Real-time Segmentation - A Study of Approaches

Christen Millerdurai[1], Matthias Christoph Kruppa[2]
[1]7009670, [2]2572548
`{chmi00002, s8makrup}@teams.uni-saarland.de`

Abstract

Contextual information as well as spatial detail is essential for semantic segmentation. However, to speed up the model inference, most methods abate either spatial detail or semantic detail, which leads to a considerable decrease in accuracy. We investigate various architecture styles and network configuration to motivate high accuracy and high efficiency for real-time semantic segmentation. We conduct experiments on Pascal-VOC and Cityscapes and we reach a maximum mIoU of 0.64 in CityScapes with 21.8fps on a RTX2070. Code and trained models will be made publicly available at here.

1 Introduction

Semantic segmentation is the task of assigning labels to each pixel in an image. It is extensively used in scene parsing, with applications including but not limited to autonomous driving[13, 16], semantic understating[53], aerial surveillance[3].

Ever since inception of fully convolutional networks (FCN)[25], scene parsing frameworks have evolved significantly to bolster the accuracy of the naïve FCN, but majority of them can be categorised into three groups, dilation backbone architecture[9, 8, 7, 6], encoder-decoder architecture[36, 9, 5, 4, 2] and multi-path architecture[41, 44, 43, 34, 19, 24]. The main objective for these methods is to acquire a high-resolution feature representation with high-level semantic context. Also, not all methods emphasize on inference speed and computational cost, since most methods does not fall under real-time segmentation. For instance, in dilation backbone architecture, the Atrous convolution is compute expensive and in encoder-decoder architecture, the down-sample;up-sample blocks brings up heavy computational complexity and memory footprint. However, real-time semantic segmentation demands for, both a low compute and a low memory footprint architecture.

In order to achieve this, the input size is reduced, i.e., a smaller size reduces the number of MACs or network pruning[10, 18, 42] is done, to reduce the number of parameters in the model architecture. Although these methods can improve the computational performance, they reduce the spatial and contextual capacity of the model, leading to a reduction in the accuracy. But, to achieve a high accuracy, both low-level details and high-level semantics are crucial to the semantic segmentation task.

2 Related Work

We review related work on generalized semantic segmentation and real-time segmentation.

2.1 Generalized Segmentation

Before the widespread adoption of FCN[25] based methods, traditional methods[37, 11, 40, 32, 1] utilized hand-crafted features to extract the semantic information. However, recent FCN based methods have dominated the state-of-the-art performance in almost all benchmarks[41, 31, 45, 12, 52].

Most of these methods can be grouped into, (i) dilation backbone architecture; (ii) encoder-decoder architecture; (iii) multi-path architecture.

Dilation backbone architectures[9, 8, 7, 6] preserves high-resolution feature representations by maintaining the feature resolution using methods like atrous spatial pyramid pooling (ASPP)[7], multi-scale context pooling (PSP)[50], etc. And also, some methods use attention mechanisms like self-attention[45, 15], spatial attention[52] and channel attention[47] to capture local regions as well as global context across the feature representation. Meanwhile, the **encoder-decoder architecture**[36, 9, 5, 4, 2] add top-down blocks along with bottom-up blocks using lateral connections to recover high-resolution feature maps at the decoder. And **multi-path architectures**[41, 44, 43, 34, 19, 24], incorporates multiple bottom-up branches to preserve different scales of the feature representation.

While these architectures are capable of representing low-level details and high-level semantics simultaneously, most methods are computationally expensive. But in this study, we want to investigate real-time segmentation, i.e., capturing high spatial and semantic representation with high inference speed.

2.2 Real-time segmentation

Demand for real-time segmentation has been on the rise, as applications like autonomous driving, aerial surveillance, etc. has increased its real-world presence. To promote this, SegNet[4] uses a light weight encoder-decoder architecture and the skip connection to achieve a fast speed. And E-Net[33] devising a lightweight backbone and early down-sampling which delivers high performance. While ICNet[51] uses the image cascade to speed up the algorithm, ERFNet[35] uses residual connections and factorized convolutions to remain efficient while maintaining the accuracy. Meanwhile, ESPNet[29] devises an efficient spatial pyramid dilated convolution for real-time semantic segmentation, while GUN[28] employing a guided upsampling module to fuse the information of the multiresolution input. R2U-Net[2] uses encoder-decoder architecture with recurrent convolutions[30] in every layer along with residual connections[17] to keep the number of parameters same while improving accuracy, and DDRNets[19] which use dual bottom-up branches, with Deep Aggregation Pyramid Pooling Module to enlarge effective receptive fields (ERF)[26] and fuse multi-scale contex.

2.3 Light weight Back-bone

Light weights backbones[49, 27, 21, 38, 20] are used in the encoder i.e., bottom-up part to capture rich contextual feature representations without sacrificing inference speed. In this study, we investigate inverted residual blocks with SE channel attention[20, 22] to reduce the compute and memory footprint, while maintaining a respectable accuracy.

3 Tasks

3.1 Task 1

For this task, a light-weight backbone, *mobilenet-v3*[20] is used as the feature extractor, and to capture multi-scale contextual feature representation, the PSP module[50] is used after stage5 of the mobilenet backbone. This architecture has the same spatial resolution of FCN-32s[25], i.e., 32x down-sampling of the input, which erodes most of finer and smaller details of the feature map across the stages thereby, diminishing the accuracy and clarity of the segmentation map. However, due to the lack of any bells and whistles, the model is light-weight with only 4.18M parameters and extremely fast, mostly due to backbone's opulent use of depth-wise and separable convolutions.

3.2 Task 2

We have used the R2U-Net[2] encoder-decoder architecture, with the configuration of encoder-decoder, $3 \rightarrow 16 \rightarrow 32 \rightarrow 64 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow nclasses$ and temporal steps, $t=3$. This configuration yields us with 1.1M parameters, lesser than the task-1's architecture. However, since we have a complete complementary trunk, i.e., the decoder block, this creates a compute bottleneck, since, the feature map resolution keeps on increasing as we process them through the decoder stages. Furthermore, similar to ENet[33],the last stage, i.e,. stage5 has been dropped in favour of lower compute complexity, and this abandons the down-sampling operations in the last stage, thereby, the ERF of the model is not enough to cover large objects, resulting in a poor discriminative ability.

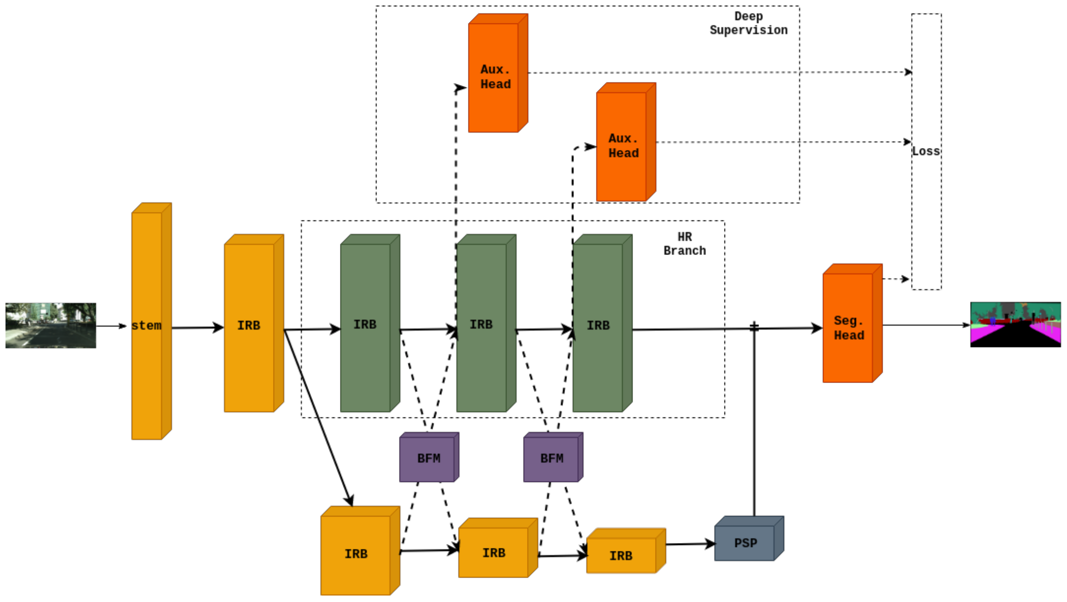


Figure 1: Overview of our network for task-3. "IRB" denotes inverted residual block[20], "BFM" is bilateral fusion module[19] , "PSP" is pyramid scene parsing module[50] and "HR branch" is the high resolution branch. After the PSP module, we concatenate the features of both HR branch and trunk and do a linear projection. These latent features are then sent to "Seg. Head", i.e., segmentation head for predicting the pixel probabilities. And also, the deep supervision block is discarded during inference.

3.3 Task 3

Taking inspirations from multi-path architecture, especially DDRNet[19], we use a dual-path network architecture, with the first few stages shared by both the paths to lower the compute complexity. Moreover, if we use two separate paths, there could be low-level features that are captured redundantly in both the paths. And Bilateral Fusion Module (BFM)[19] is used to fuse the features across the two branches at various stages. We also use the PSP Module[50] for capturing multi-scale contextual information in the semantic branch. And each stage is made up of 2 blocks of inverted residual blocks with SE channel attention[20, 22] for efficient feature encoding. We have illustrated our network in Figure 1. Even though this network yields us with 3.36M parameters, a significant increase from the R2U-Net above, it requires less computational FLOPS due to, (i) Early down-sampling; (ii) Shared branches; (iii) depth-separable convolutions[20].

4 Methodology

First, we illustrate the datasets that we use to evaluate our tasks. Second, we highlight all the implementation details for training the tasks. Finally, we explain the inference evaluation criterion.

4.1 Datasets

We perform tests on two datasets, PASCAL VOC 2012[14] for task 1, and Cityscapes[13] for task 2 and task 3. And for evaluating the tasks, we use IoU, accuracy, f1-score, sensitivity, dice scores for Cityscapes, and for PASCAL VOC, in addition to the above, we also compute the one-vs-one-combination (ovo) ROC-AUC score.

- **PASCAL VOC 2012** contains 20 foreground object classes and one background class. The dataset is split into 1464 (train), 1449 (val), and 1456 (test) pixel-level annotated images.

For each test image, we need to predict the object class of each pixel, or give it 'background' status if the object does not belong to one of the twenty specified classes.

- **Cityscapes** is tasked for urban scene understanding, which contains 30 classes and only 19 classes of them are used for scene parsing evaluation. The dataset contains 5,000 high quality pixel-level finely annotated images and 20,000 coarsely annotated images. The finely annotated 5,000 images are divided into 2,975/500/1,525 images for training, validation and testing respectively. And in our tests, we only use the finely annotated images.

4.2 Implementation Details

4.2.1 Training settings

For all experiments on Cityscapes, we train with the input as random image crops at 0.5 scale of the native resolution of 2048x1024. Each experiment in trained until the validation accuracy remains stable for 5 epochs, with at most 100 epochs per experiment. We have developed a prototyping framework for models, where any change to the model, will generate appropriate model/validation/training logs automatically. We extensively use this to log our experiments.

For the experiment on PASCAL VOC 2012, we train the model to 100 epochs with random crops at 512x512 image resolution and plot the metrics for each epoch.

For both the datasets, we train on a RTX2070 GPU with batch size of, 8 for PASCAL VOC and 4 for Cityscapes. And we use adam optimizer[23] with initial learning rate set as 0.001 for training. Similar to [50, 46], we also apply data augmentation techniques like random horizontal flips, random scale[0.5, 2], and random crops. While we first trained the model with variant of IoU Loss[54] suitable for multi-class segmentation, we finalized the loss to be cross-entropy wit Online Hard Example Mining (OHEM)[39]. We also incorporated Deep Supervision like [44, 19] with auxiliary segmentation heads, and minimize their loss along with the main loss. The auxiliary losses are weighted with 0.6 and 0.4 respectively. And during inference, we deactivate these auxiliary branches.

4.2.2 Inference evaluation criterion

Our test bench consists of a Nvidia RTX 2070, 32GB RAM and 6-core CPU. We use FLOPs as a measure of inference speed. A single inference is a batch consisting of a single image with a resolution of 2048x1024, i.e the native resolution of Cityscapes. We also record the frames per second (fps) for a model by running the forward pass for 100 times, to prevent any inconsistencies. Also, all experiments are evaluated in single-scale mode.

5 Experiments

For Pascal VOC 2012, we use mean f1-score, mean AUC score, mean dice score for estimating the models performance. While for Cityscapes, we calculate the mean as well as class-wise metrics for accuracy(ACC), f1-score(F1), sensitivity(S), jaccard score(JS), dice score(DS) and intersection-over-union(IoU).

Also for Pascal VOC 2012, we consider the background class(0) is negative, whereas 1-20 classes are positives. Meanwhile for Cityscapes, we consider ignore-class(255) as a negative, and 0-19 classes as positives. We use these assumptions to calculate the evaluation metrics. And, for IoU calculation, we ignore the background class to promote an equal comparison to other benchmarks in Cityscapes.

We benchmark task-1 with PASCAL VOC 2012, with input resolution of 512x512, and achieve a mean f1-score of 87.57, evaluated on the train set. The f1-score, AUC and dice metrics for every epoch is shown in Figure 2. We also achieve an average of 125 fps on PASCAL VOC's 512x512 input size. The inference speed of task-1's network is shown in Table 3. Even though this network contains the most number of parameters, it also has the least computational requirement of 9.01 GFLOPs.

We evaluate task-2 and task-3 on Cityscapes, with an input resolution of 2048x1024. The class-wise metrics and mean metrics for task-2 are illustrated in Table 1a and Table 2 respectively. From, task-3's class-wise metrics Table 1b and mean metrics Table 2, we see that task-3's network achieves higher metrics in all areas compared to task-2. Moreover, as shown in Table 3, task-3 has a higher inference speed than task-2 even though task-2 has lesser number of parameters, ascribing to task-3's

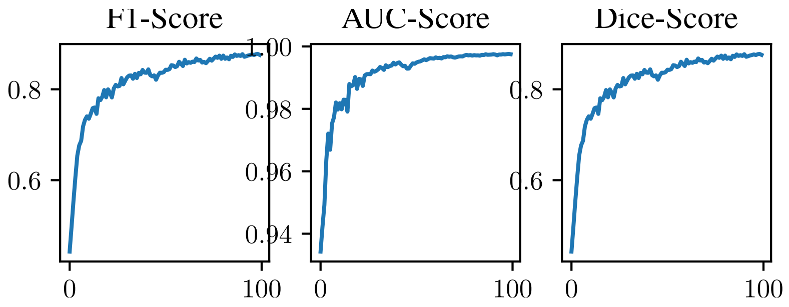
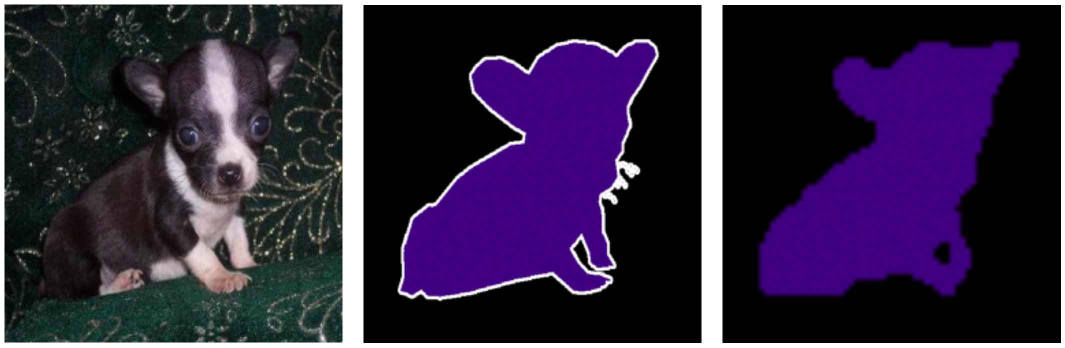
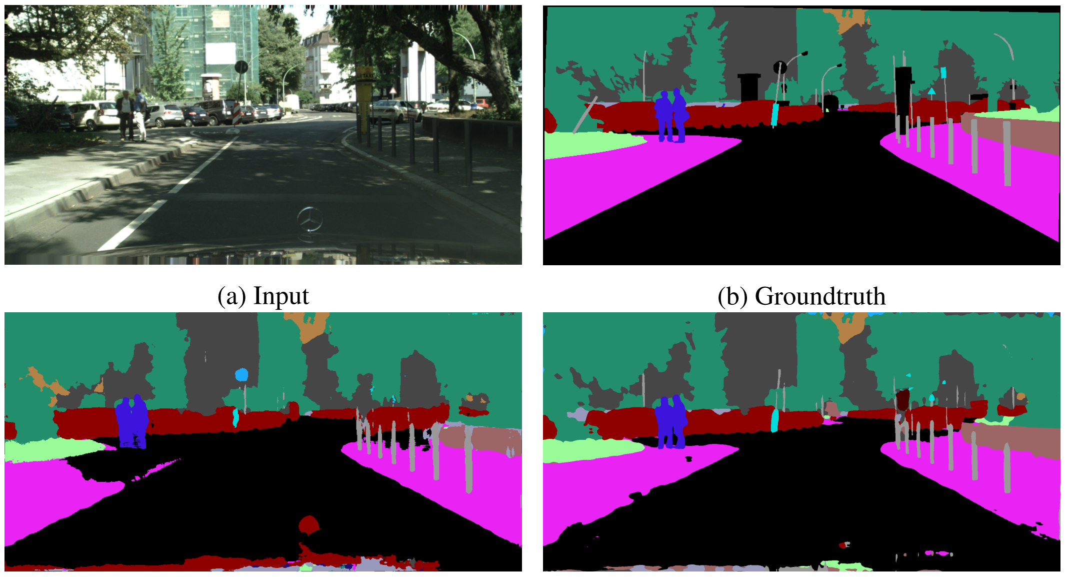


Figure 2: Mean f1-score, mean AUC, mean dice scores of task-1 on PASCAL VOC 2012. The metrics are plotted against the number of epochs, i.e., 100.



(a) Input (b) Groundtruth (c) Prediction

Figure 3: An example illustrating Task 1's performance in PASCAL VOC 2012.



(a) Input (b) Groundtruth

(c) Prediction of Task 2 (d) Prediction of Task 3

Figure 4: Visualization examples on the Cityscapes validation set produced from Task 2 and Task3. The Task 2 model has difficulty in capturing the roads, but task 3 model captures it properly, and this could be due to the absence of larger contextual representation in task 2's model.

lesser computational demand. Also, the encoder-decoder architecture in task-2, causes a huge strain in memory bandwidth due to the large resolution of the feature maps, which further reduces the inference speed.

Table 1: Class-wise metrics of Cityscapes. We report the results on validation set. Notation: ACC - Accuracy, F1 - F1score, S - sensitivity, JS - Jaccard score, DS - Dice score, IoU - Intersection over Union.

(a) Task 2

class	ACC	F1	S	JS	DS	IoU
road	0.8	0.89	0.83	0.8	0.88	0.96
sidewalk	0.6	0.75	0.8	0.57	0.7	0.7
building	0.78	0.88	0.91	0.76	0.86	0.85
wall	0.15	0.27	0.79	0.1	0.15	0.16
fence	0.22	0.37	0.52	0.13	0.19	0.28
pole	0.35	0.52	0.88	0.33	0.48	0.37
traffic light	0.25	0.4	0.89	0.16	0.25	0.26
traffic sign	0.46	0.63	0.82	0.43	0.58	0.51
vegetation	0.85	0.92	0.96	0.83	0.9	0.88
terrain	0.37	0.54	0.71	0.21	0.29	0.43
sky	0.84	0.91	0.92	0.76	0.83	0.9
person	0.56	0.71	0.93	0.4	0.53	0.58
rider	0.05	0.09	0.98	0.03	0.06	0.05
car	0.67	0.8	0.77	0.61	0.74	0.83
truck	0.01	0.02	0.7	0.01	0.01	0.01
bus	0.23	0.37	0.78	0.09	0.12	0.24
train	0.12	0.22	0.61	0.01	0.02	0.13
motorcycle	0.11	0.2	0.58	0.04	0.06	0.12
bicycle	0.48	0.65	0.74	0.33	0.45	0.58

(b) Task 3

class	ACC	F1	S	JS	DS	IoU
road	0.79	0.88	0.81	0.79	0.88	0.96
sidewalk	0.61	0.76	0.78	0.63	0.76	0.74
building	0.81	0.89	0.9	0.8	0.88	0.89
wall	0.31	0.48	0.59	0.26	0.37	0.4
fence	0.37	0.54	0.69	0.25	0.36	0.44
pole	0.43	0.6	0.84	0.41	0.57	0.46
traffic light	0.47	0.64	0.74	0.43	0.57	0.56
traffic sign	0.6	0.75	0.83	0.59	0.73	0.68
vegetation	0.88	0.94	0.97	0.87	0.93	0.91
terrain	0.52	0.68	0.87	0.38	0.51	0.56
sky	0.86	0.93	0.93	0.84	0.91	0.92
person	0.68	0.81	0.92	0.61	0.75	0.73
rider	0.45	0.62	0.94	0.37	0.5	0.46
car	0.86	0.93	0.95	0.84	0.91	0.9
truck	0.42	0.59	0.88	0.21	0.27	0.44
bus	0.67	0.81	0.97	0.36	0.42	0.69
train	0.51	0.68	0.94	0.09	0.11	0.53
motorcycle	0.27	0.43	0.82	0.16	0.23	0.29
bicycle	0.6	0.75	0.82	0.51	0.66	0.69

Table 2: Mean-metrics on Cityscapes. We report the results on validation set.

Method	mACC	mF1	mS	mJS	mDS	mIoU
Task 2	0.42	0.53	0.8	0.35	0.43	0.47
Task 3	0.58	0.72	0.85	0.49	0.6	0.64

Table 3: Compute requirement and speed comparison on Cityscapes. We calculate GFLOPs with a 2048×1024 image as input. And we use a RTX2070 for the evaluations.

Method	Params	GFLOPs	Speed (FPS)
Task 1	4.18M	9.01	74.7
Task 2	1.1M	43.23	3.9
Task 3	3.36M	26.79	21.8

5.1 Ablative Evaluation on Cityscapes

We also played with different context aggregation modules, OCModule[46], AttentionPSPModule[48], DAPPModule[19]. But they added additional flops, which could not constitute the modicum increase in accuracy. So, we stuck with the PSPModule[50] for context aggregation. We also, noted different training configurations such, single branch, loss - dice loss;cross-entropy and auxiliary loss heads. We have illustrated the various configurations in Table 4.

When we only use the semantic branch, i.e the backbone and minimize the loss with cross entropy (CE), we achieve only 0.38 jaccard score (JS). Upon visual inspection of the segmentation output, pixels of small objects like traffic lights, signs are lost. Also, most of the objects are messy blobs with no definition as illustrated in Figure 5. But after adding a high resolution (HR) branch after stage2, which maintains the spatial resolution along the forthcoming stages, we reach a substantial increase in JS with 0.45, and also the segmentation output becomes more defined. We apply Online Hard Example Mining (OHEM)[39] along with CE to boost the JS to 0.47. And adding the two auxiliary heads for Deep Supervision (DS), increases the JS to 0.48. And finally, to capture contextual features in multiple scales, we use PSP Module after stage5 of the backbone, thereby bringing the JS to 0.49.

We believe the context aggregation module (CAM) and the feature fusion module (FFM) still has scope for improvement. In our experiments, Attention-PSP (APSP) did not produce any discernible improvement in the performance, but it could be due to fact that the Q, K, V vectors were projected in

Table 4: Detailed performance analysis on the validation set of Cityscapes. Notation: Trunk - Semantic branch, HR - High resolution branch, CE - Cross entropy, DS - Dice loss, AuxH - Auxiliary heads, CAM - Context aggregation module, JS - Jaccard score.

Trunk	HR	Loss	OHEM	AuxH	CAM	JS	GFLOPs
✓		CE				0.38	19.25
✓	✓	CE				0.45	26.72
✓	✓	DS				0.45	26.72
✓	✓	CE	✓			0.47	26.72
✓	✓	CE	✓	✓		0.48	26.72
✓	✓	CE	✓	✓	PSP	0.49	26.79
✓	✓	CE	✓	✓	APSP	0.49	26.77
✓	✓	CE	✓	✓	DAPP	0.49	28.06

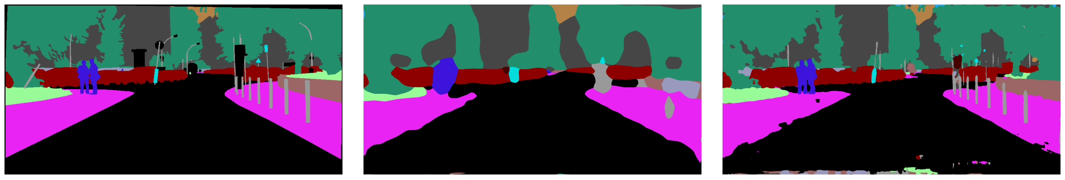

(a) Groundtruth (b) Semantic Branch (Trunk) (c) Semantic + HR Branch

Figure 5: An example illustrating the improved detail after HR branch is used along with the semantic branch.

a low feature space. The BFM, albeit inspired by DDRNets[19] might not be the most efficient FFM. The vanilla convs in the BFM could be replaced by factorized or depth-separable convs to improve the performance.

6 Concluding Remarks

We observe that the semantic segmentation task requires both low-level details and high-level semantics. Even though, task-2's R2Unet has higher feature map resolution than task-3's network, the network lacks contextual detail, due to lack of stage5 in encoder and also due to low capacity of the encoder-decoder, which creates a mIoU difference of 17%. The dual-path way network in task-3 provides achieves a more efficient method to solve semantic segmentation than task-2's encoder-decoder architecture with better accuracy as well as faster inference speed.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012. doi: 10.1109/TPAMI.2012.120.

[2] M. Z. Alom, M. Hasan, C. Yakopcic, T. M. Taha, and V. K. Asari. Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation, 2018.

[3] S. M. Azimi, C. Henry, L. Sommer, A. Schumann, and E. Vig. Skyscapes – fine-grained semantic understanding of aerial scenes, 2020.

[4] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation, 2016.

[5] A. Chaurasia and E. Culurciello. Linknet: Exploiting encoder representations for efficient semantic segmentation. *2017 IEEE Visual Communications and Image Processing (VCIP)*, Dec 2017. doi: 10.1109/vcip.2017.8305148. URL http://dx.doi.org/10.1109/VCIP.2017.8305148.

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs, 2016.

[7] 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, 2017.

[8] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation, 2017.

[9] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation, 2018.

[10] X. Chen, Y. Wang, Y. Zhang, P. Du, C. Xu, and C. Xu. Multi-task pruning for semantic segmentation networks, 2020.

[11] E. A. Chiba, M. A. G. de Carvalho, and A. L. da Costa. Graph cut and image segmentation using mean cut by means of an agglomerative algorithm. In *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, volume 1, pages 708–712, 2014.

[12] S. Choi, J. T. Kim, and J. Choo. Cars can't fly up in the sky: Improving urban-scene segmentation via height-driven attention networks, 2020.

[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 *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[15] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu. Dual attention network for scene segmentation, 2019.

[16] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[18] W. He, M. Wu, M. Liang, and S.-K. Lam. Cap: Context-aware pruning for semantic segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 960–969, January 2021.

[19] Y. Hong, H. Pan, W. Sun, S. Member, IEEE, and Y. Jia. Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes, 2021.

[20] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam. Searching for mobilenetv3, 2019.

[21] 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, 2017.

[22] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu. Squeeze-and-excitation networks, 2019.

[23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[24] S. Kumaar, Y. Lyu, F. Nex, and M. Y. Yang. Cabinet: Efficient context aggregation network for low-latency semantic segmentation, 2020.

[25] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation, 2015.

[26] W. Luo, Y. Li, R. Urtasun, and R. Zemel. Understanding the effective receptive field in deep convolutional neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/c8067ad1937f728f51288b3eb986afaa-Paper.pdf.

[27] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design, 2018.

[28] D. Mazzini. Guided upsampling network for real-time semantic segmentation, 2018.

[29] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation, 2018.

[30] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3367–3375, 2015. doi: 10.1109/CVPR.2015.7298958.

[31] R. Mohan and A. Valada. Efficientps: Efficient panoptic segmentation, 2021.

[32] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979. doi: 10.1109/TSMC.1979.4310076.

[33] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation, 2016.

[34] R. P. K. Poudel, U. Bonde, S. Liwicki, and C. Zach. Contextnet: Exploring context and detail for semantic segmentation in real-time, 2018.

[35] 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*, 19(1):263–272, 2018. doi: 10.1109/TITS.2017.2750080.

[36] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[37] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, Aug. 2004. ISSN 0730-0301. doi: 10.1145/1015706.1015720. URL https://doi.org/10.1145/1015706.1015720.

[38] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[39] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining, 2016.

[40] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991. doi: 10.1109/34.87344.

[41] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao. Deep high-resolution representation learning for visual recognition, 2020.

[42] Z. You, K. Yan, J. Ye, M. Ma, and P. Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks, 2019.

[43] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation, 2018.

[44] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation, 2020.

[45] Y. Yuan, X. Chen, and J. Wang. Object-contextual representations for semantic segmentation, 2020.

[46] Y. Yuan, L. Huang, J. Guo, C. Zhang, X. Chen, and J. Wang. Ocnet: Object context network for scene parsing, 2021.

[47] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal. Context encoding for semantic segmentation, 2018.

[48] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks, 2019.

[49] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices, 2017.

[50] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network, 2017.

[51] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia. Icnet for real-time semantic segmentation on high-resolution images. In *ECCV*, 2018.

[52] H. Zhao, Y. Zhang, S. Liu, J. Shi, C. C. Loy, D. Lin, and J. Jia. Psanet: Point-wise spatial attention network for scene parsing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[53] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ade20k dataset, 2018.

[54] D. Zhou, J. Fang, X. Song, C. Guan, J. Yin, Y. Dai, and R. Yang. Iou loss for 2d/3d object detection, 2019.