Graph-guided Architecture Search for Real-time Semantic Segmentation

Peiwen Lin[1,*]   Peng Sun[2,*]   Guangliang Cheng[1]   Sirui Xie[3]   Xi Li[2,†]   Jianping Shi[1,†]

[1]SenseTime Research   [2]Zhejiang University   [3]University of California, Los Angeles

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

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 a 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 the new state-of-the-art trade-off between accuracy and speed. In particular, on Cityscapes dataset, GAS achieves the new best performance of 73.5% mIoU with speed of 108.4 FPS on Titan Xp.

1. Introduction

As a fundamental topic in computer vision, semantic segmentation [26, 51, 9, 7] aims at predicting pixel-level labels for images. Leveraging the strong ability of CNNs [38, 18, 19, 12], many works have achieved remarkable performance on public semantic segmentation benchmarks [13, 15, 4]. To pursue higher accuracy, state-of-the-art models become increasingly larger and deeper, and thus require

---

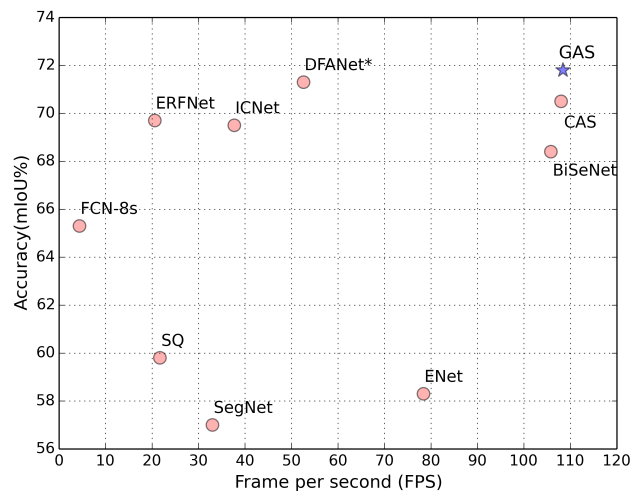*The first two authors contributed equally to this paper.
†Corresponding Author

Figure 1. The inference speed and mIoU for different networks on the Cityscapes test set with only fine training data. 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.**

high computational resources and large memory overhead, which makes it difficult to deploy on resource-constrained platforms, such as mobile devices, robotics, self-driving cars, etc.

Recently, many researches have focused on designing and improving CNN models with light computation cost and satisfactory segmentation accuracy. For example, some works [1, 34] reduce the computation cost via the pruning algorithms, and ICNet [50] uses an image cascade network to incorporate multi-resolution inputs. BiSeNet [47] and DFANet [22] utilize a light-weight backbone to speed up, and is equipped with a well-designed feature fusion or aggregation module to remedy the accuracy drop. To achieve such design, researchers acquire expertise in architecture design through enormous trial and error to carefully balance the accuracy and resource-efficiency.

To design more effective segmentation networks, some

researchers have explored automatically neural architecture search (NAS) methods [25, 53, 30, 21, 36, 5, 44] and achieved excellent results. For example, Auto-Deeplab [24] searches cell structures and the downsampling strategy together in the same round. CAS [49] searches an architecture with customized resource constraint and a multi-scale module which has been widely used in semantic segmentation field [9, 51].

Particularly, CAS has achieved state-of-the-art segmentation performance in mobile setting [50, 22, 47]. Like the general NAS methods, such as ENAS [36], DARTS [25] and SNAS [44], CAS also searches for two types of cells (*i.e.* normal cell and reduction cell) and then repeatedly stacks the identical cells to form the whole 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. As shown in Figure 2(a), the cell is prone to learn a complicated structure to pursue high performance without any resource constraint, and the whole network will result in high latency. When a low-computation constraint is applied, the cell structures tend to be over-simplified as shown in Figure 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, and thus the cells with different computation cost can be flexibly stacked to form a lightweight network in Figure 2(c). In this way, 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 structures enlarges the search space and makes the optimization more difficult, which causes accuracy degradation as shown in Figure 5(a) and Figure 5(b). To address this issue, we incorporate a Graph Convolution Network (GCN) [20] as the communication deliverer between cells. Our idea is inspired by [29] that different cells can be treated as multiple agencies, whose achievement of social welfare 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, while achieving the satis-
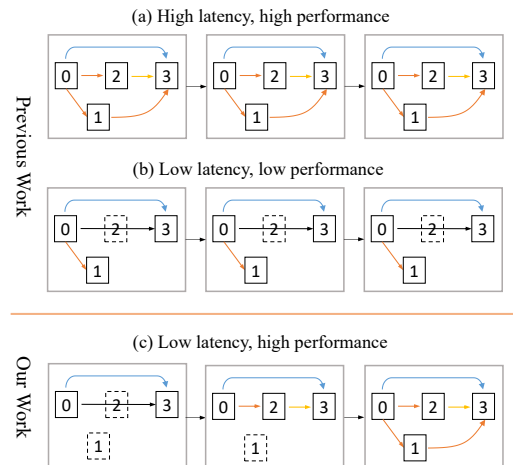

Figure 2. (a) The network stacked by complicated cells results in high latency and high performance. (b) The network stacked by simple cells leads to low latency and low performance. (c) The cell diversity strategy, *i.e.* each cell possesses own independent structure, can flexibly construct the high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

factory accuracy. We name the method as Graph-guided Architecture Search (GAS).

We conduct extensive experiments on the standard Cityscapes [13] and CamVid [4] benchmarks. Compared to other real-time methods, our method locates in the top-right area in Figure 1, which is the state-of-the-art trade-off between the performance and latency.

The main contributions can be summarized as follows:

- We propose a novel search framework, for real-time semantic segmentation, 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 independent cells.

- The lightweight segmentation network searched with GAS is customizable in real applications. Notably, GAS has achieved 73.5% mIoU on the Cityscapes test set and 108.4 FPS on NVIDIA Titan Xp with one 769×1537 image.

2. Related Work

Semantic Segmentation Methods FCN [26] is the pioneer work in semantic segmentation. To improve the segmentation performance, some remarkable works have utilized various heavy backbones [38, 18, 19, 12] or effective modules to capture multi-scale context information [51, 7, 8]. These outstanding works are designed for high-quality
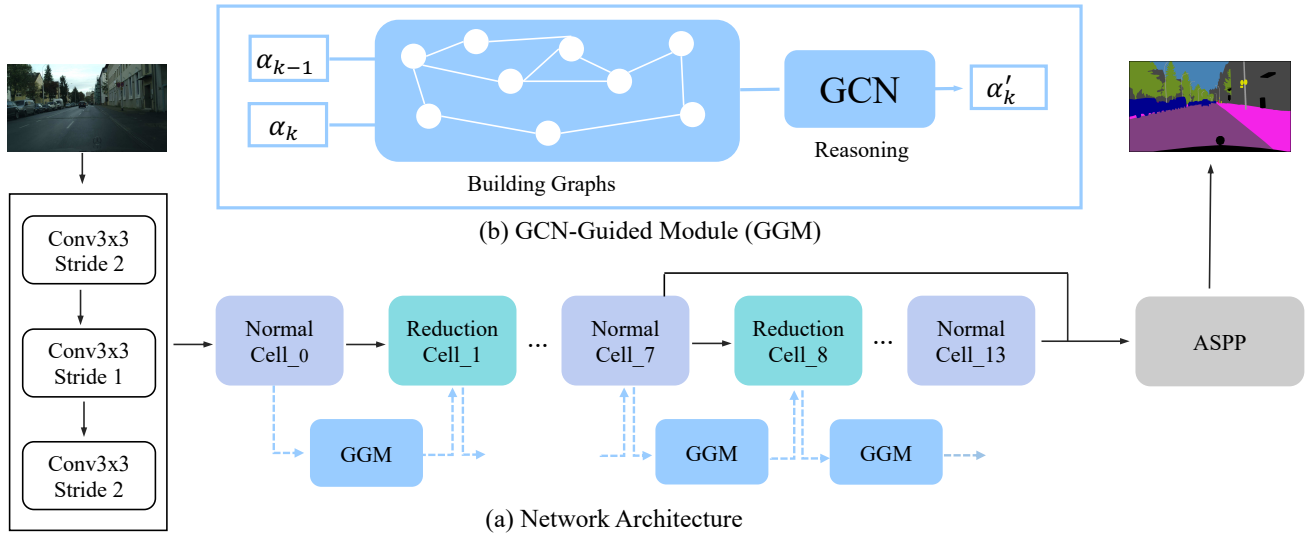
Figure 3. Illustration of our Graph-Guided Network Architecture Search. In reduction cells, all the operations adjacent to the input nodes are of stride two. (a) The backbone network, it's stacked by a series of independent cells. (b) The GCN-Guided Module (GGM), it propagates information between adjacent cells. α_k and α_{k-1} represent the architecture parameters for cell k and cell $k-1$, respectively, and α_k' is the updated architecture parameters by GGM for cell k. The dotted lines indicate GGM is only utilized in the search progress. **Best viewed in color.**

segmentation, which is inapplicable to real-time applications. In terms of efficient segmentation methods, there are two mainstreams. One is to employ relatively lighter backbone (*e.g.* ENet [34]) or introduce some efficient operations (depth-wise dilated convolution). DFANet [22] utilizes a lightweight backbone to speed up and equips with a cross-level feature aggregation module to remedy the accuracy drop. Another is based on multi-branch algorithm that consists of more than one path. For example, ICNet [50] proposes to use the multi-scale image cascade to speed up the inference. BiSeNet [47] 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 works are based on either reinforcement learning [52, 17] or evolutionary algorithm [37, 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 a parent network from which each sub-network can inherit the weights. They can be roughly divided into cell-based and layer-based methods according to the type of search space. For cell-based methods, ENAS [36] proposes a parameter sharing strategy among sub-networks, and DARTS [25] relaxes the discrete architecture distribution as continuous deterministic weights, such that they could be optimized with gradient descent. SNAS [44] proposes novel search gradients that

train neural operation parameters and architecture distribution parameters in the same round of back-propagation. What's more, there are also some excellent works [10, 32] to reduce the difficulty of optimization by decreasing gradually the size of search space. For layer-based methods, FBNet [42], MnasNet [39], ProxylessNAS [5] use a multi-objective search approach that optimizes both accuracy and real-world latency.

In the field of semantic segmentation, DPC [6] is the pioneer work by introducing meta-learning techniques into the network search problem. Auto-Deeplab [24] searches cell structures and the downsampling strategy together in the same round. More recently, CAS [49] searches an architecture with customized resource constraint and a multi-scale module which has been widely used in semantic segmentation field. And [31] over-parameterises the architecture during the training via a set of auxiliary cells using reinforcement learning. Recently, NAS also has been used in object detection, such as NAS-FPN [16], DetNAS [48] and Auto-FPN [45].

Graph Convolution Network Convolutional neural networks on graph-structure data is an emerging topic in deep learning research. Kipf [20] presents 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 propagation. After that, Graph Convolution Networks (GCNs) [20] is widely used in many domains, such as video classification [41] and action recognition [46]. In this paper, we apply the GCNs to
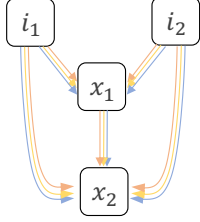
Figure 4. The structure of cell in our GAS. Each colored edge represents one candidate operation.

model the relationship of adjacent cells in network architecture search.

3. Methods

As shown in Figure 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 obtain a network with computational efficiency. This search problem can be formulated as:

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

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 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 Figure 3(a), the whole backbone takes an 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 multiscale context for the final prediction.

A cell is a directed acyclic graph (DAG) as shown in Figure 4. Each cell has two input nodes i_1 and i_2, N ordered intermediate nodes, denoted by $\mathcal{N} = \{x_1, ..., x_N\}$, and an output node which outputs the concatenation of all intermediate nodes $\mathcal{N}$. Each node represents the latent representation (*e.g.* feature map) in the network, and each directed edge in this DAG represents an candidate operation (*e.g.* conv, pooling).

The number of intermediate nodes N is 2 in our work. Each intermediate node takes all its previous nodes as input. In this way, x_1 has two inputs $I_1 = \{i_1, i_2\}$ and node x_2 takes $I_2 = \{i_1, i_2, x_1\}$ as inputs. The intermediate nodes x_i can be calculated by:

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

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

To search the selected operation $\widetilde{O}_{h,i}$, the search space is represented with a set of one-hot random variables from a fully factorizable joint distribution $p(Z)$ [44]. Concretely, each edge is associated with a one-hot random variable which is multiplied as a mask to the all possible operations $O_{h,i} = (o_{h,i}^1, o_{h,i}^2, ..., o_{h,i}^M)$ in this edge. We denote the one-hot random variable as $Z_{h,i} = (z_{h,i}^1, z_{h,i}^2, ..., z_{h,i}^M)$ where M is the number of candidate operations. The intermediate nodes during search process in such way are:

$$x_i = \sum_{c \in I_i} \widetilde{O}_{h,i}(c) = \sum_{c \in I_i} \sum_{m=1}^{M} z_{h,i}^m o_{h,i}^m(c) \qquad (3)$$

To make $P(Z)$ differentiable, reparameterization [27] is used to relax the discrete architecture distribution to be continuous:

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

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 the temperature of softmax.

For the set of candidate operations O, we only use the following 8 kinds of operations to better balance the speed and performance:

- 3×3 max pooling
- 3×3 conv
- 3×3 separable conv
- 3×3 dilated separable conv (dilation=2)
- 3×3 dilated separable conv (dilation=4)
- 3×3 dilated separable conv (dilation=8)
- skip connection
- zero operation

3.2. GCN-Guided Module

With cell independent to each other, the inter-cell relationship becomes very 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 Figure 3(b). Inspired by [41], 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 delivered to current cell by reasoning on this graph.

As stated in Section 3.1, let α_k represents the architecture parameter matrix for the cell k, and the dimension of α_k is $p \times q$ where p and q represents the number of edges and the number of candidate operations respectively. Same for cell k, the architecture parameter α_{k-1} for cell $k-1$ also is a $p \times q$ matrix. To fuse the architecture parameter information of previous cell $k-1$ into the current cell

k and generate the updated α'_k, we model the information propagation between cell $k-1$ and cell k as follows:

$$\alpha'_k = \alpha_k + \gamma \Phi_2(G(\Phi_1(\alpha_{k-1}), Adj)) \quad (5)$$

where Adj represents the adjacency matrix of the reasoning graph between cells k and $k-1$, and the function G denotes the Graph Convolution Networks (GCNs) [20] to perform reasoning on the graph. Φ_1 and Φ_2 are two different transformations by 1D convolution. Specifically, Φ_1 maps the original architecture parameter to embedding space and Φ_2 transfers it back into the source space after the GCN reasoning. γ controls the fusion of two kinds of architecture parameter information.

For the function G, we construct the reasoning graph between cell $k-1$ and cell k by their similarity. Given a edge in cell k, we calculate the similarity between this edge and all other edges in cell $k-1$ and a softmax function is used for normalization. Therefore, the adjacency matrix Adj of the graph between two adjacent cells k and $k-1$ can be established by:

$$Adj = Softmax(\phi_1(\alpha_k) * \phi_2(\alpha_{k-1})^T) \quad (6)$$

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

Based on this adjacency matrix Adj, we use the GCNs to perform information propagation on the graph as shown in Equation 7. A residual connection is added to each layer of GCNs. 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 equivalent to performing message propagation on the graph.

$$G(\Phi_1(\alpha_{k-1}), Adj) = Adj\Phi_1(\alpha_{k-1})W^g_{k-1} + \Phi_1(\alpha_{k-1}) \quad (7)$$

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

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

3.3. Latency-Oriented Optimization

To obtain a real-time semantic segmentation network, we take the real-world latency into consideration during the search process, which orients the search process toward the direction to find an optimal lightweight model. Specifically, we create a GPU-latency lookup table [5, 42, 49, 39] 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 pre-built lookup table. In this way, the total latency for cell k is accumulated as:

$$lat_k = \sum_{h,i} \sum_{m=1}^{M} z^m_{h,i} lat^m_{h,i} \quad (8)$$

where $z^m_{h,i}$ is the softened one-hot random variable as stated in Section 3.1. Given an architecture a, the total latency cost is estimated as:

$$LAT(a) = \sum_{k=0}^{K} lat_k \quad (9)$$

where K refers to the number of cells in architecture a. The latency for each operation $lat^m_{h,i}$ is a constant and thus total latency loss is differentiable with respect to the architecture parameter $\alpha_{h,i}$. The total loss function is designed as follows:

$$L(a, w) = CE(a, w_a) + \beta \log(LAT(a)) \quad (10)$$

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. The architecture parameter α and the weight w are optimized in the same round of backpropagation.

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 other works on two standard benchmarks. Secondly, we perform the ablation study for the GCN-Guided Module and latency optimization settings, and close with an 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 the Cityscapes [13] and CamVid [4] datasets. Cityscapes [13] is a public released dataset for urban scene understanding. It contains 5,000 high quality pixel-level fine annotated images (2975, 500, and 1525 for the training, validation, and test sets, respectively) with size 1024×2048 collected from 50 cities. The dense annotation contains 30 common classes and 19 of them are used in training and testing. 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 960×720 and 11 semantic categories.

Evaluation Metrics For evaluation, we use mean of class-wise intersection over union (mIoU), network forward time (Latency), and Frames Per Second (FPS) as the evaluation metrics.

4.2. Implementation Details

We conduct all experiments using Pytorch 0.4 [35] on a workstation, and the inference time in all experiments is reported on one Nvidia Titan Xp GPU.

The whole pipeline contains three sequential steps: search, pretraining and finetuning. It starts with the search progress on the target dataset and obtains the light-weight architecture according to the optimized α followed by the ImageNet [14] pretraining, and this pretrained model is subsequently finetuned on the specific dataset for 200 epochs.

In search process, the architecture contains 14 cells and each cell has $N = 2$ intermediate 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 parameters α are optimized by Adam, with initial learning rate 0.001, $\beta = (0.5, 0.999)$ 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. The search time cost on Cityscapes takes approximately 10 hours with 16 TitanXP GPU.

For finetuning details, we train the network with mini-batch 8 and SGD optimizer with 'poly' scheduler that decay learning rate from 0.01 to zero. Following [43], the online bootstrapping strategy is applied to the finetuning 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 with a fixed size for training.

For the GCN-guided Module, we use one Graph Convolution Network (GCN) [20] between two adjacent cells, and each GCN contains one layer of graph convolutions. The kernels size of the GCN parameters W in equation 7 is 64×64. We set the γ as 0.5 in equation 5 in our experiments.

4.3. Real-time Semantic Segmentation Results

In this part, we compare the model searched by GAS with other existing real-time segmentation methods on semantic segmentation datasets. The inference time is measured on an Nvidia Titan Xp GPU and the speed of other methods reported on Titan Xp GPU in CAS [49] are used for fair comparison. Moreover, the speed is remeasured on Titan Xp if the origin paper reports it on different GPU and is not mentioned in CAS [49].

Results on Cityscapes. We evaluate the network searched by GAS on the Cityscapes test set. The validation set is added to train the network before submitting to

Method	Input Size	mIoU (%)	Latency(ms)	FPS
FCN-8S [26]	512x1024	65.3	227.23	4.4
PSPNet [51]	713x713	81.2	1288.0	0.78
DeepLabV3* [7]	769x769	81.3	769.23	1.3
SegNet [1]	640x360	57.0	30.3	33
ENet [34]	640x360	58.3	12.7	78.4
SQ [40]	1024x2048	59.8	46.0	21.7
ICNet [50]	1024x2048	69.5	26.5	37.7
SwiftNet [33]	1024x2048	75.1	26.2	38.1
ESPNet [28]	1024x512	60.3	8.2	121.7
BiSeNet [47]	768x1536	68.4	9.52	105.8
DFANet A§ [22]	1024x1024	71.3	10.0	100.0
DFANet A† [22] [1]	1024x1024	71.3	19.01	52.6
CAS [49]	768x1536	70.5	9.25	108.0
CAS* [49]	768x1536	72.3	9.25	108.0
GAS	769x1537	71.8	9.22	108.4
GAS*	769x1537	73.5	9.22	108.4

Table 1. Comparing results on the Cityscapes test set. 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.

Cityscapes online server. Following [47, 49], GAS takes as an input image with size 769×1537 that is resized from origin image size 1024×2048. Overall, our GAS gets the best performance among all methods with the speed of 108.4 FPS. With only fine data and without any evaluation tricks, our GAS yields 71.8% mIoU which is the state-of-the-art trade-off between performance and speed for real-time semantic segmentation. GAS achieves 73.5% when the coarse data is added into the training set. The full comparison results are shown in Table 1. Compared to BiSeNet [47] and CAS [49] that have comparable speed with us, our GAS surpasses them along multiple performance points with 3.4% and 1.3%, respectively. Compared to other methods such as SegNet [1], ENet [34], SQ [40] and ICNet [50], our method achieves significant improvement in speed while getting performance gain over them about 14.8%, 13.5%, 12.0%, 2.3%, respectively.

Results on CamVid. We directly transfer the network searched on Cityscapes to Camvid to verify the transferability of GAS. Table 2 shows the comparison results with other methods. With input size 720×960, GAS achieves the 72.8% mIoU with 148.0 FPS which is also the state-of-the-art trade-off between performance and speed, which demonstrates the superior transferability of GAS.

---

[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 the optimized 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 is optimized correctly like DFANet.

Method	mIoU (%)	Latency(ms)	FPS
SegNet [1]	55.6	34.01	29.4
ENet [34]	51.3	16.33	61.2
ICNet [50]	67.1	28.98	34.5
BiSeNet [47]	65.6	-	-
DFANet A [22]	64.7	8.33	120
CAS [49]	71.2	5.92	169
GAS	72.8	6.53	153.1

Table 2. Results on the CamVid test set with resulotion 960 × 720. "-" indicates the corresponding result is not provided by the methods.

4.4. Ablation Study

To verify the effectiveness of each component in our framework, extensive ablation studies for the GCN-Guided Module and the latency loss are performed. In addition, we also give some insights about the role of GCN-Guided Module in the search process.

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 effectiveness of the GGM, we conduct a series of experiments with different strategies: a) network stacked by shared cell; b) network stacked by independent cell; c) based on strategy-b, using fully connected layer to infer the relationship between cells; d) based on strategy-b, using GGM to infer the relationship between cells. Experimental results are shown in Figure 5. The performance reported here is the average mIoU over five repeated experiments on the Cityscapes validation set with latency loss weight $\beta = 0.005$. The numbers below the horizontal axis are the average model size of five architectures (*e.g.* 2.18M) and the purple line is the variance of mIoU for each strategy. Overall, with only independent cell, the performance degrades a lot due to the enlarged search space which makes optimization more difficult. This performance drop is mitigated by adding communication mechanism between cells. Especially, our GCN-Guided Module can bring about 3% performance improvement compared to the fully-connected mechanism (*i.e.* setting (c)).
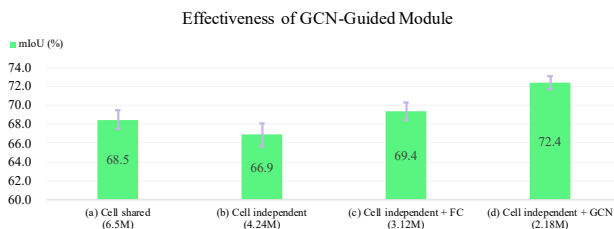

Figure 5. Ablation study for the effectiveness of GCN-Guided Module on Cityscapes validation dataset. **Best viewed in color.**

Comparison against Random Search As discussed in [23], random search is a competitive baseline for hyper-parameter optimization. To further verify the effectiveness of GCN-Guided Module, we randomly sample ten architectures from the search space and evaluate them on the Cityscapes validation set with ImageNet pretrained. Specifically, we try two types of random settings in our experiments: a) fully random search without any constraint; b) randomly select the networks that meet the speed requirement about 108 FPS from the search space. The results are shown in Table 3, in which each value is the average result of ten random architectures. In summary, the network searched by GAS can achieve an excellent trade-off between performance and latency, while random search will result in high overhead without any latency constraint or low performance with latency constraint.

Methods	mIoU (%)	FPS
GAS	72.3	108.2
Random setting (a)	69.6	61.2
Random setting (b)	65.8	105.6

Table 3. Comparison to random search on the Cityscapes validation set.

Dimension Selection The dimension selection of GCN weight W in Equation 7 is also important, thus we conduct experiments with different GCN weight dimensions (denoted by d). Experimental results are shown in Table 4 in which the values are the average mIoU over five repeated experiments on the Cityscapes validation set with latency loss weight $\beta = 0.005$. Experimental result indicates that GAS achieves the best performance when d = 64.

Methods	mIoU (%)	FPS
GCN with d = 16	71.6	108.6
GCN with d = 32	71.8	102.2
GCN with d = 64	72.4	108.4
GCN with d = 128	72.1	104.1
GCN with d = 256	71.5	111.2

Table 4. Ablation study for different GCN weight dimensions of the GCN-Guided Module.

Reasoning Graph For GCN-Guided Module, in addition to the way described in Section 3.2, we also try another way to construct the reasoning graph. Specifically, we treat each candidate operation in a cell as a node in the reasoning graph. Given the architecture parameter α_k for cell k with dimension $p \times q$, we first flatten the α_k and α_{k-1} to the one dimensional vector α'_k and α'_{k-1}, and then perform matrix multiplication to get adjacent matrix $Adj = \alpha'_k(\alpha'_{k-1})^T$. Different from the "edge-similarity" reasoning graph in Section 3.2, we call this graph "operation-identity" reasoning graph. We conduct the comparison experiment for two

types of graphs on the Cityscapes validation set under the same latency loss weight $\beta = 0.005$, the comparison results are shown in Table 5.

Reasoning Graph	mIoU (%)	FPS
Edge-similarity	72.4	108.4
Operation-identity	70.9	102.2

Table 5. The comparison results for reasoning graph for edges and operations.

Intuitively, the "operation-identity" way provides more fine-grained information about operation selection for other cells, while it also breaks the overall properties of an edge, and thus doesn't consider the other operation information at the same edge when making decision. After visualizing the network, we also found that the "operation-identity" reasoning graph tends to make cell select the same operation for all edge, which increases the difficulty of trade-off between performance and latency. This can also be verified from result in Table 5. So we choose the "edge-similarity" way to construct the reasoning graph as described in Section 3.2.

Network Visualization We illustrate the network structure searched by GAS in the supplementary material. 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.

4.4.2 Effectiveness of the Latency Constraint

As mentioned above, GAS provides the ability to flexibly achieve a superior trade-off between the performance and speed with the latency-oriented optimization. We conducted a series of experiments with different loss weight β in Equation 10. Figure 6 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 slightly falling. But when β increases from 0.005 to 0.05, the performance drops quickly while the latency decline is fairly limited. Thus 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) to search a light-
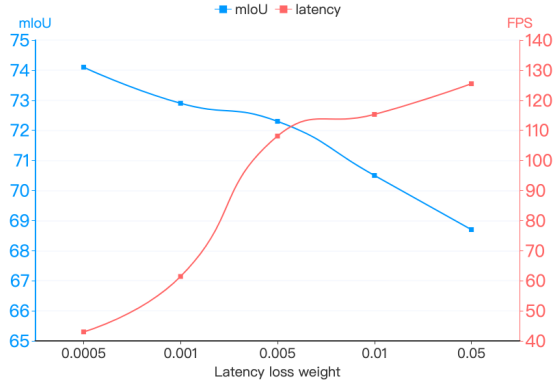

Figure 6. The validation accuracy on Cityscapes dataset for different latency constraint. **Best viewed in color.**

weight network, we do not allow the cell structures 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 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 to 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 from the existing NAS approaches that stack 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 performs much better 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.

References

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

[2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *ICML*, pages 549–558, 2018.

[3] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. *arXiv:1708.05344*, 2017.

[4] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV (1)*, pages 44–57, 2008.

[5] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv:1812.00332*, 2018.

[6] Liang-Chieh Chen, Maxwell D. Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jonathon Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *NeurIPS*, pages 8713–8724, 2018.

[7] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.

[8] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, pages 833–851, 2018.

[9] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE trans. PAMI*, 40(4):834–848, 2018.

[10] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. *arXiv:1904.12760*, 2019.

[11] Yukang Chen, Qian Zhang, Chang Huang, Lisen Mu, Gaofeng Meng, and Xinggang Wang. Reinforced evolutionary neural architecture search. *CoRR*, abs/1808.00193, 2018.

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

[13] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, pages 3213–3223, 2016.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009.

[15] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 111(1):98–136, 2015.

[16] Golnaz Ghiasi, Tsung-Yi Lin, Ruoming Pang, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, 2019.

[17] Minghao Guo, Zhao Zhong, Wei Wu, Dahua Lin, and Junjie Yan. IRLAS: inverse reinforcement learning for architecture search. *CoRR*, abs/1812.05285, 2018.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[19] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CVPR*, pages 1–9, 2016.

[20] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv:1609.02907*, 2016.

[21] Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. Dynamic evaluation of neural sequence models. *arXiv:1709.07432*, 2017.

[22] Hanchao Li, Pengfei Xiong, Haoqiang Fan, and Jian Sun. Dfanet: Deep feature aggregation for real-time semantic segmentation. In *CVPR*, pages 9522–9531, 2019.

[23] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*, page 129, 2019.

[24] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, 2019.

[25] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *ICLR*, 2019.

[26] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[27] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv:1611.00712*, 2016.

[28] Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *ECCV*, pages 552–568, 2018.

[29] Marvin Minsky. *The Society of Mind*. Simon & Schuster, 1988.

[30] Renato Negrinho and Geoff Gordon. Deeparchitect: Automatically designing and training deep architectures. *arXiv:1704.08792*, 2017.

[31] Vladimir Nekrasov, Hao Chen, Chunhua Shen, and Ian Reid. Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In *CVPR*, pages 9126–9135, 2019.

[32] Asaf Noy, Niv Nayman, Tal Ridnik, Nadav Zamir, Sivan Doveh, Itamar Friedman, Raja Giryes, and Lihi Zelnik-Manor. Asap: Architecture search, anneal and prune. *arXiv:1904.04123*, 2019.

[33] Marin Orsic, Ivan Kreso, Petra Bevandic, and Sinisa Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. *CVPR*, pages 12599–12608, 2019.

[34] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Euge-nio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv:1606.02147*, 2016.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Rai-son, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NIPS*, 2019.

[36] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, pages 4092–4101, 2018.

[37] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. *CoRR*, abs/1802.01548, 2018.

[38] Karen Simonyan and Andrew Zisserman. Very deep convo-lutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[39] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnas-net: Platform-aware neural architecture search for mobile. In *CVPR*, pages 2820–2828, 2019.

[40] Michael Treml, José Arjona-Medina, Thomas Unterthiner, Rupesh Durgesh, Felix Friedmann, Peter Schuberth, An-dreas Mayr, Martin Heusel, Markus Hofmarcher, Michael Widrich, et al. Speeding up semantic segmentation for au-tonomous driving. In *MLITS, NIPS Workshop*, volume 2, page 7, 2016.

[41] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *ECCV*, pages 399–417, 2018.

[42] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient con-vnet design via differentiable neural architecture search. In *CVPR*, pages 10734–10742, 2019.

[43] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. High-performance semantic segmentation using very deep fully convolutional networks. *CoRR*, abs/1604.04339, 2016.

[44] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *ICLR*, 2019.

[45] Hang Xu, Lewei Yao, Wei Zhang, Xiaodan Liang, and Zhen-guo Li. Auto-fpn: Automatic network architecture adap-tation for object detection beyond classification. In *ICCV*, 2019.

[46] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial tempo-ral graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018.

[47] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmenta-tion network for real-time semantic segmentation. In *ECCV*, pages 334–349, 2018.

[48] Xiangyu Zhang Gaofeng Meng Xinyu Xiao Jian Sun Yukang Chen, Tong Yang. Detnas: Backbone search for ob-ject detection. In *NeurIPS*, 2019.

[49] Yiheng Zhang, Zhaofan Qiu, Jingen Liu, Ting Yao, Dong Liu, and Tao Mei. Customizable architecture search for se-mantic segmentation. In *CVPR*, pages 11641–11650, 2019.

[50] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *ECCV*, pages 405–420, 2018.

[51] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, pages 2881–2890, 2017.

[52] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

[53] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, pages 8697–8710, 2018.