Towards Real-Time Segmentation on the Edge

Yanyu Li*[1], Changdi Yang*[1], Pu Zhao*[1], Geng Yuan[1], Wei Niu[2], Jiexiong Guan[2], Hao Tang[3], Minghai Qin[1], Qing Jin[1], Bin Ren[2], Xue Lin[1], Yanzhi Wang[1]

[1]Northeastern University
[2]College of William & Mary
[3]CVL, ETH Zurich
{li.yanyu, yang.changd, zhao.pu, yuan.geng, jin.qing, xue.lin, yanz.wang}@northeastern.edu
{wniu, jguan}@email.wm.edu, bren@cs.wm.edu, hao.tang@vision.ee.ethz.ch, qinminghai@gmail.com

Abstract

The research in real-time segmentation mainly focuses on desktop GPUs. However, autonomous driving and many other applications rely on real-time segmentation on the edge, and current arts are far from the goal. In addition, recent advances in vision transformers also inspire us to re-design the network architecture for dense prediction task. In this work, we propose to combine the self attention block with lightweight convolutions to form new building blocks, and employ latency constraints to search an efficient sub-network. We train an MLP latency model based on generated architecture configurations and their latency measured on mobile devices, so that we can predict the latency of subnets during search phase. To the best of our knowledge, we are the first to achieve over 74% mIoU on Cityscapes with semi-real-time inference (over 15 FPS) on mobile GPU from an off-the-shelf phone.

Introduction

Deep Neural Networks (DNNs) have achieved great success in various tasks with extraordinary performance. In this work, we investigate semantic segmentation. As a dense prediction task, it aims to assign a class label to each pixel, and plays an important role in many real-world applications like autonomous driving. However, segmentation models usually consume tremendous memory and computation resources, leading to difficulties for deployment on the resource-limited devices. The large feature size and complicated multi-scale feature fusion limit the efficiency of segmentation models.

From the scope of architecture design, lightweight CNNs (Yu et al. 2021) dominate the design space for efficient segmentation. Recent advances in vision transformer (Dosovitskiy et al. 2021) inspires new research other than traditional CNNs on segmentation task, specifically, based on self attention mechanism (Xie et al. 2021) with global receptive field. However, neither vision transformers nor traditional CNNs (Chen et al. 2018a; Yu et al. 2021) are computational efficient enough for edge deployment. It is especially difficult for vision transformers to handle large resolution inputs, and we observe that they even underperform CNNs in real-time computation budget. The open question is, can we incorporate the strength of them, combine the efficiency of lightweight convolutions and global receptive field of transformers, while avoiding intensive computations? In this work, we take a step toward this goal with a self attention block to extract spatial dependencies in the low resolution segment branch, and a lightweight CNN stem to reduce computation overhead.

In addition to new design, it is crucial to search for a compact network for efficient deployment. *Neural Architecture Search* (NAS) and *network pruning* have been extensively investigated to discover good architectures with lower memory occupancy, reduced energy consumption, and faster inference speed. Though NAS and automatic pruning have been proven to be successful on the classification task, it is non-trivial to migrate them to dense image prediction, due to the tremendous searching computation cost to train multiple candidate architectures in Reinforcement learning (RL)-based NAS methods (Zoph and Le 2017; Zhong et al. 2018; Zoph et al. 2018), or the huge memory cost to train all architectures simultaneously in differentiable NAS methods (Brock et al. 2017; Bender, Kindermans et al. 2018; Liu, Simonyan, and Yang 2018).

Firstly, state-of-the-art segmentation models usually incorporate complex contextual fusion, that is, utilizing information from multiple spatial resolutions. As a result, unlike the cell-level search on classification, the search space is hierarchical for the segmentation task, which is quite large and the policy is difficult to converge, and may even converge to poor local minima. Secondly, there are no well-defined proxy tasks in semantic segmentation. The general practice is to directly search on the target dataset with early stopping. The accuracy is usually very low and comparison among the candidates may not reflect the final performance. Thirdly, the current segmentation paradigm (Chen et al. 2018a), especially ViTs (Xie et al. 2021), highly relies on knowledge transformation from larger scale datasets, e.g., pretraining on ImageNet-1000 (Deng et al. 2009) or MS COCO (Lin et al. 2014). When performing architecture search, candidates are not pretrained on ImageNet first but directly trained on dense prediction, which may enlarge the accuracy gap between the early-stopped model and well-trained model. Moreover, external knowledge makes the performance assessment unfair.

In this work, we first combine the advantage of ViTs and CNNs and design a mixed supernet building block that wields

*These authors contributed equally.

global receptive field and detailed local features, and overcomes the intensive computation of pure transformers. Besides the architecture design, we propose a new search and training paradigm to resolve the aforementioned problems. (i) To address the hierarchical search space, we design a block and width search starting from a dedicate human-designed architecture, which saves considerable searching cost compared to those searching on densely connected grid (Liu et al. 2019; Zhang et al. 2021; Chen et al. 2019). Width search is achieved by automatic channel pruning from a wide supernet. We perform block search by gradient-based gamble softmax sampling, which is capable of choosing or removing building blocks through backpropagation. (ii) We perform proxyless search, which directly trains and evaluates sub-networks for image dense prediction without external knowledge. We perform self knowledge distillation and add auxiliary losses along with the search process. The distillation not only stabilizes the candidate training, but also enables us to fully exploit the potential of the sub-net and directly deploy an accurate model within single phase training. (iii) Since we aim at real-time inference on the edge, we design a new latency-aware regularization that directly assesses the inference speed/latency of candidate.

Our contributions include:

- We incorporate the merits of lightweight convolution and self attention to design a new segment branch for efficient and accurate segmentation.
- We propose an efficient search paradigm. Starting from a good dual branch supernet, we can automatically optimize the block selection and width in less than 16 GPU hours under latency constraints. The search cost is significantly reduced compared to hierarchical grid search.
- We utilize self knowledge distillation and auxiliary losses to fully exploit the potential of the subnet without external knowledge, making our method fair and reliable.
- To the best of our knowledge, we are the first to achieve semi-real-time segmentation on a mobile GPU (15 FPS) with competitive accuracy (> 74 mIoU on Cityscapes).

Related Work

Real-Time Semantic Segmentation. CNN-based semantic segmentation (Zhao et al. 2017; Chen et al. 2018b; Fu et al. 2019; Huang, Zhu, and Huang 2019) achieves great success, but generally suffers from intensive computation cost and slow inference speed. Current research in real-time segmentation mainly focuses on desktop GPUs, and can be classified into two categories, human designs (Yu et al. 2018; Li et al. 2020; Yu, Gao et al. 2020; Fan et al. 2021) or neural architecture search (NAS) methods (Liu et al. 2019; Li et al. 2019b; Chen et al. 2019; Zhang et al. 2021).

As pioneer in handcrafted real-time segmentation, ENet (Paszke et al. 2016) incorporate a lightweight network to achieve high inference speed. DeepLabV3+ (Chen et al. 2018b) utilizes atrous separable convolution to reduce FLOPs and uses the light-weight MobileNetV2 (Sandler et al. 2018) as the backbone. BiSeNet (Yu et al. 2018; Yu, Gao et al. 2020) and STDC (Fan et al. 2021) adopt a two-branch architecture, where one extracts spatial information with a deeper network, and the other utilizes a shallower network to learn details. SFNet (Li et al. 2020) uses flow alignment module to fuse context information and spatial information.

Inspired by NAS, many works investigate the potential of searching segmentation models automatically. Auto-DeepLab (Liu et al. 2019) presents a hierarchical search space to achieve extremely high segmentation performance regardless of computation budgets. FasterSeg (Chen et al. 2019) incorporates a latency regularization to search for efficient models. DCNAS (Zhang et al. 2021) proposes a densely connected search space with gradient-based searching.

Vision Transformers. Vision Transformers (ViTs) introduced in (Dosovitskiy et al. 2021) employ a self attention-based transformer architecture in visual recognition tasks to achieve comparable performance against CNN counterparts. ViT is then adopted in dense prediction tasks. SETR (Zheng et al. 2021) uses ViT as the encoder to get high-level feature map. Segmenter (Strudel et al. 2021) uses mask transformer as the decoder. Segformer (Xie et al. 2021) uses a hierarchical transformer encoder to produce multi-scale feature map and a lightweight MLP based decoder for efficient semantic segmentation. MaskFormer (Cheng, Schwing, and Kirillov 2021) abandons per-pixel prediction and used mask-based model to predict a series of masks, with a customize backbone and a transformer based decoder. Mask2Former (Cheng et al. 2021) uses a transformer decoder with masked attention and proposed a universal architecture for all segmentation tasks including instance segmentation, semantic segmentation and panoptic segmentation, with SOTA performance.

Neural Architecture Search and Pruning. NAS is proposed to identify high-performance network architectures from a given search space automatically. Reinforcement learning (RL) based NAS (Zoph and Le 2017; Pham et al. 2018) and evolution-based NAS (Elsken, Metzen, and Hutter 2018; Real et al. 2019) usually need to train and evaluate each candidate model, leading to tremendous searching cost. Another direction is the gradient-based NAS (Liu, Simonyan, and Yang 2018; Cai, Zhu, and Han 2018; Chu et al. 2019; Guo et al. 2020), which relax the discrete architecture representation into a continuous and differentiable form, to enable a more efficient search with gradients descents, at the cost of huge memory budget to cover all candidate architectures.

Network pruning is another compression technique that can effectively reduce the DNN storage and computation cost. Specifically in this work, we refer to structured pruning (Wen et al. 2016; Li et al. 2019a; Cai et al. 2019) which removes entire filters or channels of the convolution layer. Recent work (Li et al. 2022) employs reparameterization technique (Ding et al. 2021) to enable a flexible pruning policy.

Method

In this section, we introduce our new design of supernet block and search algorithm. Moreover, we utilize auxiliary losses and self-distillation to fully exploit the potential of the subnet without access to external knowledge.

Supernet Design

As discussed in BiseNetV2 (Yu, Gao et al. 2020), mixing convolution and attention in each building block is not desired

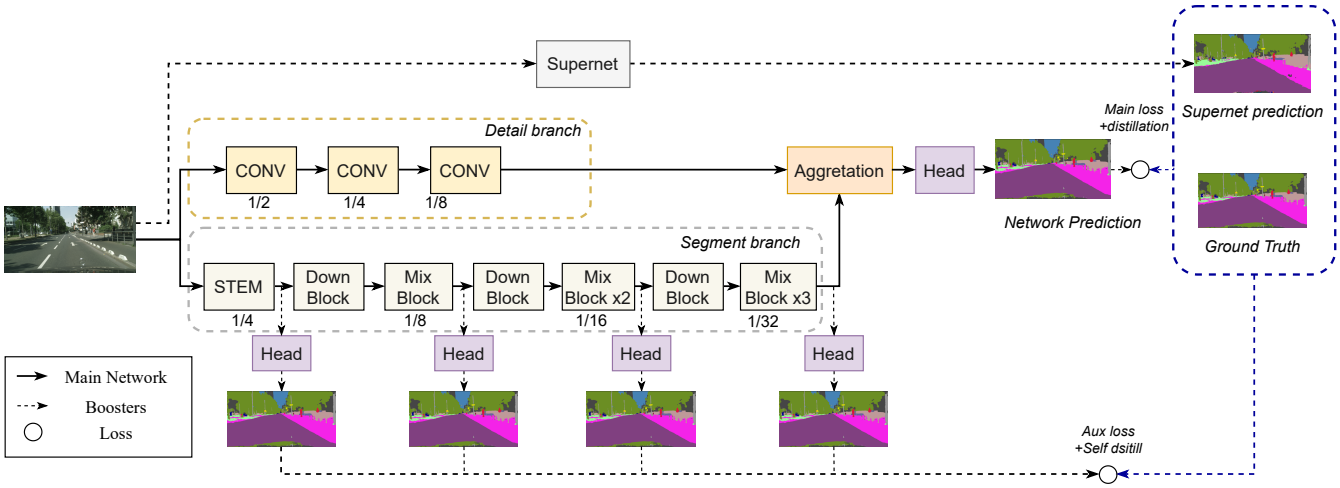


Figure 1: Illustration of the proposed method. We search from a dual branch supernet with a shallow detail path and a deep segment path with mixed block, and we demonstrate auxiliary losses and the method of self distillation.

for efficient inference. Following the dual branch design, we keep the detail branch built with convolutions and introduce mixed block only in segment branch, as shown in Fig. 1.

Detail Branch. To extract low-level features, the detail branch in supernet is built with shallow 3×3 convolutions, including 3 downsampling layers with $stride = 2$ to obtain $\frac{H}{2} \times \frac{W}{2}$, $\frac{H}{4} \times \frac{W}{4}$, $\frac{H}{8} \times \frac{W}{8}$ features.

Segment Branch. Segment branch is the key to extract spatial dependency and guarantee sufficient large receptive field. As a result, it is often deep and computation intensive. To address this, we follow BiSeNet to employ a stem block that quickly downsamples feature size to $\frac{H}{4} \times \frac{W}{4}$.

Following the stem block, we append downsampling blocks (Fig. 2 (a)), as well as the newly designed mixed block which combines lightweight inverted residual block (LIR) and multi-head self-attention (MHSA) (Fig. 2 (b)). Different from the inverted residual block in BiSeNet, we change the group-wise convolution into consecutive 1×1 and depth-wise convolution. The benefits come two-fold. We observe that this mobile-like design boosts accuracy as it preserves the 5×5 receptive field achieved by 2 consecutive 3×3 convolutions, while offering more channel combinations. In addition, group-wise convolution is not well-supported on compilers and is hard to accelerate. As for the MHSA branch, we firstly divide 2D images (4D tensor) with size (B, C, H, W) into patches (B, N, P^2C) and apply global attention,

$$X_l = MHSA(Q_l, K_l, V_l) = Softmax(\frac{Q_l \cdot K_l^T}{\sqrt{dim}}) \cdot V_l, \tag{1}$$

where the Queries (Q), Keys (K) and Values (V) are computed by linear projection

$$[Q_l : K_l : V_l] = W_{dim \times 3} \cdot X_{l-1}. \tag{2}$$

Similar to the standard configuration from ViT (Dosovitskiy et al. 2021), we apply pre-MHSA layer normalization, and post-MHSA MLP projections. We then transform the output patches back to 4D tensor and add them with the output of LIR through gamble softmax sampling to form a new dual-path supernet building block, as shown in Fig. 2 (b).

Note that our design differs from (Ding, Lian et al. 2021) for the following reasons. Firstly, we do not apply the mixed block to the shallow detail branch, as it is neither necessary for accuracy, nor computation efficient to perform MHSA on large feature sizes. Secondly, in our search pipeline, only one operator in the mixed block is preserved during inference stage, that is, either to perform the LIR or MHSA path. This is because frequent patchify and reshaping operations incur large overhead on edge devices with difficulties to parallelize.

Feature Aggregation and Decoder. Despite naive addition or concatenation, we employ Bilateral Guided Aggregation (BGA) as proposed in (Yu et al. 2021). With appropriate upsampling to match the spatial resolution of segment path and detail path, segment features are encoded as attention map by sigmoid activation and then multiplied with features from detail branch.

We use a light decoder as many efficient segmentation arts (Chen et al. 2018a; Yu et al. 2021) do, which consists of bilinear upsampling and convolutions.

Width Search by Parameterized Pruning

In width search, we aim to shrink convolution filters. Specifically, we need to determine unimportant feature channels and eliminate them. To achieve this, we create a *depth-wise binary convolution* (DWBC) layer as a trainable indicator to denote the pruning strategy of each CONV layer. The DWBC layer is built on a depth-wise 1×1 CONV layer and is inserted following each CONV layer. The width search workflow is shown in Fig. 2 (c). The forward pass of the DWBC layer can be defined as follows,

$$\boldsymbol{b}_l = \begin{cases} 1, & \boldsymbol{c}_l > T_h. \\ 0, & \boldsymbol{c}_l \leq T_h. \end{cases} \quad \text{(element-wise)} \tag{3}$$

$$\boldsymbol{a}_l = \boldsymbol{b}_l \odot (\boldsymbol{w}_l \odot \boldsymbol{a}_{l-1}), \tag{4}$$

where $\boldsymbol{c}_l \in R^{o \times 1 \times 1 \times 1}$ is the weights of the depth-wise 1×1 CONV layer with o output channels. T_h is a threshold,

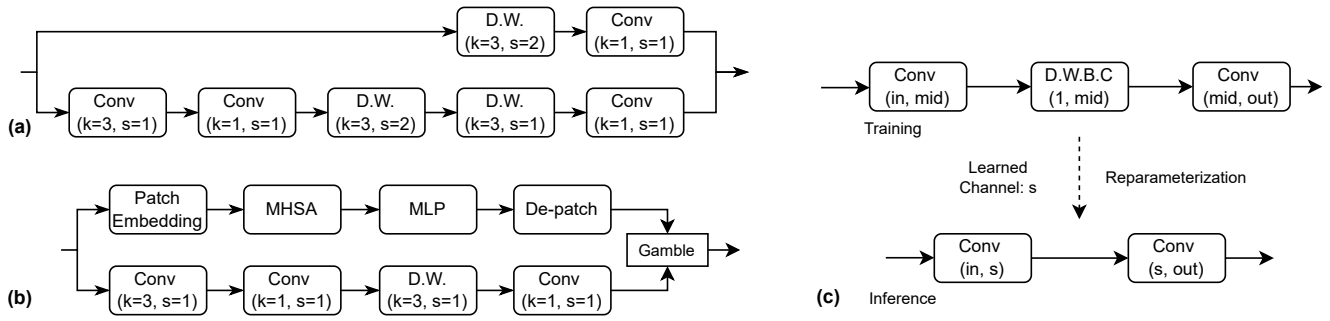


Figure 2: (a) Downsample block in segment branch. (b) Mixed branch with lightweight inverted residual and multi-head self attention. (c) Illustration of DWBC channel shrinking.

which is simply set as 0.5 here. $\odot$ is the CONV operation. $\boldsymbol{w}_l \in R^{o\times i\times k\times k}$ is the l-th CONV layer weights, with o output channels, i input channels, and kernel size $k \times k$. $\boldsymbol{a}_l \in R^{B\times o\times s\times s'}$ is the output features with $s \times s'$ feature size. B denotes the batch size.

In the forward pass, for the l-th CONV layer $\boldsymbol{w}_l$, the DWBC layer first quantize its weights of depth-wise 1×1 CONV layer into binary values $\boldsymbol{b}_l$. Then it performs CONV operation for the outputs of the previous CONV layer $\boldsymbol{w}_l \odot \boldsymbol{a}_{l-1}$ with the binarized weights b_l. Let $\boldsymbol{b}_l = \{0\}^{o_0\times 1\times 1\times 1} \oplus \{1\}^{o_1\times 1\times 1\times 1}$, where o_0 and o_1 denote the number of zeros and ones in $\boldsymbol{b}_l$, respectively ($o_0 + o_1 = o$), and $\oplus$ refers to channel-wise concatenation. Thus we have

$$\begin{aligned}\boldsymbol{a}_l =& \boldsymbol{b}_l^{o\times 1\times 1\times 1} \odot (\boldsymbol{w}_l^{o\times i\times k\times k} \odot \boldsymbol{a}_{l-1}) \\ =& (\{0\}^{o_0\times 1\times 1\times 1} \cdot \boldsymbol{w}_l^{o_0\times i\times k\times k} \odot \boldsymbol{a}_{l-1}) \\ & \oplus (\{1\}^{o_1\times 1\times 1\times 1} \cdot \boldsymbol{w}_l^{o_1\times i\times k\times k} \odot \boldsymbol{a}_{l-1}) \\ =& \boldsymbol{w}_l^{o_1\times i\times k\times k} \odot \boldsymbol{a}_{l-1},\end{aligned} \tag{5}$$

where zero channels are removed in the last equality. With DWBC, where each element in $\boldsymbol{b}_l$ corresponds to an output channel of $\boldsymbol{w}_l \odot \boldsymbol{a}_{l-1}$, non-zero elements mean that the corresponding channels are kept while zero elements denote that the corresponding channels are pruned. Thus, the DWBC layer serves as a binary mask to show the pruning status of each channel and is optimized through gradient descent.

The next problem is how to train the DWBC layers. Since $\boldsymbol{b}_l$ is determined through binarization of $\boldsymbol{c}_l$, we only need to train $\boldsymbol{c}_l$. However, the binarization operation is non-differentiable, leading to difficulties for back-propagation with gradients. To overcome this, we propose to adopt Straight Through Estimator (STE) (Bengio, Léonard, and Courville 2013) to pass gradients and enable gradient descent. Specifically, the gradient with reference to $\boldsymbol{b}_l$ is directly passed to $\boldsymbol{c}_l$ as shown below,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}_l} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_l}, \tag{6}$$

where $\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_l}$ can be obtained through back-propagation without difficulties. With STE, $\boldsymbol{c}_l$ can be updated with gradient descent optimizers in a typical training.

STE is originally proposed to avoid the non-differentiable problem in quantization tasks. We have several advantages to integrate STE with DWBC layers in channel shrinking: (i) The pruning strategy is decoupled from model parameters magnitudes. It is not determined by the weight magnitude. Instead, we have a separable variable to determine whether to prune each channel. (ii) The information in pruned channels is preserved since zeros in DWBC layers do not zero out the weights of pruned channels. Different from weight magnitude based pruning to keep all pruned weights zero, the pruned channels in our method do not necessarily become zero. As a result, the pruned channels are free to recover and contribute to accuracy if later its corresponding mask is changed from 0 to 1. (iii) We can train the model parameters $\boldsymbol{W} = \{\boldsymbol{w}_l\}$ and the pruning policy parameters $\boldsymbol{C} = \{\boldsymbol{c}_l\}$ simultaneously.

Differentiable Block Search

Besides width search, for the mixed block, we need to select desired block type or eliminate unnecessary blocks to avoid computations. Thus, we adopt Gumbel softmax sampling to automatically search the block number and block type.

To perform block search, we add a skip connection from input to output for each mixed block. Thus each block needs to choose one path from a skip connection and the other two branches (LIR and MHSA), leading to various block numbers and types in the model. The illustration of block search is shown in Fig. 2 (b). The path selection is usually non-differentiable, leading to difficulties for model training with optimizers. Therefore, we adopt the differentiable Gumbel-softmax sampling (Jang, Gu, and Poole 2016). At each optimization step, we sample one path with Gumbel-softmax and train the model with the selected path.

With Gumbel-softmax, the forward pass is expressed as

$$\boldsymbol{g}_n = h_n^s \cdot \boldsymbol{g}_{n-1} + h_n^l \cdot \boldsymbol{g}_n^l + + h_n^m \cdot \boldsymbol{g}_n^m, \tag{7}$$

where $\boldsymbol{g}_n$ is the output features of the n-th block. $\boldsymbol{g}_n^l$ is the output features of LIR and $\boldsymbol{g}_n^m$ is the MHSA output features. h_n^s, h_n^l and h_n^m (usually one hot) are the corresponding path selection variables in the n-th block. Thus $\boldsymbol{H} = \{h_n^s, h_n^l, h_n^m\}$ denotes the path selection for mixed block search.

Though pruning all channels results in an entirely vanished block, we do not employ DWBC layer and STE method in depth search because eliminating all channels of a CONV layer leads to mutation in regularization loss for adjacent layers and introduces instability.

Latency-Aware Constraints

With the proposed method, we can train the model and search a suitable width and block simultaneously with the loss,

$$\min_{\boldsymbol{W},\boldsymbol{C},\boldsymbol{H}} \mathcal{L}(\boldsymbol{W},\boldsymbol{C},\boldsymbol{H}) + \beta \cdot \mathcal{L}_{reg}(\boldsymbol{C},\boldsymbol{H}), \tag{8}$$

where $\mathcal{L}_{reg}$ is the regularization term related to the computation complexity or on-chip latency. β is a hyper-parameter to weight the relative importance.

In this work, we target to assess the model with its real latency on mobile GPUs. Prior works (Wu, Dai et al. 2019; Dai, Zhang et al. 2019; Yang, Howard et al. 2018) either collect on-device latency data to build a lookup table for latency estimate, or deploy each candidate on chip to gather real latency data. Clearly, these methods either suffer from large estimation errors or introduce additional overhead in search process waiting for real on-chip latency data. Consequently, we propose a new solution to generate a sufficient amount of different candidate building blocks in the search space and measure their latency on the mobile device. Then, we train a DNN (named as the latency model) based on the collected data to predict the speed/latency of candidate architectures. We find that a tiny DNN composed by a few fully connected layers is sufficient for this objective. Plus, this paradigm enables a once-for-all benefit, which means we can reuse the latency model as long as targeting on the same device. As a result, searching new sub-networks under different constraints will not introduce extra evaluation cost. The latency based regularization term becomes

$$\mathcal{L}_{reg}^{\text{LAT}} = \left| \sum_{b} \mathbb{S}\{o_1, i, s, s', k\} - \mathcal{S} \right|^2, \tag{9}$$

where $\mathbb{S}$ denotes the DNN to predict latency based on block characteristics (feature size, input and output channel, etc.). $\mathcal{S}$ is the target latency, and $\sum_b$ denotes latency measured by blocks. This latency model is just a multi-layer perceptron (MLP) model with 5 fully connected layers and ReLU functions. Since the latency prediction is not a very complex regression problem, the latency model can be trained fast to achieve high accuracy (<4% error-rate).

Performance Boosters

In order to further improve the performance of the searched compact network, we utilize auxiliary losses and self distillation to boost its performance without learning external knowledge from extra datasets or distillation with additional high-performance models. The performance boosters are complementary and add no additional cost at inference time. For the auxiliary loss, We plug several auxiliary segmentation heads after each downsampling stage of the semantic branch and compute the auxiliary loss with the ground truth label.

Our customized self distillation has two stages: The output of supernet will give soft constraint to both the output of the main network and the auxiliary classifiers. Self distillation transfers knowledge from the strong supernet to the searched compact subnet, at not additional access to external data or knowledge. We use label-wise distillation to transfer the knowledge from the powerful supernet T to a sub-network S. In order for network S to learn the probability distribution of teacher network T, we use Kullback–Leibler (KL) divergence to compute the loss between two outputs:

$$\mathcal{L}_d = \sum_{i \in \mathcal{R}} \text{KL}\left(\mathbf{q}_i^s \| \mathbf{q}_i^t\right), \tag{10}$$

where $\mathbf{q}_i^s$ denotes class distribution for each pixel i from the output of student network, $\mathbf{q}_i^t$ denotes class distribution for each pixel i from the output of teacher network, R denote all pixels from the output image and $\text{KL}(\cdot)$ denotes KL divergence which is calculated as,

$$\text{KL}(P\|Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right), \tag{11}$$

where P and Q are probability distributions on space $\mathcal{X}$.

The final loss function after we apply auxiliary classify losses and self distillation becomes:

$$\mathcal{L} = \min_{\boldsymbol{W},\boldsymbol{C},\boldsymbol{H}} \mathcal{L} + \beta \cdot \mathcal{L}_{reg} + \gamma \mathcal{L}_{aux} + \lambda \mathcal{L}_d, \tag{12}$$

where $\mathcal{L}_{aux}$ is the auxiliary loss between the output of branch segment heads and ground truth labels, $\mathcal{L}_d$ is the label-wise distillation loss with respect to both that output of main branch, as well as the output of branch segment heads. β, γ, λ are hyperparameters to scale loss and stabilize training.

Experiments

Datasets and Metrics

Cityscapes. Cityscapes(Cordts, Omran et al. 2016, 2015) is a dataset of urban street scenes from the perspective of cars collected in 50 cities. It includes 5000 finely annotated image, in which 2,975 images are used for training, 500 for validation, and 1,525 for testing. We exclude coarse training data. This dataset has 30 label classes and 19 of them are used for segmentation. The resolution of images are 2048×1024. Though intensively studied, it is challenging to perform real-time inference with such a high resolution.

Pascal VOC. PASCAL Visual Object Classes (VOC) 2012(Everingham et al. 2010) is a widely used dataset for semantic segmentation, classification, and object detection tasks. It includes 20 classes for 4 categories: Person, Animal, Vehicle and Indoor. In the segmentation task, there will be 21 classes including "background". There are 1464 images for training and 1449 images for validation.

ADE20K. ADE20K (Zhou et al. 2017, 2019) is a finely-annotated image dataset for object segmentations and part segmentations. It has a training set of $20,210$ images and validation set of $2,000$ dataset with 150 object and stuff classes. We use the object segmentation part of the dataset.

Metrics. For Semantic Segmentation evaluation, we use the mean of class-wise intersection-over-union (mIoU) to measure the accuracy performance.

$$mIoU = \frac{1}{n} \sum_{i}^{n} \frac{\sum P_{overlap}^i}{\sum P_{union}^i}, \tag{13}$$

where n is the class number (e.g., 19 for Cityscapes), and P_i refers to pixels that are assigned to a specific class label i.

Experimental Settings

Train Settings. We start the latency aware search from a fully pretrained supernet on the segmentation task. Note that the aforementioned booster strategies are applied throughout the gradient-based search process. We use stochastic gradient descent (SGD) optimizer and momentum is set to 0.9, and set

Method	Pretrain	Resolution	Parameters	GMACs	FPS	Val mIoU (%)	Test mIoU (%)
ENet (Paszke et al. 2016)	Y	512×1024	$354.9K$	5.9	–	–	58.3
PSPNet (Zhao et al. 2017)	Y	1024×2048	$68.07M$	525.0	0.2	–	78.4
DeepLabV3+ (Chen et al. 2018b)	Y	512×1024	$2.26M$	9.5	9.4	69.0	68.6
CAS (Zhang et al. 2019)	N	768×1536	–	–	–	71.6	70.5
DF1-Seg (Li et al. 2019b)	Y	1024×2048	–	–	–	74.1	73.0
BiSeNetV2 (Yu et al. 2021)	N	512×1024	$3.34M$	24.6	5.0	73.4	72.6
BiSeNetV2-L (Yu et al. 2021)	N	512×1024	–	–	–	75.8	75.3
SFNet (Li et al. 2020)	Y	1024×2048	$8.5M$	132	4.6	76.4	74.5
STDC1 (Fan et al. 2021)	Y	512×1024	$12.05M$	31.1	4.3	72.2	71.9
STDC2 (Fan et al. 2021)	Y	512×1024	$16.08M$	44.3	3.7	74.2	73.4
Auto-DeepLab-S (Liu et al. 2019)	N	1024×2048	$10.15M$	333.3	–	79.7	79.9(MS)
FasterSeg (Chen et al. 2019)	Y	1024×2048	–	28.2	9.8	73.1	71.5
DCNAS (Zhang et al. 2021)	N	1024×2048	–	294.6	–	–	84.3
Segformer-B0 (Xie et al. 2021)	Y	512×1024	$3.8M$	17.7	1.6	71.9	–
Mask2Former (Cheng et al. 2021)	Y	1024×2048	$44M$	568.0	–	79.4	–
TopFormer-B (Zhang et al. 2022)	Y	512×1024	$5.1M$	2.7	–	70.7	–
Supernet	N	512×1024	$\mathbf{24.43M}$	**136.0**	–	79.1	–
RTSeg-L	N	512×1024	$\mathbf{5.55M}$	**20.8**	**13.2**	75.3	74.6
RTSeg-M	N	512×1024	$\mathbf{3.56M}$	**9.2**	**15.5**	74.4	73.8
RTSeg-S	N	512×1024	$\mathbf{2.69M}$	**5.7**	**17.9**	73.1	72.3

Table 1: Comparison of our latency-driven searched model and prior arts in Cityscapes. The first segment includes popular handcraft baselines, while the second segment is NAS-based models. FPS is measured on the Qualcomm Adreno 660 GPU of Samsung Galaxy S21 mobile phone, all with our compiler support for fair comparison. Some FPS results are not available due to unsupported operations on mobile device. MS denotes for multi scale test. Detailed configurations can be found in Section .

batch size to 8 on each GPU. For Cityscapes, the learning rate is set to 0.1 initially with "poly" policy. For PASCAL VOC 2012, we set initial learning rate as 0.01. Learning rate value is determined as $\left(1 - \frac{\text{iter}}{\text{total\_iter}}\right)^{0.9}$ where iter refers to the current iteration number. The pretraining of supernet takes 160k iterations, while the search and fine-tune process both take 40k iterations. We incorporate multiple random scaling $\{0.5, 0.75, 1.0, 2.0\}$ and fixed size cropping of 512×1024 as data augmentation For Cityscapes. The crop size is chosen based on the trade-off between mobile capacity and accuracy. To enhance the training, we also use color jitter and random horizontal flip. As for PASCAL VOC 2012, we randomly crop the input image to 513×513. We set hyperparameters β as 0.01, γ as 1.0 and λ to be 0.001 in all experiments.

Test Settings. Despite some work incorporate muti-scale testing, we employ single scale test for fair comparison. We take 512×1024 as inference resolution for Cityscapes dataset, which greatly speedup inference on the edge while does not sacrifice too much accuracy. Plus, the resolution of 512×1024 serves enough for the scenario of edge sensors and monitors. We set inference resolution to 513×513 for PASCAL VOC 2012 dataset.

Experiment Environments. We search and train the neural network on 8 NVIDIA RTX TITAN GPUs, with CUDA 11.1 and PyTorch 1.9. Mobile latency is measured on the GPU of an Samsung Galaxy S21 smartphone, with Qualcomm Snapdragon 888 mobile platform integrated with Qualcomm Kryo 680 Octa-core CPU and a Qualcomm Adreno 660 GPU.

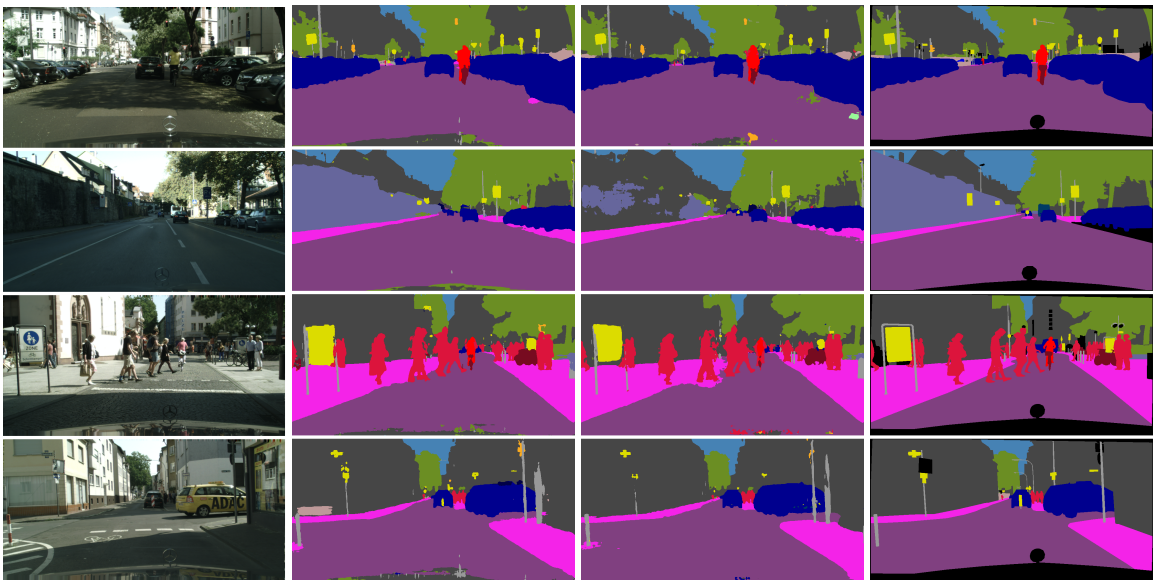
Experimental Results and Analysis

Based on our latency-driven search algorithm, we search on the proposed dual branch backbone with mixed operators.

Method	Proxyless	GPU Days	GMACs	mIoU
Auto-DeepLab	N	3	695.0	82.1
GAS	-	6.7	-	73.5
FasterSeg	N	2	28.2	71.5
Fast-NAS	N	8	435.7	78.9
SparseMask	N	4.2	36.4	68.6
DCNAS	Y	5.6	294.6	84.3
RTSeg-L	**Y**	**0.7**	**20.8**	75.3
RTSeg-M	**Y**	**0.7**	**9.2**	74.4
RTSeg-S	**Y**	**0.7**	**5.7**	73.1

Table 2: Comparison of search methods on Cityscapes.

Width search can be achieved by the proposed trainable indicators, while the blocks are searched by Gumbel-softmax. On Cityscapes, as shown in Table 1, compared with BiSeNetV2, our RTSeg-M greatly reduces the GMACs (our 9.2GMACs v.s. 24.6GMACs of BiSeNetV2) and achieves non-trivial better accuracy (our 74.4 mIoU v.s. 73.4 mIoU of BiSeNetV2). We have similar observations for STDC1 and STDC2. SFNet consumes too many computations (132GMACs) to be deployed on practical edge devices. Compared with the NAS-based method FasterSeg, our RTSeg-L, RTSeg-M and RTSeg-S achieve higher mIoU with much smaller computation costs. Other NAS methods such as Auto-DeepLab-S and DCNAS consume a huge amount of computations (about 300GMACs) which are inapplicable in practical mobile deployment. Compared with transformer-based methods such as Segformer-B0, similarly, our RTSeg-S achieves higher mIoU with less computations, while the SOTA Mask2Former is performance oriented and very computation intensive. Our RTSeg-M can achieve 15.5 FPS on the mobile device (Samsung Galaxy



(a) Input Image (b) RTSeg (c) BiSeNetV2 (d) Ground Truth

Figure 3: Visualization of segmentation results on the Cityscapes dataset.

Method	Params (M)	GMACs	mIoU
PSPNet	13.7	52.2	29.6
DeepLabV3+	15.4	25.8	38.1
Semantic FPN	12.8	33.8	35.8
RTSeg-L	5.6	10.5	38.7

Table 3: RTSeg results on ADE20K with 512×512 inputs.

S21), which approaches real-time execution with competitive segmentation performance, as shown in Table 1. Current desktop GPU designated arts can hardly achieve real-time segmentation on the edge device, with FPS lower than 10.

We show the results on ADE20K in Table 3. As observed, our method can achieve high mIoU with less computations compared with baseline methods on ADE20K. We also validate our method on PASCAL VOC dataset in Appendix. The search process takes approximately 0.7 GPU days and we further fine-tune the result sub-network for the same iterations. As shown in Table 2, our method efficiently search out a compact model while greatly save search cost compared to previous methods. Moreover, different from (Chen et al. 2018a; Xie et al. 2021), we do not incorporate pretraining on ImageNet. Thanks to our booster recipe, we successfully transfer label-wise knowledge from the supernet, saving search and training cost and meanwhile reduce human interference.

In Figure 3, we show the visualization comparison of our RTSeg-S and BiSeNetV2. RTSeg-S are better in quality than BiSeNetV2, especially for the buildings in the second row, the road under the pedestrians in the third row, and the tiny traffic sign in the fourth row, which is not well recognized by BiSeNetV2. These results indicate that our proposed method can effectively reduce the model size and computation while maintaining a good ability to recognize different objects.

Method	Designs			Search			
LIR	✓		✓	✓	✓	✓	✓
MHSA		✓	✓	✓	✓	✓	✓
Booster					✓	✓	✓
Search				✓	✓	✓	✓
GMACs	25.1	29.2	29.5	20.8	20.8	9.2	5.7
mIoU	74.0	73.8	74.2	74.5	75.3	74.4	73.1

Table 4: Ablation study on the proposed methods. The BiSeNetV2 baseline model has 24.6 GMACs and 73.4 uIoU.

Ablation Study

We conduct ablation study to verify the effects of proposed components, as shown in Table 4. From the model design level, we test the performance with new LIR block as well as MHSA. In addition, we also test the results with/without boosters. We can observe that simply adopting LIR or MHSA for the BiSeNetV2, the mIoU can be improved from 73.4 to 74.0 and 73.8, respectively, demonstrating their effectiveness. During search, adding the booster can improve the mIoU from 74.5 to 75.3, showing the necessity of our strategy.

Conclusion

In this work, we redesign the backbone of semantic segmentation task and incorporate the merits of self attention. Plus, we propose an efficient latency-driven search framework to find compact segmentation models. Our search algorithm can be finished within a simple training process and our inference speed proves to be significant faster than previous arts. Further, we utilized performance boosters including auxiliary losses and self distillation that effectively exploit the potential of the compact subnet without external knowledge.

Acknowledgments

This work is supported in part by National Science Foundation (NSF) CNS-1932351, CCF-1937500 and CMMI-2125326.

References

Bender, G.; Kindermans, P.-J.; et al. 2018. Understanding and simplifying one-shot architecture search. In *ICML*, 550–559.

Bengio, Y.; Léonard, N.; and Courville, A. C. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR*, abs/1308.3432.

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

Cai, H.; Gan, C.; Wang, T.; Zhang, Z.; and Han, S. 2019. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*.

Cai, H.; Zhu, L.; and Han, S. 2018. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*.

Chen, L.-C.; Zhu, Y.; Papandreou, G.; Schroff, F.; and Adam, H. 2018a. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *ECCV*.

Chen, L.-C.; Zhu, Y.; Papandreou, G.; Schroff, F.; and Adam, H. 2018b. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 801–818.

Chen, W.; Gong, X.; Liu, X.; Zhang, Q.; Li, Y.; and Wang, Z. 2019. Fasterseg: Searching for faster real-time semantic segmentation. *arXiv preprint arXiv:1912.10917*.

Cheng, B.; Misra, I.; Schwing, A. G.; Kirillov, A.; and Girdhar, R. 2021. Masked-attention mask transformer for universal image segmentation. *arXiv preprint arXiv:2112.01527*.

Cheng, B.; Schwing, A.; and Kirillov, A. 2021. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34.

Chu, X.; Zhang, B.; Xu, R.; and Li, J. 2019. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*.

Cordts, M.; Omran, M.; et al. 2015. The Cityscapes Dataset. In *CVPR Workshop on The Future of Datasets in Vision*.

Cordts, M.; Omran, M.; et al. 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Dai, X.; Zhang, P.; et al. 2019. Chamnet: Towards efficient network design through platform-aware model adaptation. In *CVPR*, 11398–11407.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*, 248–255. IEEE.

Ding, M.; Lian, X.; et al. 2021. HR-NAS: Searching Efficient High-Resolution Neural Architectures with Lightweight Transformers. In *CVPR*.

Ding, X.; Zhang, X.; Ma, N.; Han, J.; Ding, G.; and Sun, J. 2021. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13733–13742.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ICLR*.

Elsken, T.; Metzen, J. H.; and Hutter, F. 2018. Efficient multi-objective neural architecture search via lamarckian evolution. *arXiv preprint arXiv:1804.09081*.

Everingham, M.; Van Gool, L.; Williams, C. K. I.; Winn, J.; and Zisserman, A. 2010. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2): 303–338.

Fan, M.; Lai, S.; Huang, J.; Wei, X.; Chai, Z.; Luo, J.; and Wei, X. 2021. Rethinking BiSeNet For Real-time Semantic Segmentation. In *CVPR*, 9716–9725.

Fu, J.; Liu, J.; Tian, H.; Li, Y.; Bao, Y.; Fang, Z.; and Lu, H. 2019. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3146–3154.

Guo, Z.; Zhang, X.; Mu, H.; Heng, W.; Liu, Z.; Wei, Y.; and Sun, J. 2020. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, 544–560. Springer.

Huang, J.; Zhu, Z.; and Huang, G. 2019. Multi-stage HRNet: multiple stage high-resolution network for human pose estimation. *arXiv preprint arXiv:1910.05901*.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Li, T.; Wu, B.; Yang, Y.; Fan, Y.; Zhang, Y.; and Liu, W. 2019a. Compressing convolutional neural networks via factorized convolutional filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Li, X.; You, A.; Zhu, Z.; Zhao, H.; Yang, M.; Yang, K.; Tan, S.; and Tong, Y. 2020. Semantic flow for fast and accurate scene parsing. In *European Conference on Computer Vision*, 775–793. Springer.

Li, X.; Zhou, Y.; Pan, Z.; and Feng, J. 2019b. Partial order pruning: for best speed/accuracy trade-off in neural architecture search. In *CVPR*, 9145–9153.

Li, Y.; Zhao, P.; Yuan, G.; Lin, X.; Wang, Y.; and Chen, X. 2022. Pruning-as-Search: Efficient Neural Architecture Search via Channel Pruning and Structural Reparameterization. *arXiv preprint arXiv:2206.01198*.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, 740–755. Springer.

Liu, C.; Chen, L.-C.; Schroff, F.; Adam, H.; Hua, W.; Yuille, A. L.; and Fei-Fei, L. 2019. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, 82–92.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Paszke, A.; Chaurasia, A.; Kim, S.; and Culurciello, E. 2016. ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation. arXiv:1606.02147.

Pham, H.; Guan, M. Y.; Zoph, B.; Le, Q. V.; and Dean, J. 2018. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized evolution for image classifier architecture search. In *AAAI*, volume 33, 4780–4789.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 4510–4520.

Strudel, R.; Garcia, R.; Laptev, I.; and Schmid, C. 2021. Segmenter: Transformer for semantic segmentation. In *CVPR*, 7262–7272.

Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2016. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, 2074–2082.

Wu, B.; Dai, X.; et al. 2019. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*, 10734–10742.

Xie, E.; Wang, W.; Yu, Z.; Anandkumar, A.; Alvarez, J. M.; and Luo, P. 2021. SegFormer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34.

Yang, T.-J.; Howard, A.; et al. 2018. Netadapt: Platform-aware neural network adaptation for mobile applications. In *ECCV*, 285–300.

Yu, C.; Gao, C.; Wang, J.; Yu, G.; Shen, C.; and Sang, N. 2021. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *International Journal of Computer Vision*, 1–18.

Yu, C.; Gao, C.; et al. 2020. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *arXiv preprint arXiv:2004.02147*.

Yu, C.; Wang, J.; Peng, C.; Gao, C.; Yu, G.; and Sang, N. 2018. BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation. In *ECCV*.

Zhang, W.; Huang, Z.; Luo, G.; Chen, T.; Wang, X.; Liu, W.; Yu, G.; and Shen, C. 2022. TopFormer: Token Pyramid Transformer for Mobile Semantic Segmentation. In *CVPR*.

Zhang, X.; Xu, H.; Mo, H.; Tan, J.; Yang, C.; Wang, L.; and Ren, W. 2021. Dcnas: Densely connected neural architecture search for semantic image segmentation. In *CVPR*.

Zhang, Y.; Qiu, Z.; Liu, J.; Yao, T.; Liu, D.; and Mei, T. 2019. Customizable architecture search for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11641–11650.

Zhao, H.; Shi, J.; Qi, X.; Wang, X.; and Jia, J. 2017. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2881–2890.

Zheng, S.; Lu, J.; Zhao, H.; Zhu, X.; Luo, Z.; Wang, Y.; Fu, Y.; Feng, J.; Xiang, T.; Torr, P. H.; et al. 2021. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 6881–6890.

Zhong, Z.; Yan, J.; Wu, W.; Shao, J.; and Liu, C.-L. 2018. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2423–2432.

Zhou, B.; Zhao, H.; Puig, X.; Fidler, S.; Barriuso, A.; and Torralba, A. 2017. Scene Parsing through ADE20K Dataset. In *CVPR*, 5122–5130.

Zhou, B.; Zhao, H.; Puig, X.; Xiao, T.; Fidler, S.; Barriuso, A.; and Torralba, A. 2019. Semantic Understanding of Scenes Through the ADE20K Dataset. *Int. J. Comput. Vis.*, 127(3): 302–321.

Zoph, B.; and Le, Q. V. 2017. Neural Architecture Search with Reinforcement Learning. In *ICLR*.

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