FASSD-Net: Fast and Accurate Real-Time Semantic Segmentation for Embedded Systems

Leonel Rosas-Arias, Gibran Benitez-Garcia, *Member, IEEE*, Jose Portillo-Portillo, Jesus Olivares-Mercado, Gabriel Sanchez-Perez, and Keiji Yanai, *Member, IEEE*

***Abstract*— Recent works of real-time semantic segmentation, remove or make use of light decoders from dense deep neural networks to achieve fast inference speed. This strategy helps to achieve real-time performance; however, the accuracy is significantly compromised in comparison to non-real-time methods. In this paper, we introduce two key modules aimed to design a high-performance decoder for real-time semantic segmentation, which also reduces the accuracy gap between real-time and non-real-time networks. The first module, Dilated Asymmetric Pyramidal Fusion (DAPF), is designed to increase the receptive field on the top of the last stage of the encoder, obtaining richer contextual features. The second module, Multi-resolution Dilated Asymmetric (MDA) module, fuses and refines detail and contextual information from multi-scale feature maps coming from early and deeper stages of the network. Both modules are designed to keep a low computational complexity by using asymmetric convolutions. With these modules, we propose a network entitled "FASSD-Net," which is based on a light-weight CNN backbone. Running on a single Nvidia GTX 1080Ti, our model reaches 77.5% and 69.3% of mIoU, at 41 and 80 FPS on the Cityscapes and CamVid datasets, respectively. We present an extensive analysis of the accuracy-speed tradeoffs of three FASSD-Net variations on different embedded systems, demonstrating that a light version of our network can run on the low-power consumption Jetson Xavier NX, at 32 FPS reaching 74% of mIoU with full resolution (1024 × 2048). The source code and pre-trained models are available at github.com/GibranBenitez/FASSD-Net.**

***Index Terms*— Semantic segmentation, fully convolutional networks, spatial pyramid pooling, HarDNet, embedded systems, Jetson Xavier NX.**

I. Introduction

SEMANTIC segmentation is considered as a fundamental task in computer vision [1]–[3]. It aims to assign semantic class labels to each pixel present in a given input image. In recent years, due to the development of new deep learning techniques, semantic segmentation has been widely applied to a number of challenging fields, including: autonomous driving [4]–[8], robot sensing [9], medical imaging [10], augmented reality [11] and video surveillance [12], to name a few. Some of these applications require the inference speed to be real-time for making important decisions or actions, as a part of more complex systems. In particular, several applications in the field of intelligent transportation systems [8], [13]–[15] require keeping a balance between high accuracy prediction and real-time performance, such as processing urban scene images from autonomous or assisted driving cars [3], [16], [17]. However, speed and accuracy are two factors that seemingly contradict each other, making real-time semantic segmentation a challenging task, especially when the implementation of the system is carried on an embedded system [2], [18]. In this case, factors such as low-power consumption and memory usage become crucial [19]–[21].

In terms of accuracy, some state-of-the-art (SOTA) networks for real-time semantic segmentation [16], [22], [23] use *U-shape*-like architectures [10] to achieve high performance by recovering hierarchical features from previous stages of the network. Nevertheless, their precision is still significantly lower than non-real-time semantic segmentation networks, which instead use deeper classification architectures as a backbone, such as ResNet-101 [24]. Moreover, a common way to boost the accuracy of these networks is by leveraging the use of dilated (atrous) convolutions in the last blocks of the network. In this way, the receptive field of the convolution kernel is enlarged [25]. However, the networks designed to exploit this property require a considerable amount of floating-point operations, such as the Atrous Spatial Pyramid Pooling module (ASPP) from DeepLabV3+ [26]. On top of that, the ASPP module heavily relies on dilated convolutions, which by themselves are slow to compute due to framework optimization constraints [23].

The slow-down caused by the use of dilated convolutions can be alleviated by implementing convolution factorization [19]. Therefore, in this paper, we build upon this solution to design two modules for achieving real-time semantic segmentation with a boost of accuracy. In short, our general approach aims to exploit contextual information in multiple stages of a *U-shape* architecture. Specifically, we name the two modules: Dilated Asymmetric Pyramidal Fusion (DAPF) and Multi-resolution Dilated Asymmetric (MDA).

In order to increase the kernel receptive field and keep low computational complexity, we carefully employ 3×3 dilated

Manuscript received August 16, 2020; revised April 8, 2021 and October 4, 2021; accepted November 9, 2021. This work was supported in part by the Instituto Politécnico Nacional (IPN) and CONACyT (Mexico) and in part by the JSPS KAKENHI under Grant 15H05915, Grant 17H01745, Grant 17H06100, and Grant 19H04929. The Associate Editor for this article was S.-H. Kong. *(Leonel Rosas-Arias and Gibran Benitez-Garcia are co-first authors.) (Corresponding author: Gibran Benitez-Garcia.)*

Leonel Rosas-Arias, Jose Portillo-Portillo, Jesus Olivares-Mercado, and Gabriel Sanchez-Perez are with the Instituto Politécnico Nacional, ESIME Culhuacan, Mexico City 04440, Mexico (e-mail: gasanchezp@ipn.mx).

Gibran Benitez-Garcia and Keiji Yanai are with the Department of Informatics, The University of Electro-Communications, Tokyo 182-8585, Japan (e-mail: gibran@ieee.org).

Digital Object Identifier 10.1109/TITS.2021.3127553

convolutions factorized into two consecutive 1D dilated convolutions. From hereafter, we refer to this type of convolutions as *dilated asymmetric convolutions*, due to the asymmetric nature of the convolution kernel and the dilation implementation.

Our DAPF module is designed to significantly increase the receptive field of the last stage of the encoder network, obtaining richer contextual features. We follow a pyramidal scheme similar to the ASPP module [26], but all 3×3 dilated convolutions are replaced with dilated asymmetric convolutions. Our design, allows us to adjust the number of pyramidal feature maps of DAPF according to the number of input feature maps, which further reduces the computational complexity.

Similarly, our MDA module, fuses multi-resolution feature maps coming from previous encoder and decoder levels of the network. In this module, feature maps are processed simultaneously by two parallel branches: the asymmetric branch and the non-asymmetric branch. The asymmetric convolutional branch exploits the contextual information of the input feature maps, whereas the non-asymmetric branch focuses on recovering details. This design allows a simultaneous refinement of detail and contextual information in multiple stages of the decoder.

By combining the DAPF and MDA modules, we design a novel network entitled FASSD-Net, which effectively increases the semantic segmentation accuracy with a relatively low computational cost. FASSD-Net bridges the accuracy gap between existing real-time (around 70% mIoU on Cityscapes benchmark) [2], [19], [23], [27]–[29] and non-real-time networks (about 80% mIoU) [1], [26], [30]–[33]. Moreover, we present two light variations that provide a balanced trade-off between accuracy and inference speed. Our fastest variation can even run real-time at full resolution on low-power consumption embedded systems, such as Jetson Xavier NX ($< 15W$) [34].

Our main contributions are summarized as follows:

- We introduce DAPF, an efficient plug-and-play spatial pyramidal fusion module inspired by ASPP [26], which demands far less computational complexity, and enables its use for real-time applications.
- We introduce the MDA module, which allows better learning from two different stages of the network, refining spatial and contextual information simultaneously by keeping low computation cost.
- We propose a FASSD-Net and two light variations which obtain SOTA mIoU results on the Cityscapes and CamVid benchmarks for the task of real-time semantic segmentation. Moreover, FASSD-Net is comparable to non-real-time methods such as DeepLabV3+ [26] and PSPNet [30] in terms of accuracy, while being about $40\times$ faster.

This work is an extension of the conference paper presented at ICPR2020 [35], which introduces our FASSD-Net approach (Section III). The novel contributions of this paper include:

- Experimental analysis with more benchmark datasets, including Cityscapes [36] and CamVid [37] (Section IV).
- Additional experiments to demonstrate the performance of our two modules with different backbone networks, including MobileNetV2 [38], ShuffleNetV2 [39], and ResNet-18 [24] (Section IV-C).
- An extensive evaluation of the accuracy-speed trade-offs of our FASSD-Net variations on the NVIDIA Jetson [34] family of GPU-powered embedded systems (Section IV-E).
- An updated SOTA comparisons with recent methods for real-time semantic segmentation (Sections IV-F and IV-H).

II. Related Work

Models such as PSPNet [30] and DeepLabV3+ [26] exploit contextual information by processing the same set of feature maps. PSPNet [30] downsamples the feature maps at four different rates, and applies a series of convolutions on them to finally perform a fusion process. Likewise, DeepLabV3+ [26] processes the feature maps by applying atrous convolutions at different rates. These models have achieved top results on several segmentation benchmarks by leveraging the use of multi-scale information in a pyramidal fashion. However, even on modern GPUs, the required computational resources for these methods are considerably expensive, making them unfeasible for real-time applications [28]. In contrast, our proposed FASSD-Net handles a similar pyramidal strategy as proposed in DeepLabV3+ [26], introducing dilated asymmetric convolutions, which enables real-time applications.

In addition, fusion strategies for multi-resolution feature maps have been used in recent works such as HarDNet [22], SwiftNet [23] and FasterSeg [27]. These networks either concatenate or add two sets of feature maps and further process them with a single convolution. This straightforward fusion strategy requires a small amount of computation, but do not exploit contextual information. In contrast, our MDA module concatenates two sets of feature maps, processing them by two parallel branches simultaneously, which refines features rich in detailed information and context.

On the other hand, techniques for reducing the computational complexity of the networks such as depthwise separable convolutions [38]–[41], zoomed convolutions [27], or convolution factorization [19], [28], [29] have been proposed and applied to the task of real-time semantic segmentation. Networks that employ these techniques such as Fast-SCNN [41], ERFNet [28], and FasterSeg [27] achieve real-time performance, usually at the cost of significantly lower accuracy compared to non-real-time methods. Similarly, our three network proposals rely on factorized (asymmetric) convolutions used in the DAPF and MDA modules. However, our proposed networks outperform Fast-SCNN [41], ERFNet [28] and FasterSeg [27] in terms of accuracy.

Additional methods for accelerating neural networks include filtering or channel pruning [42], [43], network distillation [44], [45], Network Architecture Search (NAS) [27], [46], and Neural Network Quantization [47]. Such methods mainly reduce the number of parameters and weight of the model or transfer knowledge from a cumbersome network to a compact model. However, most of them, either utilize sophisticated methodologies, require a considerable amount of memory, or cannot be directly applied to more elaborated network architectures [48]. Specifically, FasterSeg [27] and CAS [46]

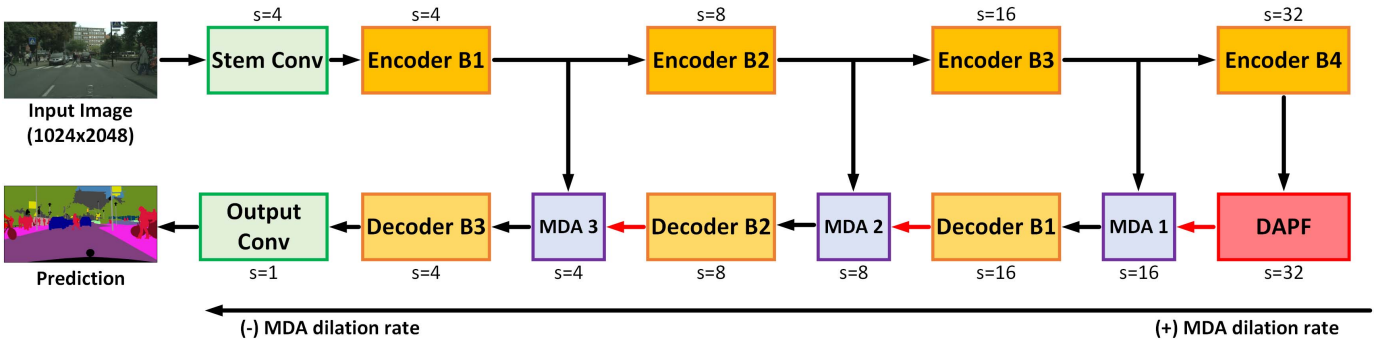


Fig. 1. Block diagram of the proposed network. "s" indicates the downsampling rate of the feature maps with respect to the original input image (e.g., *s=32* indicates output feature maps of size 32×64). The colors of each block indicate its type: standard convolution, HarDBlock, MDA and DAPF.

obtain SOTA inference speed by utilizing NAS techniques. By comparison, our proposed models are designed manually and still, outperform these NAS networks in accuracy and speed performance.

Networks such as ESPNetv2 [49], ESNet [49], and LEDNet [50] employ lightweight pyramidal multi-resolution strategies similar to our DAPF module. More specifically, LEDNet [50] and ESNet [29] incorporate asymmetric convolutions in their core modules. However, they heavily rely on it, which reduces the inference speed performance, compared to the highly optimized standard convolutions [23]. Contrary to these methods, our DAPF and MDA modules utilize asymmetric convolutions only if dilation is applied concurrently, alleviating the reduction of inference speed performance caused by the atrous convolutions [19]. When compared to our proposals, LEDNet [50] and ESNet [29] are slower and less accurate.

III. Proposed Method

Most of the existing SOTA methods for semantic segmentation are built on top of high-performance baselines for image classification such as ResNet, WiderResNet, or Xception [1], [26], [31]–[33]. Following this trend, we propose FASSD-Net (Fast and Accurate Semantic Segmentation with Dilated asymmetric convolutions) by extending the work of Chao *et al.* [22] with our DAPF and the MDA modules. The proposed network structure is shown in Figure 1. In the figure, the stem convolution block consists of four consecutive convolution layers. The core element of all encoder and decoder blocks is the HarDBlock (Harmonic Dense Block), proposed in HarDNet [22]. Our proposed DAPF is placed at the end of the encoder, while MDA modules connect each decoder block with its corresponding encoder, in a *U-shape* fashion. Finally, the last block of the network consists of a single 1×1 convolution for making the final prediction. Bilinear upsampling is used to reestablish the original input size (1024×2048). Table I shows the detailed architecture of FASSD-Net, including the output and number of channels for each element.

TABLE I

FASSD-Net Architecture. L Denotes the Number of Convolution Layers in the HarDBlock

Stage	Name	Type	Output size
Input	-	-	1024×2048×3
Encoder	Stem Conv	Conv 3×3 (s=2)*	512×1024×16
		Conv 3×3**	512×1024×24
		Conv 3×3 (s=2)	256×512×32
		Conv 3×3	256×512×48
	Encoder B1	HarDBlock (L=4)	256×512×64
	Encoder B2	2D Average Pooling	128×256×96
		HarDBlock (L=4)	
	Encoder B3	2D Average Pooling	64×128×160
		HarDBlock (L=8)	
	Encoder B4	2D Average Pooling	32×64×224
		HarDBlock (L=8)	
	DAPF	-	32×64×224
Decoder	MDA	-	64×128×192
	Decoder B1	HarDBlock (L=8)	64×128×160
	MDA	-	128×256×119
	Decoder B2	HarDBlock (L=4)	128×256×78
	MDA	-	256×512×63
	Decoder B3	HarDBlock (L=4)	256×512×48
	Output Conv	Conv 1×1	256×512×19
		Upsampling ×4	1024×2048×19

*Replaced to (s=3) in FASSD-Net-L1
**Replaced to (s=2) in FASSD-Net-L2

A. Network Overview

HarDNet (Harmonic DenseNet) [22], is a recent SOTA network inspired by DenseNet (Densely Connected Network) [51]. Its core component, the *HarDBlock* (Harmonic Dense Block), is specifically designed to address the memory traffic problems and the density of computations of the *DenseBlock*. In particular, the HarDBlock reduces most of the layer connections from a DenseBlock, which reduces concatenation cost; moreover, the input/output channel ratio is balanced by increasing the channel width of a layer according to its connections. As shown in Figure 2, the layer connections are significantly reduced, while the layer width is based on the concatenation size, being the last layer of the block the largest in terms of the number of channels. Compared to ResNet [24] and DenseNet [51], HarDNet achieves comparable accuracy with significantly lower GPU runtime for classification tasks.

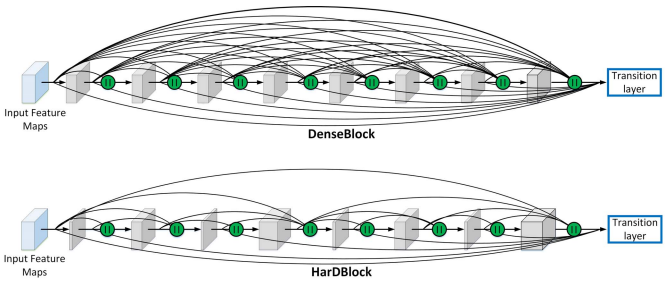


Fig. 2. Concatenation scheme comparison between the DenseBlock and the HarDBlock. The HarDBlock is named after the harmonic waves pattern that describes its concatenation scheme. "||" denotes concatenation process.

The HarDBlock proposed by Chao *et al.* [22] is based on the observation that, when the density of computation is low, DRAM (dynamic random-access memory) traffic can influence inference time more substantially than the model size and the number of operations. In detail, the density of computations (MoC) can be expressed as:

$$MoC = \frac{\text{\#Multiply-Accumulate operations (MACs)}}{\text{Convolutional Input/Output (CIO)}}, \quad (1)$$

where CIO is an approximation of the DRAM traffic proportional to the real DRAM traffic measurement. Likewise, CIO is mathematically defined by:

$$CIO = \sum_{l}(c_{in}^{(l)} \times h_{in}^{(l)} \times w_{in}^{(l)} + c_{out}^{(l)} \times h_{out}^{(l)} \times w_{out}^{(l)}), \quad (2)$$

where c, w, and h are, respectively, the number of channels, width and height of the feature maps for a given convolution layer l. Given that, the HarDBlock follows a concatenation scheme aimed to improve the throughput of the feature maps in the network, avoiding unnecessary DRAM accesses. Specifically, the layer k connects to layer $k - 2^n$ if 2^n divides k, for all non-negative n that satisfies $k - 2^n \geq 0$, being layer 0 the input layer. In this way, layers from 1 through $2^n - 1$ can be flushed from the memory once layer 2^n is processed.

As a baseline, we use the FC-HarDNet-70 model [22], which is the implementation of HarDNet for the task of semantic segmentation. FC-HarDNet-70 is a *U-shaped* architecture [10] with five encoder blocks and four decoder blocks (all of them HarDBlocks). The convolution layers in the last encoder stage of FC-HarDNet process a high number of feature maps with $1/64$ input size resolution, which is essential for classification tasks. However, we believe that this is not always the case for semantic segmentation tasks. For example, even a high-resolution 1024×2048 image that has been processed and downsampled to $1/64$ of its original size, becomes a tensor at the resolution of 16×32, which may completely lose track of small objects, heavily compromising the segmentation accuracy. Therefore, in our FASSD-Net, the last encoder block and the first decoder block of FC-HarDNet are replaced with our DAPF module, so that the smallest feature maps processed by our network are $1/32$ of the input size. Similarly, the FC-HarDNet multi-resolution fusion scheme is substituted by our MDA module, as shown in Figure 1.

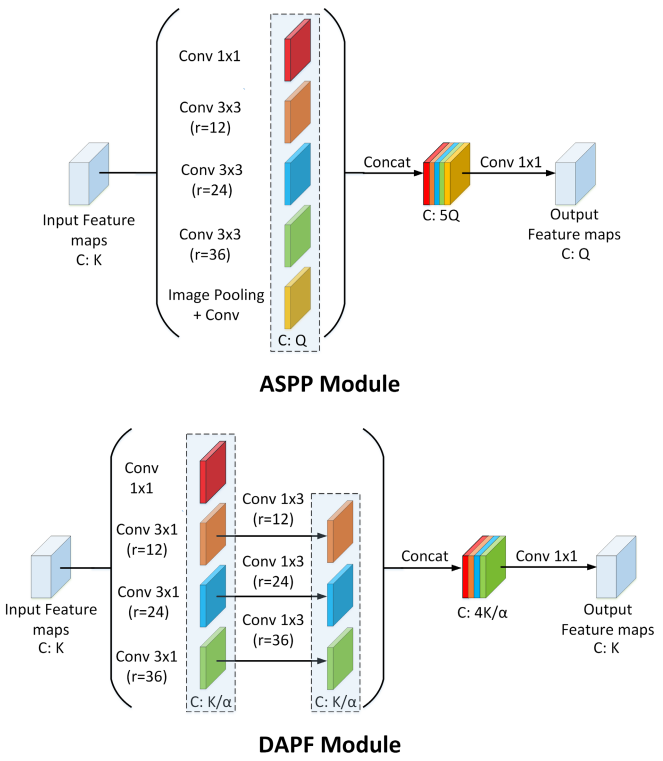


Fig. 3. ASPP and DAPF modules comparison. "C" denotes the number of feature maps.

B. Dilated Asymmetric Pyramidal Fusion Module

The original ASPP module [26] heavily relies on standard atrous convolutions and produces a fixed number of feature maps $\mathbf{Q}$ in each of its five pyramidal branches, as shown in Figure 3. Pyramidal branches consist of: $1 \times$ Conv 1×1, $3 \times$ atrous Conv 3×3 $r = (12, 24, 36)$ and $1 \times$ Pooling + Conv 1×1, where r is the dilation rate of the convolution kernel.

Inspired by ASPP, we propose the DAPF module aimed to reduce its computational burden, which consists of two key elements: (1) 3×3 atrous convolutions are factorized into two consecutive 1D atrous convolutions, specifically, a 3×1 convolution followed by a 1×3 convolution; (2) the number of feature maps generated by each pyramidal branch, is not fixed. Instead, it is defined by $\mathbf{K} \times \frac{1}{\alpha}$, where $\mathbf{K}$ is the number of input feature maps of the module, and α serves as the compression factor. Note that α can be adjusted according to the available computational budget. In our implementation, we set $\alpha = 2$. Figure 3 illustrates the differences between DAPF and the ASPP modules.

Formally, For a given set of input feature maps $\mathbf{X} \in \mathbb{R}^{C \times H \times W}$, the DAPF module can be defined as:

$$\mathbf{Y} = \mathcal{F}_{1,1}\Big(\mathcal{F}_{1,1}(\mathbf{X})||A_{3,12}(\mathbf{X})||A_{3,24}(\mathbf{X})||A_{3,36}(\mathbf{X})\Big), \quad (3)$$

where $\mathbf{Y}$ represents the output feature maps, $\mathcal{F}_{1,1}$ is a 1×1 convolution, and $A_{k,r}$ denotes the asymmetric dilated convolution branch defined by $\mathcal{F}_{(1\times k),r}(\mathcal{F}_{(k\times 1),r}(\mathbf{X}))$, where $\mathcal{F}_{(k\times 1),r}$ and $\mathcal{F}_{(1\times k),r}$ are the two consecutive convolutions with asymmetric kernel size k and dilation rate r.

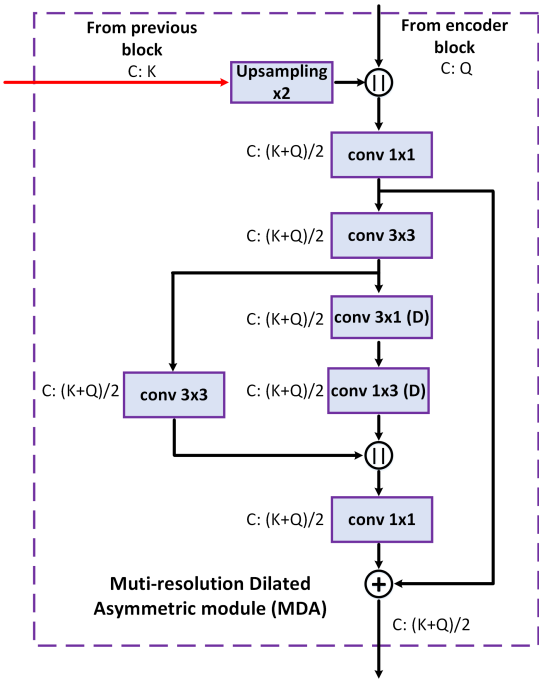


Fig. 4. MDA module. "C" denotes the number of output feature maps, "||" denotes concatenation and "D" indicates dilated convolution.

The major contributing factor for reducing the computational burden in our DAPF module is the use of asymmetric convolutions. For instance, in the pyramidal branches, for each standard 2D convolution, we would have to perform $K \times d \times d \times F$ operations, where K is the number of input channels (feature maps), d is the kernel size, and F is the number of output channels. On the other hand, following our asymmetric strategy with $\alpha = 2$, we perform $(K \times d \times \frac{1}{2}K) + (\frac{1}{2}K \times d \times \frac{1}{2}K)$ operations. For a 3×3 kernel, this factorization strategy only requires $\frac{1}{2}$ of the original number of operations, thus saving 50% of the needed computations and parameters in comparison to its non-asymmetric convolution equivalent. Moreover, factorization can also improve the learning capacity of the module as a result of the intermediate activation layers used between the two 1D convolutions [28].

C. Multi-Resolution Dilated Asymmetric Module

We design our MDA module to simultaneously exploit contextual information and recover spatial information. As shown in Figure 4, the two multi-resolution feature maps of sizes K and Q are concatenated and fused together with a 1×1 convolution. This convolution reduces the number of feature maps by half to reduce the number of computations in the module and subsequent stages.

After an additional 3×3 convolution that refines the initially fused feature maps, two parallel branches process the output feature maps. The asymmetric convolutional branch $A_{3,r}(\mathbf{X})$ aims to exploit the contextual information present in the feature maps by leveraging the use of dilated convolutions. In contrast, the non-asymmetric branch $\mathcal{F}_{(3,3)}(\mathbf{X})$ focuses on refining the details. The resulting feature maps are concatenated and processed by a 1×1 convolution to match the number of feature maps of the first 1×1 convolution. Finally, feature maps of both 1×1 convolutions are summed up through a residual connection, helping to improve the gradient flow.

The dilation rate r of the asymmetric branch gradually decreases in every MDA block from the *deepest* to the *shallowest* stage of the decoder (depicted in Figure 1 as the longest arrow at the bottom). Specifically, the dilation rates are $r = 8$ for MDA 1, $r = 4$ for MDA 2, and $r = 2$ for MDA 3. The intuition behind this idea is that inner feature maps of the network are richer in contextual information and can be leveraged by atrous convolutions with larger dilation rates.

TABLE II
ABLATION STUDY OF OUR PROPOSED MODULES ON THE CITYSCAPES VALIDATION SET. RED, GREEN, BLUE REPRESENT THE TOP-3 RESULTS

Method	GFLOPs	Parameters	Δp	FPS	mIoU
FC-HarDNet-70 [22]	35.4	4.10M	-	52.3	76.4
Baseline	32.9	1.90M	0M	56.3	75.2
+ ASPP	36.8	3.85M	1.95M	50.2	75.8
+ DAPF	33.9	2.36M	0.46M	53.9	77.7
+ MDA	44.2	2.38M	0.48M	42.2	77.4
+ ASPP + MDA	48.0	4.33M	2.43M	39.1	76.8
+ DAPF + MDA	45.1	2.85M	0.95M	41.1	78.2

IV. EXPERIMENTAL RESULTS

We evaluate our proposed network architectures using two benchmark datasets: Cityscapes [36] and CamVid [37]. The Cityscapes dataset [36] consists of 5,000 finely annotated 1024×2048 images: 2,975 for training, 1,525 for testing, and 500 images for validation. Additionally, 19,998 images with coarse annotations are also provided. On the other hand, the CamVid dataset [37] is a significantly smaller dataset with 701 720×960 images, including 367 for training, 101 for validation, and 233 for testing. For a fair comparison, we only use the fine annotated images of Cityscapes, and following the common practice of CamVid, we incorporate the value subset into train because it is too small and too easy to be useful for validation. We explicitly mention when using multi-scale evaluation in the final results. The performance is mainly measured in mean intersection-over-union accuracy (mIoU) and frames per second (FPS). Besides, we report the number of parameters and computational complexity in GFLOPs using the full resolution size of Cityscapes.

A. Implementation Details

We use PyTorch 1.2 with CUDA 10.2 for all experiments. The same training setting is used for all models, where Stochastic Gradient Descent (SGD) with weight-decay 5×10^{-4} and momentum 0.9 is used as the optimizer. We employ the "poly" learning rate strategy $lr = initial\_lr \times (\frac{iter}{total\_iter})^{0.9}$, and an initial learning rate of 0.02. Cross-entropy loss is computed following the online bootstrapping strategy [52]. Data augmentation consists of random horizontal flip, random scale in the range $[0.5, 2]$, and random cropping. We train with 1024×1024, and 512×512 crop size, for Cityscapes and for CamVid, respectively. We trained all models for

TABLE III
PER-CLASS MIOU SCORE ABLATION RESULTS ON THE CITYSCAPES VALIDATION SET

Method	road	s.walk	build.	wall	fence	pole	t.light	t.sign	veg.	terr.	sky	person	rider	car	truck	bus	train	mbik	bike	mIoU	FPS
Baseline	97.7	82.6	92.2	53.5	59.5	63.1	67.8	76.7	92.2	63.2	94.9	80.8	62.0	94.4	71.0	81.6	70.5	49.7	75.5	75.2	56.3
+ DAPF	98.3	85.6	92.6	57.4	62.7	66.0	70.6	78.3	92.4	64.8	95.0	82.3	63.9	94.9	75.2	84.3	76.3	59.2	76.2	77.7	53.9
+ MDA	98.1	84.8	92.6	54.6	60.7	66.1	70.5	79.3	92.6	65.4	94.7	82.0	62.3	95.0	73.7	84.3	76.2	60.9	76.3	77.4	42.2
+ DAPF + MDA	98.2	85.3	92.8	55.1	62.9	66.4	71.5	79.2	92.5	64.5	94.9	82.0	62.9	95.3	81.4	87.2	78.0	59.2	76.5	78.2	41.1

90000 iterations with batch size 16. Finally, we extend the same training protocol for 30000 more iterations, setting the batch size to 24 and the learning rate to 0.001.

All the encoder blocks are pre-trained on the ImageNet dataset [53], and the inference time is measured on an Intel Core i7-9700K desktop with a single NVIDIA GTX 1080Ti GPU, if not specified otherwise. For all experiments, the speed is calculated from the average FPS rate of $10,000$ iterations with $1024 \times 2048 \times 3$ images.

B. Ablation Study

In this section, we show the performance comparison between our proposed DAPF and the DeepLabV3+'s ASPP module [26]. We also evaluate the effectiveness of our MDA module and its combination with DAPF and ASPP. Table II summarizes the corresponding results. *Baseline* denotes a modified FC-HarDNet-70, where the last encoder and the first decoder blocks are removed, as previously described in Section III-A. Note that, for a fair comparison, all methods shown in Table II are trained without further fine-tuning.

Our DAPF module outperforms DeepLabV3+'s ASPP in all four metrics (GFLOPs, Parameters, FPS and mIoU). In addition, the increase of parameters (Δp) by DAPF from the *Baseline* is significantly lower than the ASPP module (0.46M vs 1.95M). In resume, our module is more than four times lighter than ASPP, and presents a precision increase of 1.9%. It can be observed that the addition of our MDA strategy outperforms the mIoU of the baseline network to almost the same degree as DAPF. Likewise, our MDA strategy consumes roughly the same number of parameters. However, since MDA is used in three different levels of the decoder, the FPS drop becomes evident compared to DAPF.

Table III shows per-class mIoU results of our proposed modules compared with the baseline. As expected, DAPF improves the accuracy of classes related to contextual information, such as wall and rider. Accordingly, MDA helps to segment small objects of the scene, such as traffic sign (t.sign) and motorbike (mbik), some of the smallest objects on the cityscapes validation set. On the other hand, the combination of our two proposals achieves the best mIoU accuracy in general. Specifically, it outperforms the baseline and the SOTA results of FC-HarDNet-70 [22] by 3% and 1.8%, respectively. Additionally, the increase of parameters (Δp) of our proposal is less than 50% compared to the ASPP model. On top of that, the total number of parameters is significantly lower than those needed by FC-HArDNet-70 (2.85M vs 4.10M). Note that *Baseline* + *DAPF* + *MDA* corresponds to our final model called FASSD-Net, as shown in Figure 1.

TABLE IV
ABLATION STUDY OF HYPERPARAMETER α OF DAPF MODULE

α	mIoU	Parameters	FPS
1	78.50	3.54M	39.8
2	78.22	2.85M	41.1
4	77.63	2.58M	41.7
6	77.61	2.51M	41.9

TABLE V
EVALUATION OF OUR PROPOSED MODULES WITH DIFFERENT BACKBONES ON THE CITYSCAPES VALIDATION SET

Method	GFLOPs	Parameters	Δp	FPS	mIoU
HarDNet (baseline)	32.9	1.90M	0M	56.3	75.2
FASSD-Net	45.1	2.85M	0.95M	41.1	78.2
U-MobileNetV2	25.5	4.64M	0M	16.4	72.4
FASSD-Mobile	24.7	5.33M	0.69M	15.6	76.5
U-ShuffleNetV2	29.3	3.17M	0M	43.5	70.9
FASSD-Shuffle	23.0	4.24M	1.07M	35.8	73.7
U-ResNet-18	142.3	14.42M	0M	32.6	74.6
FASSD-ResNet	131.4	15.71M	1.29M	28.3	77.0

Finally, we evaluate the sensitivity of the DAPF module on the hyperparameter alpha. Table IV shows the results of our FASSD-Net with four different values. As mentioned in Section III-B, alpha is meant to regulate the compression factor of each pyramidal branch of DAPF. As we can see in Table IV, if we increase the alpha value, the network runs faster at the cost of accuracy. Thus, in our implementation, we set $\alpha = 2$.

C. Evaluation With Different Backbones

To better evaluate the effectiveness of our proposed modules, we replace the HarDNet backbone to different efficient CNN architectures mainly designed for mobile applications. We implement MobileNetV2 [38], ShuffleNetV2 [39], and ResNet-18 [24] following the FASSD-Net architecture, described in Table I. So that, these architectures initially designed for classification, have been redesigned for semantic segmentation in a *U-shape* style with four encoder and three decoder blocks, plus our DAPF and MDA modules. We call these versions *FASSD-Mobile*, *FASSD-Shuffle*, and *FASSD-ResNet*, respectively. For a fair comparison, we also implement the baselines of each architecture with a simple fusion block and without pyramidal pooling blocks, same as the HarDNet baseline, described in Section III-A. We refer to these implementations as *U-MobileNetV2*, *U-ShuffleNetV2*, and *U-ResNet-18*. Table V shows the comparison results of these implementations.

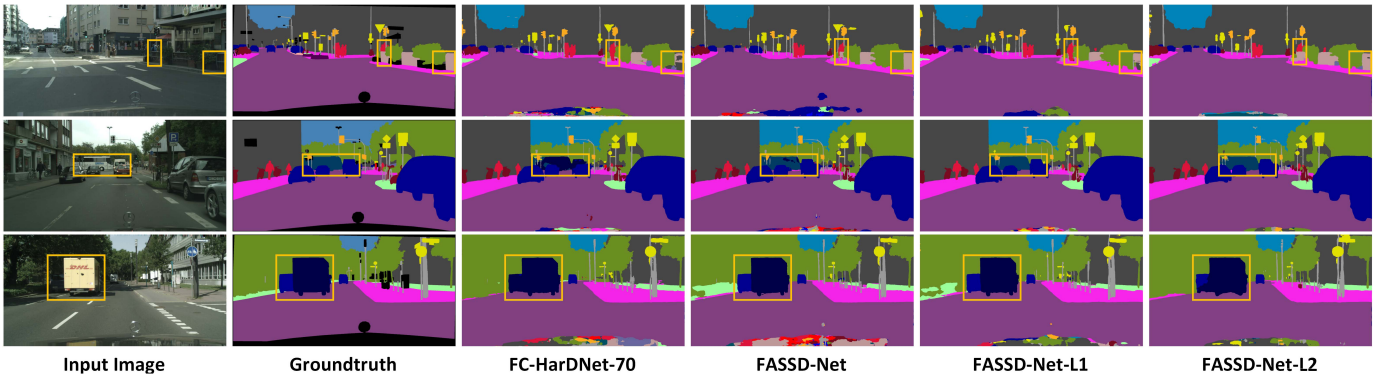


Fig. 5. Qualitative results of the proposed networks. Regions of improvement are highlighted with yellow squares.

TABLE VI
PER-CLASS MIOU SCORE COMPARISON OF OUR PROPOSALS ON THE CITYSCAPES VALIDATION SET

Method	road	s.walk	build.	wall	fence	pole	t.light	t.sign	veg.	terr.	sky	person	rider	car	truck	bus	train	mbik	bike	mIoU	FPS
FC-HarDNet-70 [22]	98.1	84.6	92.6	60.0	63.5	64.9	69.7	78.8	92.2	62.8	95.0	81.4	60.7	95.0	73.9	82.4	77.4	61.7	76.2	77.4	53.0
FASSD-Net	98.3	86.0	92.9	59.6	63.9	67.1	71.7	79.6	92.5	63.7	95.0	82.1	63.2	95.4	80.8	87.8	79.6	61.3	76.8	78.8	41.1
FASSD-Net-L1	98.2	84.6	92.4	54.7	61.3	63.3	68.2	77.1	92.1	61.8	94.9	80.0	59.8	94.9	76.0	82.7	74.7	59.1	74.6	76.3	78.0
FASSD-Net-L2	97.9	83.1	91.6	55.6	57.0	58.0	62.5	71.8	91.7	63.0	94.4	77.3	57.0	93.8	75.5	81.9	71.1	53.1	70.8	74.1	133.1

As expected, all the backbone architectures significantly increase precision when using our DAPF and MDA modules (FASSD-Net versions). Particularly, U-MobileNetV2 increases more than 4% of mIoU while keeping the number of parameters closer to the baseline (U-MobileNetV2). From Table V we can also see that the FASSD-Shuffle and FASSD-Mobile architectures have the lowest computational complexity with less than 25 GFLOPs calculated from an input size of 1024×2048, which can be useful for mobile applications. Note that the inference speed of MobileNetV2 versions is surprisingly low because of the optimization problems of depthwise separable convolutions attributed to the PyTorch framework.[1]

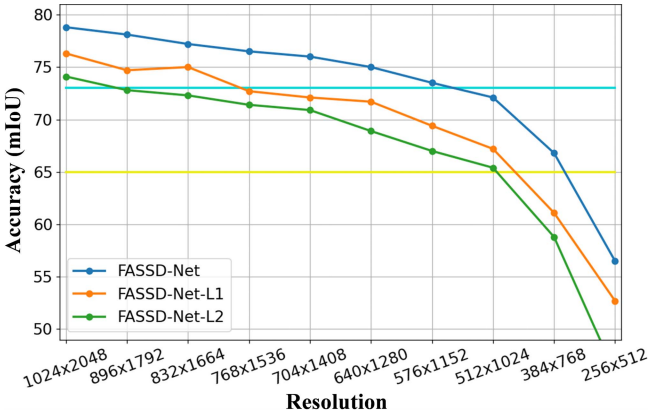


Fig. 6. Accuracy vs. input resolution of our proposals measured on the Cityscapes validation set. The cyan and yellow lines represent the average and the commonly accepted mIoU of real-time SOTA methods, respectively.

D. Light Version of FASSD-Net

In addition to our network FASSD-Net, we introduce two light versions designed to maintain a better tradeoff between speed and accuracy. We call these networks: FASSD-Net-L1 and FASSD-Net-L2. As shown in Table I, FASSD-Net-L1 differs only in the first convolution layer, where the convolution stride is increased from 2 to 3. Such modification, preserves the same number of parameters of the network and leads to a faster inference speed at the cost of a small drop in accuracy. Specifically, it is $1.9\times$ faster and 2.5% less accurate. FASSD-Net-L2, on the other hand, is designed to be the fastest among our three proposals. It adopts an additional convolution stride of 2 in the second convolution layer of the stem block. Moreover, all the HarDBlocks in the decoder are replaced by conventional 3×3 convolution layers of 64 channels. Thus, FASSD-Net-L2 is $3.2\times$ faster than FASSD-Net with only a 4.7% drop in accuracy performance.

We report the results of per-class mIoU accuracy in Table VI. Most significative improvements occur in the truck and bus classes with 6.9% and 5.4%, respectively. Similarly, FASSD-Net-L1 and FASSD-Net-L2 also obtain better results overall in these two classes compared to FC-HarDNet-70, despite being less accurate models. Qualitative results of our FASSD-Net versions are shown in Figure 5. As the quantitative results suggest, the most significant improvements occur on pixels belonging to large objects, such as trucks and buses. Compared to FC-HarDNet-70, all three FASSD-Nets better differentiate between car, bus, and truck classes. For a fair comparison, we have conducted the evaluation of our final models against FC-HarDNet-70 with its official weights from its open-source implementation.[2]

E. Evaluation on GPU-Powered Embedded Systems

The NVIDIA Jetson family [34] is a group of GPU-powered embedded devices with low power consumption designed by

[1] github.com/pytorch/pytorch/issues/18631

[2] github.com/PingoLH/FCHarDNet

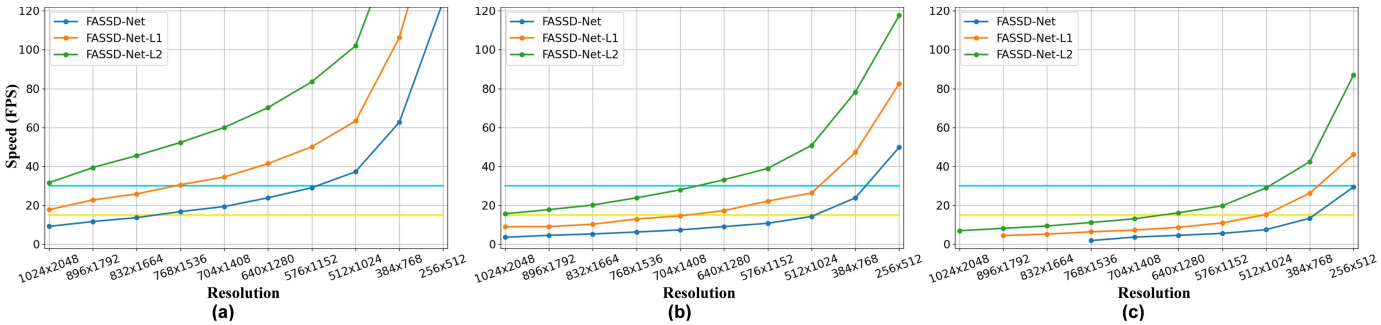


Fig. 7. Inference speed vs. input resolution of our proposals on the embedded systems. (a) Jetson Xavier NX. (b) Jetson TX2. (c) Jetson Nano. The cyan and yellow lines represent the real-time speed: 30 and 15 FPS, respectively.

TABLE VII
TECHNICAL SPECS OF THE NVIDIA JETSON EMBEDDED SYSTEMS

	Nano	TX2	Xavier NX
AI Perf.	472 GFLOPS	1.33 TFLOPS	21 TOPS
GPU	128-core Maxwell	256-core Pascal	384-core Volta
CPU (*ARM*)	Quad-core	Dual & Quad-core	Six-core
Memory	4 GB at 25.6GB/s	8 GB at 59.7GB/s	8 GB at 51.2GB/s
Storage	16 GB	32 GB	16 GB
Max. Power	10W	15W	15W

TABLE VIII
FASSD-NET'S SINGLE VS HALF PRECISION EVALUATION

Method	mIoU fp32	mIoU fp16	Error
FASSD-Net	78.791	78.789	0.002
FASSD-Net-L1	76.331	76.325	0.006
FASSD-Net-L2	74.058	74.061	-0.003

NVIDIA Corporation. Relevant specs of the three devices currently available on the market are listed in Table VII. These single-board systems can cover a wide range of possible AI applications based on requirements of computational complexity, model size, as well as power consumption, and price budget. Therefore, in this section, we present an extensive analysis of our proposals implemented on these devices: Jetson Nano, Jetson TX2, and Jetson Xavier NX.

We use JetPack 4.4 [54] (the official SDK for the Jetson family) with PyTorch 1.4 for all experiments. The SDK already incorporates TensorRT [55], an optimization tool for high performance deep learning inference. It is worth noting that this tool relies on the half-precision floating-point (fp16) as an essential speed optimization. Accordingly, we test the influence of half-precision vs. single-precision (fp32) on the accuracy of our models. As shown in Table VIII, the mIoU drop in half-precision is negligible. Specifically, the worst-case scenario presents a 0.006% of error. Hence, we use NVIDIA's TensorRT [56] with half-precision for all results on the embedded systems.

To precisely evaluate the accuracy-speed tradeoffs of our FASSD-Net variations on the embedded systems, we first have to test the degradation of the accuracy with respect to the input size. Therefore, we evaluate ten different input sizes, from 256×512 to full resolution (1024×2048) on the Cityscapes validation set. Figure 6 shows the results of our three models concerning the input resolution. We can see that FASSD-Net obtains better accuracy than the average SOTA results (73%) with a minimum resolution of 576×1152, which represents a downscale of more than three times the full resolution. On the other hand, with an input size of 512×1024, our three models achieve better results than the minimum accepted mIoU, defined as 65% [20], [57].

TABLE IX
COMPARISON WITH PREVIOUS WORKS ON THE JETSON TX2 SYSTEM

Method	Input Size	Speed (FPS)	mIoU
SwiftNet [23]	512×1024	6.1	70.2
ERFNet [28]	512×1024	7.1	71.5
SkipNet-ShuffleNet [18]	256×512	15.0	62.2
ThunderNet [20]	256×512	20.9	64.0
FASSD-Net (ours)	512×1024	14.3	72.1
FASSD-Net-L1 (ours)	576×1152	22.2	69.5
FASSD-Net-L2 (ours)	704×1408	28.5	71.0

Figure 7 shows the inference speed evaluation of our proposals on the NVIDIA Jetson embedded devices. On the new Jetson Xavier NX (Figure 7a), FASSD-Net-L2 surpasses the real-time requirement with full resolution input, achieving 74.1% of mIoU on single-board with less than 15W of power consumption. In the case of Jetson TX2 (Figure 7b), the most efficient result is obtained with a resolution of 704×1408, almost reaching real-time performance with 71% of mIoU. Similarly, on the Jetson Nano (Figure 7c), with an input size of 512×1024, FASSD-Net-L1 and FASSD-Net-L2 can run at 15 and 29 fps, respectively. Results considered outstanding for a semantic segmentation model that can run on a GPU-powered embedded device with only 128 CUDA cores.

Finally, in Table IX, we show the comparison of our network proposals against other methods for semantic segmentation implemented on the Jetson TX2. Our proposed FASSD-Net is more than 2× faster than the closest competitor in terms of accuracy, ERFNet [28]. Concerning the inference speed, FASSD-Net-L2 reaches 28.5 FPS, the best option covering the accuracy-speed tradeoffs. Note that, on the new Jetson Xavier NX, our approach can surpass the real-time requirement (31.7 FPS) with a SOTA accuracy of 74.1% of mIoU.

F. Comparison With SOTA Real-Time Methods on the Cityscapes Benchmark

Table X shows the overall comparison of our network proposals versus other SOTA methods for real-time semantic

TABLE X

COMPARISON WITH SOTA NETWORKS FOR REAL-TIME SEMANTIC SEGMENTATION ON THE CITYSCAPES BENCHMARK. CATEGORIZED BY SPEED PERFORMANCE, FROM TOP TO BOTTOM: NON-REAL-TIME, FASTER REAL-TIME (>120FPS), FAST REAL-TIME (>60FPS), AND REAL-TIME (>30FPS)

Method	Input Size	GPU	GFLOPs	Parameters	FPS	FPS (norm.)	mIoU (val)	mIoU (test)
LDN [58]	1024×2048	GTX 1080Ti	135.1	10.3M	13.6	13.6	79.0	79.3
DeepLabV3+ [26]	512×1024	Titan X (P)	-	-	≈ 1	≈ 1	79.6	81.9‡
PSPNet [30]	713×713	Titan X (P)	-	-	< 1	< 1	79.7	81.2‡
Fast-SCNN [41]	1024×2048	Titan XP	-	1.11M	123.5	110.3	69.2	68.0‡
SwiftNet [23]	512×1024	GTX 1080Ti	26.0	11.8M	134.9	134.9	70.2	-
FC-HarDNet-70 (L2)	1024×2048	GTX 1080Ti	6.6	2.86M	152.8	152.8	72.1	-
FASSD-Net-L2 (ours)	1024×2048	GTX 1080Ti	8.7	2.3M	133.1	133.1	74.1/76.7†	72.1/73.4†
ENet [59]	512×1024	Titan X (P)	-	0.37M	78.4	76.1	-	58.3
ESNet [29]	512×1024	GTX 1080Ti	-	1.66M	63.0	63.0	-	69.1
LEDNet [50]	512×1024	GTX 1080Ti	-	0.94M	71.0	71.0	-	70.6
ERFNet [28]	512×1024	Titan X (M)	-	2.10M	41.7	68.4	71.5	69.7
CAS [46]	768×1536	Titan XP	-	-	108.0	96.4	71.6	70.5
DFANet [16]	1024×1024	Titan X (P)	3.4	7.8M	100	97.1	-	71.3
DF1-Seg-d8 [42]	1024×2048	GTX 1080Ti	-	-	136.9*	82.9	72.4	71.4
FasterSeg [27]	1024×2048	GTX 1080Ti	28.2	4.4M	163.9*	99.3	73.1	71.5
FC-HarDNet-70 (L1)	1024×2048	GTX 1080Ti	15.7	4.1M	93.4	93.4	74.8	-
FASSD-Net-L1 (ours)	1024×2048	GTX 1080Ti	20.0	2.85M	78.0	78.0	76.3/77.9†	73.9/75.0†
ICNet [2]	1024×2048	Titan X (M)	28.3	26.5M	30.3	49.7	67.7	69.5
DABNet [19]	1024×2048	GTX 1080Ti	41.8	0.76M	27.7	27.7	69.1	70.1
GUN [60]	512×1024	Titan XP	-	-	33.3	29.7	69.6	70.4
LBN-FFN [61]	448×896	Titan X	49.5	6.2M	51.0	49.5	74.4	73.6
BiSeNet [3]	768×1536	Titan XP	-	49M	65.5	58.5	74.8	74.7
ShelfNet [62]	1024×2048	GTX 1080Ti	-	-	36.9	36.9	-	74.8
SwiftNet [23]	1024×2048	GTX 1080Ti	104.0	11.8M	39.9	39.9	75.4	75.5
FC-HarDNet-70 [22]	1024×2048	GTX 1080Ti	35.4	4.1M	53.0	53.0	77.4	76.0
FASSD-Net (ours)	1024×2048	GTX 1080Ti	45.1	2.85M	41.1	41.1	78.8/79.7†	76.0/77.5†

* Speed measured with TensorRT acceleration
† Using multi-scale evaluation
‡ Using coarse data

segmentation on the Cityscapes benchmark. The table is divided into three categories, based on the inference speed directly comparable to each of our three proposed networks. For fair comparisons under different GPU architectures, we follow the same protocol as Orsic *et al.* [23] and let the column *FPS (norm.)* provide a speed estimation of the model running on a GTX 1080Ti GPU. The scaling factors are: 1.0 for GTX 1080Ti, 0.61 for Titan X (Maxwell), 1.03 for Titan X (Pascal), and 1.12 for Titan XP.

Our main network, FASSD-Net, surpasses by a considerable margin the mIoU score of all other methods for real-time semantic segmentation, requiring 1.44× fewer parameters and being 1.4% more accurate than the closest competitor FC-HarDNet-70. A particular case is the LDN [58] approach, which claims to achieve real-time performance with TensorRT on a Titan XP GPU. However, with the official source code running on a GTX 1080Ti GPU without TensorRT, it reaches just 13.6 FPS.

Our second network, FASSD-Net-L1, resembles the reported results of BiSeNet [3] in mIoU accuracy and FPS. However, the speed of BiSeNet has been originally measured on 768 × 1536 images on a Titan XP. For a fair comparison, and according to Zhuang *et al.* [62], we adjust its speed to 37 FPS evaluated at 1024 × 2048 size on a GTX 1080Ti GPU. Therefore, resulting in our network being about 2.1× faster, 1.5% more accurate, and requiring 17.2× fewer parameters.

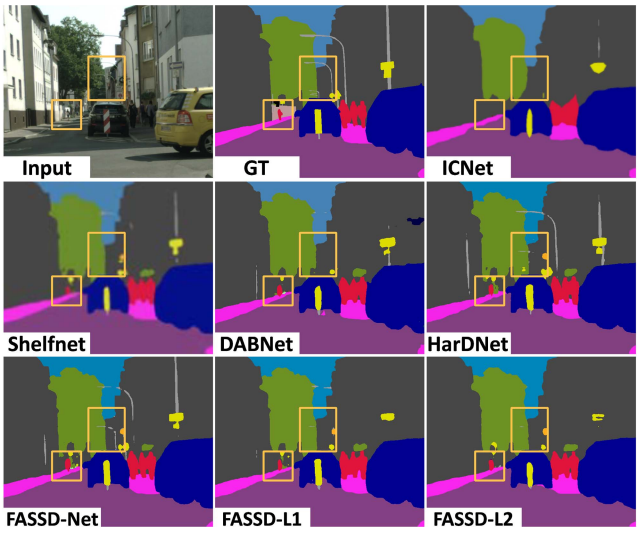


Fig. 8. Qualitative comparison with SOTA networks for real-time semantic segmentation on the Cityscapes validation set. Critical regions are highlighted with yellow squares.

Similarly, FASSD-Net-L2 can be compared to FasterSeg [27] and DF1-Seg-d8 [42], which were designed and optimized by NAS methodologies. Both methods utilize TensorRT acceleration [55] to increase their speed performance. For a fair comparison, we let 1.65× be the

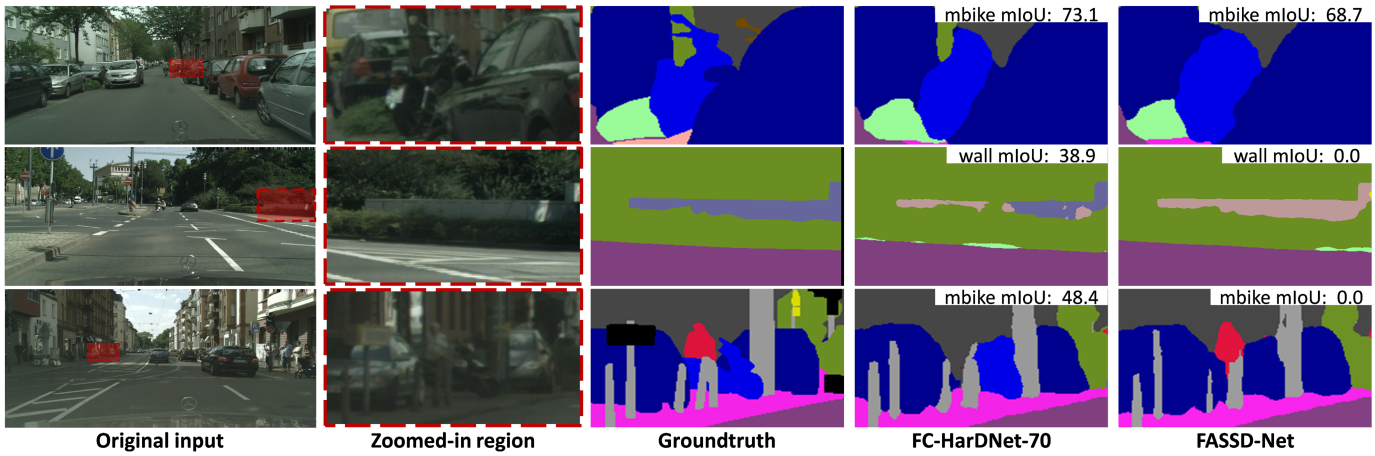


Fig. 9. Example of common errors on classes wall and motorbike. The last two rows show the worst scenarios.

acceleration factor of TensorRT. This value is approximated from works that present results with and without TensorRT, such as FasterSeg [27]. Under these assumptions, our FASSD-Net-L2 is faster than FasterSeg and DF1-Seg-d8, while being 1% and 1.7% more accurate, respectively. It is worth noting that, due to the fast inference speed of our proposal, we also present results using a multi-scale evaluation with four scales (0.5, 0.75, 1, 1.25). So that, even with four consecutive forward passes, FASSD-Net-L2 can surpass the real-time requirement (with 33.2 FPS).

Figure 8 shows the qualitative results of our proposals versus other SOTA methods for real-time semantic segmentation on the Cityscapes validation set. Note that these images are zoomed-in from the original 1024×2048 size. Thus, it is clear that our networks can segment small objects such as the person and the pole light in the highlighted regions.

G. Analysis of Common Errors

From Table VI, we observed that FASSD-Net obtains the highest score in 17 out of 19 classes, outperforming the current SOTA model, FC-HarDNet-70. Wall and motorbike are the two classes that present a deficit of 0.4 of mIoU; hence, Figure 9 shows common and worst-case errors of both. As we can see, failures appear on significantly small objects, usually misrecognized as classes from the same category,[3] wall as a fence, and motorbike as a car. Thus, the mIoU of categories *construction* and *vehicle* from FASSD-Net are higher than FC-HarDNet-70 (93.3 & 94.1 versus 92.9 & 93.7, respectively). We attribute the failure cases to the short presence of these two classes on the dataset, affecting the quality of context features learned from them. Besides, FC-HarDNet-70 presents a deeper encoder architecture with five convolutional blocks instead of four, as mentioned in Section III-A. The high efficiency of our proposal lets the option to design a deep ensemble model from different folds of training data as a straightforward solution to the problem. Note that this technique is out of the scope of this paper; details can be found in [63].

[3]see the original Cityscapes paper [36] for details on categories and classes.

TABLE XI
COMPARISON WITH SOTA NETWORKS ON THE CAMVID BENCHMARK

Method	Input Size	GPU	FPS (norm.)	mIoU
ENet [59]	360x480	Titan X (P)	59.4	51.3
SegNet [64]	360x480	Titan X (M)	49.0	55.6
DFANet [16]	720x960	Titan X (P)	116.5	64.7
FC-HarDNet-70 [22]	720x960	GTX 1080Ti	84.3	66.7
ICNet [2]	720x960	Titan X (M)	46.3	67.1
LBN-FFN [61]	720x960	Titan X (P)	38.2	68.0
BiSeNet [3]	720x960	Titan XP	63.4	68.7
LDN [58]	720x960	Titan XP	43.8	70.9
ESSN [21]	540x720	GTX 1070	49.8	71.6
SwiftNet [23]	720x960	GTX 1080Ti	74.3	71.6
FASSD-Net (ours)	720x960	GTX 1080Ti	80.0	69.3

H. Comparison With SOTA Real-Time Methods on the CamVid Benchmark

Finally, to illustrate the generality and effectiveness of our proposal, Table XI shows the results of FASSD-Net versus SOTA works evaluated on the CamVid benchmark. We can see that our proposal is within the top-3 works of the best efficiency and accuracy. Specifically, FASSD-Net is more efficient than SwiftNet [23] (7% faster), but it is less accurate (1.3% lower). On the other hand, our model is significantly more accurate than the fastest methods DFANet [16], and FC-HarDNet-70 [22]. These results demonstrate that FASSD-Net is comparable with the SOTA methods trained on small datasets, such as CamVid.

V. CONCLUSION

In this paper, we focus on reducing the accuracy gap between real-time and non-real-time semantic segmentation networks. For this purpose, we have proposed the DAPF and MDA modules, which exploit the contextual information in several stages of the decoder and boost the accuracy performance of the baseline network. Using our two proposals jointly with a highly efficient network, we have designed three architecture variations that can be chosen depending on the computational budget. Our main network, FASSD-Net, sets the new state-of-the-art mIoU accuracy for real-time semantic

segmentation on the Cityscapes at full resolution (1024 × 2048). Moreover, our proposed FASSD-Net-L2 can even run real-time on low-power consumption embedded systems such as Jetson Xavier NX. As future work, we plan to analyze channel pruning and knowledge distillation options to further speed up our proposal. Besides, we would like to evaluate it in different scenarios, such as indoor parsing and medical images.

REFERENCES

[1] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "CCNet: Criss-cross attention for semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 603–612.

[2] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for real-time semantic segmentation on high-resolution images," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 405–420.

[3] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiSeNet: Bilateral segmentation network for real-time semantic segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 325–341.

[4] P. Hu *et al.*, "Real-time semantic segmentation with fast attention," *IEEE Robot. Autom. Lett.*, vol. 6, no. 1, pp. 263–270, Jan. 2021.

[5] Y. Pei, B. Sun, and S. Li, "Multifeature selective fusion network for real-time driving scene parsing," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–12, 2021.

[6] Q. Song, K. Mei, and R. Huang, "AttaNet: Attention-augmented network for fast and accurate scene parsing," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 3, 2021, pp. 2567–2575.

[7] Q. Wang, J. Gao, and X. Li, "Weakly supervised adversarial domain adaptation for semantic segmentation in urban scenes," *IEEE Trans. Image Process.*, vol. 28, no. 9, pp. 4376–4386, Sep. 2019.

[8] Q. Wang, J. Gao, and Y. Yuan, "A joint convolutional neural networks and context transfer for street scenes labeling," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 5, pp. 1457–1470, May 2017.

[9] W. Shi *et al.*, "Multilevel cross-aware RGBD indoor semantic segmentation for bionic binocular robot," *IEEE Trans. Med. Robot. Bionics*, vol. 2, no. 3, pp. 382–390, Aug. 2020.

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent*. Cham, Switzerland: Springer, 2015, pp. 234–241.

[11] H. Zhang, B. Han, C. Y. Ip, and P. Mohapatra, "Slimmer: Accelerating 3D semantic segmentation for mobile augmented reality," in *Proc. IEEE 17th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Dec. 2020, pp. 603–612.

[12] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, "A survey on deep learning techniques for image and video semantic segmentation," *Appl. Soft Comput.*, vol. 70, pp. 41–65, Sep. 2018.

[13] K. Yang, X. Hu, L. M. Bergasa, E. Romera, and K. Wang, "Pass: Panoramic annular semantic segmentation," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 10, pp. 4171–4185, Oct. 2019.

[14] L. Sun, K. Yang, X. Hu, W. Hu, and K. Wang, "Real-time fusion network for RGB-D semantic segmentation incorporating unexpected obstacle detection for road-driving images," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5558–5565, Oct. 2020.

[15] J. Zhang, K. Yang, and R. Stiefelhagen, "ISSAFE: Improving semantic segmentation in accidents by fusing event-based data," 2020, *arXiv:2008.08974*.

[16] H. Li, P. Xiong, H. Fan, and J. Sun, "DFANet: Deep feature aggregation for real-time semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9522–9531.

[17] J. Fu *et al.*, "Dual attention network for scene segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3146–3154.

[18] M. Siam, M. Gamal, M. Abdel-Razek, S. Yogamani, M. Jagersand, and H. Zhang, "A comparative study of real-time semantic segmentation for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 587–597.

[19] G. Li and J. Kim, "DABNet: Depth-wise asymmetric bottleneck for real-time semantic segmentation," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2019, pp. 1–12.

[20] W. Xiang, H. Mao, and V. Athitsos, "ThunderNet: A turbo unified network for real-time semantic segmentation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 1789–1796.

[21] D. S. Kim, M. Arsalan, M. Owais, and K. R. Park, "ESSN: Enhanced semantic segmentation network by residual concatenation of feature maps," *IEEE Access*, vol. 8, pp. 21363–21379, 2020.

[22] P. Chao, C.-Y. Kao, Y. Ruan, C.-H. Huang, and Y.-L. Lin, "HarDNet: A low memory traffic network," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3552–3561.

[23] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic, "In defense of pre-trained ImageNet architectures for real-time semantic segmentation of road-driving images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12607–12616.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[25] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds., 2016, pp. 1–13.

[26] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 801–818.

[27] W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li, and Z. Wang, "FasterSeg: Searching for faster real-time semantic segmentation," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–14.

[28] E. Romera, J. M. Álvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient residual factorized ConvNet for real-time semantic segmentation," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 263–272, Jan. 2018.

[29] Y. Wang, Q. Zhou, J. Xiong, X. Wu, and X. Jin, "ESNet: An efficient symmetric network for real-time semantic segmentation," in *Proc. Chin. Conf. Pattern Recognit. Comput. Vis. (PRCV)*, 2019, pp. 41–52.

[30] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2881–2890.

[31] T. Takikawa, D. Acuna, V. Jampani, and S. Fidler, "Gated-SCNN: Gated shape CNNs for semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5229–5238.

[32] Z. Tian, T. He, C. Shen, and Y. Yan, "Decoders matter for semantic segmentation: Data-dependent decoding enables flexible feature aggregation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3126–3135.

[33] X. Li, Z. Zhong, J. Wu, Y. Yang, Z. Lin, and H. Liu, "Expectation-maximization attention networks for semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9167–9176.

[34] *NVIDIA Jetson Systems*. Accessed: Jun. 30, 2020. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/

[35] L. Rosas-Arias, G. Benitez-Garcia, J. Portillo-Portillo, G. Sanchez-Perez, and K. Yanai, "Fast and accurate real-time semantic segmentation with dilated asymmetric convolutions," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, 2021, pp. 2264–2271.

[36] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3213–3223.

[37] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognit. Lett.*, vol. 30, no. 2, pp. 88–97, 2009.

[38] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[39] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 116–131.

[40] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.

[41] R. P. K. Poudel, S. Liwicki, and R. Cipolla, "Fast-SCNN: Fast semantic segmentation network," 2019, *arXiv:1902.04502*.

[42] X. Li, Y. Zhou, Z. Pan, and J. Feng, "Partial order pruning: For best speed/accuracy trade-off in neural architecture search," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9145–9153.

[43] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4340–4349.

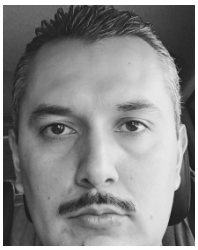
[44] Y. Liu, K. Chen, C. Liu, Z. Qin, Z. Luo, and J. Wang, "Structured knowledge distillation for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2604–2613.

[45] K. Yang, X. Hu, Y. Fang, K. Wang, and R. Stiefelhagen, "Omnisupervised omnidirectional semantic segmentation," *IEEE Trans. Intell. Transp. Syst.*, early access, Sep. 23, 2020, doi: 10.1109/TITS.2020.3023331.
[46] Y. Zhang, Z. Qiu, J. Liu, T. Yao, D. Liu, and T. Mei, "Customizable architecture search for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11641–11650.
[47] R. Zhao, Y. Hu, J. Dotzel, and Z. Zhang, "Improving neural network quantization without retraining using outlier channel splitting," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7543–7552.
[48] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–13.
[49] S. Mehta, M. Rastegari, L. Shapiro, and H. Hajishirzi, "ESPNetv2: A light-weight, power efficient, and general purpose convolutional neural network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9190–9200.
[50] Y. Wang *et al.*, "LEDNet: A lightweight encoder-decoder network for real-time semantic segmentation," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 1860–1864.
[51] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.
[52] Z. Wu, C. Shen, and A. van den Hengel, "High-performance semantic segmentation using very deep fully convolutional networks," 2016, *arXiv:1604.04339*.
[53] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[54] *JetPack SDK*. Accessed: Jun. 30, 2020. [Online]. Available: https://developer.nvidia.com/embedded/jetpack
[55] *TensorRT*. Accessed: Jun. 30, 2020. [Online]. Available: https://developer.nvidia.com/tensorrt
[56] *PyTorch to TensorRT*. Accessed: Jun. 30, 2020. [Online]. Available: https://github.com/NVIDIA-AI-IoT/torch2trt
[57] R. P. K. Poudel, U. Bonde, S. Liwicki, and C. Zach, "ContextNet: Exploring context and detail for semantic segmentation in real-time," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2018, pp. 1–12.
[58] I. Kreso, J. Krapac, and S. Segvic, "Efficient ladder-style DenseNets for semantic segmentation of large images," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 4951–4961, Aug. 2021.
[59] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A deep neural network architecture for real-time semantic segmentation," 2016, *arXiv:1606.02147*.
[60] D. Mazzini, "Guided upsampling network for real-time semantic segmentation," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2018, pp. 1–12.
[61] G. Dong, Y. Yan, C. Shen, and H. Wang, "Real-time high-performance semantic image segmentation of urban street scenes," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3258–3274, Jun. 2021.
[62] J. Zhuang, J. Yang, L. Gu, and N. Dvornek, "ShelfNet for fast semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV) Workshops*, Oct. 2019, pp. 847–856.
[63] J. M. Kaminski, I. Allodi, R. Monta nana-Rosell, R. Selvan, and O. Kiehn, "Deep ensemble model for segmenting microscopy images in the presence of limited labeled data," in *Proc. Med. Imag. Deep Learn. (MIDL)*, 2021, pp. 22–33.
[64] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

Leonel Rosas-Arias received the B.S. degree in electronics engineering from the Tecnológico Nacional de México in 2018. He is currently pursuing the M.Eng. degree with the Mechanical Engineering School, National Polytechnic Institute (IPN), Mexico. He has also done a research internship at The University of Electro-Communications (UEC), Japan, where he focused on the field of real-time semantic segmentation. His current research interests are in the areas of computer vision, autonomous vehicles, and robotics.

Gibran Benitez-Garcia (Member, IEEE) received the B.S. and M.S. degrees from the Mechanical Engineering School, National Polytechnic Institute, Mexico City, in 2011 and 2014, respectively, and the Ph.D. degree from The University of Electro-Communications (UEC), Tokyo, Japan, in 2017. From 2017 to 2019, he was a Post-Doctoral Fellow with the IIM Laboratory, Toyota Technological Institute, Nagoya, Japan. He is currently a Post-Doctoral Fellow with the Yanai Laboratory, UEC. His main research interests include face/facial expression recognition and human–computer interaction.

Jose Portillo-Portillo received the B.S. degree in electronic engineering with specialty in instrumentation and control, the M.S. degree in electronic engineering, and the Ph.D. degree in electronic and communications from the National Polytechnic Institute in 2007, 2012, and 2017, respectively. He is a member of the National Researchers System of Mexico. His current research interests include biometrics, AI, deep learning, and computer vision.



Jesus Olivares-Mercado received the B.S., M.S., and Ph.D. degrees from the National Polytechnic Institute, Mexico, in 2006, 2008, and 2012, respectively. In 2009, he received the Best Student Award from the National Polytechnic Institute, for his master's research work in the biometrics area. He realized post-doctoral studies at the CINVESTAV Tamaulipas Unit from 2012 to 2013. He is a member of the National Researchers System of Mexico.

Gabriel Sanchez-Perez received the B.S. degree in computer science engineering and the Ph.D. degree in electronic and communications from the National Polytechnic Institute, Mexico City, in 1999 and 2005, respectively. From 2007 to 2008, he realized post-doctoral studies at INAOE, Puebla. In January 2003, he joined the Mechanical and Electrical Engineering School, National Polytechnic Institute, where he is currently a Professor. He is a member of the National Researchers System of Mexico.

Keiji Yanai (Member, IEEE) received the B.Eng., M.Eng., and D.Eng. degrees from The University of Tokyo in 1995, 1997, and 2003, respectively. From 1997 to 2006, he was a Research Associate and an Associate Professor (till 2015) with the Department of Computer Science, The University of Electro-Communications, Tokyo, Japan. From November 2003 to September 2004, he was a Visiting Scholar with the Department of Computer Science, University of Arizona, USA. He is currently a Professor with the Department of Informatics, The University of Electro-Communications. His research interests include object recognition, deep learning, and Web multimedia mining.