Graph-guided Architecture Search for Real-time Semantic Segmentation

Peiwen Lin[†*] Peng Sun[‡*] Guangliang Cheng[†] Sirui Xie[†] Xi Li[‡] Jianping Shi[†]

[†]SenseTime Research [‡]Zhejiang University

{linpeiwen, chengguangliang, xiesirui, shijianping}@sensetime.com
{sunpeng1996, xilizju}@zju.edu.cn

Abstract

Designing a lightweight semantic segmentation network often requires researchers to find a trade-off between performance and speed, which is always empirical due to the limited interpretability of neural networks. In order to release researchers from these tedious mechanical trials, we propose a Graph-guided Architecture Search (GAS) pipeline to automatically search real-time semantic segmentation networks. Unlike previous works that use a simplified search space and stack a repeatable cell to form a network, we introduce a novel search mechanism with new search space where a lightweight model can be effectively explored through the cell-level diversity and latency-oriented constraint. Specifically, to produce the cell-level diversity, the cell-sharing constraint is eliminated through the cell-independent manner. Then a graph convolution network (GCN) is seamlessly integrated as a communication mechanism between cells. Finally, a latency-oriented constraint is endowed into the search process to balance the speed and performance. Extensive experiments on Cityscapes and CamVid datasets demonstrate that GAS achieves new state-of-the-art trade-off between accuracy and speed. In particular, on Cityscapes dataset, GAS achieves the new best performance of 73.3% mIoU with speed of 102 FPS on Titan Xp.

1. Introduction

As a fundamental topic in computer vision, semantic image segmentation [24, 44, 9, 7] aims at predicting pixel-level labels for an image. Leveraging the strong ability of CNNs, many works have achieved state-of-the-art performance in popular semantic segmentation benchmarks [13, 15, 4]. To achieve higher accuracy, state-of-the-art models become increasingly larger and deeper that require high computational resources and large memory overhead, which makes it difficult to deploy on resource-constrained platforms, such as mobile devices, robotics, and self-driving cars, etc.

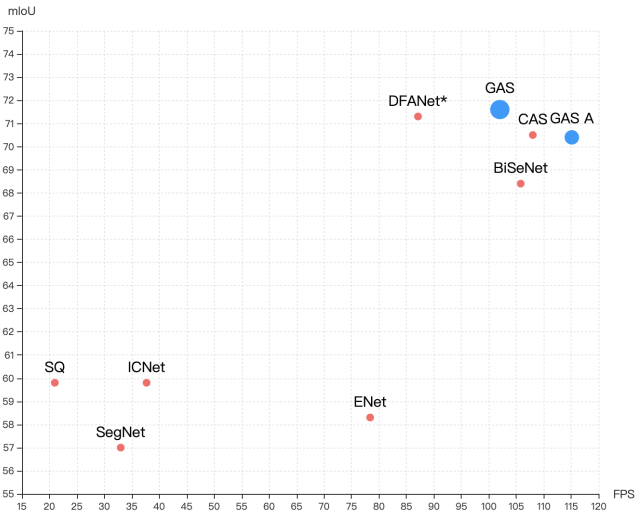

Figure 1. The inference speed and mIoU for different networks on the Cityscapes test set. Our GAS achieves the state-of-the-art trade-off between speed and performance. The Mark $*$ denotes the speed is remeasured on Titan Xp. **Best viewed in color.**

Recently, many researches have focused on designing and improving CNN models with light computation cost and high segmentation accuracy. For example, [1, 30] reduce the computation cost via the pruning algorithms, and [43] uses an image cascade network to incorporate multi-resolution input. BiSeNet [41] and DFANet [21] utilize a light-weight backbone to speed up, and equip with a well-designed feature fusion or aggregation module to remedy

[*]equal contribution

the accuracy drop. Normally, researchers acquire expertise in architecture design through enormous trial and error to carefully balance accuracy and resource-efficiency.

To design more effective segmentation network for embedded devices, some researchers have explored automatically neural architecture search (NAS) methods [23, 46, 27, 20, 32, 5, 39] and achieved excellent results. For example, Auto-Deeplab [22] searches cell structure and the downsampling strategy together in the same round. CAS [42] searches an architecture with customized resource constraints and a multi-scale module that has been widely used in semantic segmentation field [9, 44].

Particularly, CAS has achieved state-of-the-art segmentation performance in lightweight community [43, 21, 41]. Like the general NAS methods, such as ENAS [32], DARTS [23] and SNAS [39], CAS also searches for a few types of cells (i.e. normal cell and reduction cell) and then repeatedly stacks the same cells through the network. This simplifies the search process, but also increases the difficulties to find a good trade-off between performance and speed due to the limited cell diversity. For example, the cell is prone to learn a complicated structure to pursue high performance without any resource constraint. As shown in Fig.2(a), the whole network stacked with complicated cell will result in high latency. When a low-computation constraint is applied, the cell structure tends to be over-simplified as shown in Fig.2(b), which may not achieve satisfactory performance.

Different from the traditional search algorithms with simplified search space, in this paper, we propose a novel search mechanism with new search space, where a lightweight model with high performance can be fully explored through the well-designed cell-level diversity and latency-oriented constraint. On one hand, to encourage the cell-level diversity, we make each cell structure independent, thus the cells with different computation cost can be flexibly stacked to form a lightweight network shown in Fig.2(c). For example, simple cells can be applied to the stage with high computation cost to achieve low latency, while complicated cells can be chosen in deep layers with low computation for high accuracy. On the other hand, we apply a real-world latency-oriented constraint into the search process, through which the searched model can achieve better trade-off between the performance and latency.

However, simply endowing cells with independence in exploring its own structure enlarges the search space and makes the optimization more difficult, which causes accuracy degradation as shown in Table 3. To address this issue, we incorporate a Graph Convolution Network (GCN) [19] as the communication deliverer between cells. We name the method as Graph-guided Architecture Search (GAS). Our idea is inspired by [26] that different cells can be treated as multiple agencies, whose achievement of social welfare

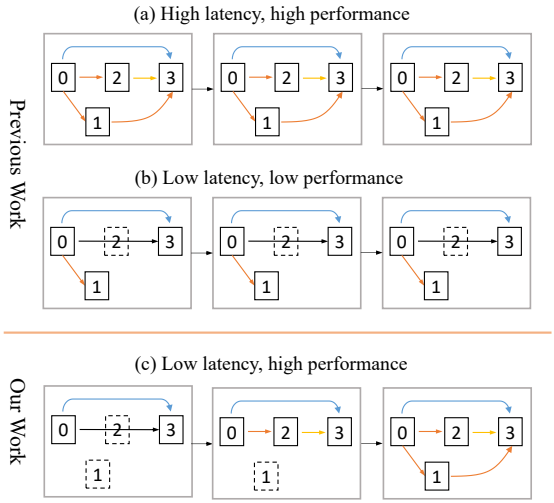

Figure 2. (a) The network stacked by complicated cells results in high latency and high performance. (b) The network stacked by simple cells leads to low latency and low performance. (c) The cell diversity strategy, i.e., each cell possesses own independent structure, can flexibly construct the high accuracy lightweight network. **Best viewed in color.**

may require communication between them. Specifically, in the forward process, starting from the first cell, the information of each cell is propagated to the next adjacent cell with a GCN. Our ablation study exhibits that this communication mechanism tends to guide cells to select less-parametric operations, thus achieving the balance between accuracy and latency.

We conduct extensive experiments on the standard Cityscapes [13] and CamVid [4] benchmarks. Compared to other state-of-the-art methods, the proposed method achieves the new best performance while maintaining competitive latency. Particularly, our method locates in the top-right area shown in Fig. 1, which achieves the state-of-the-art trade-off between speed and performance.

The main contributions can be summarized as follows:

- We propose a novel search framework, for real-time semantic segmentation task, with a new search space in which a lightweight model with high performance can be effectively explored.
- We integrate the graph convolution network seamlessly into neural architecture search as a communication mechanism between cells.
- The lightweight segmentation network searched with GAS is customizable in real applications. Notably, GAS has achieved 73.3% mIoU on Cityscapes test dataset and 102FPS on NVIDIA Titan Xp with an 769×1537 image.

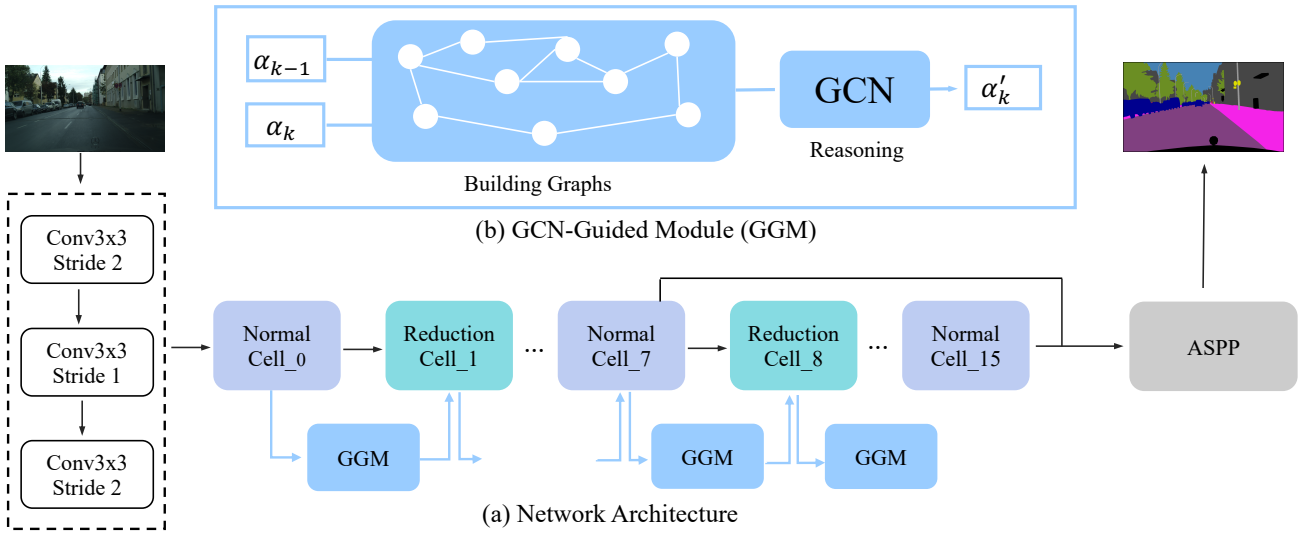

Figure 3. Illustration of our Graph-Guided Network Architecture Search. In reduction cells, all the operations adjacent to the input nodes are of stride two. (a) The backbone network, it's stacked by a series of independent cells. (b) The GCN-Guided Module (GGM), it propagates information between adjacent cells. α_k and α_{k-1} represent the architecture parameters for cell k and cell $k-1$ respectively, and $\alpha_k^{'}$ is the updated architecture parameters by GGM for cell k. **Best viewed in color.**

2. Related Work

Efficient Semantic Segmentation Methods Fully convolutional neural networks [24] is the pioneer work in semantic segmentation. Some remarkable network have achieved state-of-the-art performance by introducing heavy network backbones (VGGNet [34], ResNet [17], DenseNet [18], Xception [12]). And some outstanding works introduce effective modules to capture multi-scale context information [44, 7, 8]. In terms of efficient segmentation methods, there are two mainstreams: One is to employ relatively lighter backbone (e.g. ENet [30]) or introduce some efficient operations (depth-wise dilated convolution). DFANet [21] utilizes a lightweight backbone to speed up and equips with a cross-level feature aggregation module to remedy the accuracy drop. Another is multi-branch based algorithm that consists of more than one path. For example, ICNet [43] proposed to use the multi-scale image cascade to speed up the inference. BiSeNet [41] decouples the extraction for spatial and context information using two paths.

Neural Architecture Search Neural Architecture Search (NAS) aims at automatically searching network architectures. Most existing architecture search papers are based on either reinforcement learning [45, 16] or evolutionary algorithm [33, 11]. Though they can achieve satisfactory performance, they need thousands of GPU hours. To solve this time-consuming problem, one-shot methods [2, 3] have been developed to greatly solve the time-consuming problem by training an parent network from which each sub-network can inherit the weight. They can be roughly divided into cell-based and layer-based methods according to the type of search space. For cell-based methods, ENAS [32] proposes a parameter sharing strategy among sub-networks, DARTS [23] relaxes the discrete architecture distribution as continuous deterministic weights, such that they could be optimized with gradient descent. SNAS [39] propose novel search gradients that train neural operation parameters and architecture distribution parameters in same round of back-propagation. What's more, there are also some excellent works [10, 29] to reduce the difficult of optimization by decreasing gradually the size of search space. For layer-based methods, FBNet [37], MnasNet [35], ProxylessNAS [5] use a multi-objective search approach that optimizes both accuracy and real-world latency. In the field of semantic segmentation, [6] is the pioneer work by introducing meta-learning techniques into the network search problem. Auto-Deeplab [22] search cell structure and the downsampling strategy together in same round. More recently, CAS [42] search an architecture with customized resource constraints and a multi-scale module that has been widely used in semantic segmentation field. And [28] over-parameterise the architecture during the training via a set of auxiliary cells using reinforcement learning.

Graph Convolution Network Convolution neural networks on graph-structure data is an emerging topic in deep learning research. Kipf [19] present a scalable approach for graph-structured data that is based on an efficient variant of convolutional neural networks which operate directly on graphs, for better information transfer. After that, Graph

Convolution Networks (GCNs) [19] is widely used in many domains, such as video classification [36] and action recognition [40]. In this paper, we apply the GCNs [19] to model the relationship of adjacent cells in network architecture search. As far as we know, we propose a novel mechanism which is the first that applies graph-based neural networks for the network architecture search task.

3. Methods

As shown in Fig. 3, GAS searches for, with GCN-Guided module (GGM), an optimal network constructed by a series of independent cells. In the search process, we take the latency into consideration to get a network with computational efficiency. This searching problem can be formulated as:

$$\min_{a \in \mathcal{A}} L_{val} + \beta * L_{lat} \tag{1}$$

where $\mathcal{A}$ denotes the search space, L_{val} and L_{lat} are the validation loss and the latency loss, respectively. Our goal is to search an optimal architecture $a \in \mathcal{A}$ that can achieves the best trade-off between the performance and speed.

In this section, we will describe three main components in GAS: 1) Network Architecture Search; 2) GCN-Guided Module; 3) Latency-Oriented Optimization.

3.1. Network Architecture Search

As shown in Fig. 3 (a), the whole backbone takes one image as input which is first filtered with three convolutional layers followed by a series of independent cells. The ASPP [9] module is subsequently used to extract the multi-scale context for the final prediction.

A cell is a directed acyclic graph (DAG) as shown in Fig. 4. Each cell consists of N ordered nodes, denoted by $\mathcal{N} = \{x_1, ..., x_N\}$, and each node represents the latent representation (e.g. feature map) in network. Each directed edge in this DAG represents an operation transformation (e.g. conv, pooling). Each cell has two input nodes, represented as i_1, and i_2, and output the concatenation of all intermediate nodes $\mathcal{N}$. In our work, we set N=2. So for intermediate node x_1, it has two input $I_1 = \{i_1, i_2\}$. For intermediate node x_2, it has three input $I_2 = \{i_1, i_2, x_1\}$. The intermediate nodes x_i can be calculated by:

$$x_i = \sum_{c_h \in I_i} \widetilde{O}_{h,i}(c_h) \tag{2}$$

where $\widetilde{O}_{h,i}$ is the final operation at edge (h, i).

To search the final operation $\widetilde{O}_{h,i}$, we use the method described in SNAS [39], where the search space is represented with a set of one-hot random variables from a fully factorizable joint distribution $p(Z)$. Concretely, each edge is associated with a one-hot random variable which is multiplied as a mask to the all possible operations $O_{h,i}$=($o^1_{h,i}$, $o^2_{h,i}$, ..., $o^M_{h,i}$) in this edge. We denote one-hot random variable as $Z_{h,i}$ = ($z^1_{h,i}$, $z^2_{h,i}$, ..., $z^M_{h,i}$) where M is the number of candidate operations. The intermediate nodes during search process in such way are:

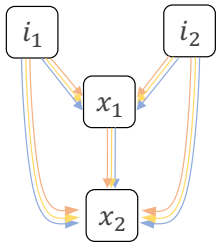

Figure 4. The structure of cell in our GAS. Each colored edge represents one candidate operation.

$$x_i = \sum_{c_h \in I_i} \widetilde{O}_{h,i}(c_h) = \sum_{c_h \in I_i} \sum_{m=1}^{M} z^m_{h,i} o^m_{h,i}(c_h) \tag{3}$$

To make $P(Z)$ differentiable, reparameterization trick [25] is used to relax the discrete architecture distribution to be continuous:

$$Z_{h,i} = f_{\alpha_{h,i}}(G_{h,i}) = softmax((log\alpha_{h,i} + G_{h,i})/\lambda) \tag{4}$$

where $\alpha_{h,i}$ is the architecture parameters at the edge (h, i), and $G_{h,i} = -log(-log(U_{h,i}))$ is a vector of Gumbel random variables, $U_{h,i}$ is a uniform random variable and λ is used to control the temperature of softmax.

For the set of candidate operations O, we only use the following 7 kinds of operations to better balance the speed and performance: 3 $\times$ 3 separable conv, 3 $\times$ 3 max pooling, 3 $\times$ 3 conv, skip connection, zero operation, 3 $\times$ 3 dilated separable conv (dilation=2), and 3 $\times$ 3 dilated separable conv (dilation=4).

3.2. GCN-Guided Module

With cell independent to each other, the inter-cell relationship becomes every important for searching efficiently. We propose a novel GCN-Guided Module (GGM) to naturally bridge the operation information between adjacent cells. The total network architecture of our GGM is shown in Fig. 3(b). Inspired by [36], the GGM represents the communication between adjacent cells as a graph and perform reasoning on the graph for information delivery. Specifically, we utilize the similarity relations of edges in adjacent cells to construct the graph where each node represents one edge in cells. In this way, the state changes for previous cell can be delivery to current cell by reasoning on this graph.

Let α_k represents the architecture parameters matrix for the cell k, and the dimension of α_k is $i \times j$ where i and j represents the number of edges and the number of candidate operations respectively. Same for cell k, the architecture parameters α_{k-1} for cell $k-1$ also is a $i \times j$ dimension matrix.

Given a edge in cell k, we calculate the similarity between this edge and all other edges in cell $k-1$. Therefore, the adjacency matrix Adj of the graph between two adjacent cells k and $k-1$ can be established by

$$Adj = \phi_1(\alpha_k) * \phi_2(\alpha_{k-1})^T \tag{5}$$

where we have $\phi_1 = w_1\alpha_k$ and $\phi_2 = w_2\alpha_{k-1}$ for two different transformations of the original matrixes, and parameters w_1 and w_2 are both $i \times i$ dimensions weights which can be learned via back propagation. The result Adj is an $i \times i$ matrix.

Based on this adjacency matrix Adj, we use Graph Convolution Networks (GCNs) [19] to perform reasoning on the graph, efficiently propagating information from cell $k-1$ to cell k. The reasoning process includes the following three steps.

Firstly, to get the graph node feature representation, we apply the convolutional operation Φ_1 to the architecture parameters α_{k-1} and then obtain its $i \times d$ dimension node feature representation matrix F_{k-1} in the embedding space:

$$F_{k-1} = \Phi_1(\alpha_{k-1}; w_{k-1}) \tag{6}$$

where w_{k-1} represents the convolutional operation weight.

Secondly, with the graph node representations F_{k-1} and the Adj, we use the GCNs [19] to perform information propagation on the graph as shown in Equation 7. A residual connection is added to each layer of GCN. The GCNs allow us to compute the response of a node based on its neighbors defined by the graph relations, so performing graph convolution is equal to performing message propagation on the graphs.

$$O_{k-1} = AdjF_{k-1}W^g_{k-1} + F_{k-1} \tag{7}$$

where the W^g_{k-1} denotes the GCNs weight with dimension $d \times d$, which can be learned via back propagation.

Finally, the output of each GCNs is still in $i \times d$ dimensions, so we use another convolutional operation Φ_2 to map the representation from the embedding space to source space as the $\Delta\alpha$ in Equation 8. So we then add the $\Delta\alpha$ and the original α_k in element-wise manner as the updated α'_k in Equation 9, where γ control the weight between α'_k and $\Delta\alpha$. Then the current cell k has fused the parameter information of previous cell $k-1$. We use the new α'_k as the new architecture parameter of cell k. And the e_{k-1} denotes the weight for Φ_2.

$$\Delta\alpha = \Phi_2(O_{k-1}; e_{k-1}) \tag{8}$$

$$\alpha'_k = \alpha_k + \gamma\Delta\alpha \tag{9}$$

Through the proposed well-designed GGM that seamlessly integrates the graph convolution network and neural architecture search, which can bridge the operation information between adjacent cells.

3.3. Latency-Oriented Optimization

Similar to many excellent NAS works [5, 37, 42, 35], we also take real-world latency for a network into consideration during the search process, which orients the search process toward the direction to find a optimal lightweight model. Specifically, we create a GPU-latency lookup table which records the inference latency of each candidate operation. During the search process, each candidate operation m at edge (h, i) will be assigned a cost $lat^m_{h,i}$ given by the designed lookup table. In this way, the total latency for cell k is accumulated as:

$$lat_k = \sum_{h,i} \sum_{m=1}^{M} z^m_{h,i} lat^m_{h,i} \tag{10}$$

where $z^m_{h,i}$ is the architecture parameter for operation m at edge (h, i) and M is the number of candidate operations. Given a architecture a, the total latency cost is estimated as:

$$LAT(a) = \sum_{k=0}^{K} lat_k \tag{11}$$

where K refers to the number of cells in architecture a. The latency for each operation $z^m_{h,i}$ is a constant and thus total latency loss is differentiable with respect to the architecture parameters $z^m_{h,i}$.

Different from the exponent coefficient latency loss [37], we briefly define the total loss function as follows:

$$L(a, w) = CE(a, w_a) + \beta\, log(LAT(a)) \tag{12}$$

where $CE(a, w_a)$ denotes the cross-entropy loss of architecture a with parameter w_a, $LAT(a)$ denotes the overall latency of architecture a, which is measured in microsecond, and the coefficient β controls the balance between the accuracy and latency. During the search phrase, we directly optimize the architecture parameter a and the weight w in same round of back-propagation rather than using iterative optimization [37].

4. Experiments

In this section, we conduct extensive experiments to verify the effectiveness of our GAS. Firstly, we compare the network searched by our method with the state-of-the-art works on two standard benchmarks. Secondly, we perform the ablation study for the GCN-Guided Module and latency optimization settings and close with a insight about GCN-Guided module.

4.1. Benchmark and Evaluation Metrics

Datasets In order to verify the effectiveness and robustness of our method, we evaluate our method on Cityscapes [13] and CamVid [4] datasets. The Cityscapes [13] is a public released dataset for semantic urban scene understanding. It contains 5,000 high quality pixel-level fine annotated images (2975, 500, and 1525 for the training, validation, and testing sets respectively) with size 1024 x 2048 collected from 50 cities. The dense annotation contains 30 common classes and 19 of them are used in training and testing following [13]. CamVid [4] is another public released dataset with object class semantic labels. It contains 701 images in total, in which 367 for training, 101 for validation and 233 for testing. The images have a resolution of 960 x 720 and 11 semantic categories.

Evaluation Metrics For evaluation, we use three metrics, including mean of class-wise intersection over uniou (mIOU), network forward time (Latency), and Frames Per Second (FPS).

4.2. Implementation Details

We conduct all experiments based on Pytorch 0.4 [31]. All experiments are on a workstation with Titan Xp GPU cards under CUDA 9.0, and the inference time in all experiments is also reported on Nvidia Titan Xp GPU.

We first conduct architecture search using GAS on segmentation dataset and then obtain the target light-weight network architecture according to the optimized α. We then utilize the ImageNet [14] dataset to pretrain the searched network from scratch. We finally finetune the network on the specific segmentation dataset for 200 epochs.

In search process, the architecture contains 16 cells and each cell has $N = 2$ nodes. With the consideration of speed, the initial channel for network is 8. For the training hyper-parameters, the mini-batch size is set to 16. The architecture distribution parameters α are optimized by Adam, with initial learning rate 0.001, β = (0.5, 0.999) and weight decay 0.0001. The network parameters are optimized using SGD with momentum 0.9, weight decay 0.001, and cosine learning scheduler that decays learning rate from 0.025 to 0.001. For gumbel softmax, we set the initial temperature λ in equation 4 as 1.0, and gradually decrease to the minimum value of 0.03.

For finetuning details, we train the network with mini-batch 8 and SGD optimizer with poly learning rate scheduler that decay learning rate from 0.01 to zero. Following [38], The online bootstrapping strategy has been applied to the training process. For data augmentation, we use random flip and random resize with scale between 0.5 and 2.0. Finally, we randomly crop the image into a fixed size for training.

For the GCN-guided Module, we use one Graph Convolution Network (GCN) [19] between every adjacent cells, and each GCN contains one layer of graph convolutions. The kernels size of the parameters W in graph convolutions operation is 64x64.

Method	InputSize	mIOU	Latency(ms)	FPS
FCN-8S	512x1024	65.3	227.23	4.4
PSPNet	713x713	81.2	1288.0	0.78
DeepLabV3*	769x769	81.3	769.23	1.3
SegNet	640x320	57.0	30.3	33
ENet	640x320	58.3	12.7	78.4
SQ	1024x2048	59.8	46.0	21.7
ICNet	1024x2048	69.5	26.5	37.7
BiSeNet	768x1536	68.4	9.52	105.8
DFANet A§	1024x1024	71.3	10.0	100.0
DFANet A† [1]	1024x1024	71.3	11.48	87.1
CAS	768x1536	70.5	9.25	108.0
CAS*	768x1536	72.3	9.25	108.0
GAS	769x1537	71.6	9.80	102.0
GAS A	769x1537	70.4	8.68	115.1
GAS*	769x1537	73.3	9.80	102.0

Table 1. Comparing results on Cityscapes test dataset. Methods trained using both fine and coarse data are marked with $*$. The mark § represents the speed on TitanX, and the mark † represents the speed is remeasured on Titan Xp.

4.3. Real-time Semantic Segmentation Results

In this part, we compare the model searched by GAS with other existing state-of-the-art real-time segmentation models on semantic segmentation datasets. The inference time is calculated on one Nvidia Titan Xp GPU and the speed of other methods reported in the paper [42] are used for comparing. Moreover, the speed is measured again on the Titan Xp if the origin paper reports the speed on different GPU.

Results on Cityscapes. We evaluate the network searched by GAS on Cityscapes test sets. The validation set is added to train network before submitting to Cityscapes server. Following [41, 42], GAS takes as an input image with size 769x1537 that resize from origin image size 1024x2048. Overall, our GAS get the best performance among all methods while maintains the comparable speed with 102 FPS. With only fine data and without any evaluation tricks, our GAS yields 71.6% mIoU which is the state-of-the-art trade-off for light-weight semantic segmentation. The performance achieve 73.3% when coarse data is added into the training dataset. The full comparison results are shown in Table 1. Compared to BiSeNet and CAS which have a slight speed advantage, our GAS beat them along multiple performance points with 3.2% and 1.1% respectively. Compared to other methods such as SegNet, ENet, SQ and ICNet, our method achieves significant improvement in speed while get performance improvement over

them about 14.6%, 13.3%, 11.8%, 2.1% respectively. GAS A is searched by the latency constraint 0.01.

Results on CamVid. We also conduct the whole GAS pipeline on CamVid dataset to further verify our method's ability. Table 2 shows the comparison results with other methods. With input size 720x960, we achieve the 71.9% mIoU with 142 FPS which is also the state-of-the-art trade-off between accuracy and speed.

Method	mIOU	Latency(ms)	FPS
SegNet	55.6	34.01	29.4
ENet	51.3	16.33	61.2
ICNet	67.1	28.98	34.5
BiSeNet	65.6	-	-
DFANet A	64.7	8.33	120
CAS	71.2	5.92	169
GAS	71.9	7.04	142.0

Table 2. Results on CamVid test set with resulotion 960x720. "-" indicates the corresponding result is not provided by the methods.

4.4. Ablation Study

In this part, we detailedly verify the effect of each component in our framework, we perform the ablation study experiments for the GCN-Guided Module and the latency loss. Furthermore, we give a insight about what role does the GCN-Guided Module play in the search process.

Methods	mIOU
a) Cell shared	63.0
b) Cell independent	60.7
c) Cell independent + FC	63.6
d) Cell independent + GCN	66.3

Table 3. Ablation study for the effectiveness of GCN-Guided Module on Cityscapes dataset.

4.4.1 Effectiveness of the GCN-Guided Module

We propose the GCN-Guided Module (GGM) to build the connection between cells. To verify the advantage of the GGM, we conducted a series of experiments with different strategies: a) network stacked by shared cell; b) network stacked by independent cell; c) Based on b, using fully connected layer to infer the relationship between cells; d) Based on b, using GCN-Guided Module to infer the relationship between cells. Experiment results are shown in Table 3. The performance reported here is the average mIoU over five repeated experiments on Cityscapes validation dataset during search phrase without latency loss. Overall, with only independent cell, the performance degrades due to the enlarge search space which make optimization more difficult. This reduction is mitigated through adding communication mechanism between cells by GCN. Specially, our GCN-guided module can bring about 2.7 points performance improvement compare to the setting (c).

We illustrate the network structure searched by GAS in the Fig. 6. An interesting observation is that the operations selected by GAS with GGM have fewer parameters and less computational complexity than GAS without GGM, where more dilated or separated convolution kernels are preferred. This exhibits the emergence of concept of burden sharing in a group of cells when they know how much others are willing to contribute. It also explains why GAS with GGM on is less overfitted.

[1]After merging the BN layers for DFANet, there still has a speed gap between the original paper and our measurement. We suspect that it's caused by the inconsistency of implementation platform in which DFANet has optimized the depth-wise convolution (DW-Conv). GAS also have many candidate operations using DW-Conv, so the speed of our GAS is still capable of beating it if the DW-Conv be optimized correctly like DFANet or BiSeNet.

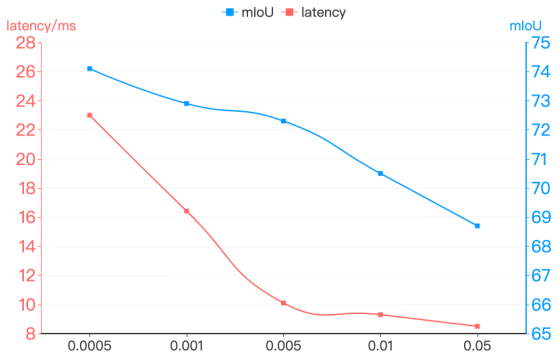

Figure 5. The validation accuracy on Cityscapes dataset for different latency constraint. **Best viewed in color.**

4.4.2 Effectiveness of the Latency Constraint

As mentioned earlier, GAS provides the ability to flexibly achieve a good trade-off between the performance and speed with the latency-oriented optimization. We conduct a series of experiments with different loss weight β in Equation 12. Fig. 5 shows the variation of mIoU and latency as β changes. With smaller β, we can obtain a model with higher accuracy, and vice-versa. When the β increases from 0.0005 to 0.005, the latency decreases rapidly and the performance is slowly falling. But when β increases from 0.005 to 0.05, the performance drops quickly and the decline of latency is fairly limited. So in our experiments, we set β as 0.005. We can clearly see that the latency-oriented optimization is effective for balancing the accuracy and latency.

4.4.3 Analysis of the GCN-Guided Module

One concern is about what kind of role does GCN play in the search process. We suspect that its effectiveness is derived from the following two aspects: 1) In order to learn a light-weight network, we allow the cell structures not to share with each other to encourage structure diversity. Apparently, learning cell independently makes the search more difficult and does not guarantee better performance, thus the GCN-Guided Module can be regraded as a regularization term to regularize the search process. 2) We have discussed that $p(Z)$ is a fully factorizable joint distribution in above section. As shown in Equation 4, $p(Z_{h,i})$ for current cell becomes a conditional probability if the architecture parameter $\alpha_{h,i}$ depends on the probability $\alpha_{h,i}$ for previous cell. In this case, the GCN-Guided Module plays a role that model the condition in probability distribution $p(Z)$.

5. Conclusion & Discussion

In this paper, a novel Graph-guided architecture search (GAS) framework is proposed to tackle the real-time semantic segmentation task. Different to the existing NAS approaches that stacks the same searched cell into a whole network, GAS explores to search different cell architectures and adopts the graph convolution network to bridge the information connection among cells. In addition, a latency-oriented constraint is endowed into the search process for balancing the accuracy and speed. Extensive experiments have demonstrated that GAS has achieved much better performance than the state-of-the-art real-time segmentation approaches.

In the future, we will extend the GAS to the following directions: 1) We will search networks directly for the segmentation and detection tasks without retraining. 2) We will explore some deeper research on how to effectively combine the NAS and the graph convolution network. 3) Exploring other approaches to apply latency constraint.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE trans. PAMI*, 39(12):2481–2495, 2017. 1

[2] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le. Understanding and simplifying one-shot architecture search. In *ICML*, pages 549–558, 2018. 3

[3] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Smash: one-shot model architecture search through hypernetworks. *arXiv:1708.05344*, 2017. 3

[4] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV (1)*, pages 44–57, 2008. 1, 2, 6

[5] H. Cai, L. Zhu, and S. Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv:1812.00332*, 2018. 2, 3, 5

[6] L. 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 *NeurIPS*, pages 8713–8724, 2018. 3

[7] L. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017. 1, 3

[8] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, pages 833–851, 2018. 3

[9] 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 trans. PAMI*, 40(4):834–848, 2018. 1, 2, 4

[10] X. Chen, L. Xie, J. Wu, and Q. Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. *arXiv:1904.12760*, 2019. 3

[11] Y. Chen, Q. Zhang, C. Huang, L. Mu, G. Meng, and X. Wang. Reinforced evolutionary neural architecture search. *CoRR*, abs/1808.00193, 2018. 3

[12] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, pages 1800–1807, 2017. 3

[13] 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 *CVPR*, pages 3213–3223, 2016. 1, 2, 6

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009. 6

[15] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 111(1):98–136, 2015. 1

[16] M. Guo, Z. Zhong, W. Wu, D. Lin, and J. Yan. IRLAS: inverse reinforcement learning for architecture search. *CoRR*, abs/1812.05285, 2018. 3

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 3

[18] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CVPR*, pages 1–9, 2016. 3

[19] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv:1609.02907*, 2016. 2, 3, 4, 5, 6

[20] B. Krause, E. Kahembwe, I. Murray, and S. Renals. Dynamic evaluation of neural sequence models. *arXiv:1709.07432*, 2017. 2

[21] H. Li, P. Xiong, H. Fan, and J. Sun. Dfanet: Deep feature aggregation for real-time semantic segmentation. In *CVPR*, pages 9522–9531, 2019. 1, 2, 3

[22] C. Liu, L. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. *CoRR*, abs/1901.02985, 2019. 2, 3

[23] H. Liu, K. Simonyan, and Y. Yang. DARTS: differentiable architecture search. In *ICLR*, 2019. 2, 3

[24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 1, 3

[25] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv:1611.00712*, 2016. 4

[26] M. Minsky. *The Society of Mind*. Simon & Schuster, 1988. 2

[27] R. Negrinho and G. Gordon. Deeparchitect: Automatically designing and training deep architectures. *arXiv:1704.08792*, 2017. 2

[28] V. Nekrasov, H. Chen, C. Shen, and I. Reid. Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In *CVPR*, pages 9126–9135, 2019. 3

[29] A. Noy, N. Nayman, T. Ridnik, N. Zamir, S. Doveh, I. Friedman, R. Giryes, and L. Zelnik-Manor. Asap: Architecture search, anneal and prune. *arXiv:1904.04123*, 2019. 3

[30] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv:1606.02147*, 2016. 1, 3

[31] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017. 6

[32] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean. Efficient neural architecture search via parameter sharing. In *ICML*, pages 4092–4101, 2018. 2, 3

[33] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. *CoRR*, abs/1802.01548, 2018. 3

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 3

[35] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, pages 2820–2828, 2019. 3, 5

[36] X. Wang and A. Gupta. Videos as space-time region graphs. In *ECCV*, pages 399–417, 2018. 4

[37] 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 *CVPR*, pages 10734–10742, 2019. 3, 5

[38] Z. Wu, C. Shen, and A. van den Hengel. High-performance semantic segmentation using very deep fully convolutional networks. *CoRR*, abs/1604.04339, 2016. 6

[39] S. Xie, H. Zheng, C. Liu, and L. Lin. SNAS: stochastic neural architecture search. In *ICLR*, 2019. 2, 3, 4

[40] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018. 4

[41] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *ECCV*, pages 334–349, 2018. 1, 2, 3, 6

[42] Y. Zhang, Z. Qiu, J. Liu, T. Yao, D. Liu, and T. Mei. Customizable architecture search for semantic segmentation. In *CVPR*, pages 11641–11650, 2019. 2, 3, 5, 6

[43] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia. Icnet for real-time semantic segmentation on high-resolution images. In *ECCV*, pages 405–420, 2018. 1, 2, 3

[44] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, pages 2881–2890, 2017. 1, 2, 3

[45] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 3

[46] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, pages 8697–8710, 2018. 2

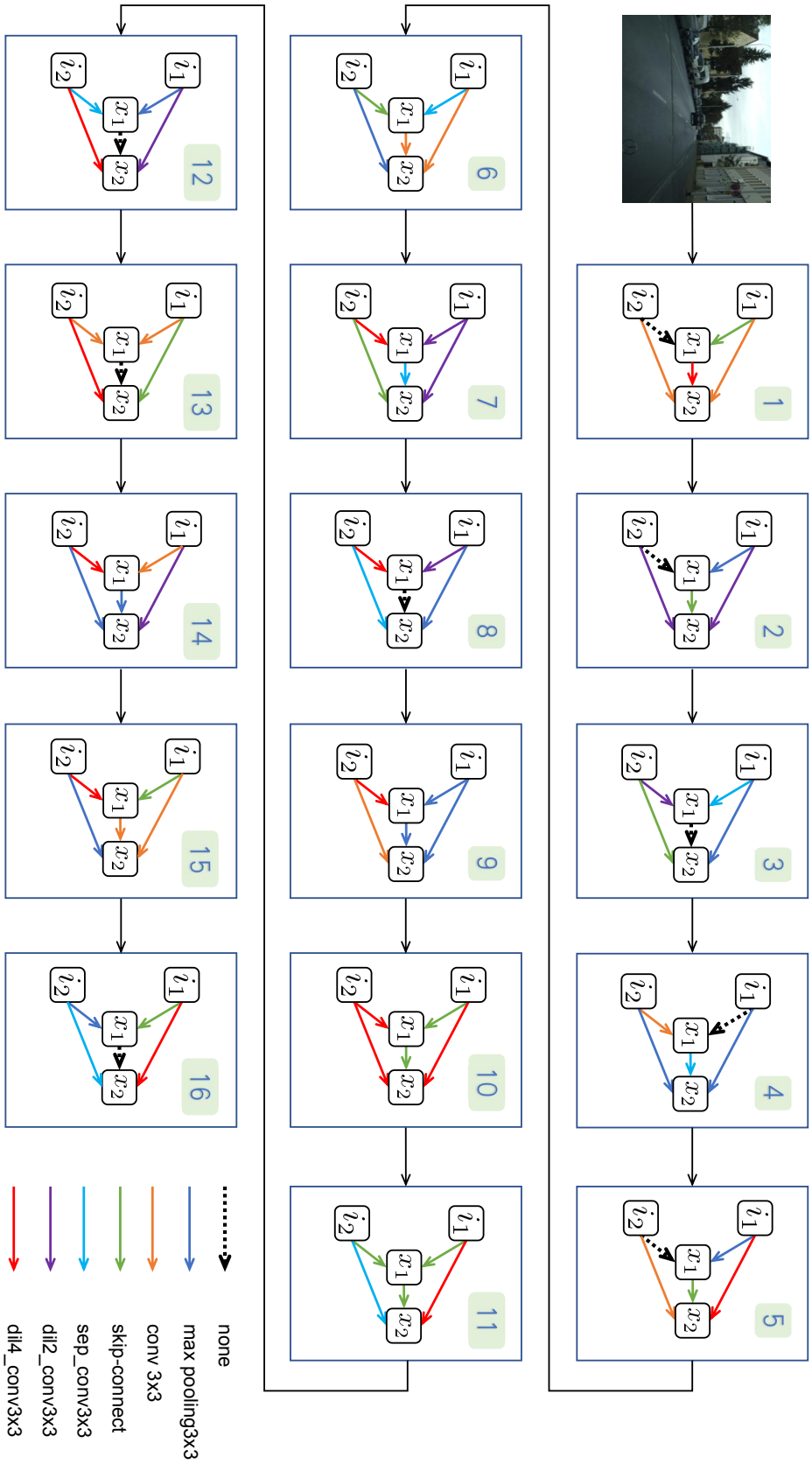

Figure 6. The model structure searched by GAS. **Best viewed in color.**