentation effect of our model is better than BiseNetV2 [16] and DFANet [15] in detail, especially in the edge parts of roads, persons, and trees. In the second column, we can see that our model successfully distinguishes cars and trucks. It also has a relatively precise segmentation effect for objects with a small number of training samples, such as the fence in the third column.

V. Conclusion

In this paper, we concentrate on adopting neural network architecture search (NAS) to find an optimal real-time semantic segmentation model. Most NAS methods generally search and share one or two cells in the entire network. We think it is more reasonable for the feature maps with different resolutions to be processed by the network blocks with diverse structures.

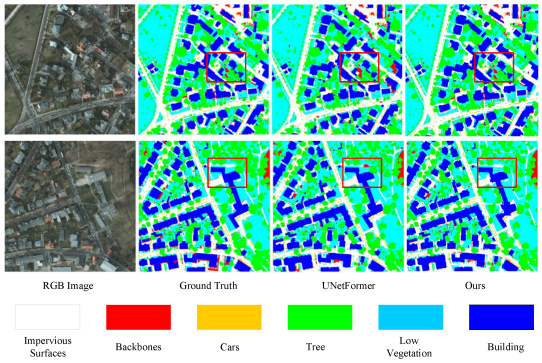

Fig. 11: Results of the proposed model on Potsdam dataset (Image ID 3_14 and Image ID 2_13).

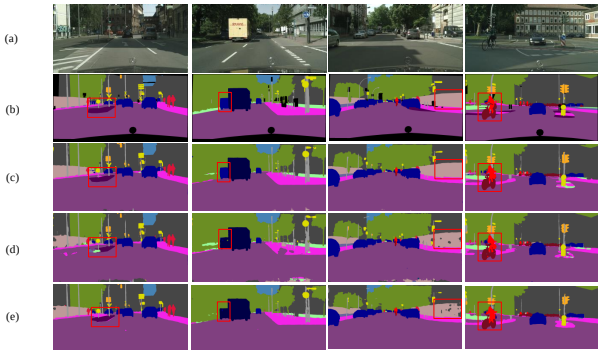

Fig. 12: Results of the proposed model on CityScapes validation set. (a) is input images, and (b) is ground truth. (c), (d), (e) display the outputs of our HAS model, BiseNetV2, and DFANet.

Therefore, a hierarchical shared search strategy is proposed, in which only the network blocks on the same layer share the cell with the same structure. Existing semantic segmentation methods have achieved great progress in feature extraction, and recently most of them have focused on effectively fusing the features from various layers to increase accuracy. Because of these, a real-time semantic segmentation network is proposed, consisting of a lightweight backbone for multilevel feature extraction and a module for feature aggregation. The proposed hierarchical shared search strategy is adopted to find an optimal feature aggregation module. The lightweight backbone is carefully designed using as little computing cost as possible and takes advantage of the latest excellent semantic segmentation network. With comparative experiments, our method shows better performance than traditional searching methods. Finally, experimental results on five datasets show that the searched optimal model obtains the state-of-the-art trade-off between accuracy and speed.

Limited by the memory of the GPU, our search space only contains the structure of separate cells. The connection mode between cells is artificially fixed. In the future, we plan to expand our search space on the basis of this article. We will no longer only search the operation types inside cells but consider bringing the connection mode between cells into the search scope to search for a more flexible network structure with better performance. In addition, designing a more general

network-level search space is also a problem worth exploring.

REFERENCES

[1] Y. Chen, G. Zhang, H. Cui, X. Li, S. Hou, J. Ma, Z. Li, H. Li, and H. Wang, "A novel weakly supervised semantic segmentation framework to improve the resolution of land cover product," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 196, pp. 73–92, 2023.

[2] W. Li, P. Ma, H. Wang, and C. Fang, "Sar-tscc: A novel approach for long time series sar image change detection and pattern analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–16, 2023.

[3] R. Rahman, C. Indris, T. Zhang, K. Li, B. McCornack, D. Flippo, A. Sharda, and G. Wang, "On the real-time semantic segmentation of aphid clusters in the wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 6298–6305.

[4] W. Bo, J. Liu, X. Fan, T. Tjahjadi, Q. Ye, and L. Fu, "Basnet: Burned area segmentation network for real-time detection of damage maps in remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–13, 2022.

[5] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2017.

[6] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 552–568.

[7] G. Li, S. Jiang, I. Yun, J. Kim, and J. Kim, "Depth-wise asymmetric bottleneck with point-wise aggregation decoder for real-time semantic segmentation in urban scenes," *IEEE Access*, vol. 8, pp. 27 495–27 506, 2020.

[8] R. Poudel, S. Liwicki, and R. Cipolla, "Fast-scnn: Fast semantic segmentation network," in *30th British Machine Vision Conference 2019, BMVC*, 2019.

[9] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 325–341.

[10] Q. Sun, Z. Zhang, and P. Li, "Second-order encoding networks for semantic segmentation," *Neurocomputing*, vol. 445, pp. 50–60, 2021.

[11] S. Hao, Y. Zhou, Y. Guo, R. Hong, J. Cheng, and M. Wang, "Real-time semantic segmentation via spatial-detail guided context propagation," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[12] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[13] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 405–420.

[14] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic, "In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12 607–12 616.

[15] H. Li, P. Xiong, H. Fan, and J. Sun, "Dfanet: Deep feature aggregation for real-time semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9522–9531.

[16] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation," *arXiv preprint arXiv:2004.02147*, 2020.

[17] R. Li, S. Zheng, C. Zhang, C. Duan, L. Wang, and P. M. Atkinson, "Abcnet: Attentive bilateral contextual network for efficient semantic segmentation of fine-resolution remotely sensed imagery," *ISPRS journal of photogrammetry and remote sensing*, vol. 181, pp. 84–98, 2021.

[18] W. Shi, Q. Meng, L. Zhang, M. Zhao, C. Su, and T. Jancsó, "Dsanet: A deep supervision-based simple attention network for efficient semantic segmentation in remote sensing imagery," *Remote Sensing*, vol. 14, no. 21, p. 5399, 2022.

[19] L. Wang, R. Li, C. Zhang, S. Fang, C. Duan, X. Meng, and P. M. Atkinson, "Unetformer: A unet-like transformer for efficient semantic segmentation of remote sensing urban scene imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 190, pp. 196–214, 2022.

[20] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019.

[21] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[22] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2019, pp. 4780–4789.

[23] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[24] W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li, and Z. Wang, "Fasterseg: Searching for faster real-time semantic segmentation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. [Online]. Available: https://openreview.net/forum?id=BJgqQ6NYvB

[25] A. Shaw, D. Hunter, F. Landola, and S. Sidhu, "Squeezenas: Fast neural architecture search for faster semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[26] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 82–92.

[27] P. Lin, P. Sun, G. Cheng, S. Xie, X. Li, and J. Shi, "Graph-guided architecture search for real-time semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4203–4212.

[28] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[29] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[30] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[31] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, no. 4, pp. 834–848, 2017.

[32] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[33] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3146–3154.

[34] C. Li, X. Kang, L. Zhu, L. Ye, P. Feng, and A. Ming, "Hdnet: Hybrid distance network for semantic segmentation," *Neurocomputing*, vol. 447, pp. 129–144, 2021.

[35] M. A. Elhassan, C. Huang, C. Yang, and T. L. Munea, "Dsanet: Dilated spatial attention for real-time semantic segmentation in urban street scenes," *Expert Systems with Applications*, vol. 183, p. 115090, 2021.

[36] M.-H. Guo, C.-Z. Lu, Q. Hou, Z. Liu, M.-M. Cheng, and S.-M. Hu, "Segnext: Rethinking convolutional attention design for semantic segmentation," *arXiv preprint arXiv:2209.08575*, 2022.

[37] Y. Liu, Q. Ren, J. Geng, M. Ding, and J. Li, "Efficient patch-wise semantic segmentation for large-scale remote sensing images," *Sensors*, vol. 18, no. 10, p. 3232, 2018.

[38] H. Pan, Y. Hong, W. Sun, and Y. Jia, "Deep dual-resolution networks for real-time and accurate semantic segmentation of traffic scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 3, pp. 3448–3460, 2022.

[39] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[40] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8697–8710.

[41] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2017, pp. 2902–2911.

[42] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," in *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[43] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 4095–4104.

[44] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1294–1303.

[45] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "Pc-darts: Partial channel connections for memory-efficient architecture search," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[46] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[47] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10 734–10 742.

[48] L.-C. Chen, M. D. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens, "Searching for efficient multi-scale architectures for dense image prediction," in *Advances in Neural Information Processing Systems (NIPS)*, 2018, pp. 8713–8724.

[49] Y. Zhang, Z. Qiu, J. Liu, T. Yao, D. Liu, and T. Mei, "Customizable architecture search for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 11 641–11 650.

[50] J. Wu, H. Kuang, Q. Lu, Z. Lin, Q. Shi, X. Liu, and X. Zhu, "M-fasterseg: An efficient semantic segmentation network based on neural architecture search," *Engineering Applications of Artificial Intelligence*, vol. 113, p. 104962, 2022.

[51] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[52] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213–3223.

[53] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2008, pp. 44–57.

[54] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[55] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 801–818.

[56] A. Kirillov, R. Girshick, K. He, and P. Dollár, "Panoptic feature pyramid networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 6399–6408.

[57] Z. Zheng, Y. Zhong, J. Wang, and A. Ma, "Foreground-aware relation network for geospatial object segmentation in high spatial resolution remote sensing imagery," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4096–4105.

[58] A. Ma, J. Wang, Y. Zhong, and Z. Zheng, "Factseg: Foreground activation-driven small object semantic segmentation in large-scale remote sensing imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–16, 2021.

[59] L. Wang, R. Li, D. Wang, C. Duan, T. Wang, and X. Meng, "Transformer meets convolution: A bilateral awareness network for semantic segmentation of very fine resolution urban scene images," *Remote Sensing*, vol. 13, no. 16, p. 3065, 2021.

[60] J. Chen, Y. Lu, Q. Yu, X. Luo, E. Adeli, Y. Wang, L. Lu, A. L. Yuille, and Y. Zhou, "Transunet: Transformers make strong encoders for medical image segmentation," *arXiv preprint arXiv:2102.04306*, 2021.

[61] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for semantic segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 7262–7272.

[62] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[63] J. Xu, Z. Xiong, and S. P. Bhattacharyya, "Pidnet: A real-time semantic segmentation network inspired by pid controllers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 19 529–19 539.

[64] X. Zhang, Z. Zhao, L. Ran, Y. Xing, W. Wang, Z. Lan, H. Yin, H. He, Q. Liu, B. Zhang *et al.*, "Fasticenet: A real-time and accurate semantic segmentation model for aerial remote sensing river ice image," *Signal Processing*, p. 109150, 2023.

[65] J. Zhuang, J. Yang, L. Gu, and N. Dvornek, "Shelfnet for fast semantic segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision workshops*, 2019, pp. 0–0.

[66] P. Hu, F. Perazzi, F. C. Heilbron, O. Wang, Z. Lin, K. Saenko, and S. Sclaroff, "Real-time semantic segmentation with fast attention," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 263–270, 2020.

[67] X. Zheng, L. Huan, G.-S. Xia, and J. Gong, "Parsing very high resolution urban scene images by learning deep convnets with edge-aware loss," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 170, pp. 15–28, 2020.

[68] R. Li, S. Zheng, C. Duan, J. Su, and C. Zhang, "Multistage attention resu-net for semantic segmentation of fine-resolution remote sensing images," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2021.

[69] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr *et al.*, "Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 6881–6890.

[70] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "Ccnet: Criss-cross attention for semantic segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 603–612.

[71] H. Yan, C. Zhang, and M. Wu, "Lawin transformer: Improving semantic segmentation transformer with multi-scale representations via large window attention," *arXiv preprint arXiv:2201.01615*, 2022.

[72] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 077–12 090, 2021.

[73] Y. Wu, J. Jiang, Z. Huang, and Y. Tian, "Fpanet: Feature pyramid aggregation network for real-time semantic segmentation," *Applied Intelligence*, pp. 1–18, 2022.

[74] X. Tang, W. Tu, K. Li, and J. Cheng, "Dffnet: An iot-perceptive dual feature fusion network for general real-time semantic segmentation," *Information Sciences*, vol. 565, pp. 326–343, 2021.

[75] X. Hu, L. Jing, and U. Sehar, "Joint pyramid attention network for real-time semantic segmentation of urban scenes," *Applied Intelligence*, vol. 52, no. 1, pp. 580–594, 2022.

[76] G. Gao, G. Xu, J. Li, Y. Yu, H. Lu, and J. Yang, "Fbsnet: A fast bilateral symmetrical network for real-time semantic segmentation," *IEEE Transactions on Multimedia (TMM)*, 2022.

[77] P. Wang, L. Li, F. Pan, and L. Wang, "Lightweight bilateral network for real-time semantic segmentation," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 27, no. 4, pp. 673–682, 2023.

[78] Y. Dong, K. Zhao, L. Zheng, H. Yang, Q. Liu, and Y. Pei, "Refinement co-supervision network for real-time semantic segmentation," *IET Computer Vision*, 2023.

[79] E. Arani, S. Marzban, A. Pata, and B. Zonooz, "Rgpnet: A real-time general purpose semantic segmentation," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, pp. 3009–3018.

[80] Y. Dong, H. Yang, Y. Pei, L. Shen, L. Zheng, and P. Li, "Compact interactive dual-branch network for real-time semantic segmentation," *Complex & Intelligent Systems*, pp. 1–14, 2023.

[81] J. Liu, X. Xu, Y. Shi, C. Deng, and M. Shi, "Relaxnet: Residual efficient learning and attention expected fusion network for real-time semantic segmentation," *Neurocomputing*, vol. 474, pp. 115–127, 2022.

[82] G. Xu, J. Li, G. Gao, H. Lu, J. Yang, and D. Yue, "Lightweight real-time semantic segmentation network with efficient transformer and cnn," *IEEE Transactions on Intelligent Transportation Systems*, 2023.