NAS-based Recursive Stage Partial Network (RSPNet) for Light-Weight Semantic Segmentation

Anonymous Authors

Abstract

Current NAS-based semantic segmentation methods focus on accuracy improvements rather than light weight design. In this paper, we propose a two-stage framework to design our NAS-based RSPNet model for light-weight semantic segmentation. The first architecture search determines the inner cell structure, and the second architecture search considers exponentially growing paths to finalize the outer structure of the network. It was shown in the literature that the fusion of high- and low-resolution feature maps produces stronger representations. To find the expected macro structure without manual design, we adopt a new path-attention mechanism to efficiently search for suitable paths to fuse useful information for better segmentation. Our search for repeatable micro-structures from cells leads to a superior network architecture in semantic segmentation. The proposed architecture is very efficient, simple, and effective that both the macro- and micro- structure searches can be completed in five days of computation on two V100 GPUs. The light-weight NAS architecture with only 1/4 parameter size of SoTA architectures can achieve SoTA performance on the Cityscapes dataset without using any backbones.

Introduction

Network Architecture Search (NAS) (Elsken, Metzen, and Hutter 2019b) is a computational approach for automating the optimization of the neural architecture design. As deep learning has been widely used for medical image segmentation, the most common deep networks used in practice are still designed manually. In this work, we focus on applying NAS for medical image segmentation as the targeted application. To optimize the NAS that looks for the best architecture for image segmentation, the search task can be decomposed into three parts: (i) a supernet to generate all possible architecture candidates, (ii) a global search of neural architecture paths from the supernet, and (iii) a local search of the cell architectures, namely operations including the conv/deconv kernels and the pooling parameters. The NAS space to explore is exponentially large w.r.t. the number of generated candidates, the paths between nodes, the number of depths, and the available cell operations to choose from. The computational burden of NAS for image segmentation is much higher than other tasks such as image classification, so each architecture verification step takes longer to complete. As a result, there exist fewer NAS methods that work successfully for image segmentation. In addition, none is designed for light-weight semantic segmentation, which is very important for AV(Automobile Vehicle)-related applications.

The main challenge of NAS is on how to deal with the exponentially large search space when exploring and evaluating neural architectures. We tackle this problem based on a formulation regarding what needs to be considered in priority and how to effectively reduce search complexity. Most segmentation network designs (Ronneberger, Fischer, and Brox 2015; Fourure et al. 2017; Weng et al. 2019; Liu et al. 2019) use U-Nets to achieve better accuracies in image segmentation. For example, AutoDeepLab (Liu et al. 2019) designs a level-based U-Net as the supernet whose search space grows exponentially according to its level number \( L \) and depth parameter \( D \). Joining the search for network-level and cell-level architectures creates huge challenges and inefficiency in determining the best architecture. To avoid exponential growth in the cell search space, only one path in AutoDeepLab is selected and sent to the next node. This limit is unreasonable since more input to the next node can generate richer features for image segmentation.

A “repeatable” concept is adopted in this paper to construct our model. Similar to repeatable cell architecture design, our model contains repeated units that share the same structure. The proposed model architecture for image segmentation is shown in Fig. 1. It is based on differential learning and is as efficient as the DARTS (Differentiable Architecture Searching) method (Liu, Simonyan, and Yang 2019), compared to the other NAS methods based on RL and EV. In addition, we modify the concept of CSPNet (Wang et al. 2020) to recursively use only half of the channels to pass through the cell we searched for. The RSPNet (Recursive Stage Partial Network) makes our search procedure much more efficient and results in a light weight architecture for semantic segmentation. The proposed architecture is simple, efficient, and effective in image segmentation. Both the macro- and micro- structure searches can be completed.
in two days of computation on two V100 GPUs. The lightweight NAS network with only 1/4 parameter size of SoTA architectures achieves state-of-the-art performance on the Cityscapes datasets without any backbones. Main contributions of this paper are summarized in the following:

- We propose a two-stage search method to decrease memory usage and speed up search time.
- We designed a cell-based architecture that can construct a complex model by stacking the cell we searched for.
- RSPNet makes our search procedure much more efficient and results in our light-weight architecture for semantic segmentation.
- The proposed PAM selects the paths and fuses more inputs than Auto-Deeplab (Liu et al. 2019) and Hi-NAS (Cho et al. 2020) to generate richer features for better image segmentation.
- Without using any backbone, our architecture outperforms SoTA methods with improved accuracy and efficiency on the Cityscapes (Cordts et al. 2016) dataset.

**Related Works**

Mainstream NAS algorithms generally consist of three basic steps: supernet generation, architecture search, and network cell parameter optimization. Approaches for architecture search can be organized into three categories (Elsken, Metzen, and Hutter 2019b): reinforcement learning (RL) based, evolution (EV) based, and gradient based. In what follows, the details of each method are discussed.

**RL-based methods** (Baker et al. 2017, Zoph and Le 2017) use a controller to sample neural network architectures (NNAs) to learn a reward function to generate better architectures from exploration and exploitation. Although an RL-based NAS approach can construct a stable architecture for evolution, it needs a huge number of tries to get a positive reward for updating architectures and thus is very computationally expensive. For example, in (Zhong et al. 2018), a cell-based Q-learning approach is evaluated on ImageNet, and the search takes up to 9 GPU-days to run.

**Evolution-based methods** (Real et al. 2017, Real et al. 2019, Liu et al. 2018, Elsken, Metzen, and Hutter 2019a) perform evolution operators (e.g., crossover and mutation) based on the genetic algorithm (GA) to continuously adjust NNAs and improve their qualities across generations. These methods suffer from high computational cost in optimizing the model generator when validating the accuracy of each candidate architecture. Compared to the methods, which rely on optimization over discrete search spaces, gradient-based methods can optimize much faster via search in continuous spaces.

**Gradient-based methods**: The Differentiable ARchiTecture Search (DARTS) (Liu, Simonyan, and Yang 2019) significantly improves search efficiency by computing a convex combination of a set of operations where the best architecture can be optimized by gradient descent algorithms. In DARTS, a supernet is constructed by placing a mixture of candidate operations on each edge rather than applying a single operation to a node. An attention mechanism on the connections is adopted to remove weak connections, such that all supernet weights can be efficiently optimized jointly with a continuous relaxation of the search space via gradient descent. The best architecture is found efficiently by restricting the search space to the subgraphs of the supernet. However, this acceleration comes with a penalty when limit-
Figure 2: The Cell architecture we propose.

Figure 3: Network forward and backward procedures, where \( X_i \) denotes the output of the \( i \)th layer with the weight matrix \( W_{i+1} \) and the output function \( f_{i+1} \).

Figure 4: Comparisons between ResNet and CSPNet. (a) ResNet architecture (b) CSP architecture.
For the \((L-i)\text{th}\) layer, based on the form of Eq.\((4)\), the gradient of \(W_{L-i}\) to \(C\) can be recursively updated by a backpropagation method, \(i.e.,\)
\[
\frac{\partial C}{\partial W_{L-i}} = \frac{\partial C}{\partial X_L} \left( \prod_{k=1}^{i} \frac{\partial f_{L-k+1}}{\partial X_{L-k}} \right) \frac{\partial f_{L-i}}{\partial W_{L-i}}.
\]
(5)

When the activation function used in \(f_i\) is a Sigmoid function, the recursive term in Eq.\((5)\) will close to zero and result in the problem of vanishing gradient; that is,
\[
\prod_{k=1}^{i} \frac{\partial f_{L-k+1}}{\partial X_{L-k}} \rightarrow 0.
\]
(6)

To solve this problem in Eq.\((6)\), one solution is to change the activation function to the ReLU function. Another solution proposed in ResNet \([He\ et\ al.\ 2016]\) explicitly preserves information through additive identity transformation, \(i.e.,\) skip connection. With the skip connection, Eq.\((1)\) can be rewritten as
\[
X_{i+1} = f_{i+1}(X_i, W_{i+1}) + X_i.
\]
(7)

Fig.\((4)a\) shows the architecture of ResNet \([He\ et\ al.\ 2016]\). Based on Eq.\((7)\), Eq.\((5)\) can be rewritten as
\[
\frac{\partial C}{\partial W_{L-i}} = \frac{\partial C}{\partial X_L} \left( \prod_{k=1}^{i} \frac{\partial f_{L-k+1}}{\partial X_{L-k}} + 1 \right) \frac{\partial f_{L-i}}{\partial W_{L-i}}.
\]
(8)

Based on Eq.\((8)\), the recursive term in Eq.\((5)\) will not converge to zero, and thus the problem of vanishing gradient can be solved. Unlike ResNet \([He\ et\ al.\ 2016]\), CSPNet \([Wang\ et\ al.\ 2020]\) adopts a split-transform-merge strategy to separate the feature map of the base layer into two parts; one part will go through a convolution block and a transition layer; the other part is then combined with the transmitted feature map to the next stage. Then, Eq.\((7)\) can be modified as
\[
X_{i+1} = f_{i+1} \left( \frac{1}{2} X_i, W_{i+1} \right) + \frac{1}{2} X_i.
\]
(9)

It is another way to make the recursive term in Eq.\((5)\) not converge to zero, and thus the problem of vanishing gradient is solved. Eq.\((9)\) inspires the design of our RSPNet. The architecture of this CSPNet is shown in Fig.\((4)b\). Due to the split-transform-merge strategy, rich features can be extracted and thus enhance both the efficiency and accuracy of object detection. However, CSPNet separates the feature map into two parts only on the base layer. We can apply the split-transform-merge strategy recursively not only on the base layer but also to other layers. The recursively mixed two-path way can preserve feature reusing characteristics, but at the same time prevents an excessive amount of duplicate gradient information by truncating the gradient flow. This is named as RSPNet (Recursive Stage Partial Network).

**Methods**

A new concept is proposed in this paper to use two-stage method from which the best architecture can be searched via NAS. Fig.\((1)\) shows the architecture created in our method.

At first, the cell-structure will be first searched with lightweight structure by NAS \((see\ Bottom\ Right)\). Then, we fix the cell-structure and repeat it four times to construct our architecture. In what follows, the details of our method are described.

**Cell Architecture Search**

For cell structure search, only one type of cell is used: the Normal Cell (Norm-Cell) as in Fig.\((2)\). We use two intermediate nodes to construct each cell structure. Each cell keeps the number of channels and neither reduces the spatial dimension from \(W \times H\) to \(\frac{W}{2} \times \frac{H}{2}\) nor enlarges the spatial dimension from \(W \times H\) to \(2W \times 2H\).

Referring to Fig.\((1)\) \((bottom\ left)\) and Fig.\((2)\) for each Norm-Cell with one input, feature maps derived by previous layer with different shapes are reshaped by convolution. The input maps are then fed into subsequent operations after adding together, which are searched by DARTS to generate a feature map with the same size \(C \times H \times W\).

The set of operation types consists of the following 5 operators, all prevalent in modern CNNs:

- \(3 \times 3\) Convolution
- \(5 \times 5\) Convolution
- \(3 \times 3\) Dilated Convolution
- Identity

We list all primitive operations used for cell search that appear in Fig.\((2)\). For all edges between two intermediate nodes, their weights are calculated by applying softmax to obtain real-valued architecture parameters.

**Two Stage Search**

The main challenge of NAS is on how to deal with the exponentially large search space when exploring and evaluating neural architectures. We use a two-stage method to search and train our model. In the first stage, we adopt the lightweight architecture by using fewer cells to search. In Fig.\((1)\) \((bottom\ left)\), we only use one cell in each layer. After the search was completed, we stack the cells searched using DARTS \([Liu,\ Simonyan,\ and\ Yang\ 2019]\) four times to construct our model. Our experiment result shows that the method can not only get better efficiency during the search process, but also outperform SoTA methods with improved accuracy.

**Path Attention Module**

To avoid the problem of combination exploration, we propose the Path Attention Module (PAM) to reduce the path search problem into a linear weight optimization. Our PAM considers the outputs from all previous layers and the current layer. It chooses the best two of them as inputs for the next cells to generate better segmentation results.

Fig.\((5)\) depicts the details of this PAM. Given a node, the idea of PAM aims at calculating the attention value of each path and choosing the number of outputs you want (denoted by red color in Fig.\((5)a))\) with higher attention to construct
Figure 5: Details of the proposed Path-Attention NAS components. (a) The outputs of cells are concatenated as inputs to the Path Attention Module (PAM). Its output channels are divided into the specified number of outputs (red color) and pass through the subsequent cell. (b) Path-Attention Module is mainly to choose the number of feature maps we want, then it pass through the next cell.

Figure 6: Details of the proposed Channel Attention Module (CAM).

Figure 7: DARTS searched architecture on the Cityscapes dataset.

Experimental Results

Experiments are conducted in the Cityscapes (Cordts et al. 2016) urban scene understanding dataset for evaluation. Auto-DeepLab (Liu et al. 2019), U-Net (Ronneberger, Fischer, and Brox 2015) and NasUnet (Weng et al. 2019) are...
Table 1: Performance evaluations of the model on Cityscapes validation set.

| Methods                      | Backbone            | Coarse | ImageNet | mIOU(%) | Params(M) |
|------------------------------|---------------------|--------|----------|---------|-----------|
| PSPNet (Hengshuang et al. 2018) | Dilated-ResNet-101  | ×      | ✓        | 76.2    | ×         |
| PSANet (Hengshuang et al. 2018) | Dilated-ResNet-101  | ×      | ✓        | 77.3    | ×         |
| PADNet (Xu et al. 2018)      | Dilated-ResNet-101  | ×      | ✓        | 78.1    | ×         |
| DenseASPP (Maoke et al. 2018) | WDenseNet-161       | ×      | ✓        | 77.8    | ×         |
| DeepLabv3 (Chen et al. 2017) | ResNet-101          | ✓      | ✓        | 79.5    | ×         |
| Auto-DeepLab (Liu et al. 2019)|                     | ×      | ✓        | 80.3    | 44.42     |
| HRNetV2 (Sun et al. 2019)   |                     | ×      | ✓        | 80.9    | 69.06     |
| Ours                        | ×                   | ×      | ✓        | 81.0    | 17.20     |
| Ours + RSP                   | ×                   | ×      | ✓        | 81.0    | 17.20     |

Table 2: Performance evaluations of the model on Cityscapes validation set. Training with the Mapillary Vistas dataset.

| Methods                      | Backbone            | ImageNet | mIOU(%) | Params(M) |
|------------------------------|---------------------|----------|---------|-----------|
| Mapillary (Neuhold et al. 2017) | ResNeXt-101       | ✓        | 80.6    | ×         |
| HANet (Sungha, Joanne, and Jaegul 2020) | ResNet-101       | ✓        | 81.7    | ×         |
| HRNetV2-OCR (Sun et al. 2019) | HRNetV2            | ✓        | 81.8    | 70.37     |
| DecoupleSegNets              | Wide-ResNet        | ✓        | 81.6    | ×         |
| DCNAS (Zhang et al. 2020)    |                     | ✓        | 81.3    | ×         |
| Ours                        | ×                   | ×        | 81.4    | 68.67     |
| Ours + RSP                   | ×                   | ✓        | 82.1    | 68.67     |

Table 1 shows that our NAS model outperforms the SoTA on the Cityscapes. Without any pretraining, our best model significantly outperforms all the SoTA method. Last row of Table 1 shows that the light-weigh RSP design uses only 1/4 parameter size of SoTA methods but still outperforms them. Fig. 8 shows that the visualization of our model on Cityscapes (Cordts et al. 2016) validation and test set.

The Mapillary Vistas Dataset is a large-scale street-level image dataset containing 25,000 high-resolution images annotated into 66/124 object categories of which 37/70 classes are instance-specific labels (v.1.2 and v2.0, respectively). Annotation is performed in a dense and fine-grained style by using polygons for delineating individual objects. Dataset contains images from all around the world, captured at various conditions regarding weather, season and daytime. Images come from different imaging devices (mobile phones, tablets, action cameras, professional capturing rigs) and differently experienced photographers. We also adopted the Mapillary Vistas Dataset (Neuhold et al. 2017) during our training procedure. Because of the class number of cityscapes is less than Mapillary Vistas Dataset, we have to map the category into the corresponding ones in Cityscapes. Table 2 shows that our NAS model outperforms the SoTA on the Cityscapes with adopting Mapillary Vistas Dataset (Neuhold et al. 2017). Clearly, our light-weight RSP design still outperform other SoTA methods even with only 1/4 parameter size.

To better aggregate the context, we also adopted the Object-Contextual Representations (OCR) (Yuan et al. 2019). Table 3 shows that our NAS model outperforms the other model with pretraining and Object-Contextual Representations (OCR) (Yuan et al. 2019) on the Cityscapes dataset. Without using any backbone, our light-weight architecture still outperforms HRNET and DCNAS.

**Results on Cityscapes scene segmentation**

The Cityscapes dataset (Cordts et al. 2016) is a recent large-scale urban scene dataset containing a diverse set of stereo video sequences from 50 cities. Cityscapes dataset contains high quality pixel-level annotations of 5,000 images with size 1,024 × 2,048. There are 2,975, 500, and 1,525 for training, validation, and test images, respectively, and additional 20,000 weakly annotated frames. It is an order of magnitude larger than similar previous datasets.

We consider 2 intermediate nodes in all cells with one input. For each cell, we keep the channel numbers and the height and width of the feature tensor. For each cell, we keep the channel numbers and the height and width of the feature tensor. Fig. 7 shows the searched cell architectures by NAS on the Cityscapes dataset. During our training procedure. Because of the class number of cityscapes is less than Mapillary Vistas Dataset, we have to map the category into the corresponding ones in Cityscapes. Table 2 shows that our NAS model outperforms the SoTA on the Cityscapes with adopting Mapillary Vistas Dataset (Neuhold et al. 2017). Clearly, our light-weight RSP design still outperforms other SoTA methods even with only 1/4 parameter size.

We consider 2 intermediate nodes in all cells with one input. For each cell, we keep the channel numbers and the height and width of the feature tensor. For each cell, we keep the channel numbers and the height and width of the feature tensor. Fig. 7 shows the searched cell architectures by NAS on the Cityscapes dataset. During our training procedure. Because of the class number of cityscapes is less than Mapillary Vistas Dataset, we have to map the category into the corresponding ones in Cityscapes. Table 2 shows that our NAS model outperforms the SoTA on the Cityscapes with adopting Mapillary Vistas Dataset (Neuhold et al. 2017). Clearly, our light-weight RSP design still outperforms other SoTA methods even with only 1/4 parameter size.

Table 2 shows that our NAS model outperforms the SoTA on the Cityscapes. Without any pretraining, our best model significantly outperforms all the SoTA method. Last row of Table 1 shows that the light-weigh RSP design uses only 1/4 parameter size of SoTA methods but still outperforms them. Fig. 8 shows that the visualization of our model on Cityscapes (Cordts et al. 2016) validation and test set.

**Results on Cityscapes scene segmentation**

The Cityscapes dataset (Cordts et al. 2016) is a recent large-scale urban scene dataset containing a diverse set of stereo video sequences from 50 cities. Cityscapes dataset contains high quality pixel-level annotations of 5,000 images with size 1,024 × 2,048. There are 2,975, 500, and 1,525 for training, validation, and test images, respectively, and additional 20,000 weakly annotated frames. It is an order of magnitude larger than similar previous datasets.

We consider 2 intermediate nodes in all cells with one input. For each cell, we keep the channel numbers and the height and width of the feature tensor. For each cell, we keep the channel numbers and the height and width of the feature tensor. Fig. 7 shows the searched cell architectures by NAS on the Cityscapes dataset. During our training procedure. Because of the class number of cityscapes is less than Mapillary Vistas Dataset, we have to map the category into the corresponding ones in Cityscapes. Table 2 shows that our NAS model outperforms the SoTA on the Cityscapes with adopting Mapillary Vistas Dataset (Neuhold et al. 2017). Clearly, our light-weight RSP design still outperforms other SoTA methods even with only 1/4 parameter size.

We consider 2 intermediate nodes in all cells with one input. For each cell, we keep the channel numbers and the height and width of the feature tensor. For each cell, we keep the channel numbers and the height and width of the feature tensor. Fig. 7 shows the searched cell architectures by NAS on the Cityscapes dataset. During our training procedure. Because of the class number of cityscapes is less than Mapillary Vistas Dataset, we have to map the category into the corresponding ones in Cityscapes. Table 2 shows that our NAS model outperforms the SoTA on the Cityscapes with adopting Mapillary Vistas Dataset (Neuhold et al. 2017). Clearly, our light-weight RSP design still outperforms other SoTA methods even with only 1/4 parameter size.

We consider 2 intermediate nodes in all cells with one input. For each cell, we keep the channel numbers and the height and width of the feature tensor. For each cell, we keep the channel numbers and the height and width of the feature tensor. Fig. 7 shows the searched cell architectures by NAS on the Cityscapes dataset. During our training procedure. Because of the class number of cityscapes is less than Mapillary Vistas Dataset, we have to map the category into the corresponding ones in Cityscapes. Table 2 shows that our NAS model outperforms the SoTA on the Cityscapes with adopting Mapillary Vistas Dataset (Neuhold et al. 2017). Clearly, our light-weight RSP design still outperforms other SoTA methods even with only 1/4 parameter size.
Table 3: Performance evaluation on the Cityscapes validation set.

| Methods                  | OCR | ImageNet | mIOU(%) |
|--------------------------|-----|----------|---------|
| HRNetV2 (Sun et al. 2019)| ×   | ×        | 76.16   |
| HRNetV2                  | ✓   | ×        | 78.2    |
| HRNetV2                  | ×   | ✓        | 80.9    |
| HRNetV2                  | ✓   | ✓        | 81.6    |
| DCNAS (Zhang et al. 2020)| ×   | ×        | 81.9    |
| Ours                     | ×   | ✓        | 81.4    |
| Ours                     | ✓   | ✓        | 83.2    |
| Ours + RSP               | ✓   | ✓        | 82.5    |

Discussions

We discuss practical considerations in deploying our method. First, is the sensitivity of the number $N$ of cells that is stacked during training. We use a fixed $N = 4$ in this paper. More search parameters are involved and more time will be consumed for larger $N$, however better image segmentation performance might be obtained. Secondly, the potential negative impact of our work is on the environmental aspect, that automatic model search requires significant GPU computation. Again, the proposed micro search can be completed in two day of computation on modern GPUs.

Conclusion

This paper designs a two-stage architecture that allows us to search micro structure with less memory resource for image segmentation. The complexity reduction in search space and path selection makes the entire architecture search optimization very efficient. It takes about 2 days on two V100 GPUs to find the desired micro architecture. RSPNet makes our search procedure much more efficient and leads to our light-weight design for outstanding semantic segmentation.

Future work includes the generalization and consideration of additional computation structures and cells as a basis that can be incorporated into the proposed framework. The model can be replaced with other simple structures for memory- or power-aware applications. Another issue is the selection of the number $N$. Current version sets the same $N$ for each module. If the number of each module is different, the system performance will be further improved. In addition, we envision works to be carried out straightforwardly on the NAS for other computer vision and NLP tasks.
