When Humans Meet Machines: Towards Efficient Segmentation Networks

Peike Li
peike.li@student.uts.edu.au
Xuanyi Dong
xuanyi.dong@student.uts.edu.au
Xin Yu
xin.yu@uts.edu.au
Yi Yang
yi.yang@uts.edu.au

ReLER Lab
Australian Artificial Intelligence Institute
University of Technology Sydney
Sydney, AUSTRALIA

Abstract

In this paper, we investigate how to achieve a high-performance yet lightweight segmentation network for real-time applications. By analyzing three typical segmentation networks, we observe that the segmentation backbones and heads are often imbalanced which restricts network efficiency. Thus, we develop a lightweight context fusion (LCF) module and a lightweight global enhancement (LGE) module to construct our lightweight segmentation head. Specifically, LCF fuses multi-resolution features to capture image details and LGE is designed to enhance feature representations. In this manner, our lightweight head facilities network efficiency and significantly reduces network parameters. Furthermore, we design a Multi-Resolution Macro Segmentation structure (MRMS) to incorporate human knowledge into our network architecture composition. Given the resource-aware constraint (*e.g.*, latency time), we optimize our network with network architecture search while considering the relationships among atomic operators, network depth and feature resolution in segmentation tasks. Since MRMS embeds the segmentation-specific knowledge, it also provides a better architecture search space. Our Human-Machine collaboratively designed Segmentation network (HMSeg) achieves better performance and faster inference speed. Experiments demonstrate that our network achieves 71.4% mean intersection over union (mIOU) on Cityscapes dataset with only 0.7M parameters at 172.4 FPS on NVIDIA GTX1080Ti.

1 Introduction

Semantic segmentation is broadly applied in the fields of autonomous driving [19], video analysis [11], and virtual reality [17]. Since many applications require segmentation networks to run in real time, it is desirable to design an efficient network. Moreover, as the development of portable intelligence devices, deploying efficient networks to those devices also becomes highly demanded. Compared with high-end GPUs, those devices often have limited computational resources and thus require lightweight networks.

Previous methods manually design convolutional neural networks (CNN) [37, 39] to achieve real-time segmentation models. In general, these manually designed CNN-based segmentation networks are divided into two main components, *i.e.*, a *backbone encoder* and a *segmentation decoder head*. Previous works usually employ lightweight classification networks as backbones to reduce computational cost. However, this might lead to weak localization ability since segmentation targets the pixel-level classification rather than yielding an image-level prediction [10].

Recent works [4, 38] have been proposed to automatically choose a segmentation network from several architecture candidates. However, those methods highly rely on the neural architecture search (NAS) to select candidates. The architecture candidates are mainly built based on image classification macro-structure and do not take task-specific characteristics into account. In addition, those methods oversight the resource-aware constraints [4] or restrict them implicitly [38]. Therefore, previous works are not suitable to obtain a lightweight high-accuracy segmentation network.

In this work, we observe some key factors for designing an efficient segmentation network:

The segmentation backbone and head are often imbalanced. As illustrated in Figure 1, we analyze three typical efficient segmentation networks. Although those three methods all employ lightweight backbones to achieve real-time performance, their segmentation heads still remain heavy in terms of both parameter size and computational cost. This imbalance between the backbone and segmentation head is often overlooked in previous methods, preventing them from attaining efficient segmentation networks. Thus, designing a lightweight segmentation head is desirable to achieve an efficient segmentation network. Towards this goal, we not only employ a lightweight backbone, but also propose a lightweight segmentation head. To be specific, we introduce two extremely efficient modules in the segmentation head, *i.e.*, the Lightweight Context Fusion (LCF) module and the Lightweight Global Enhancement (LGE) module. Our LCF is designed to preserve image detailed spatial information via fusing different multi-resolution branch features. Our LGEM is developed to capture semantic context information by highlighting class-specific features. By doing so, our segmentation head can segment class-wise details more accurately.

The computational cost of a layer grows quadratically as feature resolution or channel numbers increase. We profile the GPU running time of different layers with different resolution in Figure 2. When the resolution of a layer becomes larger, the computational cost increases quadratically as seen in Figure 2. Similarly, increasing the channel numbers (*e.g.*, from stage 4 to stage 5) with the same feature resolution also leads to quadratical growth of computational cost. Moreover, we observe that most of the computational cost lies in the layers with large input resolution (stage 1) and large channel numbers (stage 4 and 5). In order to improve the inference efficiency, existing methods either restrict input resolutions [25, 30, 39] or reduce the channel numbers [1, 28, 39]. Although inference speed has improved to some extent, these methods sacrifice the spatial detail information or the potential capacity, leading to performance degradation. To pursue efficiency and effectiveness simultaneously, we design a Multi-Resolution Macro Segmentation (MRMS) structure to balance the architecture design of feature resolutions and channel numbers while maintaining network efficiency and effectiveness. In particular, we adopt our MRMS in different stages of our network. For instance, we employ more resolution while less channel numbers to extract low-level features while less resolution and more channels to represent high-level class-specific information.

As aforementioned, by integrating the human expert knowledge into our network design,

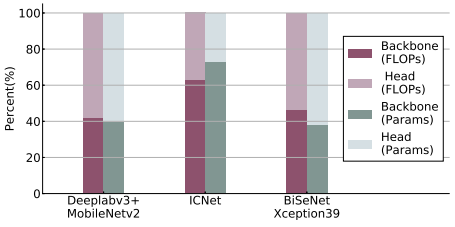

Figure 1: Computational cost (FLOPs) and model size (Params) for real-time segmentation methods, *i.e.*, Deeplabv3+ (MobileNetV2) [6], ICNet [39], BiSeNet [37].

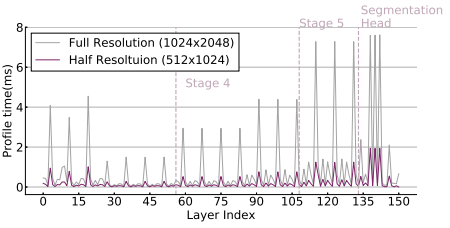

Figure 2: GPU profile time of Deeplabv3+ (MobilNetv2) on Cityscapes full resolution and half resolution inputs.

we propose MRMS to achieve a real-time semantic segmentation as illustrated in Figure 3. Although our hand-crafted network has incorporated human expert knowledge [10], the network might be a sub-optimal solution to the trade-off between the accuracy and inference speed. Moreover, it is challenging and time-consuming to manually optimize the topology under a reference resource-aware constraint, *e.g.*, satisfying a latency time. We leverage machine automatically search to optimize our network architecture within human prior based network design space.

Furthermore, our human knowledge based network architecture space is more flexible compared to NAS based segmentation methods [4, 26]. Since we observe that the feature resolution plays a key role in the semantic segmentation, we optimize along the dimension of the resolution. Specifically, our approach allows a network to be optimized with respect to atomic operator in each layer, the depth and the feature resolution for each branch with an explicit resource-aware constraint. Our human-machine collaboratively designed Segmentation network (HMSeg) achieves superior segmentation performance and inference speed. Experiments demonstrate that our network achieves 71.4% mean intersection over union (mIOU) on Cityscapes with only 0.7M parameters at 172.4 FPS on NVIDIA GTX1080Ti.

2 Related Work

Prior works on designing efficient segmentation networks can be roughly divided into two categories, *i.e.*, manually designed architecture with human knowledge and machine-driven architecture optimization.

Hand-crafted Efficient Segmentation Networks. The works [1, 37, 39] designed lightweight networks from scratch to achieve real-time performance. MobileNetV2 [33] designed an efficient block by applying depth-wise separable convolutional operations. However, the network was designed for image classification tasks rather than segmentation tasks. ICNet [39] adopted the image cascaded structure to speed up and reduce computational cost. BiSeNet [37] introduced a shallow spatial branch to process full resolution images while learning context information by a deep branch. Unlike previous works, our method presents two lightweight modules (*i.e.*, LFM and LGE) to achieve higher performance while remaining lightweight. We further design the MRMS to better incorporate human knowledge.

Machine-driven Architecture Optimization. Neural Architecture Search (NAS) [8, 9] is an effective technique to switch the labor-intensive architecture design to an automatic

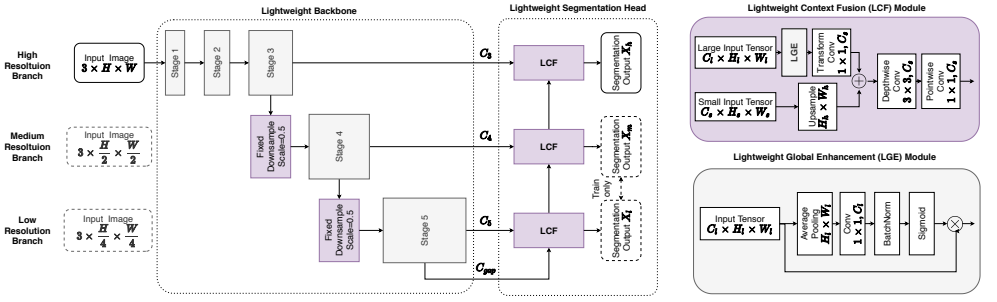

Figure 3: **The architecture of MRMS**. Our method instantiates MobileNetV2 as the lightweight backbone. Starting from the high resolution branch, we step-wisely reuse the intermediate feature maps and downsample them. The downsampled features are fed to medium and low resolution branches. We introduce two efficient lightweight modules, *i.e.*, *LCF* and *LGE* (in the LCF upper branch) to balance our backbone and segmentation head.

machine-driven optimization process. Prior arts [4, 24, 26, 32, 38] have exceeded the performance of manually designed networks. DPC [4] and AuxCell [26] searched configuration of a multi-scale context module while their backbone networks still remained the same as existing classification networks. However, those might not be suitable for segmentation tasks. Auto-Deeplab [24] searched a segmentation-specific network composed of a shared cell structure. These works only focus on segmentation accuracy without considering the computational cost. Recently, CAS [38] introduced a customized resource-aware constraint in searching an efficient segmentation network. However, the backbones of previous works are either fixed or repeat the same searching cell structure. In contrast, our approach allows each layer to be optimized under resource-aware constraints while taking into account the relationships among kernel sizes, network depths and feature resolution. Thus, our network achieves a better trade-off between inference speed and performance.

3 Methodology

3.1 Human-driven Prototype Design

In our human-driven design process, we leverage the expert knowledge to achieve an efficient segmentation network. As illustrated in Figure 3, our MRMS includes a lightweight backbone and a lightweight segmentation head.

Multi-resolution Lightweight Backbone. Segmentation tasks require a higher localization ability than classification tasks. Thus, we improve such ability in the following two aspects. First, we observe that *a large receptive field* is crucial for encoding long-term relation between pixels and leveraging context information [18]. Since an efficient backbone (*e.g.*, MobileNetV2) is often shallow, we enlarge the receptive field by utilizing the dilated convolutions in stage 4 and 5. Second, *fusing the low-level features and high-level features* is also critical for efficient segmentation networks. Generally, low-level features contain detailed localization information while high-level features involve more semantic information. To better aggregate these information while reducing the computational cost, we propose a multi-resolution macro segmentation structure (see Figure 3).

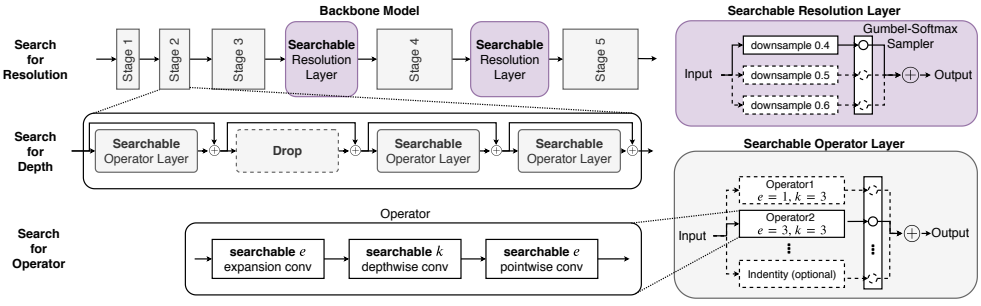

Figure 4: **Configuration of architecture optimization space.** Our macro-search space builds upon the backbone in our human-designed network. We aim to optimize the resolution downsampling rate of each branch via a *searchable resolution layer*, the depth in each stage, and the types of inverted residual bottleneck [33] via a *searchable operator layer*. A layer will be skipped by choosing an identity transformation. In our searchable resolution and operator layers, only one single path is selected each time by our gumbel-softmax sampler.

As depicted in Figure 3, we step-wisely share the model weights and reuse the intermediate feature maps. We input a full-size high resolution image to the high resolution branch (*i.e.*, stage 1, 2 and 3 in backbone). Thus, we are able to discover the detailed spatial information with fine object boundaries. To capture more semantic information [12], we feed the feature maps to the medium resolution branch (*i.e.*, stage 4) and the low resolution branch (*i.e.*, stage 5). Although the medium and low branch contain more channels, they only introduce a small computation overhead due to the low feature resolution. Unlike the image cascade structure [39], our mulit-resolution branch structure shares the weights among branches as well as reuses the intermediate features to save computational cost. With this simple yet effective design, our MRMS becomes an efficient architecture for semantic segmentation.

Lightweight Segmentation Head. The imbalance between the backbone and head (as we explained in § 1) leads to redundant computational cost and thus prevents from attaining an efficient segmentation network. To tackle this issue, we design a *Lightweight Context Fusion* (LCF) module to aggregate the output features C_3, C_4, C_5 from multi-resolution branches and the global average pooling feature C_{gap} from C_5, as illustrated in Figure 3. Instead of concatenating those features, we add the intermediate features from different resolution branches to ensure efficiency. With negligible computational cost, we further design the *Lightweight Global Enhancement* (LGE) module by adopting a global average pooling to capture global context as well as applying channel-wise attention to enhance the feature representation. Taking advantages of depth-wise separable convolutions, our LCF and LGE achieve higher computational efficiency and model compactness. The details of the segmentation head are shown in Figure 3.

Intermediate Boosting Strategy. To stabilize the training process and improve the segmentation accuracy, we introduce a boosting training strategy. Our intermediate boosting strategy can enhance the feature representation learning in the training phase and does not introduce any extra computation overhead in the inference phase. Specially, we upsample the outputs X_m and X_l from the medium and low resolution branches as intermediate boosting signals for

segmentation. Thus, we minimize the total loss $\mathcal{L}$ as,

$$\mathcal{L}(X;\theta) = l_h(X_h;\theta) + l_m(X_m;\theta) + l_l(X_l;\theta). \tag{1}$$

By doing this, we enhance the feature extraction ability of our backbone network even though it is shallow and lightweight.

3.2 Machine-driven Architecture Optimization

MRMS has incorporated the human expert knowledge. However, accuracy and inference speed of the network might be sub-optimal. Inspired by the prevailing architecture search methods [8, 9, 35, 36], we optimize our MRMS framework based on strong human priors to achieve a hardware resource-aware efficient segmentation network. Considering the characteristics of semantic segmentation tasks (*e.g.*, resolution often matters), we optimize different aspects of our backbone model, including the operators in each single layer, the depth in each stage and the resolution down-sample ratio for each branch. Our architecture optimization space is depicted in Figure 4.

Operator optimization. We seek the best operator configuration in each layer considering two most important aspects, *i.e.*, the expansion ratio and the kernel size. In each *searchable operator layer*, it consists of inverted residual blocks with various expansion ratios $e \in \{1,3,6\}$ and various kernel size $k \in \{3,5,7\}$ as candidates. We denote the distribution of each operator among total $|\mathbb{O}|$ candidate operators as $\alpha \in \mathbb{R}^{|\mathbb{O}|}$. The probability of selecting i-th operator is $p_i = \sigma(\alpha_i)$, where σ is a softmax function. To reduce the memory cost, our aim is to sample one operator $\mathbb{I} \subseteq \mathbb{O}$ each time. However, sampling a discrete operator from the candidate set is non-differentiable, preventing gradients back-propagation to α. Instead, we employ Straight-Through Gumbel-Softmax approximation [15] to soften the sampling procedure,

$$\hat{p}_i = \frac{\exp\left(\left(\log\left(p_i\right) + o_i\right)/\tau\right)}{\sum_{k=1}^{|\mathbb{O}|} \exp\left(\left(\log\left(p_k\right) + o_k\right)/\tau\right)}, \quad s.t. \;\; o_i = -\log(-\log(u)), \tag{2}$$

where $u \sim \mathcal{U}(0,1)$ is sampled from a uniform distribution. τ is a temperature parameter to control the sparsity. When $\tau \to 0$, the distribution $\hat{p}$ becomes a one-hot vector. When $\tau \to \infty$, $\hat{p}$ becomes a uniform distribution. To enable the network explore all the potential candidates and then converge to a certain one, the temperature τ anneals from 10 to 0.1. We discretize $\hat{p}$ using $\arg\max$ and only forward the sampled path $\mathbb{I}$ in the forward pass [8]. During the gradient back-propagation, we use the continuous approximation. Therefore, we sample one operator once a time and learn the operator weights in an end-to-end manner.

Depth optimization. Another aspect for optimizing the architecture is to determine the number of layers in each stage. To achieve this goal, we allow a whole layer to be replaced by an identity mapping, known as a skip connection. For example, as illustrated in Figure 4, there are four potential searchable operator layers in stage 2 and the second layer has been skipped by the identity transformation. In our method, we enable all layers without a channel number change to be skipped by the *identity transformation* operator d. Then, our searchable candidate set becomes $\mathbb{O} \cup \{d\}$. This newly added operator assists us to optimize the depth of each resolution branch.

Resolution optimization. In our human designed MRMS, the input resolution of each branch is fixed. However, this configuration might be sup-optimal since semantic segmentation pays more attention to spatial context information. Thus, we optimize the resolution

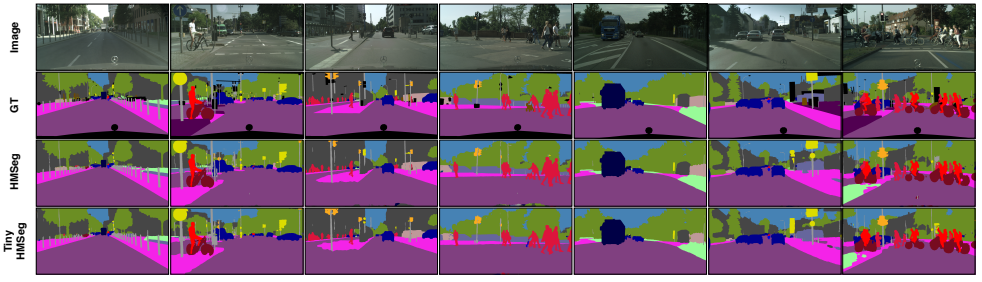

Figure 5: Visualization results of our HMSeg and TinyHMSeg on Cityscape validation set.

from a set $\mathbb{L}$ for each branch. Specifically, we search the *searchable resolution layer* set $\mathbb{L}$ with various downsampling ratios $\{0.4,0.5,0.6\}$. We denote the distribution of each resolution downsampling ratio by $\beta \in \mathbb{R}^{|\mathbb{L}|}$. We adopt the strategy proposed in the operator optimization in Eq. (2) to make β differentiable.

Resource-Aware Constraint. Our overall architecture optimization space can be regarded as a directed acyclic graph (DAG). Any path in the graph corresponds to a specific architecture. The goal of our optimizing procedure is to find the best architecture a from architecture parameters $\mathbb{A} = \{\alpha, \beta\}$. The optimizing procedure is formulated as follows,

$$\min_{a} \mathcal{L}_{seg}(\theta_a^*, a) + \lambda_{cost}\mathcal{L}_{cost}(\theta_a^*, a), \quad s.t.\ \theta_a^* = \arg\min_{\theta} \mathcal{L}_{\text{seg}}(\theta, a), \tag{3}$$

where $\mathcal{L}_{seg}$ is the segmentation loss defined in Eq. (1), $\mathcal{L}_{cost}$ is a loss accounting for the computational cost, and λ_{cost} controls the weight between the two losses. Since we treat both of these two objectives equally important, we set λ_{cost} to 1 in all the experiments. During the network learning procedure, we alternatingly optimize the architecture parameters $\mathbb{A}$ and the model weight parameters θ.

In Eq. (3), we introduce the loss $\mathcal{L}_{cost}$ to account for the computational cost since we aim to achieve a lightweight segmentation network. This computational cost loss encourages our network to converge to a specific hardware resource requirement R. Towards this goal, we pre-calculate a lookup table (LUT) for the computational cost of each candidate in the search space. The table is constructed based on the latency time of each candidate. By doing so, we can also optimize the network under different hardware constraints, *e.g.*, hardware-agnostic computational cost (FLOPs), the number of parameters, *etc.* Therefore, the objective $\mathcal{L}_{cost}$ is expressed as,

$$\mathcal{L}_{cost} = sign(F_{cost}(a) - R)\log(\mathbb{E}_{cost}(\mathbb{A})) \tag{4}$$

where $sign(\cdot)$ is a sign function, $F_{cost}(a)$ is the cost of derived architecture and $\mathbb{E}_{cost}$ is the expected computational cost of architecture parameters $\mathbb{A}$ based on the probability of each candidate. The final optimized architecture a is obtained *by selecting the candidate with the highest probability* from the architecture parameters $\mathbb{A}$.

4 Experiment

We conduct our experiments and ablation studies on the Cityscapes [7] dataset. To further valid the effectiveness and generalization ability of our method, we also report the perfor-

mance and speed on different semantic segmentation application scenarios, *i.e.*, CamVid [2] for urban street understanding and LIP [13] for human parsing [21, 22].

Implementation Details. We train our network on Cityscapes *from scratch* with random initialization. Our training scheme partially follows [37]. In brief, our network is trained on Cityscapes images of 768x1536 pixels for 720 epochs. SGD is used to optimize our network with a base learning rate 0.01 and weight-decay $5e^{-4}$. We employ Inplace Synchronized BatchNorm [31] to ensure the batch size large enough for training stability. Benefiting from our lightweight base model, our network can be trained only on two GTX 1080Ti GPUs. For all our experiments, we use the prefix *Tiny* to denote a more efficiency version of our network with the half channel capacity.

To deploy on diverse devices, we test the inference speed of our network with the high-performance inference framework TVM [6]. All the batch normalization layers are merged with the convolution layers. Note all of our results are reported in FP32 mode. The mean Intersection over Union (mIoU) is chosen as the performance metric. For real-time segmentation tasks, we measure the mIoU *without* using any test-time augmentation *e.g.*, flipping and multi-scale testing.

4.1 Real-time Semantic Segmentation Comparisons

Results on Cityscapes. We evaluate both our proposed network MRMS with human expert knowledge and machine optimized network HMSeg on Cityscapes. Different from most of other methods, our model is trained from scratch without leveraging any extra data *e.g.*, ImageNet pre-train weights or coarse annotations of Cityscapes dataset. Results are reported in Table 1. With our effective human design, TinyMRMS outperforms ICNet in terms of mIoU while reducing almost 90.1% FLOPs and 97.7% parameters. Our network achieves the fastest speed 139 FPS and outperforms BiSeNet and DFANet. Furthermore, HMSeg achieves superior results compared with other NAS-based methods. It is noteworthy that our HMSeg achieves considerably better mIoU and speedup in comparison to MobileNetV3. Our TinyHMSeg (with 0.7M parameters) only requires storage space about 2.8MB, making the deployment of our network possible in almost any embedded devices. All these results demonstrate the effectiveness of HMSeg and its ability to balance the performance and resource-aware constraints.

Results on CamVid. We compare our method with other real-time segmentation approaches on CamVid dataset in Table 2. Note that, our architecture is optimized from Cityscapes and we only fine-tune the network weights on Camvid. As indicated in Table 2, our HMSeg outperforms all the other real-time methods. Compared with CAS, our model runs 64% faster and still achieves better segmentation performance. This impressive result also demonstrates the generalization ability of our network.

Results on LIP. Human parsing is another vital application that acquires real-time performance. To the best of our knowledge, previous works mainly focus on the mIoU performance while overlook the real-time efficiency. Here, we compare with some mIoU-driven methods in Table 3. The performance of our efficient HMSeg even surpasses the network of [3] that employs an extremely heavy backbone. Notably, our TinyHMSegachieves 15× speedup (455.6 *vs* 29.3 FPS) and 90× less parameters (0.7M *vs* 66.7M) with a sacrifice of mere 4% performance decrease.

Table 1: Comparisons with the state-of-the-art real-time semantic segmentation methods on Cityscapes dataset. We report the computation cost (FLOPs), latency time and model size (Params). The symbol † indicates that the method use ImageNet pre-train weights.

	Method	mIoU(%)↑	FPS↑	Resolution	FLOPSs↓	Latency(ms)↓	Params↓
Human	SegNet† [1]	57.0	16.7	360x640	286G	59.9	29.5M
	ICNet† [39]	69.5	30.3	1024x2048	28.3G	33.0	26.5M
	ERFNet† [30]	68.0	11.2	512x1024	27.7G	89.3	20M
	SwiftNet [27]	69.4	27.7	1024x2048	41.0G	36.1	2.4M
	FastSCNN [29]	68.0	106.2	1024x2048	11.7G	9.4	1.1M
	DFANet† [18]	67.1	120	1024x1024	**2.1G**	8.3	4.8M
	BiSeNet† [37]	68.4	105.8	768x1536	14.8G	9.5	5.8M
	MRMS	**73.7**	67.7	768x1536	7.3G	14.7	2.1M
	TinyMRMS	69.9	**139.5**	768x1536	2.8G	**7.2**	**0.6M**
Machine	MobileNetV3 [14]	72.4	46.6	1024x2048	9.74G	21.5	1.51M
	DF1-Seg-d8† [20]	71.4	136.9	1024x2048	28.2G	7.3	6.7M
	CAS† [33]	70.5	108.0	768x1536	12.0G	9.3	1.7M
	HMSeg	**74.3**	83.2	768x1536	8.1G	12.0	2.3M
	TinyHMSeg	71.4	**172.4**	768x1536	3.0G	**5.8**	0.7M

Table 2: Results on CamVid. The input resolution is 720x960.

Method	mIoU(%)↑	FPS↑
ICNet [39]	67.1	34.5
ENet [28]	51.3	61.2
BiseNet [37]	65.6	165.4
CAS [33]	71.2	169.0
HMSeg	**75.1**	130.8
TinyHMSeg	71.8	**278.5**

Table 3: Results on LIP. The input resolution is 512x512.

Method	mIoU(%)↑	FPS↑	FLOPs↓	Params↓
DeepLab [3]	44.80	12.7	183.1G	42.5M
JPP [23]	51.37	6.6	374.0G	93.4M
CE2P [31]	**53.10**	30.3	87.7G	66.7M
HMSeg	49.43	292.6	1.8G	2.3M
TinyHMSeg	47.71	**455.6**	**0.7G**	**0.7M**

4.2 Ablation Study

Component Analysis of Human-Designed Network. We investigate the components of our human design framework in Table 4: (a) We directly upsample and simply use the outputs C_3, C_5 from the backbone to predict segmentation results. (b) We insert the LGE module, and achieve an improvement of 1.99% mIoU. (c) We employ the LCF instead of simply fusing the multi-resolution features. LCF significantly improves mIoU by a large margin of 4.18%. This demonstrates the importance of aggregating low-level and high-level features. Our final network employs both modules, with less than 0.3M extra parameters and marginal increase of computation cost, but improves the segmentation performance significantly.

Effects of Different Architecture Optimization Strategies. Under the same resource-aware constraint, we investigate the effect of different architecture optimization aspects, including the operator (O), the depth (D) and the resolution (R). Table 6 indicates that optimizing the architectures in larger search space brings more performance gain. Specially, due to the importance of spatial information in segmentation tasks, optimizing R brings significant performance improvements without increasing latency time. Since our network is manually designed and the input images are in high resolution, we found that our network cannot satisfy the desired latency time. Therefore, we did not solely optimize the architecture in the dimension of the resolution. Table 6 verifies the effectiveness of our MRMS.

Inference Speed *w.r.t* Resolution. Figure 6 presents the inference speed of our model on various input image resolution. Theoretically, the inference speed (FPS) is proportional to the

Table 4: Analysis of our human-designed segmentation head, *i.e.*, LCF and LGE.

Exp	LCF	LGE	mIoU	Latency(ms)	FlOPs	Params
(a)	-	-	68.77	12.9	6.35G	1.8M
(b)	-	✓	70.76	13.0	6.37G	1.8M
(c)	✓	-	72.95	14.7	7.32G	2.0M
(d)	✓	✓	**73.67**	14.7	7.33G	2.1M

Table 5: Inference speed on Snapdragon835 (Mobile), XeonE5-2682v4 (CPU) and GeForce 1080Ti (GPU). The input resolution is 256x512.

Model	Mobile	CPU	GPU
HMSeg	15.2	25.5	332.5
TinyHMSeg	**30.1**	**59.1**	**612.1**

pixel numbers of input images. Although lower resolution inputs tend to have low arithmetic intensities, the testing results may not strictly follow the theoretical rules. Our TinyHMSeg achieves inference speed of 349.5 FPS, 105.4 FPS for half-resolution (512×1024 pixels) and full-resolution (1024x2048 pixels) inputs, respectively. We also report the speed for commonly used video resolutions 360p, 720p and 1080p respectively in Figure 6.

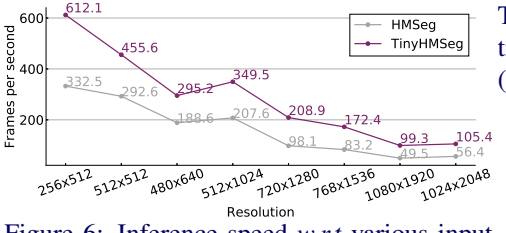

Figure 6: Inference speed *w.r.t* various input resolutions on GTX 1080 Ti.

Table 6: Impacts of different architecture optimization strategies, including the Operators (O), Depth (D) and Resolution (R).

Exp	O	D	R	mIoU↑	Latency(ms)↓
(a)	✓	-	-	73.44	11.9
(b)	✓	-	✓	74.01	12.1
(c)	-	✓	-	73.18	12.2
(d)	-	✓	✓	73.72	12.0
(e)	✓	✓	✓	**74.27**	12.0

Inference Speed *w.r.t* Hardware. We also evaluate the inference speed of our model on different hardware platforms , including CPU, GPU and mobile devices. The results are shown in Table 5. Our HMSeg achieves an outstanding speed for 332.5 FPS on the GPU platform. Our TinyHMSeg achieves real-time segmentation on both CPU and mobile devices at 59.1 and 30.1 FPS, respectively. All these evaluations demonstrate our HMSeg is practical in real-world applications.

5 Conclusion

In this work, we proposed a simple yet effective network HMSeg for real-time semantic segmentation. We introduced a new paradigm for designing an efficient network by incorporating the task-specific expert knowledge. Based on the expert knowledge, we are able to balance the feature extraction backbone and segmentation head, thus achieving a lightweight segmentation network. To enable our network to run on embedded devices, we employ architecture search to optimize our network configuration and further improve segmentation performance. Extensive experiments demonstrate our HMSeg attains superior performance in comparison to state-of-the-art real-time segmentation approaches.

Acknowledgment

This work is in partly supported by ARC DP200100938.

References

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

[2] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 40(4):834–848, 2017.

[4] Liang-Chieh Chen, Maxwell Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jon Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *NeurIPS*, pages 8699–8710, 2018.

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, pages 801–818, 2018.

[6] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. TVM: An automated end-to-end optimizing compiler for deep learning. In *13th Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 578–594, 2018.

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

[8] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *CVPR*, pages 1761–1770, 2019.

[9] Xuanyi Dong and Yi Yang. Network pruning via transformable architecture search. In *NeurIPS*, 2019.

[10] Qianyu Feng, Guoliang Kang, Hehe Fan, and Yi Yang. Attract or distract: Exploit the margin of open set. *ICCV*, 2019.

[11] Qianyu Feng, Zongxin Yang, Peike Li, Yunchao Wei, and Yi Yang. Dual embedding learning for video instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019.

[12] Qianyu Feng, Yu Wu, Hehe Fan, Chenggang Yan, Mingliang Xu, and Yi Yang. Cascaded revision network for novel object captioning. *IEEE Transactions on Circuits and Systems for Video Technology*, 2020.

[13] Ke Gong, Xiaodan Liang, Dongyu Zhang, Xiaohui Shen, and Liang Lin. Look into person: Self-supervised structure-sensitive learning and a new benchmark for human parsing. In *CVPR*, pages 932–940, 2017.

[14] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. *arXiv preprint arXiv:1905.02244*, 2019.

[15] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2017.

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

[17] Peike Li, Yunqiu Xu, Yunchao Wei, and Yi Yang. Self-correction for human parsing. *arXiv preprint arXiv:1910.09777*, 2019.

[18] Peike Li, Pingbo Pan, Ping Liu, Mingliang Xu, and Yi Yang. Hierarchical temporal modeling with mutual distance matching for video based person re-identification. *IEEE Transactions on Circuits and Systems for Video Technology*, 2020.

[19] Peike Li, Yunchao Wei, and Yi Yang. Meta parsing networks: Towards generalized few-shot scene parsing with adaptive metric learning. In *Proceedings of the 28th ACM International Conference on Multimedia*. ACM, 2020.

[20] Xin Li, Yiming Zhou, Zheng Pan, and Jiashi Feng. Partial order pruning: for best speed/accuracy trade-off in neural architecture search. In *CVPR*, pages 9145–9153, 2019.

[21] Xiaodan Liang, Xiaohui Shen, Jiashi Feng, Liang Lin, and Shuicheng Yan. Semantic object parsing with graph lstm. In *ECCV*, pages 125–143, 2016.

[22] Xiaodan Liang, Xiaohui Shen, Donglai Xiang, Jiashi Feng, Liang Lin, and Shuicheng Yan. Semantic object parsing with local-global long short-term memory. In *CVPR*, pages 3185–3193, 2016.

[23] Xiaodan Liang, Ke Gong, Xiaohui Shen, and Liang Lin. Look into person: Joint body parsing & pose estimation network and a new benchmark. *IEEE transactions on pattern analysis and machine intelligence*, 41(4):871–885, 2018.

[24] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, pages 82–92, 2019.

[25] Davide Mazzini. Guided upsampling network for real-time semantic segmentation. *BMVC*, 2018.

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

[27] Marin Orsic, Ivan Kreso, Petra Bevandic, and Sinisa Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. In *CVPR*, pages 12607–12616, 2019.

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

[29] Rudra PK Poudel, Stephan Liwicki, and Roberto Cipolla. Fast-scnn: fast semantic segmentation network. *arXiv preprint arXiv:1902.04502*, 2019.

[30] Eduardo Romera, José M Alvarez, Luis M Bergasa, and Roberto Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):263–272, 2017.

[31] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. In *CVPR*, pages 5639–5647, 2018.

[32] Tao Ruan, Ting Liu, Zilong Huang, Yunchao Wei, Shikui Wei, and Yao Zhao. Devil in the details: Towards accurate single and multiple human parsing. In *AAAI*, volume 33, pages 4814–4821, 2019.

[33] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018.

[34] Albert Shaw, Daniel Hunter, Forrest Landola, and Sammy Sidhu. Squeezenas: Fast neural architecture search for faster semantic segmentation. In *Proceedings of the IEEE international conference on computer vision workshops*, 2019.

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

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

[37] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *ECCV*, pages 325–341, 2018.

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

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