

Local-to-Global Information Communication for Real-Time Semantic Segmentation Network Search

Guangliang Cheng*, Peng Sun*, Ting-Bing Xu, Shuchang Lyu and Peiwen Lin[✉]

Abstract—Neural Architecture Search (NAS) has shown great potentials in automatically designing neural network architectures for real-time semantic segmentation. Unlike previous works that utilize a simplified search space with cell-sharing way, we introduce a new search space where a lightweight model can be more effectively searched by replacing the cell-sharing manner with cell-independent one. Based on this, the communication of local to global information is achieved through two well-designed modules. For local information exchange, a graph convolutional network (GCN) guided module is seamlessly integrated as a communication deliver between cells. For global information aggregation, we propose a novel dense-connected fusion module (cell) which aggregates long-range multi-level features in the network automatically. In addition, a latency-oriented constraint is endowed into the search process to balance the accuracy and latency. We name the proposed framework as Local-to-Global Information Communication Network Search (LGCNet). Extensive experiments on Cityscapes and CamVid datasets demonstrate that LGCNet achieves the new state-of-the-art trade-off between accuracy and speed. In particular, on Cityscapes dataset, LGCNet achieves the new best performance of 74.0% mIoU with the speed of 115.2 FPS on Titan Xp.

Index Terms—Neural Architecture Search, Real-Time Semantic Segmentation, Graph Convolutional Network.

I. INTRODUCTION

SEMANTIC segmentation [1]–[6] has been a fundamental vision task that aims at predicting pixel-level semantic categories for images. With the recent advances in deep learning technology [7]–[12], many works focus on the sophisticated model design regarding depth, width and attention mechanism to pursue higher accuracy, which involves many time-consuming operations. Although these methods achieve impressive results on the public semantic segmentation benchmarks [13]–[15], they are difficult to be deployed on the resource-constrained applications, such as the auto-driving vehicles and the navigation robots, which require high computational efficiency without incurring accuracy drop. To overcome this problem, researchers have designed some low-computation CNN models to harvest the satisfactory segmentation accuracy. ICNet, ENet and SegNet [16]–[18] reduce the computational cost by incorporating some specifically designed modules with reduced input size or filter numbers.

Guangliang Cheng is with SenseTime Research Group of Beijing, and Department of Computer Science, in the University of Liverpool.

Peng Sun is with Bytedance Data Group

Ting-Bing Xu and Shuchang Lyu are with Beihang University

Peiwen Lin is with SenseTime Research Group of Beijing

* The first two authors contributed equally to this paper.

✉ Corresponding author: Peiwen Lin

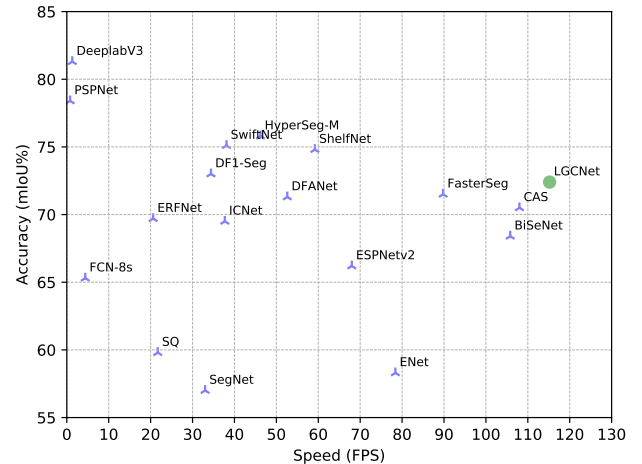


Fig. 1: The inference speed (FPS) and mIoU performance among different networks on the Cityscapes test set. Our LGCNet achieves the state-of-the-art trade-off between the speed and performance.

BiSeNet [19], [20] decouples the context and spatial information with bilateral path, and then fuses them to achieve a satisfactory trade-off between accuracy and latency. DFANet [21] utilizes lightweight depth-wise separable convolutions and remedies its accuracy drop by incorporating an aggregation module. Although achieving remarkable results, these human-designed strategies usually require expertise in architecture design through enormous trial and error to carefully balance the accuracy and resource-efficiency. Such human-designed process need to be redone when the hardware setting changes, which further increases the difficulty in applying to the actual applications.

Different from manually designed architectures, network architecture search methods have drawn extensive attention [22]–[30] and achieved remarkable performance. They mainly focus on the block-based (also called as cell) search for the backbone and utilize the multi-scale module such as ASPP [31] to fuse the global information. For the building block, some works directly design the search space on the targeted platform. CAS [32] searches for two cell types (normal and reduction cell) and then stacks the identical cells repeatedly to form a network. FBNet [33] and SqueezeNet [34] search the hyper-parameters including number of blocks, channel numbers of each layer for the network based on the efficient blocks in human-designed network such as ResNet and MobileNet

[8], [35]. For multi-scale module, AutoRTNet [36] and CAS [32] automatically aggregate features at different levels with a multi-scale fusion cell, and [29] utilizes a recurrent neural network (i.e. controller) to decide which layer and what kind of operations will be employed.

Although the searched building block and multi-scale cell have achieved satisfactory performance with above methods, some indispensable aspects for a remarkable real-time segmentation network are ignored:

1) Difficult to achieve a good trade-off between latency and accuracy due to the limited cell diversity with the identical cell. As shown in Figure 2 (a), the cell is prone to learn a complicated structure to achieve high performance without any resource constraint, and the network stacked with it will result in high latency. When a low-computation constraint is applied, the cell structure tends to be over-simplified as shown in Figure 2 (b), which may not achieve satisfactory performance. We thus modify the cell setting from cell-sharing manner to cell-independent one, which can be flexibly stacked to form a lightweight network with cell diversity as shown in Figure 2 (c).

2) Lack of local information exchange. Above methods only consider the global information fusion using the multi-scale module during the search progress, while the local information exchange between adjacent cells are very important to achieve a good trade-off between latency and accuracy because different cells can be treated as multiple agencies, whose achievement of social welfare may require information exchange between them inspired by [37].

3) Global information fusion. CAS [32] and [29] search a multi-scale module in which only the short-range features from stride=8 to stride=16 are taken into account to reduce the inference time cost, but it would be helpful to improve the accuracy if the lower level information in stride=4 could be incorporated since some boundary clues from this feature are essential for achieving more fine-grained segmentation results.

Based on the independent cell mechanism, it's worth further exploring how to effectively establish the local-to-global information communication among cells of the whole network during the search process. With this aim in mind, in this article we present a new lightweight segmentation network search method by integrating the local information exchange and global information fusion together from the local and global aspects. First, to address the local exchange, we utilize a Graph Convolutional Network [38] guided module (GGM) as the local information exchange deliverer among cells, through which the information of each cell can be propagated to the next adjacent cell. Second, a dense-connected fusion cell is proposed to perform global information aggregation, in which the long-range multi-level features (low-level spatial details and high-level semantic context) in the network can be effectively exploited and fused. A latency-oriented constraint for target computing platform is embedded into the search process, thus the searched model can achieve better trade-off between the accuracy and latency.

To verify the effectiveness of the proposed search method, extensive experiments and detailed analysis are provided on two public segmentation datasets, i.e., the standard Cityscapes

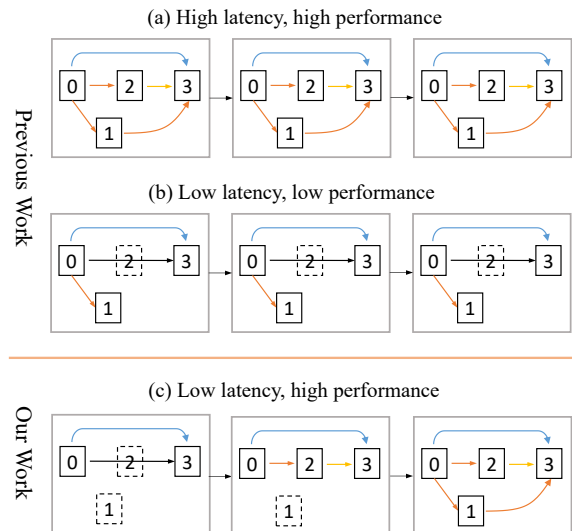


Fig. 2: (a) The network stacked with complicated cells results in high latency and high performance. (b) The network stacked with simple cells leads to low latency and low performance. (c) The cell diversity strategy, i.e., each cell possesses its own independent structure, can flexibly construct the high accuracy lightweight network.

[13] and CamVid [15] benchmarks. The experimental results show that our method achieves much better performance and faster inference speed than GAS [39], which is a prior state-of-the-art trade-off between accuracy and speed. Compared to other real-time methods, our method also locates in the top-right area in Figure 1, which indicates that our method obtains the new state-of-the-art trade-off between accuracy and latency.

This article is an extension of our conference version [39], and the major contributions can be summarized as follows.

- We extend previous segmentation network architecture search from only graph-guided local information exchange between adjacent cells to the whole information communication of local perception and global fusion.
- The global information aggregation is implemented via the dense-connected fusion cell, which aggregates multi-level features automatically to effectively fuse the low-level spatial details and high-level semantic context.
- More detailed algorithm descriptions, deeper analyses, and more comparison experiments are presented in this paper to demonstrate the effectiveness of the proposed method for the lightweight segmentation network architecture search.
- The lightweight segmentation model via the proposed search method is customizable in practical applications. Notably, it achieves 74.0% mIoU on the Cityscapes test set and 115.2 FPS on NVIDIA Titan Xp for one 769×1537 image.

The remainder of this article is organized as follows. Section II reviews the related works; Section III describes the details of the proposed method; Section IV presents experimental results and discussions, and Section V draws concluding remarks.

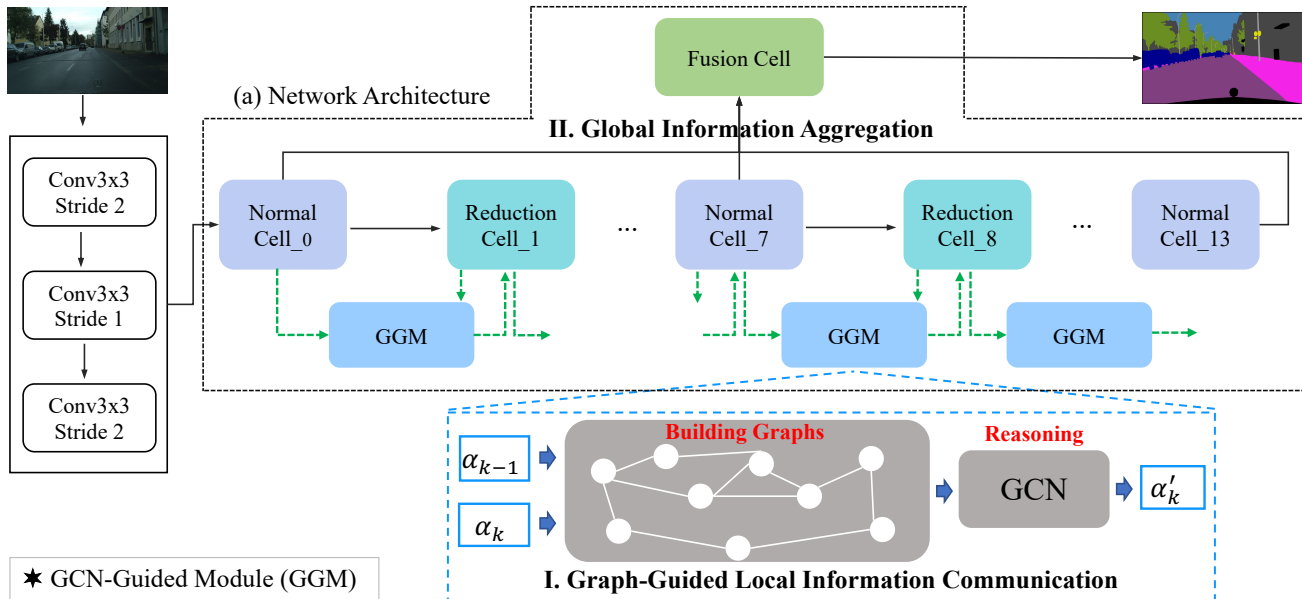


Fig. 3: Illustration of our local-to-global information communication network. In reduction cells, all the operations adjacent to the input nodes are of stride two. (a) The backbone network, which is stacked by a series of independent cells. (I) The GCN-Guided Module (GGM), which performs *local* information exchange between adjacent cells. α_k and α_{k-1} represent the architecture parameters for cell k and cell $k - 1$, respectively. α'_k is the updated architecture parameters by GGM for cell k . The dotted lines indicate that GGM is only utilized in the search progress. (II) The Dense-Connected Fusion Cell, which aggregates the multi-scale feature to capture *global* context information.

II. RELATED WORK

a) *Semantic Segmentation Methods*: FCN [1] is the pioneer work to achieve the end-to-end semantic segmentation task. Since then, to improve the segmentation performance, some remarkable works have utilized various heavy backbones [7]–[10] or well-designed network modules to capture multi-scale context information [2], [4], [31]. These outstanding works are designed for high-performance segmentation, which is inapplicable to the real-time applications. In terms of efficient segmentation methods, there are two mainstreams. One is to employ the human-designed and relatively light backbone (e.g. ENet [18]) or introduce some efficient operations (e.g. 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 based on a multi-branch algorithm that consists of more than one path. For example, ICNet [16] proposes to use the multi-scale image cascade to speed up the inference. BiSeNet [19] decouples the extraction for spatial and context information using two paths.

b) *Information Communication for Semantic Segmentation*: Capturing local details and global context information in images and properly combining them is critical for semantic segmentation performance. A post-process module of conditional random field (CRF) [40] is proposed to improve the ability of capturing local details. After that, the dilated convolution [3] is utilized to remedy the resolution loss, and the ASPP [3] and PSPNet [2] modules enable efficient combination of local detail information and global context information based on the multi-scale features.

c) *Neural Architecture Search*: Neural Architecture Search (NAS) aims at automatically searching network architectures. Most existing architecture search works are based on either reinforcement learning [41], [42] or evolutionary algorithm [43], [44]. Though they can achieve satisfactory performance, they need thousands of GPU hours. To solve this time-consuming problem, one-shot methods [45], [46] have been developed to train 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 [26] proposes a parameter sharing strategy among sub-networks, and DARTS [22] relaxes the discrete architecture distribution as continuous deterministic weights, such that they could be optimized with gradient descent. SNAS [28] proposes novel search gradients that train neural operation parameters and architecture distribution parameters in the same round of back-propagation. Additionally, there are some fantastic works [47], [48] that gradually lower the size of the search space in order to lessen the complexity of optimization. For the layer-based methods, FBNet [33], MnasNet [49], ProxylessNAS [27] use a multi-objective search approach that optimizes both accuracy and real-world latency.

d) *NAS for Segmentation*: In the field of semantic segmentation, DPC [50] is the pioneer work by introducing meta-learning techniques into the network architecture search problem. Auto-Deeplab [51] searches cell structures and the downsampling strategy together in the same round. More recently, CAS [32] searches an architecture with customized resource constraint and a multi-scale module that has been

widely used in the semantic segmentation field. [29] over-parameterizes the architecture during the training via a set of auxiliary cells using reinforcement learning.

e) *Graph Convolutional Network*: Convolutional neural networks on graph-structure data is an emerging topic in deep learning research. Kipf [38] 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 Convolutional Networks (GCNs) [38] is widely used in many domains, such as video classification [52] and action recognition [53]. Recently, Zhang et al. [54] propose the Graph HyperNetwork to amortize the search cost: given an architecture, it directly generates the weights by running inference on a graph neural network. In this paper, we apply the GCNs to model the relationship of adjacent cells in network architecture search. Specifically it takes the combination of local and global information into consideration during the search progress through the communication between adjacent cells and dense-connected fusion.

III. METHODS

In this section, we first formulate the search problem and the overall framework, and then introduce the cell architecture search. Later we describe the proposed local to global information communication method, which consists of GCN-Guided Module (GGM) for local information exchange between adjacent cells, dense-connected fusion module for global information aggregation and latency-oriented search for a lightweight model, respectively.

A. Problem Formulation

a) *Overview*: As shown in Figure 3(a), the input image is firstly processed by three convolutional layers followed by a series of independent cells, then the searchable dense-connected fusion cell fuses long-range multi-scale features for producing the representation feature by considering the local detail information and global context information before the pixel level classification. The GCN-Guided module is embedded into the search framework to bridge the information between adjacent cells. Then the search process is directed towards the goal of a lightweight network by the latency-oriented optimization loss.

The overall loss function in the training stage can be formulated as:

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

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 latency.

b) *Notation Table*: As shown in Table I, we build the following notation table for clear representation.

TABLE I: The symbols and notations used in this paper.

N	\triangleq	the intermediate node number of the cell
M	\triangleq	the number of candidate operations
$\tilde{O}_{h,i}$	\triangleq	the selected operation at edge (h, i)
$Z_{h,i}$	\triangleq	the one-hot random variable at edge (h, i)
$O_{h,i}$	\triangleq	all possible operations at edge (h, i)
$\alpha_{h,i}$	\triangleq	the architecture parameter at edge (h, i)
α_k	\triangleq	the architecture parameter matrix of cell k
$lat_{h,i}^m$	\triangleq	the latency cost of candidate operation m at edge (h, i)

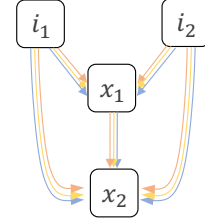


Fig. 4: The structure of cell in our LGCNet. Each colored edge represents one candidate operation.

B. Preliminaries: Cell Architecture Search

A cell is the basic component of a network, which 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 $\mathcal{N} = \{x_1, \dots, x_N\}$, and an output node that 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 a 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:

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

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

To search the selected operation $\tilde{O}_{h,i}$, the search space is represented with a set of one-hot random variables from a fully factorizable joint distribution $p(Z)$ [28] and is optimized by the single level optimization as opposed to bilevel optimization employed by DARTS [22]. Concretely, each edge is associated with a one-hot random variable which is multiplied as a mask to the all possible operations $O_{h,i} = (o_{h,i}^1, o_{h,i}^2, \dots, o_{h,i}^M)$ in this edge. We denote the one-hot random variable as $Z_{h,i} = (z_{h,i}^1, z_{h,i}^2, \dots, z_{h,i}^M)$ where M is the number of candidate operations. The intermediate nodes during search process in such way are:

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

An essential issue is how to make the one-hot random variables z (or $P(Z)$) differentiable in Equation 3. We use

reparameterization [55] to relax the discrete architecture distribution to be continuous:

$$Z_{h,i} = f_{\alpha_{h,i}}(G_{h,i}) = \text{Softmax}((\log \alpha_{h,i} + G_{h,i})/\lambda) \quad (4)$$

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.

To better balance the speed and performance, we only employ the following 8 types of operations in Table II as the set of candidate operations O :

TABLE II: The candidate operations for the cell-independent search.

3×3 max pooling
skip connection
3×3 conv
zero operation
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)

C. Local to Global Information Communication

a) *GCN-Guided Module for Local Information Exchange*: With cell independent mechanism, 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 (I). Inspired by [52], the GGM represents the communication between adjacent cells as a graph and performs reasoning on the graph for information exchange. For more through explanation on what role that GGM plays, please refer to the section *D* of Network Visualization and Analysis.

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 the previous cell can be delivered to the current cell by reasoning on this graph. As stated before, 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 as cell k , the architecture parameter α_{k-1} for cell $k-1$ is also 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:

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

where \mathbf{A} represents the adjacency matrix of the reasoning graph between cells k and $k-1$, and the function G denotes the Graph Convolutional Networks (GCNs) [38] to perform reasoning on the graph. Φ_1 and Φ_2 are two different transformations with two fully connected (FC) layers. Specifically, Φ_1 maps the original architecture parameter to the embedding space and Φ_2 transfers it back into the source space after

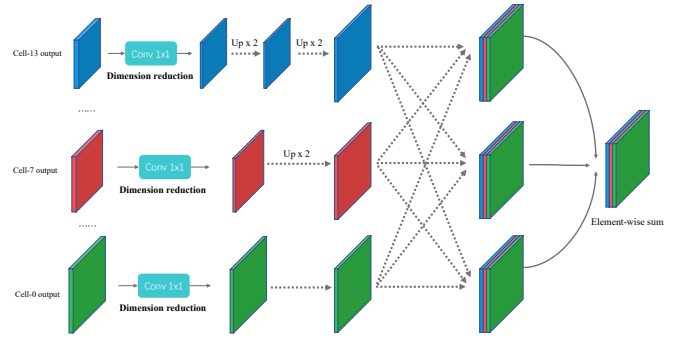


Fig. 5: Overview of the dense-connected fusion cell for automatic multi-scale feature aggregation. The fusion cell contains E edges (dotted arrows), each edge is equipped with some candidate operations. “Up $\times 2$ ” means upsampling operation.

the GCN reasoning. γ controls the fusion of two types of architecture parameter information.

For the function G , we construct the reasoning graph between cell $k-1$ and cell k by their similarity. Given an 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 \mathbf{A} of the graph between two adjacent cells k and $k-1$ can be established by:

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

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 \mathbf{A} is a $p \times p$ matrix.

Based on this adjacency matrix \mathbf{A} , 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.

$$G(\Phi_1(\alpha_{k-1}), \mathbf{A}) = \mathbf{A} \Phi_1(\alpha_{k-1}) W_{k-1}^g + \Phi_1(\alpha_{k-1}) \quad (7)$$

where the W_{k-1}^g denotes the GCNs weight with dimension $d \times d$. In fact, the weight W in GCN is embedded into the updated architecture parameter (i.e. α'), as shown in Equation 5, thus the weight W can be learned via back propagation.

The proposed GGM seamlessly integrates the graph convolutional network into neural architecture search, which can bridge the operation information between adjacent cells. Moreover, the GCN-guided Module is only used in the training phase, thus it introduces no extra parameters and computational cost to the searched model in the inference stage.

b) *Dense-Connected Fusion Cell for Global Information Aggregation*: As we mentioned before, the effective combination of local details and global context information is crucial for semantic segmentation, which is usually accomplished by

the multi-scale module such as ASPP in Deeplab and multi-scale cell in CAS. However, multi-scale cell in CAS only considers short range features from stride=8 and stride=16 to reduce time cost, while the ASPP also takes more latency due to multiple convolutional layers. We thus propose the dense-connected fusion cell to tackle these two issues. On one hand, our dense-connection cell aggregate long range features from stride=4 to stride=16 in the network, which can capture more fine-grained clues to further improve accuracy. On the other hand, we search operations for each feature scales, which can obtain more lightweight operations guided by the latency oriented loss (convolutional layer will obtain larger loss than lightweight operations such as pooling).

The structure of the proposed dense-connected fusion cell is shown in Figure 5. As shown in Figure 3, the fusion cell takes the outputs of cell-0, cell-7, and cell-13 with different resolutions as its inputs, thus the dense-connected fusion cell is designed to combine multi-scale features (i.e., low-level spatial details and high-level semantic context). The dense-connected fusion cell is designed as a directed acyclic graph consisting of M nodes and E edges, each node is a latent representation (i.e., feature map) and each directed edge is associated with some candidate operations. As shown in Figure 5, the number of channels of three inputs will be reduced to 48 through a 1×1 convolution layer to save time cost in subsequent process. Each edge is able to search specific operators from the search space, unless explicitly specified by “Up $\times 2$ ”, in which only deconvolutional layer and upsample layer (interpolate) can be searched. The middle feature maps of three branches are cross-branch connected densely by concatenation. Then the output of the dense-connected fusion cell is designed as the element-wise sum of the final feature maps from three branches. We use the same sampling and optimization process as the normal/reduction cell to optimize the fusion cell’s architecture parameter.

Given the candidate operation set, the dense-connected fusion cell also efficiently enlarges the receptive field of the network. For the operation set of the aggregation cell, we collect the following 10 kinds of operations in Table III:

TABLE III: The candidate operations for the aggregation cell.

1×1 conv
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)
3×3 dilated separable conv (dilation=12)
Global pooling with output size $1 \times 1 + 1 \times 1$ conv + upsampling
Global pooling with output size $2 \times 2 + 1 \times 1$ conv + upsampling
Global pooling with output size $5 \times 5 + 1 \times 1$ conv + upsampling

c) *Latency-Oriented Search*: To obtain a real-time semantic segmentation network, we take the real-world latency into consideration during the search process, which guides the search process towards the direction to find an optimal lightweight model. Specifically, we create a GPU-latency lookup table [27], [32], [33], [49] 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 $lat_{h,i}^m$ given by the pre-built lookup table. In this way, the total latency for cell k is accumulated as:

$$lat_k = \sum_{h,i} \sum_{m=1}^M z_{h,i}^m lat_{h,i}^m \quad (8)$$

where $z_{h,i}^m$ is the softened one-hot random variable as stated in Equation 3. 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 $lat_{h,i}^m$ is a constant and thus total latency loss is differentiable with respect to the architecture parameter $\alpha_{h,i}$.

The total loss function in LGCNet is designed as follows:

$$\begin{aligned} L(a, w) &= \min_{a \in \mathcal{A}} L_{val} + \beta * L_{lat} \\ &= \min_{a \in \mathcal{A}} CE(a, w_a) + \beta \log(LAT(a)) \end{aligned} \quad (10)$$

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 micro-second, 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 back-propagation.

IV. EXPERIMENTS

In this section, we conduct extensive experiments to verify the effectiveness of our LGCNet and compare the network searched by our method with other works on two standard benchmarks.

A. Experiment Setup

a) *Datasets and Evaluation Metrics*: In order to verify the effectiveness and robustness of our method, we evaluate it on the Cityscapes [13] and CamVid [15] datasets. Cityscapes [13] is a public released dataset for urban scene understanding. It contains 5000 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 [15] 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. For evaluation, we use the mean of class-wise intersection over union (mIoU), network forward time (Latency), and Frames Per Second (FPS) as the evaluation metrics.

b) Implementation Details: We conduct all the experiments using Pytorch 0.4 [56] on a workstation, and the inference time for all the 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 process on the target dataset and obtains the light-weight architecture according to the optimized α followed by the ImageNet [57] pretraining, and this pretrained model is subsequently finetuned on the specific dataset for 200 epochs.

In the search process, the architecture contains 14 cells and each cell has $N = 2$ intermediate nodes. With the consideration of speed and accuracy, we set the initial channel for the network as 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 it gradually decreases to the minimum value of 0.03. The search time cost on Cityscapes takes approximately 10 hours with 16 TitanXP GPU cards.

For finetuning details, we train the network with mini-batch 8 and SGD optimizer with ‘poly’ scheduler that the learning rate decays from 0.01 to zero. Following [58], the online bootstrapping strategy is applied to the finetuning process. For data augmentation, we use random flip and random resize with a 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 Convolutional Network (GCN) [38] between two adjacent cells, and each GCN contains one layer of graph convolutions. The kernel 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.

For the dense-connected fusion cell, we set the node number M and edge number E as 9 and 13, respectively.

B. Real-time Semantic Segmentation Results

In this part, we compare the model searched by LGCNet 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 [32] 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 [32].

a) Results on Cityscapes: We evaluate the network searched by LGCNet on the Cityscapes test set. The validation set is added to train the network before submitting it to Cityscapes online server. Following [19], [32], LGCNet takes an input image with size 769×1537 that is resized from origin size 1024×2048 . Overall, our LGCNet gets the best performance among all methods with the speed of 115.2 FPS. With only fine data and without any evaluation tricks, our LGCNet yields 72.4% mIoU which is the state-of-the-art trade-off between performance and speed for real-time semantic segmentation. LGCNet achieves 74.0% when

TABLE IV: Comparing results on the Cityscapes test set. Methods trained using both fine and coarse data are marked with *. The mark § represents that the speed is measured on TitanX, and the mark † denotes the speed is remeasured on Titan Xp. The mark τ indicates that TensorRT acceleration has been used.

Method	Input Size	mIoU (%)	Latency(ms)	FPS
FCN-8S [1]	512×1024	65.3	227.23	4.4
PSPNet [2]	713×713	78.4	1288.0	0.78
DeepLabV3* [4]	769×769	81.3	769.2	1.3
AutoDeepLab* † [51]	769×769	81.2	303.0	3.3
SegNet [17]	640×360	57.0	30.3	33
ENet [18]	640×360	58.3	12.7	78.4
SQ [59]	1024×2048	59.8	46.0	21.7
ICNet [16]	1024×2048	69.5	26.5	37.7
SwiftNet [60]	1024×2048	75.1	26.2	38.1
DF1-Seg [61]	768×1536	73.0	29.1	34.4
ESPNet [62]	1024×512	60.3	8.2	121.7
ESPNetV2 [63]	1024×512	66.2	14.7	68.0
ERFNet [64]	1024×512	69.7	48.5	20.6
BiSeNet-Xception39 [19]	768×1536	68.4	9.5	105.8
BiSeNet-Res18 [19]	768×1536	74.7	15.3	65.5
DFANet A§ [21]	1024×1024	71.3	10.0	100.0
DFANet A † [21] ¹	1024×1024	71.3	19.0	52.6
ShelfNet [65]	768×1536	74.8	16.9	59.2
CAS [32]	768×1536	70.5	9.25	108.0
CAS* [32]	768×1536	72.3	9.25	108.0
GAS [39]	769×1537	71.8	9.22	108.4
GAS* [39]	769×1537	73.5	9.22	108.4
FasterSeg [66] ²	1024×2048	71.5	11.13	89.8
HyperSeg-M [30]	512×1024	75.8	21.6	46.2
BiSenet v2 τ [20]	512×1024	72.6	5.6	179.2
STDC1-Seg75 τ [67]	768×1536	75.3	6.98	143.1
LGCNet	769×1537	72.4	8.68	115.2
LGCNet*	769×1537	74.0	8.68	115.2
LGCNetτ	769×1537	74.0	4.67	205.6

the coarse data is utilized. The full comparison results are shown in Table IV. Compared to BiSeNet-Xception39 [19] and CAS [32] that have comparable running speed with us, our LGCNet surpasses them by a large margin with 4.0% and 1.9% improvement, respectively. Compared to other methods such as SegNet [17], ENet [18], SQ [59] and ICNet [16], our method achieves significant improvement in speed while achieving the performance gain over them about 15.4%, 14.1%, 12.6%, 2.9%, respectively. To fairly compare with BiSeNet v2 [20] and STDC1-Seg75 [20], we remeasured our method with TensorRT 5.1.5 acceleration. Although STDC1-Seg75 [20] performs better than our method in accuracy, our running speed is much faster (205.6 vs 143.1 in FPS), which demonstrates that the proposed method can achieve either better performance or faster running speed when comparing with the co-occurrent works.

also have many candidate operations using DW-Conv, so the speed of our LGCNet is still capable of beating it if the DW-Conv is optimized correctly like DFANet.

¹After merging the BN layers for DFANet, there still has a speed gap between the original paper and our measurement. We suspect that it is caused by the inconsistency of the implementation platform in which DFANet has the optimized depth-wise convolution (DW-Conv). LGCNet

²The speed, which is tested by Pytorch without TensorRT, is provided by the authors.

b) *Results on CamVid*: To evaluate the transfer capacity of LGCNet, we transfer the network searched on Cityscapes to Camvid directly. Table V shows the comparison results with other methods. With input size 720×960 , LGCNet achieves the 73.8% mIoU with 158.6 FPS that is also the state-of-the-art trade-off between performance and speed, which demonstrates the superior transferability of LGCNet.

TABLE V: Results on the CamVid test set with resolution of 960×720 . "-" indicates the corresponding result is not provided by the methods.

Method	mIoU (%)	Latency(ms)	FPS
SegNet [17]	55.6	34.01	29.4
ENet [18]	51.3	16.33	61.2
ICNet [16]	67.1	28.98	34.5
BiSeNet [19]	65.6	-	-
DFANet A [21]	64.7	8.33	120
CAS [32]	71.2	5.92	169
FasterSeg [66]	71.1	7.59	131.8
GAS [39]	72.8	6.53	153.1
LGCNet	73.8	6.31	158.6

c) *Visual Segmentation Results*: We provide some visual prediction results on the Cityscapes validation set. As shown in Figure 6, the columns correspond to the input image, ground truth, the prediction of LGCNet. It demonstrates that LGCNet can achieve satisfactory and consistent visual results.



Fig. 6: Results of the proposed LGCNet on the Cityscapes validation set. The first column is input image, the second column is ground truth, and the final column displays the output of LGCNet.

C. Ablation Study

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

a) *Effectiveness of 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 cells, i.e., one type of normal cell and one type of reduction cell; b) network stacked by independent cells, i.e., each cell has its own structure; 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. The corresponding experiment results are shown in Table VI. The performance reported here is the average mIoU over five repeated experiments on the Cityscapes validation set with latency loss weight $\beta = 0.005$. \pm denotes the variance of mIoU for each strategy. Overall, performance suffers greatly when there is just one independent cell since there is a larger search space, which makes optimization much more challenging. This performance drop is mitigated by adding a 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)).

TABLE VI: Ablation study for the effectiveness of GCN-Guided Module on Cityscapes validation dataset.

Methods	mIoU (%)	Params	FPS
a) cell shared	68.6 (± 0.9)	6.60 M	100.1
b) cell independent	67.1 (± 1.0)	4.18 M	118.9
c) cell independent + FC	69.7 (± 0.6)	3.24 M	112.8
d) cell independent + GCN	73.0 (± 0.5)	2.09 M	115.2

b) *Comparison against Random Search*: As discussed in [68], random search is a competitive baseline for hyper-parameter optimization. To further verify the effectiveness of the GCN-Guided Module, we randomly sample ten architectures from the search space and evaluate them on the Cityscapes validation set with ImageNet pretrain. 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 over 100 FPS from the search space. The results are shown in Table VII, in which each value is the average result of ten random architectures. In summary, the network searched by LGCNet 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.

TABLE VII: Comparison to random search on the Cityscapes validation set.

Methods	mIoU (%)	FPS
Random setting (a)	69.8 (± 0.5)	61.9 (± 4.8)
Random setting (b)	66.3 (± 0.4)	107.6 (± 3.4)
LGCNet	73.0 (± 0.5)	115.2 (± 3.7)

c) *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 VIII in which the values are the average mIoU over five repeated experiments on the Cityscapes validation set with latency

loss weight $\beta = 0.005$. The experimental result indicates that LGCNet achieves the best performance when $d = 64$.

TABLE VIII: Ablation study for different GCN weight dimensions of GGM.

Methods	mIoU (%)	FPS
without GGM	67.1	112.9
GGM with $d = 16$	72.1	111.6
GGM with $d = 32$	72.2	109.2
GGM with $d = 64$	73.0	115.2
GGM with $d = 128$	72.5	107.1
GGM with $d = 256$	72.6	116.3

d) Different 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 $\mathbf{A} = \alpha'_k(\alpha'_{k-1})^T$. 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 IX.

TABLE IX: The comparison results for reasoning graph for edges and operations.

Reasoning Graph	mIoU (%)	FPS
Operation-identity	71.2	106.4
Edge-similarity	73.0	115.2

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 a 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 the trade-off between performance and latency. This can also be verified from the result in Table IX. So we choose the “edge-similarity” way to construct the reasoning graph as described in Section 3.2.

e) Effectiveness of the Dense-connected Fusion Cell: To demonstrate the effectiveness of the proposed dense-connected fusion cell (DCFC), we conduct a series of experiments with different strategies: a) without multi-scale feature aggregation; b) with ASPP module [4]; c) with PPM module [2]; d) with searched MSCell in [32]; e) with searched dense-connected fusion cell (DCFC). The results are shown in Table X. Overall, the searched dense-connected fusion cell successfully boosts up the mIoU from 69.5% to 73.0% on the Cityscapes validation set. Particularly, the searched aggregation cell surpasses the ASPP module and PPM module, searched MSCell by 0.6% and 0.5%, 0.3% performance gains with faster inference speed.

TABLE X: The performance for different multi-scale modules on the Cityscapes validation set.

Methods	mIoU (%)	FPS
a) LGCNet	69.5	124.1
b) LGCNet with ASPP	72.4	108.4
c) LGCNet with PPM	72.5	114.1
d) LGCNet with MSCell	72.7	112.7
e) LGCNet with DCFC	73.0	115.2

We also conduct a ablation study for which features should be aggregated, and the result is shown in the Table XI. Specifically, the cell 0 is fixed for all experiment settings because there is only one cell with low level information (stride=4), and we explore different choices for other feature scales (i.e., stride=8 and stride=16). The numbers in the table is the average result of three repeated experiments with the same random seed on the Cityscapes validation set. The results show that cell 0, cell 7 and cell 13 can achieve best trade-off between accuracy and latency comparison to other settings.

TABLE XI: Ablation study for which features should be aggregated on Cityscapes validation dataset.

Cells to aggregate	mIoU (%)	FPS
a) [0, 5, 13]	72.5	109.8
b) [0, 6, 13]	71.8	113.4
c) [0, 7, 13]	72.9	115.1
e) [0, 7, 10]	72.2	115.2
f) [0, 7, 11]	72.1	114.6
g) [0, 7, 12]	72.4	112.8

f) Effectiveness of the Latency Constraint: As mentioned above, LGCNet provides the ability to flexibly achieve a superior trade-off between performance and speed with latency-oriented optimization. We conducted a series of experiments with different loss weights β in Equation 10. Figure 7 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 *beta* grows from 0.0005 to 0.005. When the β increases from 0.0005 to 0.005, the latency dramatically decreases and performance somewhat degrades slightly. However, as the β increases from 0.005 to 0.05, the performance declines rapidly while the latency decline is fairly limited. Thus in our experiments, we set β as 0.005. It is obvious that the latency-oriented optimization works well for achieving a balance between accuracy and latency.

D. Network Visualization and Analysis

a) Network Visualization: As shown in Table VI, the network searched by our LGCNet with GGM has a smaller parameter size while achieving much higher performance. The visualization results can effectively help to analyze which component brings in the performance improvement. We thus visualize the networks searched on Cityscapes by the three methods: 1) LGCNet with GGM; 2) LGCNet with fully connected layer; and 3) random search in Figure 9, Figure 10 and Figure 11, respectively.

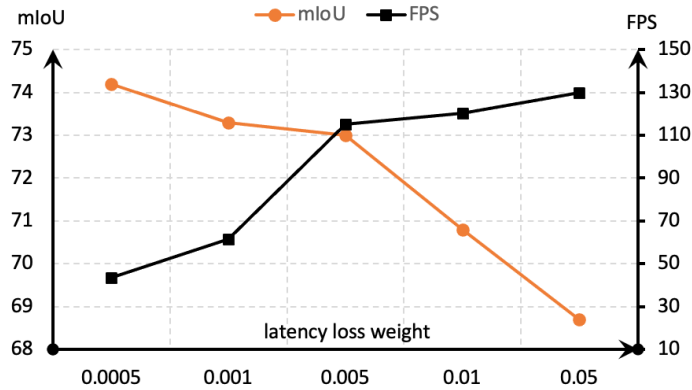


Fig. 7: The accuracy on the Cityscapes validation set for different latency constraint settings.

Compared to the other methods, the network searched by our LGCNet with GGM shows the following three advantages:

1) The cells in the shallow stage tend to choose light-weight operations (i.e., none, max pooling, skip connection), while the cells in the deep stage prefer the complicated ones, which is the goal of pursuing both high speed and performance as mentioned in the introduction of our study. Specifically, under the same latency loss weight, the network searched by our LGCNet with GGM comprises thirty light-weight operations (dashed-line arrow in the figure) with lower latency, whereas the other two methods utilize twenty-one and twenty-three light-weight operations, respectively. Our LGCNet with GGM, on the other hand, achieves superior performance, demonstrating the effectiveness of the concept of burden-sharing in a group of cells when they are aware of how much others are willing to contribute.

2) The deeper layers tend to utilize larger receptive field operations (e.g. conv with dilation = 4 or 8), which plays a key role to improve performance in semantic segmentation [3], [4]. Specifically, the network searched by our LGCNet with GGM uses 11 large receptive field operations (denoted by the green arrow) in the last four cells and the other methods only use 4 or 8 operations, respectively.

3) As we anticipated, the final structure searched by our LGCNet with GGM exhibits sufficient cell-level diversity. On the contrary, the network search by LGCNet with fully connected layer tends to employ similar structures, for instance, cell 7 is similar to cell 8 and 9, and cell 1 is similar to the cell 2, 3 and 4.

Moreover, we also show the architecture of the searched dense-connected fusion cell in Figure 8. We clearly found that the selection of operations in the shallow stage (i.e. low-level features) tends to choose more dilated convolution operations for achieving enough receptive field when performing the multi-scale feature fusion. As we expected, we observe that the choice of dilated convolution operations in the low-level stage narrows the semantic gaps among the features from different scales.

b) 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 find a light-weight network, we do not allow the cell structures to share with each other to encourage structure diversity. Apparently, learning cells independently makes the search process much more difficult due to the enlarged search space and does not guarantee better performance, thus the GCN-Guided Module can be regarded as a regularization term to the search process. 2) We have discussed that $p(Z)$ is a fully factorizable joint distribution in the 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 of modeling the condition in probability distribution $p(Z)$.

V. CONCLUSION

In this paper, a novel Graph-guided and Dense-Connected Fusion Architecture Search (LGCNet) framework is proposed to tackle the real-time semantic segmentation task. In contrast to the existing NAS approaches, which stack the same searched cell into a whole network, LGCNet explores to search different cell architectures and adopts the graph convolutional network to bridge the information connection among cells. In addition, we propose a novel dense-connected fusion cell that aggregates multi-level features in the network automatically to effectively fuse the low-level spatial details and high-level semantic context. Finally, a latency-oriented constraint is endowed into the search process for balancing accuracy and speed. Extensive experiments have demonstrated that LGCNet performs much better than the state-of-the-art real-time segmentation approaches.

REFERENCES

- [1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015, pp. 3431–3440. 1, 3, 7
- [2] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *CVPR*, 2017, pp. 2881–2890. 1, 3, 7, 9
- [3] 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*, vol. 40, no. 4, pp. 834–848, 2018. 1, 3, 10
- [4] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017. 1, 3, 7, 9, 10
- [5] X. Li, L. Zhang, G. Cheng, K. Yang, Y. Tong, X. Zhu, and T. Xiang, "Global aggregation then local distribution for scene parsing," *IEEE Trans. Image Process.*, vol. 30, pp. 6829–6842, 2021. 1
- [6] X. Li, X. Li, A. You, L. Zhang, G. Cheng, K. Yang, Y. Tong, and Z. Lin, "Towards efficient scene understanding via squeeze reasoning," *IEEE Trans. Image Process.*, vol. 30, pp. 7050–7063, 2021. 1
- [7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. 1, 3
- [8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778. 1, 2, 3
- [9] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CVPR*, pp. 1–9, 2016. 1, 3
- [10] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *CVPR*, 2017, pp. 1800–1807. 1, 3
- [11] X. Li, H. He, H. Ding, K. Yang, G. Cheng, J. Shi, and Y. Tong, "Improving video instance segmentation via temporal pyramid routing," *CoRR*, vol. abs/2107.13155, 2021. 1

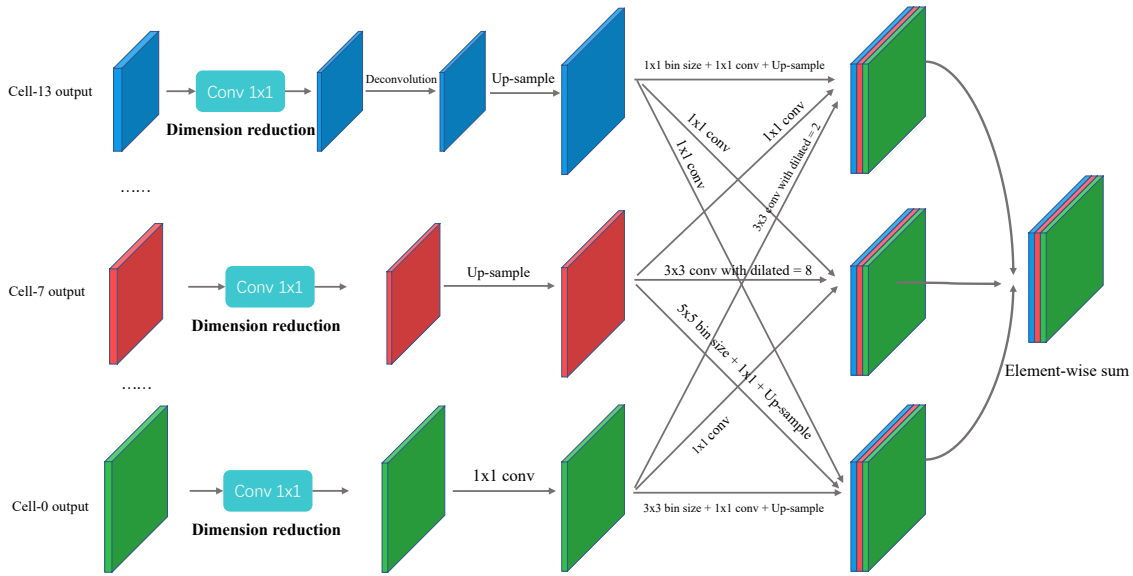


Fig. 8: Illustration of the searched dense-connected fusion cell architecture.

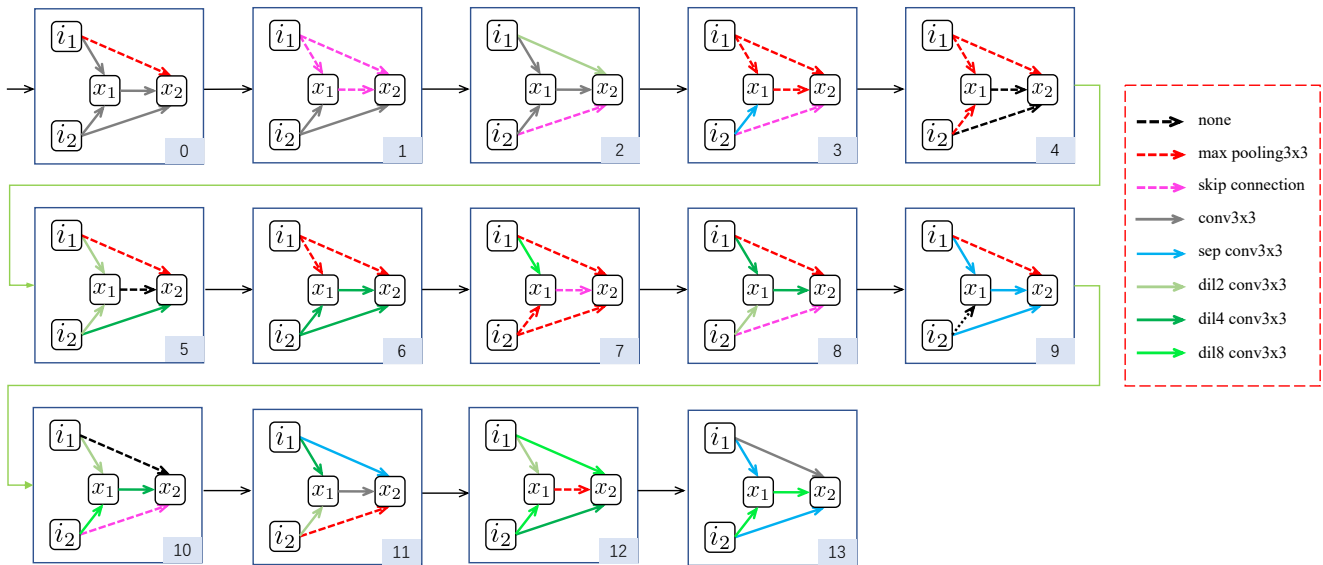


Fig. 9: The network searched by our LGCNet with GGM exhibits the benefited property (e.g. more dilation convolution operations in deep layers and more low computational operations for fast speed) for real-time semantic segmentation.

[12] X. Li, X. Li, L. Zhang, G. Cheng, J. Shi, Z. Lin, S. Tan, and Y. Tong, "Improving semantic segmentation via decoupled body and edge supervision," in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XVII*, ser. Lecture Notes in Computer Science, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., vol. 12362. Springer, 2020, pp. 435–452. 1

[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*, 2016, pp. 3213–3223. 1, 2, 6

[14] 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*, vol. 111, no. 1, pp. 98–136, 2015. 1

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

[16] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," in *ECCV*, 2018, pp. 405–420. 1, 3, 7, 8

[17] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE trans. PAMI*, vol. 39, no. 12, pp. 2481–2495, 2017. 1, 7, 8

[18] 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, 7, 8

[19] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," in *ECCV*, 2018, pp. 334–349. 1, 3, 7, 8

[20] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "Bisenet v2: Bilateral network with guided aggregation for real-time semantic

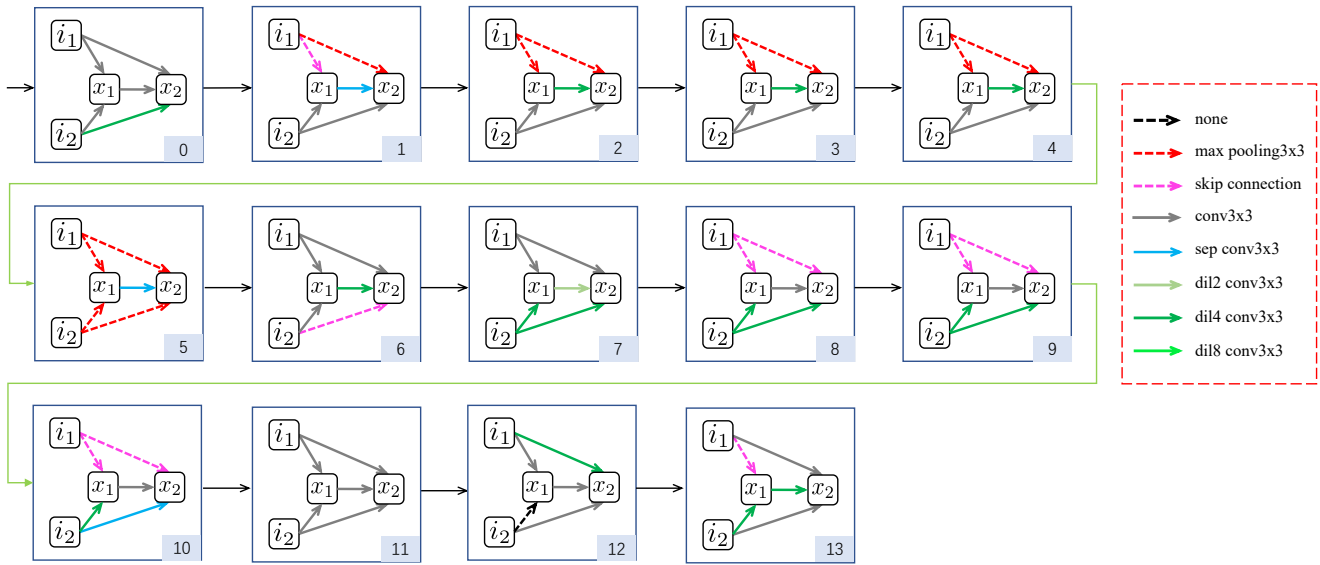


Fig. 10: The network searched by our LGCNet with fully connected layer.

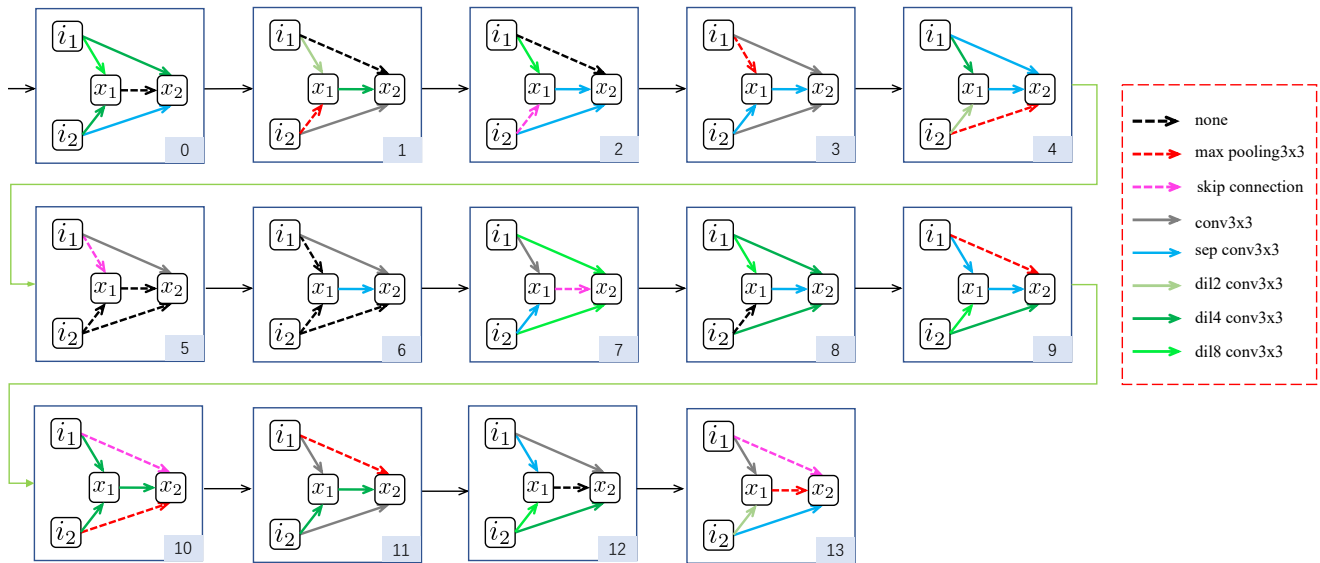


Fig. 11: The randomly sampled network.

segmentation,” *International Journal of Computer Vision*, pp. 1–18, 2021. 1, 7

[21] H. Li, P. Xiong, H. Fan, and J. Sun, “Dfanet: Deep feature aggregation for real-time semantic segmentation,” in *CVPR*, 2019, pp. 9522–9531. 1, 3, 7, 8

[22] H. Liu, K. Simonyan, and Y. Yang, “DARTS: differentiable architecture search,” in *ICLR*, 2019. 1, 3, 4

[23] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in *CVPR*, 2018, pp. 8697–8710. 1

[24] R. Negrinho and G. Gordon, “Deeparchitect: Automatically designing and training deep architectures,” *arXiv:1704.08792*, 2017. 1

[25] B. Krause, E. Kahembwe, I. Murray, and S. Renals, “Dynamic evaluation of neural sequence models,” *arXiv:1709.07432*, 2017. 1

[26] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, “Efficient neural architecture search via parameter sharing,” in *ICML*, 2018, pp. 4092–4101. 1, 3

[27] H. Cai, L. Zhu, and S. Han, “Proxylessnas: Direct neural architecture search on target task and hardware,” *arXiv:1812.00332*, 2018. 1, 3, 6

[28] S. Xie, H. Zheng, C. Liu, and L. Lin, “SNAS: stochastic neural architecture search,” in *ICLR*, 2019. 1, 3, 4

[29] V. Nekrasov, H. Chen, C. Shen, and I. Reid, “Fast neural architecture search of compact semantic segmentation models via auxiliary cells,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9126–9135. 1, 2, 4

[30] Y. Nirkin, L. Wolf, and T. Hassner, “Hyperseg: Patch-wise hypernetwork for real-time semantic segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4061–4070. 1, 7

[31] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in *ECCV*, 2018, pp. 833–851. 1, 3

[32] Y. Zhang, Z. Qiu, J. Liu, T. Yao, D. Liu, and T. Mei, “Customizable architecture search for semantic segmentation,” in *CVPR*, 2019, pp.

- 11 641–11 650. 1, 2, 3, 6, 7, 8, 9
- [33] 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*, 2019, pp. 10 734–10 742. 1, 3, 6
- [34] A. Shaw, D. Hunter, F. Iandola, and S. Sidhu, “SqueezeNAS: Fast neural architecture search for faster semantic segmentation,” in *ICCV Neural Architects Workshop*, 2019. 1
- [35] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv:1704.04861*, 2017. 2
- [36] P. Sun, J. Wu, S. Li, P. Lin, J. Huang, and X. Li, “Real-time semantic segmentation via auto depth, downsampling joint decision and feature aggregation,” *International Journal of Computer Vision*, vol. 129, no. 5, pp. 1506–1525, 2021. 2
- [37] M. Minsky, *The Society of Mind*. Simon & Schuster, 1988. 2
- [38] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv:1609.02907*, 2016. 2, 4, 5, 7
- [39] P. Lin, P. Sun, G. Cheng, S. Xie, X. Li, and J. Shi, “Graph-guided architecture search for real-time semantic segmentation,” in *CVPR, to appear*, 2020. 2, 7, 8
- [40] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017. 3
- [41] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” in *ICLR*, 2017. 3
- [42] M. Guo, Z. Zhong, W. Wu, D. Lin, and J. Yan, “IRLAS: inverse reinforcement learning for architecture search,” *CoRR*, vol. abs/1812.05285, 2018. 3
- [43] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search,” *CoRR*, vol. abs/1802.01548, 2018. [Online]. Available: <http://arxiv.org/abs/1802.01548> 3
- [44] Y. Chen, Q. Zhang, C. Huang, L. Mu, G. Meng, and X. Wang, “Reinforced evolutionary neural architecture search,” *CoRR*, vol. abs/1808.00193, 2018. [Online]. Available: <http://arxiv.org/abs/1808.00193> 3
- [45] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, “Understanding and simplifying one-shot architecture search,” in *ICML*, 2018, pp. 549–558. 3
- [46] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, “Smash: one-shot model architecture search through hypernetworks,” *arXiv:1708.05344*, 2017. 3
- [47] 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
- [48] A. Noy, N. Nayman, T. Ridnik, N. Zamir, S. Doherty, I. Friedmann, R. Giryes, and L. Zelnik-Manor, “Asap: Architecture search, anneal and prune,” *arXiv:1904.04123*, 2019. 3
- [49] 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*, 2019, pp. 2820–2828. 3, 6
- [50] 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*, 2018, pp. 8713–8724. 3
- [51] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. Yuille, and L. Fei-Fei, “Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation,” in *CVPR*, 2019. 3, 7
- [52] X. Wang and A. Gupta, “Videos as space-time region graphs,” in *ECCV*, 2018, pp. 399–417. 4, 5
- [53] S. Yan, Y. Xiong, and D. Lin, “Spatial temporal graph convolutional networks for skeleton-based action recognition,” in *AAAI*, 2018. 4
- [54] C. Zhang, M. Ren, and R. Urtasun, “Graph hypernetworks for neural architecture search,” *ICLR*, vol. abs/1810.05749, 2019. 4
- [55] C. J. Maddison, A. Mnih, and Y. W. Teh, “The concrete distribution: A continuous relaxation of discrete random variables,” *arXiv:1611.00712*, 2016. 5
- [56] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *NIPS*, 2019. 7
- [57] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *CVPR. IEEE*, 2009, pp. 248–255. 7
- [58] Z. Wu, C. Shen, and A. van den Hengel, “High-performance semantic segmentation using very deep fully convolutional networks,” *CoRR*, vol. abs/1604.04339, 2016. 7
- [59] M. Trembl, J. Arjona-Medina, T. Unterthiner, R. Durgesh, F. Friedmann, P. Schuberth, A. Mayr, M. Heusel, M. Hofmarcher, M. Widrich *et al.*, “Speeding up semantic segmentation for autonomous driving,” in *MLITS, NIPS Workshop*, vol. 2, 2016, p. 7. 7
- [60] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic, “In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images,” *CVPR*, pp. 12 599–12 608, 2019. 7
- [61] X. Li, Y. Zhou, Z. Pan, and J. Feng, “Partial order pruning: for best speed/accuracy trade-off in neural architecture search,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 7
- [62] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, “Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation,” in *ECCV*, 2018, pp. 552–568. 7
- [63] S. Mehta, M. Rastegari, L. Shapiro, and H. Hajishirzi, “Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network,” *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9182–9192, 2019. 7
- [64] E. Romera, J. M. Álvarez, L. M. Bergasa, and R. Arroyo, “Erfnet: Efficient residual factorized convnet for real-time semantic segmentation,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, pp. 263–272, 2018. 7
- [65] J. Zhuang, J. Yang, L. Gu, and N. Dvornik, “Shelfnet for fast semantic segmentation,” *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 847–856, 2019. 7
- [66] W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li, and Z. Wang, “Fasterseg: Searching for faster real-time semantic segmentation,” *ICLR*, vol. abs/1912.10917, 2020. 7, 8
- [67] M. Fan, S. Lai, J. Huang, X. Wei, Z. Chai, J. Luo, and X. Wei, “Re-thinking bisenet for real-time semantic segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9716–9725. 7
- [68] L. Li and A. 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*, 2019, p. 129. 8