FENet: Fast Real-time Semantic Edge Detection Network

Yang Zhou[1], Rundong Ge[1], Gary McGrath[2], and Giuseppe Loianno[1]

***Abstract*— Semantic edge is a geometric-aware semantic feature that can be leveraged in robotic perception systems for increased situational awareness and high-level environment understanding. This sparse representation nicely encapsulates the semantic information (object categories) within the geometric object boundaries. State-of-the-art semantic edge detection approaches require significant computation power and fail to approach real-time performance on embedded devices for robotic applications. In this paper, we present FENet (Fast Real-time Semantic Edge Detection Network), a semantic edge detection approach for robots with Size, Weight, and Power (SWaP) constraints. Specifically, we adopt MobileNetV2 as a lightweight backbone network, and we utilize joint pyramid upsampling to improve the system performance. We further reduce the model complexity and latency through network pruning and multiple upsampling strategies to adapt the model on embedded devices such as NVIDIA Jetson TX2. The proposed method is evaluated on Cityscapes with accuracy performances close to the state-of-the-art methods, but with substantially reduced computational complexity that speeds up the network by a factor of 10. To the best of our knowledge, FENet is the first real–time semantic edge detection network for robotic platforms.**

I. Introduction

Semantic perception is crucial to increase robots' situational awareness and high-level environment understanding by providing a compact and lightweight geometric and semantic environment representation as shown in Fig. 1. This facilitates missions' execution in complex and unstructured environments by enabling robots to navigate in various outdoor and indoor settings and solve complex tasks such as inspection, environment mapping, and search and rescue. Semantic scene understanding can also be beneficial in human-robot interaction to increase the operator's awareness of environment regions containing relevant object categories.

In the past decades, different levels of scene understanding, including object detection and semantic segmentation, have been explored by the computer vision and robotics communities. Recently, semantic edge as a semi-dense semantic representation started to gain attention. Semantic edge provides pixel-wise category information on object boundaries, which produce semi-dense scene understanding as shown in Fig. 1. CASENet [1] formulated the task as a multi-label problem by associating each pixel with multiple semantic categories, which represent the intersection of object boundaries. Other works [2], [3], [4] improved the performance by introducing different alignment mechanisms and feature fusion strategies. Moreover, [5] stepped further to provide an instance-level semantic edge representation.

There are several key advantages provided by semantic edge detection compared to other semantic representations. First, semantic edge provides an ideal data association with semantic information and geometric edge representation, which can lead to semantic-geometric consistent semi-dense mapping for search and rescue tasks. Second, semantic edges provide more details on the objects' boundary, which is crucial for accurate localization and mapping as well as planning. Third, since the edges often belong to several different categories, the multi-class nature of semantic edges provides useful uncertainty property for robotic applications, which is relatively hard to obtain by post-processing semantic segmentation results. Overall, these properties clearly show the advantage of this compact but meaningful geometric-semantic representation for robotic systems in complex scenarios such as monitoring and search and rescue.

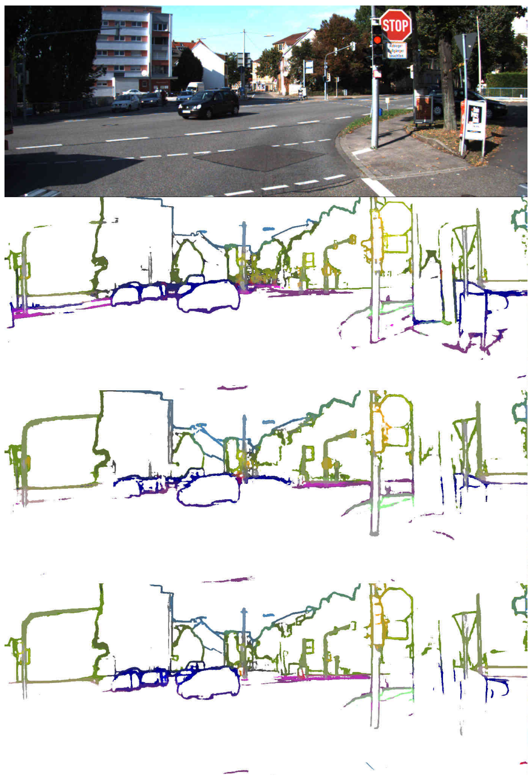
Fig. 1: From top to bottom: DFF (ResNet-50), DFF (MobileNetV2), FENet (ours). Semantic edge maps on KITTI dataset are produced by the network trained on Cityscapes dataset without any finetuning. Our real–time network shows competitive performances compared to existing methods.

This work was supported by Qualcomm Research, the ARL grant DCIST CRA W911NF-17-2-0181, and the Technology Innovation Institute.

[1]The authors are with the New York University, Tandon School of Engineering, Brooklyn, NY 11201, USA. email: {yangzhou, rundong.ge, loiannog}@nyu.edu.

[2]The author is with Qualcomm Technologies, Inc., 5775 Morehouse Drive, San Diego, USA. email: gmcgrath@qti.qualcomm.com.

The current state-of-the-art methods are computationally expensive and rely on large backbone networks such as ResNet-50 and ResNet-101. These require high-end GPUs which are generally not available on robotic platforms preventing to obtain real-time performance. This is necessary in robotics to deal with real-time control and planning. To the best of our knowledge, there is no prior work addressing the efficiency of semantic edge detection. Without a lightweight approach, it is infeasible to deploy this approach in real-world settings on computationally constrained robots.

In this work, we propose the first lightweight semantic edge detection network with a reasonable balance between efficiency and performance. This paper presents multiple contributions. First, we employ a MobileNet-V2 based encoder to extract features, and we improve the efficiency of the multi–scale adaptive weight fusion module by optimizing the memory access operation. Second, inspired by FastFCN [6], we introduce a joint pyramid upsampling module to improve the algorithm performance on embedded devices. Finally, we further compress the encoder network by adopting NetAdapt [7] to reduce the inference time and the number of parameters. We also propose two different variants of the network to address the latency issue and memory usage. FENet approaches real–time performance on NVIDIA Jetson TX2 with 320×320 resolution image.

The paper is organized as follows. In Section II, an overview of the state of the art semantic segmentation approaches is provided. Section III introduces the strategy to obtain real–time semantic edge segmentation. Section IV presents extensive results on open-source datasets. Section V concludes the work and presents multiple future scenarios.

II. Related works

A. Semantic Edge Detection

Category-aware edge detection was first proposed in [9] assigning semantic class to each boundary pixel. The authors used an inverse detector to fuse top-down detector information and bottom-up edge information. CASENet [1] instead formulated the problem by associating the multi-class label with each boundary pixel. CASENet fused the low-level feature maps with the last high-level feature maps to produce category-aware semantic edges. Semantic edge has been used as well in different applications. VLASE [10] employed semantic edges to achieve on-road localization for vehicles. UPI-Net [11] further applied semantic edge detection in medical ultrasound image analysis. Dedge-AGMNet [12] introduced depth edge, which is a binary semantic edge of instance-sensitive as a feature of the stereo matching branch to improve the performance of stereo depth estimation. PEN [5] blended the instance-level object detection with the detected semantic edge to generate an instance-level edge map. To improve the accuracy of semantic edge detection, SEAL [2] formulated a probabilistic model to align edges using latent variable optimization. However, in this approach, it is very time-consuming to align ground truth edges and learn semantic edge simultaneously. STEAL [3] reasons the annotation noise during training and enforces the network to predict the edges along with the normal directions by a novel network layer and loss function. This approach can produce sharper edge prediction results and can be used to refine the dataset's noisy annotations. DFF [4] proposed an adaptive weight fusion module to adjust the weight of feature maps from different levels according to the input image and locations, which improved the performance significantly. Recently, [13] presented a joint multi-task learning method for semantic segmentation and semantic edge detection, which utilized a novel spatial gradient fusion to suppress unrelated edges.

However, all the aforementioned works rely on deep backbone deep convolution networks, which are computationally expensive and memory consuming. These aspects prevent deploying them on embedded robotic platforms with SWaP constraints. In this work, we propose a lightweight semantic edge network. Our approach presents reduced latency and computational complexity, while obtaining comparable performance and accuracy with respect to existing state-of-the-art approaches as shown in Fig. 1.

B. Efficient network for platforms with SWaP constraints

Efficient network design has been explored widely in prior works. The goal of the efficient network structure is to reach an ideal trade-off between accuracy and efficiency. SqueezeNet [14] used 1×1 convolutions with squeeze and expansion operations to reduce the number of parameters. MobileNetV1 [15], which was proposed for image classification, is comparable to VGG-16 [16] with a substantially reduced number of parameters. To achieve this result, MobileNetV1 utilized depthwise separable convolution. MobileNetV2 [8] further reduced the amount of computation resources with respect to MobileNetV1 by introducing inverted residuals block and linear bottlenecks. To maximize the channel information, ShuffleNet [17] proposed group convolution and channel shuffle operations, which reduce the computation while exploiting the potential of channels. ShuffleNetV2 [18] employed point-wise group convolution instead of group convolution and proposed to evaluate the direct metric on target platform instead of considering FLOPs.

We adopt MobileNetV2 as lightweight backbone network to speed up the feature extraction module which takes, in the semantic edge detection, most computation resources.

III. Methodology

A. Network Architecture Overview

The overview of the network architecture is illustrated in Fig. 2. We design a lightweight backbone based on MobileNetv2 [8] in order to keep real–time performance. The feature extraction module extracts features from multiple levels. The joint pyramid upsampling module upsamples the high–level feature maps utilizing information from the last three feature maps. We use deconvolution to upsample these edge maps to the original resolution, and we propose as well two different upsampling strategies to recover the original resolution and reducing latency and memory consumption. The adaptive weight fusion learns the fusion weight of multi–level features and produces the final semantic edge map.

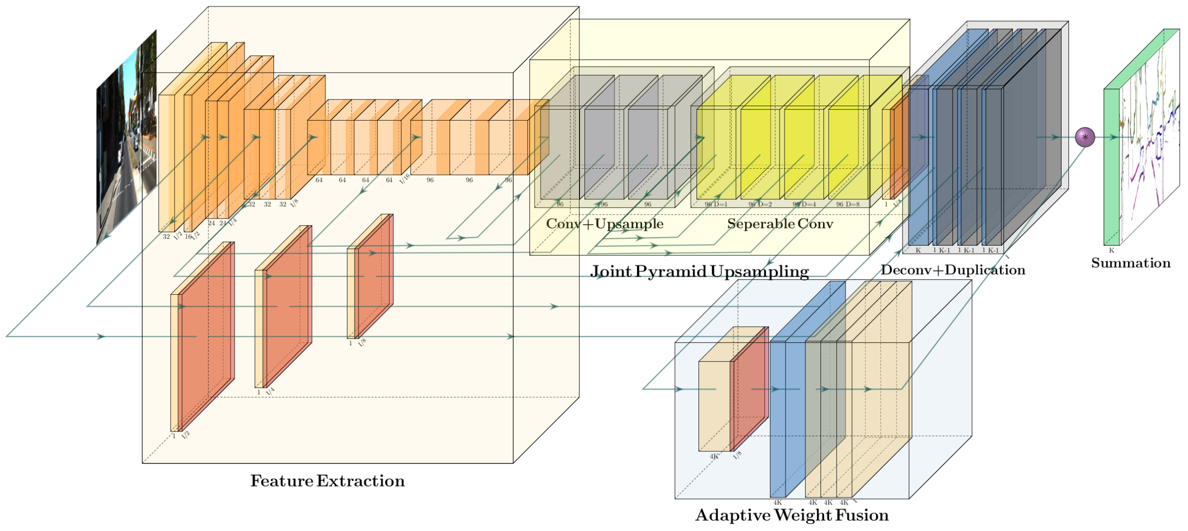

Fig. 2: FENet Network Structure: convolution layer (yellow block), bottleneck module from MobileNetV2 [8] (orange block), batch normalization layer (red block), bilinear upsampling module followed by convolution layer (purple block), separable convolution (yellow block), deconvolution layer (blue block), duplication of upsampled edge map (black block), and summation operation (green block). In FENet-fast, the deconvolution is substituted by bilinear sampling. In FENet-small, bilinear sampling only upsamples to half size, and recover the full resolution with an additional bilinear upsampling.

B. *Feature extraction module*

To enable the network to run in real-time on embedded devices such as NVIDIA Jetson TX2, we employ a lightweight feature extraction module based on MobileNetV2 backbone to extract feature maps from different stages of the encoder. Although ResNet-50 and ResNet-101 are commonly used in other semantic edge detection, these backbone networks can hardly execute real–time semantic segmentation with 320×320 resolution images. MobileNetV2 uses depthwise separable convolutions, which require less computation than standard convolutional layers. For example, the computation cost of a depthwise separable convolution is only 12% compared to a 3×3 standard convolution. In addition, shortcuts are also used between bottlenecks to improve gradient propagation across multiple layers.

The first convolution layer uses a 3-by-3 convolution with stride 2 to downsample the input image at the starting stage. In the robotic system's mobile embedded platform, the GPU memory or shared memory condition is critical because it has to be shared with other real–time modules as well (planning, control, and mapping). Apart from the limited memory condition, doubling the size of feature maps will also lead to a quadratic increment of execution time [19]. Since the resolution of the last layer of feature maps is small, and it contains fewer location information than previous layers of feature maps, we follow the design of CASENet [1] to generate edge maps from the first, second, third, and fifth stages (according to the size of feature maps) using a convolution as the encoder. The channel sizes of the edge maps from the first three stages are 1, and the channel size of the edge map from the fifth stage is the number of semantic categories. These low-level edge maps will guide the final semantic-aware edge map by providing rich location and detailed information to generate sharp semantic edges.

C. *Joint pyramid upsampling module*

To enlarge the feature map of the last layer to compensate for information loss during upsampling, we introduce a joint pyramid upsampling module shown in yellow in Fig. 2, which was first proposed by FastFCN [6]. We consider the feature maps from the last three stages as the input of the joint pyramid upsampling module to generate a multi–scale feature with the same size of the feature map as the third feature map. In this application, this operation will double the size of feature maps to be passed to the next module. Initially, each feature map goes through a convolution layer with the same channels. Then each feature map is upsampled to the final size layer and concatenated together. The ensembled features are processed by four separable convolutions with dilation rates 1, 2, 4, and 8 and $1/4$ channels of the ensembled feature map. The final multi–scale feature map of the joint pyramid upsampling module is obtained by concatenating these four feature maps.

The joint pyramid upsampling module builds a multi–scale feature map with a larger size compared to the fifth stage one. This stage embeds the location information from the third and fourth feature maps into the semantic-aware feature map and speeds up the upsampling procedure.

D. *Upsampling Strategies*

In Fig. 2, we use deconvolution to upsample edge maps to the original resolution. However, the embedded platform has generally limited computation resources, memory size, and speed. To further address the SWaP constraints, we propose two different upsampling strategies. The feature map consumes a substantial amount of memory and computational resources with deconvolution, especially when the upsample scale factor is large, requiring large kernel deconvolution. For example, the deconvolution layer after the fifth feature map of a MobileNetV2-based DFF needs to use a 32×32 kernel to upsample by 16 times. Bilinear upsampling requires no extra parameter, and this simple operation is easy to compute even without the GPU's support. The first variant FENet-fast uses a bilinear upsampling operator instead of deconvolution in the network. The parameter size of the network is not the only metric related to the memory issue. The forward memory size is a crucial metric for the embedded platform such as NVIDIA Jetson TX2, which only has 8 GB of memory shared across CPU and GPU. Therefore, we propose a second variant FENet-small, which only upsamples the feature map to half of the original resolution before the final fusion module. The original resolution size is upsampled at the last step. FENet- small substantially reduces the forward memory size as shown in Section. IV-D.

E. Memory-friendly adaptive weight fusion module

DFF [4] first proposed the adaptive weight fusion module (light blue block in Fig. 2). After using the deconvolution layer to upsample edge maps from stage 1, 2, 3, and multi–scale edge maps from the joint pyramid upsampling module, DFF follows CASENet to use shared concatenation to share low–level feature maps with semantic–aware feature map of different classes. However, the operation of shared concatenation is expensive in terms of memory resources and access. In this work, to avoid this expensive operation, we utilize the idea of weighting. Let us denote K as the number of classes, H as the height, and W as the width. We duplicate these three low-level edge maps with $1 \times H \times W$ channel by K times and concatenate them with the semantic-aware edge map, which has $K \times H \times W$ channels. After obtaining the $4K \times H \times W$-channel features, we follow the design of the original adaptive fusion module to generate $4K \times H \times W$-channel weight maps from the multi-scale feature map using convolution layers. To be noticed, the result $4K \times H \times W$-channel map needs to be reordered to $K \times 4 \times H \times W$ after multiplying the concatenated $4K \times H \times W$-channel feature with the adaptive weight maps. Finally, summation operation across the second channel can produce the final semantic edge map with channels of $K \times H \times W$. Applying channel duplication instead of shared concatenation increases the cache hit rate and improve the speed by 20.8%.

F. Loss Function

The network is trained with the same re–weighted cross-entropy loss introduced by CASENet [1], where $\mathbf{Y}_k$ is the predicted edge probability of class k, $\overline{\mathbf{Y}}_k$ is the binary edge ground truth of class k, $\mathbf{p}$ represents the pixel location, and γ is the non-edge pixels percentage

$$\begin{aligned}\mathcal{L} = \sum_k \mathcal{L}_k = \sum_k \sum_{\mathbf{p}} \{ & -\gamma \overline{\mathbf{Y}}_k(\mathbf{p}) \log \mathbf{Y}_k(\mathbf{p}) \\ & -(1-\gamma)\left(1 - \overline{\mathbf{Y}}_k(\mathbf{p})\right) \log\left(1 - \mathbf{Y}_k(\mathbf{p})\right)\}\end{aligned} \tag{1}$$

We combine the result of loss function on the fifth–level feature and the final edge maps with the same weight to get the final training loss.

G. Network Pruning

The efficiency of the proposed network can be further improved by network pruning, which removes redundant parameters according to the task's characteristics to solve the over-parameterized issue. From the perspective of weight, the approach presented in [20] used L1–norm of the filters' weight to identify redundant parameters. From the activation perspective, [21] removed filters according to the average percentage of zeros in the activation. However, these two works are not guided by direct metrics on the target device. We perform network pruning on the encoder network using NetAdapt [7], which builds a latency table of each layer on target device to remove filters in a greedy strategy. After the whole network is trained, NetAdapt removes redundant filters in each iteration until the target latency is reached. It generates proposals that reduce filters in each layer. After fine–tuning them with few epochs, the best one in terms of accuracy-resource trade–off is adopted in the next iteration. The algorithm ends once the target latency is reached. The pruned model is further fine–tuned for better accuracy.

IV. Experimental results

A. Implementation

We implement our proposed method in PyTorch with the support of synchronized batch normalization [22]. We evaluate our methods on fine–annotated subsets of Cityscapes [23] dataset, which includes 2975 training samples, 500 validation samples, and 1525 test samples. We treat all training samples as the training set, and all validation samples as the test set since semantic segmentation ground truth of test samples are not provided. We generate semantic edge ground truth following the methods in CASENet [1]. We employ 640×640 image resolution to train our network. Various data augmentation strategies are applied in our method: we resize the image with a scaling factor from 0.75 to 2 randomly, and we also apply random mirroring and random cropping. We test our network with an original image resolution of 2048×1024. We pre–train our backbone network on ImageNet [24] first. Subsequently, we train the network for 200 training epochs by employing the SGD optimizer. The base learning rate of the backbone network and other modules are 0.08 and 0.8. We train the network with synchronized batch normalization with batch size 8. The selected learning policy is polynomial learning rate decay with 0.9 decay weight. This training strategy is applied to all the experiments for a fair comparison.

B. Evaluation

We adopt the evaluation protocol in SEAL [2], which is stricter than other protocols [1], [9]. This protocol matches the raw predictions with unthinned ground truths in addition to matching the thin predictions with thinned ground truths. We set the matching distance tolerance to 0.0035 for the Cityscapes dataset. We use two metrics to evaluate our method's performance: maximum F-measure at optimal dataset scale (MF-ODS) [25] and average precision (AP).

Method	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic Light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle	Mean
CASENet	54.58	65.44	67.75	37.97	39.93	57.28	64.65	69.38	71.27	50.28	73.99	72.56	59.92	66.84	35.91	56.04	41.19	46.88	63.54	57.65
STEAL	90.86	78.94	77.36	43.01	42.33	71.13	75.57	77.60	81.60	56.98	87.30	83.21	66.79	91.59	45.33	66.64	46.25	52.07	74.41	68.89
DFF (ResNet-101)	37.85	66.65	65.28	42.36	44.81	52.34	60.27	71.58	70.67	49.22	79.12	71.68	60.57	57.53	51.03	66.58	45.23	49.20	63.42	58.18
DFF (ResNet-50)	29.30	53.41	58.35	31.68	38.49	35.56	64.56	50.72	64.37	49.40	68.83	53.11	58.01	24.91	31.63	55.17	32.06	47.49	58.71	47.67
DFF (MobileNetV2)	**40.73**	**58.11**	**57.77**	36.15	35.09	**40.49**	**54.64**	**56.43**	**61.08**	**47.33**	**58.79**	**58.99**	47.07	**38.37**	13.55	34.01	9.75	28.87	**52.91**	43.69
FENet (Ours)	35.28	48.89	55.62	**37.29**	**38.40**	35.48	53.96	55.20	59.12	46.92	53.43	51.65	**49.36**	31.92	**22.75**	**46.50**	**23.16**	**38.59**	52.03	**43.98**

TABLE I: AP score (%)

Method	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic Light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle	Mean
CASENet	87.06	75.95	75.74	46.87	47.74	73.23	72.70	75.65	80.42	57.77	86.69	81.02	67.93	89.10	45.92	68.05	49.63	54.21	73.74	68.92
SEAL	87.6	77.5	75.9	47.6	46.3	75.5	71.2	75.4	80.9	60.1	87.4	81.5	68.9	88.9	50.2	47.8	44.1	52.7	73.0	69.1
STEAL	88.94	78.21	77.75	50.59	50.39	75.54	76.31	77.45	82.28	60.19	87.99	82.48	70.18	90.40	53.31	68.50	53.39	56.99	76.14	71.42
DFF (ResNet-101)	89.24	79.23	79.6	52.25	55.12	78.78	76.4	79.25	82.47	60.80	86.67	81.99	72.92	85.43	57.17	71.72	51.21	60.18	75.64	72.42
DFF (ResNet-50)	88.37	78.38	77.85	47.64	52.11	81.33	79.63	78.90	82.57	61.93	88.37	81.13	73.05	83.64	51.26	69.26	44.00	57.41	75.28	71.16
DFF (MobileNetV2)	85.86	75.64	73.83	43.80	44.39	74.20	**68.68**	74.22	80.19	57.59	84.95	75.58	58.98	**81.30**	23.98	43.58	21.03	39.69	69.49	61.95
FENet (Ours)	**86.92**	**76.58**	**75.15**	**47.34**	**47.93**	**78.14**	68.06	**74.86**	**80.72**	**59.62**	**86.68**	**77.67**	**65.97**	81.24	**34.69**	**51.51**	**34.33**	**57.41**	**70.55**	**65.66**

TABLE II: MF-ODS score (%)

Method	Params (M)	FLOPs (G)	Madds (G)	Forward Size(M)	MF-ODS (%)	Latency (ms)	Frequency (Hz)
DFF (MobileNetV2)	6.854	2.329	9.602	415.88	61.95	77.2	12.95
FENet	2.459	2.684	10.307	440.48	**65.66**	29.91	33.43
FENet (pruned)	2.243	2.481	9.913	378.22	63.04	28.5	35.08
FENet-fast	**0.888**	2.684	5.278	440.48	64.78	8.84	113.12
FENet-small	**0.888**	**1.289**	**2.534**	**233.54**	62.28	**8.72**	**114.68**

TABLE III: Ablation Study. The latency is tested on NVIDIA Jetson TX2 with TensorRT in Python.

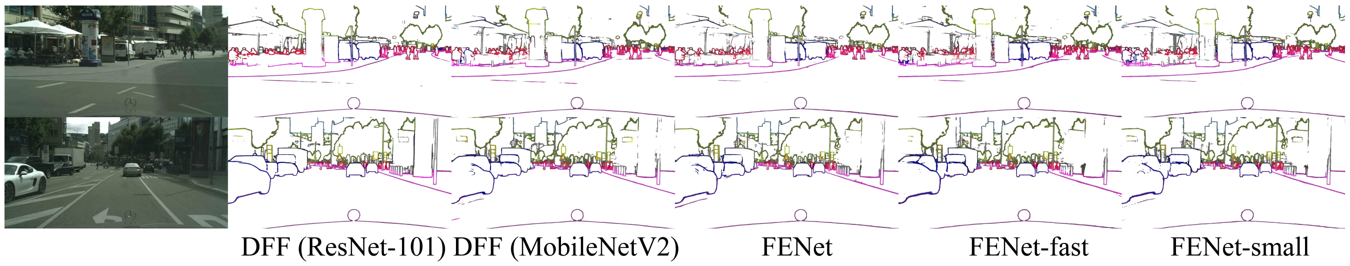

Fig. 3: Qualitative comparison of multiple semantic edge detection approaches on Cityscapes.

C. Quantitative Result

To evaluate the performance of our approach with respect to the state of the art semantic edge detectors, we compare it to CASENet [1], SEAL [2], STEAL [3], and DFF [4]. These methods are based on ResNet-101, which is a large backbone network. To establish a fair comparison, we also compare our method with DFF method with ResNet-50 and MobileNetV2 backbone network. The AP score results and MF-ODS score are illustrated in Table I and II respectively. The MobileNetV2 backbone has similar performance compared to ResNet-50 and ResNet-101 backbone, which strikes a good balance between performance and speed. However, the AP score of MobileNetV2-based DFF on some categories, which rarely appears in the dataset such as Truck, Train, and Motorcycle drops significantly compared to the large backbone network due to the small capacity of the network.

FENet is superior to MobileNetV2-based DFF from the aspect of mean AP ($+0.29\%$) and mean MF-ODS ($+3.71\%$) scores. It outperforms MobilNetV2-based DFF for most categories on MF-ODS metric. The AP score of FENet has much better performance than MobileNetV2-based DFF on rare categories such as Truck, Train, and Motorcycle. In Fig. 4, we benchmark the methods' speed on a NVIDIA Jetson TX2 platform. The experiments use MAX-N mode which has the highest GPU frequency (1.3 GHz) and CPU frequency (2 GHz). We optimize all methods by TensorRT for NVIDIA Jetson TX2. We use a batch size of one and 32-bit floating point precision to evaluate the throughput. State-of-the-art methods barely reach 2 Hz. FENet and its variants reach real-time performance while keeping competitive results.

D. Ablation study

We conduct an ablation study focusing on the model size and latency. In Table. III, we compare two variants of our model with the baseline method, which is the DFF with MobileNetV2 as the backbone network. Comparing the result of MobileNetV2-based DFF and FENet, the joint pyramid upsampling module improves the MF-ODS score significantly by embedding high-level feature maps with multi-scale features. The latency is dramatically reduced as well by introducing the joint pyramid upsampling module. The size of the output feature map of this module is two times the fifth-level feature map of the feature extraction module. It reduces the kernel size of the deconvolution layer, which is known to be a computational expensive operation for large kernels, and used to recover the resolution of feature map to original size from 32×32 to 16×16. Therefore, the latency can be reduced to the level of real–time performance. To be noticed, the FLOPS and Madds of FENet are higher than

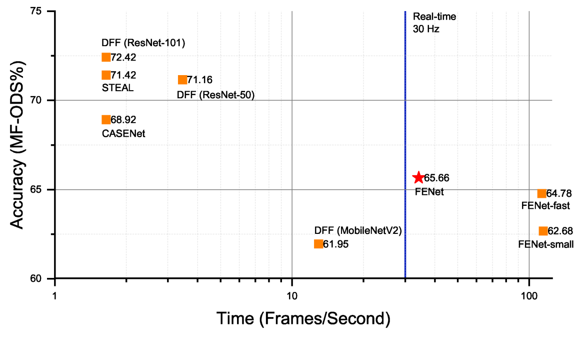

Fig. 4: Speed Benchmark.

MobileNetV2-based DFF since the joint pyramid upsampling module introduces extra computation. However, we argue that the metric of latency on embedded platforms is more important than FLOPs and Madds to evaluate the methods' speed. We also evaluate the variant FENet-fast, which is 8.7 times faster than MobileNetV2-based DFF. FENet uses the bilinear upsampling instead of deconvolution. This variant has much fewer parameters, which were due to large kernel deconvolution and is ideal for devices even slower than NVIDIA Jetson TX2 solving the critical latency issue.

To address the memory issue, we first use NetAdapt to prune the network to reduce the forward memory size by 14% comparing to FENet. However, pruned FENet still requires too much memory for resource-limited embedded robotics devices as shown in Table III. The robotic system has to run multiple tasks simultaneously, and tasks are competing to use the device's memory. FENet-small has the lowest parameter size, FLOPs, Madds, and latency among all the methods. FENet-small reduces the forward memory size (actual memory usage of the network and feature map) by 47% with a 3.38% loss of Maximum F-measure at Optimal Dataset scale (MF-ODS) comparing with FENet, making it ideal for the memory-critical devices.

V. Conclusion

We proposed the first real-time semantic edge detection network for SWaP constraints robots. We achieved this result by leveraging (a) the use of a lightweight MobileNetV2 backbone, (b) multi-scale features, and (c) a joint pyramid upsampling module. We optimize the network on NVIDIA Jetson TX2 using TensorRT, and further reduce the parameter size by adopting the NetAdapt pruning technique. The results show that our method has comparable performance with respect to existing approaches while being real–time. We provide two variants of the method to address the latency issue and forward memory size. This enables to fit our approach to extremely constrained computation resources.

Future works will leverage the semantic/geometric sparse representation in a real–time semi-dense visual odometry or SLAM system to boost localization and mapping performances. Finally, we would like to investigate semantic edge detection and depth estimation multi–task learning.

References

[1] Z. Yu, C. Feng, M.-Y. Liu, and S. Ramalingam, "CASENet: Deep Category-Aware Semantic Edge Detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5964–5973.

[2] Z. Yu, W. Liu, Y. Zou, C. Feng, S. Ramalingam, B. V. K. Vijaya Kumar, and J. Kautz, "Simultaneous Edge Alignment and Learning," in *The European Conference on Computer Vision (ECCV)*, 2018, pp. 388–404.

[3] D. Acuna, A. Kar, and S. Fidler, "Devil is in the Edges: Learning Semantic Boundaries from Noisy Annotations," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[4] Y. Hu, Y. Chen, X. Li, and J. Feng, "Dynamic Feature Fusion for Semantic Edge Detection," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2019-Augus, pp. 782–788, 2 2019.

[5] Y. Hu, Y. Zou, and J. Feng, "Panoptic Edge Detection," *arXiv preprint*, vol. arXiv:1906, 6 2019.

[6] H. Wu, J. Zhang, K. Huang, K. Liang, Y. Deepwise, and A. I. Lab, "FastFCN: Rethinking Dilated Convolution in the Backbone for Semantic Segmentation," in *arXiv preprint arXiv:1903.11816*, 2019.

[7] T.-J. Yang, A. Howard, B. Oc H E N, X. Zhang, V. Sze, and H. Adam, "NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications," in *The European Conference on Computer Vision (ECCV)*, 2018, pp. 285–300.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.

[9] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011, pp. 991–998.

[10] X. Yu, S. Chaturvedi, C. Feng, Y. Taguchi, T. Y. Lee, C. Fernandes, and S. Ramalingam, "VLASE: Vehicle Localization by Aggregating Semantic Edges," in *IEEE International Conference on Intelligent Robots and Systems*. Institute of Electrical and Electronics Engineers Inc., 12 2018, pp. 3196–3203.

[11] H. Qi, S. Collins, and J. A. Noble, "UPI-Net: Semantic Contour Detection in Placental Ultrasound," *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, pp. 416–424, 8 2019.

[12] W. Yang, "Dedge-AGMNet: A Robust Multi-task Learning Network for Stereo Matching and Depth Edge Detection," in *European Conference on Artificial Intelligence*, 8 2020.

[13] M. Zhen, J. Wang, L. Zhou, S. Li, T. Shen, J. Shang, T. Fang, and L. Quan, "Joint Semantic Segmentation and Boundary Detection Using Iterative Pyramid Contexts," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 13 666–13 675.

[14] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and $<$0.5MB model size," *arxiv preprint*, vol. arXiv:1602, 2 2016.

[15] 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," *CoRR*, vol. abs/1704.0, 4 2017.

[16] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations (ICRL)*, 9 2015, pp. 1–14.

[17] X. Zhang, X. Zhou, and M. Lin, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6848–6856.

[18] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design," in *The European Conference on Computer Vision (ECCV)*, 2018, pp. 116–131.

[19] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for Real-Time Semantic Segmentation on High-Resolution Images," in *The European Conference on Computer Vision (ECCV)*, 2018, pp. 405–420.

[20] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures," in *arXiv preprint arXiv:1607.03250*, 7 2016.

[21] Y. He, "Channel Pruning for Accelerating Very Deep Neural Networks," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 1389–1397.

[22] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, "Context Encoding for Semantic Segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7151–7160.

[23] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, D. A. R&d, and T. U. Darmstadt, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213–3223.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, 3 2010, pp. 248–255.

[25] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530–549, 5 2004.