IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.20XX.DOI

Multi-encoder Context Aggregation Network for Structured and Unstructured Urban Street Scene Analysis

TANMAY SINGHA[1], DUC-SON PHAM[1], and ANEESH KRISHNA[1]
[1]School of Electrical Engineering, Computing, and Mathematical Sciences, Curtin University, Perth, WA 6102, Australia
Corresponding author: Duc-Son Pham (e-mail: dspham@ieee.org).

ABSTRACT Developing computationally efficient semantic segmentation models that are suitable for resource-constrained mobile devices is an open challenge in computer vision research. To address this challenge, we propose a novel real-time semantic scene segmentation model called Multi-encoder Context Aggregation Network (MCANet), which offers the best combination of low model complexity and state-of-the-art (SOTA) performance on benchmark datasets. While we follow the multi-encoder approach, our novelty lies in the varying number of scales to capture both global context and local details effectively. We introduce suitable lateral connections between sub-encoders for improved feature refinement. We also optimize the backbone by exploiting the residual block of MobileNet for resource-constrained applications. On the decoder side, the proposed model includes a new Local and Global Context Aggregation (LGCA) module that significantly enhances semantic details in the segmentation output. Finally, we use several known efficient convolution techniques for the classification module to make the model more computationally efficient. We provide a comprehensive evaluation of MCANet on multiple datasets containing structured and unstructured urban street scenes. Among the existing real-time models with less than 3 million parameters, the proposed model is more competitive as it achieves the SOTA performance without ImageNet pre-trained weights on both structured and unstructured environments while being more compact for resource-constrained applications.

INDEX TERMS semantic segmentation, feature scaling, feature aggregation, deep learning, scene understanding, convolutional neural networks

I. INTRODUCTION

Scene understanding is a crucial task in many learning systems and has numerous applications, including self-driving vehicles [1], [2], human-computer interaction, virtual reality [3], object detection [4], [5], medical image analysis [6], [7], [8] and online video surveillance [9]. Semantic scene segmentation is a fundamental step towards achieving scene understanding. The goal of semantic segmentation is to recognize and localize different categories in a scene, assigning a class or a label to every pixel. The categories can vary depending on the specific application, as shown in Figure 1.

A semantic segmentation model usually follows an encoder-decoder structure where the encoder extracts semantic information, and the decoder projects it back to the input space for individual pixel classification. Inspired by the success of Deep Convolution Neural Networks (DCNNs) in general classification tasks, many off-line semantic segmentation models have been developed with deep architectures [10], [11], based on well-known backbone networks, usually ResNet [12], suitable for the segmentation task. For instance, DeepLab [11] is an approach that exploits ResNet by removing the striding operation from the last few ResNet blocks. Additionally, by utilizing high dilation rates in the feature scaling module, DeepLab ensures a wider field of view for context engrossment.

Later, the ResNet model pre-trained on ImageNet with dilated convolutions has become a popular choice as a feature extractor for scene segmentation models, including DeepLabV3 [13], PSPNet [14], HANet [15], and OCR [16]. Although these DCNN-based models have shown outstanding performance, many of them are not designed for mobile

Figure 1. Labelling class to each pixel in semantic segmentation. (a) Original input image, (b) colored annotation of the input image where each pixel of the image has a specific color code.

devices and other resource-constrained applications. Therefore, they cannot achieve satisfactory real-time performance on embedded devices.

In many practical applications, such as mobile and IoT devices [17], [18], the available computing resources are limited. Therefore, it is prohibitive to deploy large models that can achieve satisfactory real-time performance. There has been a growing interest in developing lightweight semantic segmentation models [19], [20], [21] to target these specific applications. These real-time models aim to reduce the computational cost of existing offline models while still achieving satisfactory segmentation performance.

One major challenge with developing real-time models for mobile devices and other resource-constrained applications is the high input resolution with a large field of view, which can easily increase memory usage. To overcome this problem, some real-time models, such as BiseNet [21], ICNet [22], RefineNet [23], and ContextNet [24], have introduced a new encoder design, called a *multi-branch* encoder. This consists of a shallow branch for a high-resolution input and a deep branch for a low-resolution input. As the shallow branch has fewer layers, this approach effectively reduces the computational cost while still controlling the field of view (via the shallow branch) and maintaining global contextual information (through the deep branch). Although several models have achieved computational efficiency, there is still a considerable performance gap between existing offline and real-time semantic scene segmentation models. Developing real-time lightweight models suitable for mobile devices and resource-constrained applications is still an open research question.

In this work, we address the above challenge for mobile devices and other embedded devices with inadequate hardware facilities through a novel architecture. Our solution starts with the observation from the literature on real-time semantic segmentation that the input needs to be processed at multiple scales with larger receptive fields to achieve better contextual details for improved scene understanding. We introduce a completely new multi-encoder architecture in this study. Here, the number of stages in each successive sub-encoder is reduced; however, the repetition of inverted residual blocks is increased in the successive sub-encoder. Consequently, each sub-encoder gets deeper and deeper than the previous sub-encoder and can extract more semantic information of the scene. Lateral connections are also used

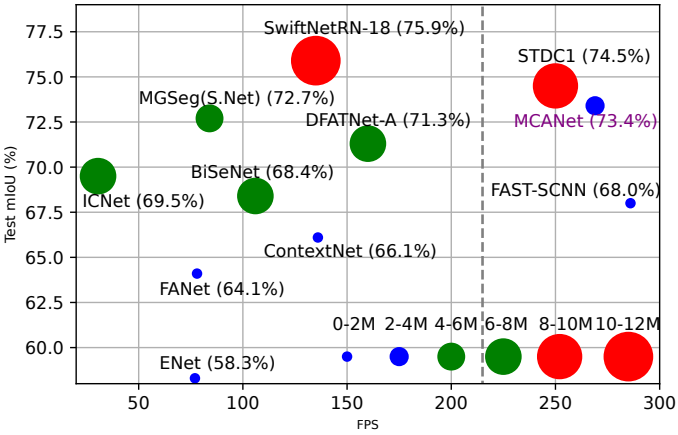

Figure 2. Test accuracy vs Parameters among real-time models

at the same stage. After the complete encoding process, we obtain five rich global feature maps at different scales, which we then use for feature fusion at the decoder end. We prove that MCANet allows us to achieve excellent semantic segmentation performance while keeping the model complexity relatively low. Our design is at least two times smaller than existing real-time scene segmentation models that are producing the SOTA result, giving our model a competitive advantage in resource-constrained applications.

We make three major contributions in this work. Firstly, we introduce a novel architecture, named MCANet, with multiple sub-encoders designed specifically for optimal feature scaling. At the same time, we also reduce the number of stages in each successive sub-encoders to control the number of model parameters. Secondly, we introduce an effective multi-stage module for local and global feature aggregation at the decoder which combines feature maps at different levels produced by the proposed backbone network. Finally, we provide comprehensive experiments on both structured and unstructured environments with various number of classes and demonstrate the model's superior performance in all circumstances among the existing real-time semantic segmentation models having less than 3 million (M) parameters. On structured dataset such as CamVid [25], BDD100K [26] and KITTI [27], and on unstructured dataset such as IDD-lite, the proposed model produces the SOTA performance among the existing real-time semantic segmentation models. On Cityscapes [28], the proposed model generates 73.4% test accuracy without ImageNet [29] or any pre-trained weights, which is the best performance among the existing real-time semantic models of having less than 3 M parameters. It can be visualized in Figure 2 which plots Cityscapes test mIoU (%) against FPS. The size of the circle in Figure 2 depicts the size of the model.

The paper is organized as follows. In Section II, we present related work in semantic segmentation. Section III details our design and Section IV discusses numerous experiments. Concluding remarks are given in Section V.

II. BACKGROUND

A. ONE-BRANCH DESIGN

As shown in Figure 3(a), this was a simple encoder-decoder architecture [10] used in the early days of semantic segmentation, such as FCN [10], DeepLab [11], BiseNet [21], and UNet [30]. The encoder typically contains a deep neural network to extract the contextual details of the scene and large backbone networks such as ResNet [12], VGG-16 [31], and Xception [32] are common choices. An extension of the one-branch approach is to include feature scaling (see Figure 3(b)) which is known to capture contextual details through a larger receptive field. For instance, PSPNet [14] introduced Pyramid Pooling Module (PPM) which uses four image pooling branches with different bin sizes. DeepLabV3+ [13] introduced Atrous Spatial Pyramid Pooling (ASPP) which utilizes five dilation branches with different dilation rates. Using ResNet-101 [33] as the backbone, it achieved 82.1% test accuracy on Cityscapes [28]. Similar to feature scaling, few semantic segmentation models [21], [34], [35], [36] started to use the attention mechanism for guiding the feature learning process using high-level information.

Recently, a new model known as MGSeg [37] improves model efficiency further by using a lightweight backbone (ResNet-18), a hybrid feature attention and feature scaling module. It produces 77.8% test accuracy on Cityscapes whilst having 13.3 M parameters and 96.5 GFLOPs (Giga Floating Point Operations) at 1024×1024 input resolution. Due to the large number of parameters in their design, the computation can be more prohibitive for high-resolution input images and hence they are not suitable for resource-constrained applications.

B. MULTI-BRANCH DESIGN

The multi-branch encoder approach has been introduced recently to address the computational burden of the one-branch approach. Figure 3(c) shows the general architecture of multi-branch encoders used in ICNet [22], RefineNet [23],ContextNet [24], and SwiftNet [38]. These models typically have a dedicated deep branch that accepts lower resolution input images and produces rich global feature maps. Additionally, several parallel shallow branches are deployed to extract low-level feature maps at higher resolutions. Therefore, a multi-branch encoder can handle higher resolution input images without incurring high computational costs. Despite being more efficient, the open challenge with the multi-branch encoder approach is to close the performance gap with the one-branch deep encoder approach. For instance, recently a new multi-branch semantic segmentation model, called SwiftNet [38], is introduced which has almost 4 times less parameters than $DeepLabV3+$ [13]. However, its test accuracy on Cityscapes is still 10% lower.

C. DUAL-BRANCH WITH DOWN-SAMPLING TECHNIQUE

This approach, as shown in Figure 3(d), is a deviation from the multi-branch approach. Here, the encoder network accepts only one input and a few convolution layers are used to down-sample the input image. After that, two branches are created: a deep branch is designed to extract rich global feature maps by exploiting a series of residual blocks whilst a shallow branch is to uproot local features using few convolution layers. Models such as such as BiseNet [21], Fast-SCNN [39], FANet [40], ESPNet [41] achieve better accuracy and efficiency due to less data pre-processing.

Table 1. Bottleneck residual block

Input	Operator	Output
$h \times w \times c$	1×1 Conv,1/1, Relu	$h \times w \times tc$
$h \times w \times tc$	3×3 DwConv, 3/s, Relu	$h/s \times w/s \times tc$
$h/s \times w/s \times tc$	1×1 Conv,1/1, -	$h/s \times w/s \times c'$

D. FEATURE REUSE IN SUB-ENCODERS

It is known that deep convolution layers learn more contextual details than the layers at the initial stage, and problems like vanishing gradient can be diminished. Based on this fact, DFANet [42] has introduced the concept of feature reuse in its sub-encoders. Figure 3(e) demonstrates that the global feature of the first sub-encoder is used as input for the second sub-encoder. Before being fed to the next sub-encoder, the global feature is upsampled 2^3 times and passed through a fully connected (FC) attention module for better refinement. Thus, the features from a previous sub-encoder are reused in the next subsequent sub-encoder. Due to feature reuse, DFANet achieves 71.3% test accuracy on the Cityscapes test set while having 7.8 M parameters.

In contrast to these designs, we propose a multi-encoder network for reusing feature maps through dynamic sub-encoders and producing refined output by multiple decoding paths. Figure 3(f) shows a layout of feature reuse by multi-encoder design. The details of the proposed design are discussed in the next section.

III. PROPOSED METHOD

Figure 4 depicts the end-to-end design of our proposed MCANet. The encoder architecture is based on the concept of reusing feature maps in a multi-encoder design. Specifically, features at lower resolutions contain rich semantic details that require multiple refinements to capture the full context. To achieve this, we employ multiple encoders that effectively utilize these features.

A. MULTI-ENCODER

Figure 4(a) outlines the design of our proposed multi-encoder. In this design, we carefully select individual components to target mobile devices and other resource-constrained applications and optimally construct an encoder network to achieve the best extraction of semantic features.

Empirically, it has been shown that MobileNet [43] bottleneck residual convolution blocks (MBConv) are more efficient for mobile devices than other existing residual blocks [44], [43], [45]. The optimized architecture of these residual

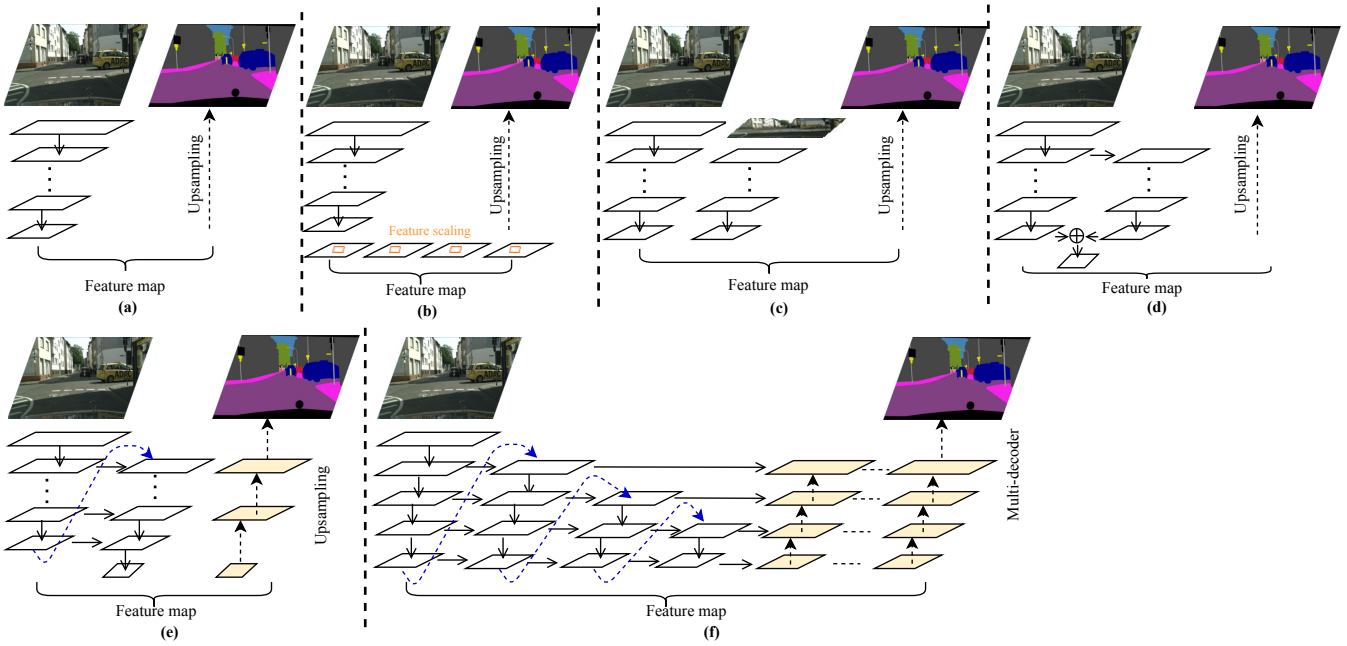

Figure 3. Different architectures of semantic segmentation model: (a) One-branch, (b) One-branch with feature scaling technique, (c) Multi-branch, (d) Dual-branch with down-sampling, (e) Feature re-use in sub-encoders with increasing stages, (f) Feature re-use in multiple sub-encoders with decreasing stages, but increasing depth in each sub-encoder and feature refinement through multiple decoding paths.

Table 2. Layer architecture of the proposed multi-encoder

Stage	Input	Operators	Width multiplier (M_w)	Depth multiplier (M_d)	stride	Output
Layer architecture of first sub-encoder						
1	1024×2048×3	Conv, k3×3	-	1	2	512×1024×32
2	512×1024×32	MBConv1, k3×3	0.75	1	2	256×512×24
-	256×512×24	MBConv6, k3×3	1.34	2	1	256×512×32
3	256×512×32	MBConv6, k3×3	1.5	2	2	128×256×48
4	128×256×48	MBConv6, k3×3	1.34	2	2	64×128×64
5	64×128×64	MBConv6, k3×3	1.5	2	2	32×64×96
6	32×64×96	MBConv6, k3×3	1.34	2	2	16×32×128
Layer architecture of second sub-encoder						
4	128×256×48	MBConv6, k3×3	1.34	3	2	64×128×64
5	64×128×64	MBConv6, k3×3	1.5	2	2	32×64×96
6	32×64×96	MBConv6, k3×3	1.34	2	2	16×32×128
Layer architecture of third sub-encoder						
5	64×128×64	MBConv6, k3×3	1.5	3	2	32×64×96
6	32×64×96	MBConv6, k3×3	1.34	2	2	16×32×128
Layer architecture of fourth sub-encoder						
6	32×64×96	MBConv6, k3×3	1.34	3	2	16×32×128

*Feature F7 is created using a MaxPooling operation after the forth sub-encoder.

blocks and the utilization of depth-wise separable convolution layers in the bottleneck intermediate expansion stage make MBConv much more computationally efficient. For that reason, we decided to use MBConv blocks to build our multi-encoder. The layered architecture of an MBConv block is shown in Table 1. As can be seen, an input feature F_i with spatial dimensions $h \times w$ and channel dimension c is first filtered by a standard convolution layer and produces an output of size $h \times w \times tc$. The number of channels of the input feature is increased by an expansion factor t. The intermediate expansion stage uses a lightweight depth-wise convolution layer that reduces the computational cost by 8 to 9 times compared to a standard convolution layer. Thus, it optimizes the overall number of model parameters and GFLOPs count. Following [43], we utilize MBConv6 blocks to design our backbone. Except for the first residual block, we use an expansion ratio of 6 for other blocks. To better preserve contextual details, we do not apply ReLU non-linearity in the last layer of each MBConv block.

After down-sampling the original input image using a standard convolution layer, we use 11 MBConv blocks of varying expansion ratios to construct the first sub-encoder of our proposed multi-encoder. The literature suggests that using MBConv blocks of different expansion ratios at the initial stage can retain more contextual and spatial details due to its squeeze and excitation architecture [43]. The depth

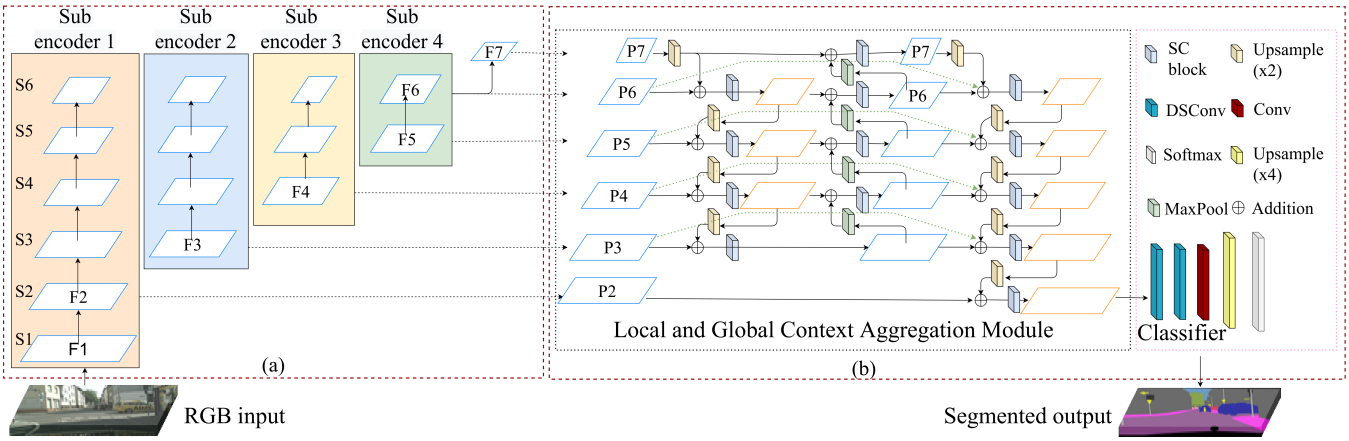

Figure 4. Complete pipeline of our proposed MCANet: (a) Multi-encoder design: Feature re-use in sub-encoders. Black dotted lines define the lateral connections from the previous sub-encoder at the same levels and green dotted lines show reusing of global feature map in the next sub-encoder. Feature F7 is produced using a simple pooling operation. (b) Decoder Design: Local and Global Context Aggregation (LGCA) module and Classifier module.

and width of each block are controlled by two tunable hyper-parameters: the width (M_w) and depth (M_d) multipliers. For an input feature map F_i of size $h_i \times w_i \times c_i$, each MBConv block produces an output feature map F_o of size $h_i/s \times w_i/s \times M_w.c_i$. Here, h_i, w_i, and c_i denote the height, width, and number of channels of the input feature map, respectively. The stride s is used to control the number of stages in the encoder, and whenever s becomes 2, one new stage is created by down-sampling the input feature map size by half. Thus, we generate all six stages in the first sub-encoder. Hyper-parameter M_d controls the depth of the encoder by controlling the number of repetitions of each MBConv block. To keep our design simple, we repeat each block in the first sub-encoder twice, except the first block. Following [43], we set the range of the width multiplier from 0.75 to 1.5 and generate a maximum of 128 channels for the global feature maps. Models like DeepLab [13], PSPNet [14], and HANet [15] have global features with a maximum of 2048 channels, which contribute to large-scale parameters and GFLOPs. By setting the lower range for the width multiplier, we achieve the best trade-off between model performance and efficiency for mobile devices. Thus, the first sub-encoder generates rich spatial and global features without contributing a large number of parameters and GFLOPs. The complete layer architecture of our proposed multi-encoder is illustrated in Table 2.

Empirically, it has been shown that high-level features at lower resolutions contain rich semantic details that are easy for attention weight learning [13], [37], [42]. As a result, many models such as DRANet [35] and DFANet [42] introduce an attention module in later stages of their encoder network. The attention mechanism employs global pooling to capture global contexts and guides the feature learning process by computing an attention vector. In DFANet, the final global feature of each sub-encoder passes through the fully connected (FC) attention module before being fed as an input feature map for the next sub-encoder. It has been demonstrated that by deploying the FC module, model performance is enhanced by 4-6%. However, the effective design of our proposed multi-encoder eliminates the need for an attention module on top of each sub-encoder [42], thereby reducing the number of parameters and computational cost. In DFANet, the layered architecture of each sub-encoder is the same, but the spatial dimensions of the input feature map of each sub-encoder are different. Consequently, an additional deep stage is created after each sub-encoder. The drawback of this architecture is that the deep features are not optimally reused. For instance, features at the fourth, fifth, and sixth stages in DFANet are reused three, two, and one times, respectively. However, high-level features (the sixth stage) should be processed more compared to intermediate and shallow features. Moreover, the uniform layered architecture of each sub-encoder is ineffective in gaining any additional knowledge while reusing deep feature maps.

$$F^2_{l_3} = Conv(Upsample8(F^1_{l_6}) \tag{1}$$

$$F^2_{l_3} = F^1_{l_3} + F^2_{l_3}. \tag{2}$$

In contrast to DFANet, our first sub-encoder consists of all six stages, producing both local and deep global features. The final feature at the sixth stage of the first sub-encoder is upsampled 2^3 times and added to the output of the third stage of the first sub-encoder, as described mathematically in Equations 1 and 2. In a feature map $F^i_{l_j}$, i represents the sub-encoder number and j represents the level (l) position. Therefore, features $F^1_{l_6}$ and $F^1_{l_3}$ denote the output of the sixth (l_6) and third (l_3) stages of the first sub-encoder ($i = 1$), respectively. The feature map $F^1_{l_6}$ from the sixth stage of the first sub-encoder is used to produce the feature map $F^2 l_3$ which is subsequently used as a input for the second sub-encoder and refined through its fourth, fifth, and sixth stages. Lateral connections are used to reuse features from the last three stages of the first sub-encoder. Similar to this process, the third and fourth sub-encoders are designed, and their

operations are described mathematically in Equations 3, 4, 5, and 6, which define the operations performed before processing feature maps with the third and fourth sub-encoders. 'Upsample8,' 'Upsample4,' and 'Upsample2' refer to scaling up the feature map by 2^3, 2^2, and 2^1 times, respectively.

$$F_{l_4}^3 = Conv(Upsample4(F_{l_6}^2) \tag{3}$$

$$F_{l_4}^3 = F_{l_4}^2 + F_{l_4}^3 \tag{4}$$

$$F_{l_5}^4 = Conv(Upsample2(F_{l_6}^3) \tag{5}$$

$$F_{l_5}^4 = F_{l_5}^3 + F_{l_5}^4. \tag{6}$$

We show in Table 2 that the layered architecture of each sub-encoder is different. In the first sub-encoder, we have 11 MBConv blocks, whereas in the successive sub-encoders, we have 7, 5, and 3 MBConv blocks respectively. The reason for the reduction of MBConv blocks in subsequent encoders is that repetition of shallow and intermediate stages in the successive sub-encoders does not contribute in context assimilation as shallow and intermediate features contain more spatial details than the contextual information. Whilst repetition of intermediate stages in the successive sub-encoder is reduced, reusing deep features in the successive sub-encoder is strategically increased for better context engrossment. Furthermore, Table 2 also illustrates that by increasing the value of depth multiplier M_d, the repetition of deep MBConv6 block is increased in the succeeding sub-encoder. Hence, the depth of the model is successively increased without creating any additional stages. Compared to DFANet, our proposed multi-encoder design reuses deep semantic features more effectively and avoids the need of FC attention modules as it scales the feature maps at different levels through a specially designed multi-encoder network.

Figure 4(a) shows that the four sub-encoders produce rich semantic feature maps F_6, F_5, F_4, and F_3. Lateral connections between encoders at the same level are used to address the gradient vanishing problem. We downsample the feature map F_6 by a pooling operation to create an additional feature map F_7. At this stage, the feature map may lose the contextual details of tiny objects in the scene due to the smaller spatial dimensions; however, it retains the context of large objects. To make the model more efficient, we utilize a simple pooling operation to create the feature map F_7 as this operation does not add any additional parameters. The rich features F_7 to F_3 will then be utilized by our decoder network.

B. DECODER NETWORK

Like most semantic segmentation models, our decoder is deployed to produce an output of the same size as the input by employing a series of upsampling techniques. Rich semantic features at different scales from the output of the encoder network need to be fused together at different levels. Similar to feature reusing in the encoder network, fusing features from multiple paths in both directions enhances the ability of object localization in the scene. Figure 4(b) displays the complete architecture of our proposed decoder network. Next, we describe key innovative steps.

1) Local and Global Context Aggregation Module

To motivate our proposed design, we first make an important observation from the literature [4], [46], [47] that aggregating features at different scales enhances the entire feature hierarchy with accurate object localization in the scene. Hence, we propose a novel component, termed Local and Global Context Aggregation (LGCA) module, for this purpose. The blue dotted box in Figure 4(b) displays the complete architecture of LGCA. First, it takes deep and intermediate features (F_3 to F_7) produced by the multi-encoder network described previously. We adopt a channel reduction mechanism in which all feature maps at various stages will be filtered by a standard point-wise convolution layer to generate features with reduced channel P_{l_i} = Conv(F_{l_i}). It is required to provide similar depth of each feature map before being fused with each other. Moreover, by reducing the depth of the deep feature maps, the model complexity is also reduced. Later on, high-level semantic features are propagated downward through a top-down path to boost the semantic representation and improve multi-scale in-variance. Equation 7 shows that before fusing a higher-level feature P_{l_i} with a previous level feature map $P_{l_{i-1}}$, P_{l_i} is bi-linearly upsampled. This top-down path provides the first decoder path which helps in achieving context assimilation in the feature hierarchy. However, the spatial details of local feature maps need to be added with rich semantic details of the global feature maps for better object localization. This necessitates one bottom-up path to send the accurate localization signals from a lower level to a higher level. Equation 8 shows that a low level feature P_{l_i-1} is down-sampled before it gets added with the next higher-level feature map P_i. The downward arrows define top-down path and the upward arrows signify the bottom-up path in Figure 4(b).

$$P_{l_{i-1}} = P_{l_{i-1}} + Upsample2(P_{l_i}) \tag{7}$$

$$P_{l_i} = P_{l_i} + MaxPooling(P_{l_{i-1}}). \tag{8}$$

Every downsampling or upsampling operation typically causes a loss of spatial details. To minimize this loss, we deploy a separable convolution (SC) block after every pooling operation. This block contains a depth-wise separable convolution (DSConv) layer followed by a batch normalization layer (BN). DSConv first filters the feature map along its depth, then deploys a point-wise standard convolution for better refinement. By standardizing the output of the DSConv layer, the BN layer enhances the independent learning ability of every layer of the network. We set the dilation rate to 2 for the DSConv layer in order to achieve a better receptive field while refining the feature maps.

After the bottom-up path, we finally introduce another top-down path for final context engrossment. Similar to feature reuse in the multi-encoder, we aggregate semantic features through multiple channels for better context accumulation

and accurate object localization. Some skip connections among the paths of the decoder are introduced to address the degradation problem and help the loss function converge quickly. At the end of the second top-down path, we receive a semantically rich feature map, which is upsampled 2 times to fuse with the coarse local feature map F_2. This completes the pipeline of LGCA.

2) Classifier

This final module of our proposed decoder network assigns a class label to every pixel based on their contextual details. The literature has shown that adding few layers in the classifier module supplements model performance [24], [39]. Hence, we deploy two depth-wise separable convolution layers, one standard convolution layer, one upsample and one softmax layers. In each DSConv layer, we use a 3×3 filter with a dilation rate of 2 as this provides a better receptive field while refining the feature map. As the input feature map in the classifier module has 64 channels, the choice of DSConv layers helps in reducing the number of parameters and GFLOPs. One standard convolution layer is implemented for the finest segmentation and we set the number of channels the same as the number of classes of the target dataset. The spatial dimensions of the feature map in the classifier module is one forth of the original input. To provide equal height and width, a bi-linear upsampling layer is utilized which pools up the feature map by 2^2 times. We also employ one Dropout layer to address model over-fitting. Finally, the softmax activation function is used to assign a class label to every individual pixel.

IV. EXPERIMENTS

As this work targets resource-constrained mobile devices, so we mainly compare the proposed model's performance with the existing real-time semantic segmentation models having a less than 5 M model parameters.

A. DATASETS

To benchmark our proposed model against others, we extensively carried out our experiments on structured and unstructured public datasets. We strictly follow the evaluation protocols of these datasets for training, validation, and testing.

1) Structured datasets

Cityscapes [28] is the most widely-used dataset for semantic segmentation. It provides urban street scene images at 1024×2048 resolution in which objects are classified into 35 classes and grouped into 8 different categories. Following the protocols used in the literature for Cityscapes, we use 19 classes for pixel annotations. the dataset consists of around 5,000 fine annotated images out of which, 2,975 images are used for training, 500 samples are used for validation and remaining 1,525 images are used for testing. However, annotations for test set are not given by the dataset. It provides an online evaluation server for test set evaluation. We submitted the proposed model's test set results to Cityscapes server for test evaluation and the result is published in the server.

CamVid [25] is a small structured dataset which provides 267 images for training, 101 for validation and 233 for testing. Consistent with the evaluation protocols in the literature, we used only 11 fine-tune classes out of 32 classes of the dataset. We utilize transfer learning to improve the performance on CamVid by pre-training the model on Cityscapes dataset with suitable mapping between the classes of two datasets.

Similar to Cityscapes, the BDD100K [26] and KITTI [27] datasets use the same class labeling technique (19 classes) for training and testing. Due to this compatibility in class labeling, we can use Cityscapes pre-trained weights to train the model with BDD100K and KITTI datasets. BDD100K provides a total of 10,000 images, out of which 7,000 images are used for training, 1,000 for validation, and the remaining 2,000 images for testing. It provides fine-grained annotations only for the training and validation sets. The original input resolution of this dataset is 720×1280 px. In comparison, KITTI is a small urban street scene dataset that provides only 200 training images with fine-tuned annotations and 200 test images without annotations. The resolution of each input image is 375×1280 px. Similar to Cityscapes, KITTI also provides an online evaluation server for the test set. We submitted the proposed model's test set results to the KITTI evaluation server to obtain the test results.

2) Unstructured dataset

The four datasets mentioned above mainly focus on urban street scenes captured in western countries, such as Europe or the USA, where the road environment is well-structured and has fewer variants of objects in the scene. However, such well-defined traffic environments cannot be found in Asian countries, such as India. To evaluate the proposed model's performance in unstructured road conditions, we trained the model with the IDD-lite (Indian Driving Dataset lite version) [48]. The dataset consists of 1,404 urban and rural training images, 204 validation samples, and 404 test samples, each having a resolution of 227×320. The entire object space is divided into seven classes: drivable, non-drivable, living things, vehicles, roadside objects, far objects, and sky, and we reported the proposed model's performance on each class.

Furthermore, we also trained the proposed model with IDD part 1 and part 2, which contain around 14,027 training samples and 2,036 validation samples. Similar to IDD-lite, we reported the proposed model's performance on the seven classes of both IDD and Cityscapes validation sets.

B. IMPLEMENTATION DETAILS

All compared models were trained on a server equipped with three Nvidia GeForce TITAN RTX GPUs, each with 24GB of memory. For effective utilization of all GPUs in data-parallel distributed training, we used the `horovod` framework [49]. The software components included CUDA 10.2 for parallel processing, `tensorflow` 2.1.0, and `keras` 2.3.1. We em-

Table 3. Results of ablation study

Encoder	ASPP	LGCA	Param. (M)	GFLOPs	Val. mIoU (%)
1	-	-	0.76	18.8	60.1
1,2	-	-	1.44	24.0	65.2
1,2,3	-	-	2.11	26.2	69.4
1,2,3,4	-	-	2.68	27.0	70.7
1,2,3,4	✓		2.69	27.0	69.9
1,2,3,4	-	✓	2.72	31.2	71.8

ployed the polynomial learning rate strategy, with a base rate of 0.045 and power of 0.9. Using a polynomial scheduler, we found the optimal learning rate at the steepest slope of the training loss vs. learning rate plot for 5 epochs. We used the distributed synchronous stochastic gradient descent (SGD) optimizer, which divides SGD mini-batches over a pool of parallel GPUs to find the best learning rate. Following [50], we also employed a gradual warm-up strategy in the `horovod` distributed framework to overcome optimization challenges, especially in the early stages of the training process.

To improve training, we employed various on-the-fly data augmentation techniques such as resizing, cropping, clipping by value, horizontal and vertical flipping, adjusting brightness, saturation and contrast of the input images to increase the effective size of the training set. We also employed different regularization techniques to address model over-fitting, such as ℓ_2 regularisation for all top layers and a dropout layer with a dropout rate of 0.3.

C. ABLATION STUDY

In this ablation study, we justify that every component of our proposed model is important for achieving the best possible segmentation performance. First, we demonstrate the significance of the multi-encoder design. To do so, we first trained the proposed model with first sub-encoder (Refer Table 2). Later on, we added second, third and forth sub-encoder progressively. The validation result reported in the Table 3 was obtained after training the model on Cityscapes training set for 500 epochs at full input resolution (1024×2048 px). In the result section, we reported the best validation and test mIoU produced by our proposed model after fine-tuning the model and trained the model for large number of epochs. Initially, we did not use multiple paths at the decoder side. We only used first top-down path of the LGCA module to fuse the features at different levels. Thereby, the first 5 rows in the Table 3 do not include LGCA. Table 3 clearly shows that with the addition of each sub-encoder, the performance of the model improves noticeably and reaches 70.7% validation mIoU after 500 epochs.

We further explored ASPP [13] for feature scaling on top of the forth encoder. ASPP is known to filter the feature map with various sizes of the receptive field and may lead to improved segmentation performance. However, we observe that the performance actually dropped slightly. This could be due to the fact that the model's backbone produces a low-resolution feature map which is 2^6 times smaller than the original input size. Due to this performance reduction, we did not consider ASPP in our model design. Moreover, the different layered architectures of each sub-encoder provides the feature scaling facility and therefore it can replace ASPP. The last row of Table 3 shows that with the utilization of LGCA on top of the multi-encoder network, our proposed model MCANet achieved 71.8% validation mIoU after 500 epochs while having only 2.72 M parameters and 31.2 GFLOPs. If we upsample the global feature map by 2^3 times at the decoder end, then GFLOPs count will be reduced to 27.5 at full input resolution. However this may cause boundary degeneration effects in the output. Hence, we upsample the global feature map at two stages: the first upsampling by 2^1 times occurs inside LGCA and the second upsampling by 2^2 times occurs inside the classifier module (see Figure 4(b)). From Table 3, one can notice the model's performance improvement with the successive addition of each sub-encoder and LGCA module. To graphically illustrate this improvement, we generated score maps (before the softmax function) at various stages using feature maps at different levels and presented them in Figure 5. In each score map, pixels with similar features are highlighted by a hotter color. Feature maps after the first, second, third, fourth, and after LGCA (refer to Figure 4) at 16×32 resolution are used to produce the score maps. In Figure 5, we highlighted two main sections (car and pedestrian) in the input image by red and white boxes. It clearly demonstrates that with the successive addition of each sub-encoder and LGCA module, the model can identify more pixels with similar features.

Hence, the quantitative and qualitative studies clearly demonstrate the effectiveness of multiple sub-encoder design for feature extraction at various levels and LGCA for constructing the segmented output using the extracted features at different levels.

D. MODEL EVALUATION

The proposed model is evaluated on the four urban street scenes datasets. Following the literature and evaluation servers, we present the following metrics: class and category-wise mean Intersection over Union (IoU), mean instance-level Intersection over Union (iIoU), model parameters, GFLOPs and Frame Per Second (EPS). As the proposed backbone is designed from scratch, so we did not use any existing pre-trained weight to train the model with Cityscapes. Moreover, We did not train the proposed backbone with ImageNet [29] dataset like other exiting models.

1) Performance on Cityscapes

We trained the proposed model with Cityscapes dataset for 1000 epochs with batch size 4 in each GPU. While measuring the performance on the validation set, we trained the model using the training set. However, for improving the test set accuracy, we considered merging both the training

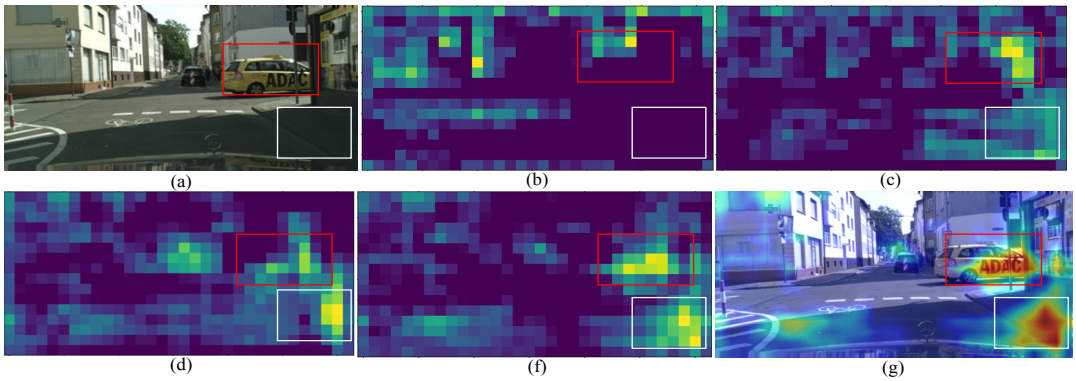

Figure 5. Illustration of feature similarities of a group of pixels using score map. Bright and hotter color means more similar feature among the pixels. Weights of intermediate Conv layers are used to produce score map at various lavels for an (a) input image. (b) Score map after first sub-encoder, (c) Score map after secod sub-encoder, (d) Score map after third sub-encoder, (e) Score map after forth sub-encoder, (f) Score map after LGCA, and (g) Overlay of score map with input image.

Table 4. Class-wise MCANet performance on Cityscapes validation and test sets

dataset	Road	S.walk	Build.	Wall	Fence	Pole	T.light	T.sign	Veg.	Terrain
Validation set	**98.7**	85.3	**93.8**	51.6	52.8	64.0	69.1	73.9	**94.1**	71.2
	Sky	Person	Rider	Car	Truck	Bus	Train	M.cycle	Bicycle	**mIoU**
	96.2	81.8	61.1	**95.6**	59.2	77.2	65.3	58.5	71.3	**74.8**
dataset	Road	S.walk	Build.	Wall	Fence	Pole	T.light	T.sign	Veg.	Terrain
Test set	**98.3**	84.2	**92.0**	49.5	51.6	62.1	67.8	73.2	**92.6**	70.3
	Sky	Person	Rider	Car	Truck	Bus	Train	M.cycle	Bicycle	**mIoU**
	95.3	81.4	59.2	**94.6**	57.8	75.9	64.2	56.3	69.1	**73.4**

Table 5. Category-wise MCANet performance on Cityscapes validation and test sets

dataset	Flat	Construction	Object	Nature	Sky	Human	Vehicle	mIoU
Validation set	**98.7**	**93.2**	69.7	**93.5**	**92.3**	82.1	**94.6**	**89.2**
Test set	**98.5**	**92.2**	68.4	**95.3**	**91.96**	82.3	**93.9**	**88.9**

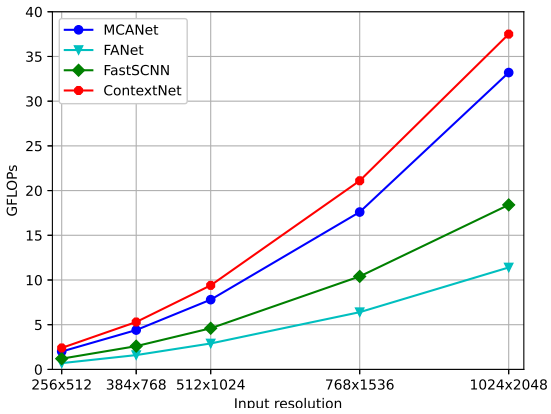

Figure 6. Input size vs GLFOPs

and validation sets. We also used additional coarse images of Cityscapes for small number of epochs. It enhances the model test performance just by 0.4%. Table 4 shows the model's class-wise performance on Cityscapes validation and test set. We observe that the proposed model performed particularly well over the top 5 classes (including road, building, vegetation, sky, car) as its accuracy was above 90% on both validation and test set. Overall, MCANet achieved 74.8% and 73.4% mIoU on Cityscapes validation and test set respectively. The performance on the test set was independently measured by Cityscapes evaluation server and the result is available on the evaluation server.

All 19 classes of Cityscapes are grouped into 7 categories and the corresponding category-wise mIoU is displayed in Table 5. It shows that MCANet performed extremely well in 5 categories out of all 7 categories. Due to the less occurrence of classes in object and human categories in the entire dataset, the model's performance is average in these two categories. This is common across all the existing models as the distribution of classes in the dataset is not uniform. Overall, the proposed model achieved almost 89% category-wise mIoU on both sets, which is outstanding.

Performance comparison To illustrate the effectiveness of our proposed model, we compared its performance with existing real-time semantic segmentation models. The general consensus in the literature is that offline models have a large number of parameters due to their deep network architecture, while real-time models have much fewer. Hence, to have a meaningful comparison, we did not include the performance

Table 6. Performance evaluation of different models on Cityscapes validation and test set

Type	Model	Parameter (M)	GFLOPs	Val. Class mIoU(%)	Val. Cat. mIoU (%)	Test Class mIoU(%)	Test Class iIoU(%)	Test Cat. mIoU (%)	Test Cat. iIoU (%)	FPS
Real time	SwiftNetRN-18†‡ [38]	11.8	114	72.2	-	75.9	-	-	-	134.9
	$STDC1$†‡ [51]	8.4	0.8	74.5	-	75.3	-	-	-	250.4
	DFANet†‡ [42]	7.8	3.4	71.9	-	71.3	-	-	-	160
	ICNet † [22]	6.7	28.3	-	-	69.5	-	-	-	30.5
	BiseNet†‡ BiseNet [21]	5.8	14.8	-	-	68.4	-	-	-	105.8
	MGSeg(S.Net)†‡ [37]	4.5	16.2	-	-	72.7	-	-	-	84
	Fast-SCNN* [39]	1.2	14.9	63.3*	82.2*	68.0	37.9	84.7	63.5	285.8
	FANet* [40]	1.1	11.4	65.9*	83.6*	64.1	33.2	83.1	61.1	78
	ContextNet* [24]	1.0	37.5	60.4*	81.5*	66.1	36.8	82.8	64.3	136.2
	ENet [19]	0.4	3.8	-	-	58.3	34.4	80.4	64.0	76.9
	QNet-attention [36]	0.2	19.8	-	-	49.2	-	70.1	-	18.2
Real time	**MCANet***	**2.7**	**31.2**	**74.8**	**89.2**	**73.4**	**45.8**	**88.9**	**72.8**	**269**

† sign means test result is not available at Cityscapes evaluation server. ‡ sign means pre-trained with ImageNet dataset.

Table 7. GFLOPs and FPS at different input resolution

	256×512		384×768		512×1024		768×1536		1024×2048	
	GFLOPs	FPS	GFLOPs	FPS	GFLOPs	FPS	GFLOPs	FPS	GFLOPs	FPS
MCANet	**2.0**	**432**	**4.4**	**392**	**7.8**	**269**	**17.6**	**129**	**31.2**	**75**
FANet	0.7	309	1.6	141	2.9	78	6.4	35	11.4	22
Fast-SCNN	1.2	494	2.6	231	4.6	124	10.4	58	18.4	34
ContextNet	2.4	397	5.3	182	9.4	101	21.1	44	37.5	27

of existing offline models in Table 6. In general, offline semantic models have more than 40 M parameters and produce $80 - 84\%$ mIoU on the Cityscapes test set. For instance, $DeepLabV3+$ [13] and PSPNet [14] generate 82.1% and 81.2% test mIoU, respectively, while having 43 and 250.8 M parameters. In comparison, most of the existing real-time semantic segmentation models have less than 10 M parameters and generate 68-72% test mIoU on Cityscapes. For example, STDC1 [51], MGSeg [37], DFANet [42], ICNet [22], and Bisenet [21] generate 75.3%, 72.7%, 71.3%, 69.5%, and 68.4% test mIoU, respectively. However, all these models still have moderately large numbers of parameters, ranging from $4.5 - 8.4$ M. Another model, called SwiftNet [38], which uses pre-trained ResNet-18 (RN18) as a backbone, produces 75.9% test mIoU on Cityscapes while having 11.8 M parameters. Although all these models produce good accuracy, they still have a moderately large number of parameters. On the other hand, models like ENet [19], ContextNet [24], Fast-SCNN [39], and FANet [40] have $0.4 - 1.2$ M parameters and produce $58 - 68\%$ test class mIoU. All these models are more efficient in a real-time environment, but their efficacy lags by $4 - 6\%$ compared to the moderately large real-time semantic models. Keeping a balance between model size and model performance, our proposed model, MCANet, generates 74.8% and 73.4% class mIoU on the Cityscapes validation and test sets, respectively, while having only 2.7 M parameters. This clearly shows the superiority of our model's performance on the Cityscapes dataset.

For consistency, we decided to replicate few existing real-time models based on publicly available implementations on GitHub for a more meaningful comparison. These models are marked by the sign * in the following tables. We trained these models under the same system configurations with full input resolution. The results obtained from our experiment on Cityscapes validation set are displayed in the Table 6 and marked by sign *. Our experimental results of the existing models may differ from the actual literature, but it is a fair comparison based on the same system settings.

Table 6 presents the validation and test set class and category-based mIoU, as well as the mean iIoU provided by the Cityscapes evaluation server, for different existing models, along with their model parameters and GFLOPs count. However, some existing real-time semantic segmentation models did not publish their results on the Cityscapes evaluation server, so their iIoU results are not available. In Table 6, the sign '-' means that the information is not available in the literature or on the evaluation server. Our Cityscapes test set result is available on the benchmark server.

While class mIoU is our primary metric for comparison, as it comes from the Cityscapes evaluation server, we also discuss GFLOPs count for completeness. However, we note that this metric is not optimal because it depends on the model size and input resolution. With an increase in input resolution, GFLOPs count increases somewhat polynomially, as shown in Figure 6. This has led to inconsistent GFLOPs counts being presented in previous work. For instance, ENet has 3.8 GFLOPs at 360×360 input resolution and 0.4M

parameters, whereas STDC1 [51] claims 0.8 GFLOPs at 224×224 input resolution and 8.4 M parameters.

As we trained three existing models (FANet [40], Fast-SCNN [39], ContextNet [24]) under the same system configuration, thereby we measured GFLOPs of all these models at different input resolutions and presented the results in Figure 6 and Table 7. Our proposed model produces 31.2 and 2.0 GFLOPs at an input resolution of 1024×2048 and 256×512 respectively.

Another metric often mentioned in semantic segmentation evaluation is frames per second (FPS). However, it is obvious that this is very hardware and input resolution dependent. For completeness, we also discuss it here. To make a meaningful comparison, we measured FPS of all the four trained models listed in Table 7 under the same system configuration at different input resolutions and presented the experimental results. As FPS also depends on the type of the model, so first we convert the Tensorflow model to TensorRT optimized model and then measure the FPS using a single TESLA T4 GPU with 16GB memory. We used a batch size of 4 and iterate the loop for 10 times to get the average value. It clearly displays that among all models in Table 7, Fast-SCNN [39] produces a better FPS at all different input resolutions. We note that our own measurements may differ from the figures published in the respective original papers, possibly due to different hardware and the measurement methods. What more important here is the relative performance based on the most intuitive way to measure the overall computation of the entire pipeline. Compared to our proposed model, all the three models in Table 7 are $2-3$ times smaller. They all used a computationally cheaper way to upsample the global feature map at the decoder than ours, which comes at the cost of boundary degradation in the output and overall poorer segmentation performance. Our effective upsampling solution as described earlier introduces extra computational cost, causes the FPS to be lower. However, we believe that such a trade-off is worthy for a much improved segmentation quality. In Table 6, we reported our proposed model FPS at 512×1024 input resolution.

2) Performance on CamVid dataset

We also trained our proposed model along with a few existing semantic segmentation models with CamVid dataset and present the results in Table 8. To ensure tensor size compatibility, we used an input size of 640×896 px instead of the full input resolution of 720×960 px. Table 8 clearly illustrates that our proposed model MCANet achieved the state-of-the-art (SOTA) performance on the CamVid validation and test sets among the existing real-time semantic models. It achieved 81.4% and 80.2% validation and test mIoU, respectively, which is even higher than many existing offline models such as DeepLab [11] and PSPNet [14]. The dual deep model $DeepLabV3+$ with SDCNetAug [52] achieved the SOTA performance (81.7%) on the CamVid test set due to its large backbone, joint strategies, and large synthetic datasets. In comparison, real-time models such as

Table 8. Performance evaluation on CamVid validation and test sets

Model	Input size	Val. class mIoU(%)	Test class mIoU (%)	FPS
DeepLabV3Plus SDCNetAug [52]	720×960	-	81.7	-
PSPNet [14]	-	-	69.1	-
DeepLab [13]	720×960	-	61.6	5
STDC1 [51]	720×960	-	73.0	197.6
MGSeg (R18) [37]	720×960	-	72.7	127
ICNet [22]	720×960	-	67.1	28
FANet* [40]	640×896	73.8	66.9	71
Fast-SCNN* [39]	640×896	73.3	66.7	120
ContextNet* [24]	640×896	69.6	65.8	96
BiseNet [21]	720×960	-	65.6	-
DFANet [42]	720×960	-	64.7	120
ENet [19]	360×480	-	51.3	-
MCANet*	**640×896**	**81.4**	**80.2**	**31**

Table 9. Performance evaluation on validation set of BDD100K dataset

Model	Param. (M)	GFLOPs	Class mIoU (%)	FPS
HANet-R101	64.2	2137.8	64.8	-
HANet-MV2 [15]	14.8	142.7	58.9	-
FANet* [40]	1.1	5.4	50.0	40
Fast-SCNN* [39]	1.2	8.6	47.9	70
ContextNet* [24]	1.0	17.5	44.5	56
MCANet*	**2.7**	**20.7**	**58.8**	**28**

Table 10. Performance evaluation on test set of KITTI

Model	Class mIoU (%)	Class iIoU (%)	Cat. mIoU (%)	Cat. iIoU (%)
DeepLabV3Plus + SDCNetAug [52]	72.8	48.7	88.9	75.3
SGDepth (Seg.) [53]	53.0	24.4	78.7	55.9
SDNet [54]	51.1	17.7	79.6	50.5
PAG [55]	47.9	17.9	78.1	49.2
MCANet	**58.5**	**24.0**	**83.0**	**54.1**

STDC1 [51] and MGSeg [37] achieved 73.0% and 72.7% test mIoU on the CamVid set, respectively. Literature [37] did not report the performance of the smaller variant of MGSeg (ShuffleNetV2) on the CamVid dataset. Hence, we compared the proposed model's performance with the higher variant of MGSeg (ResNet-18) (refer to Table 8). In terms of size, both of these models (MGSeg (R18) and STDC1) are 3 to 5 times bigger than our proposed model. Despite being a smaller network, our proposed MCANet achieved more than 7% test accuracy on the CamVid dataset. Thus, Table 8 shows the superior performance of our proposed model among real-time semantic models.

3) Performance on BDD100K

Table 9 displays models performance on the BDD100K dataset. We trained the model with 768×1280 input resolution for better compatibility with tensor dimensions. To improve model performance on BDD100K dataset, we used Cityscapes pre-trained weight. Due to the diverse and com-

plex nature of this data set, not many existing models are trained with this dataset. The only work we could find is [15] which introduced two different variants of HANet- HANet with MobileNetV2 (MV2) as backbone and HANet with ResNet-101 (R101) as backbone, both of which are clearly off-line models. We present both the variants' performance along with the proposed model. HANet R101 variant produces the SOTA result (64.8%) on BDD100K validation set while having as many as 64.2M parameters and 2137.8 GFLOPs. The smaller variant of HANet (MV2) which has 14.8M parameters, generates 58.9% validation mIoU. In comparison, the proposed model generates 58.8% validation mIoU while having 5 to 24 times less parameters than the both variants of HANet. It clearly shows the superior performance of the proposed model on BDD100K dataset. We also trained few existing models (marked by * sign) with BDD100K dataset and presented the results in Table 9. It can be observed that among the real-time semantic models, the proposed model produces the SOTA result on BDD100K validation set.

4) Performance on KITTI

We follow the same training protocol like BDD100k [26] to train the proposed model with KITTI [27] dataset and the result on KITTI test set is exhibited in Table 10. KITTI dataset is mainly used for stereo, visual odometry and depth analysis. Hence, we did not find any existing semantic segmentation models in real-time category which are trained and tested by KITTI fine-tune dataset and submitted the result on evaluation server. We found few literature from the KITTI server such as $DeepLabV3Plus + SDCNetAug$ [52], SGDepth [53], SDNet [54], and PAG [55] which are mainly used for depth analysis. Although, along with the depth analysis decoder head, all these models deploy a semantic head to boost model's performance. Among these models, SGDepth [53] produces comparatively better results (53.0% class mIoU) on KITTI test set, followed by SDNet [54] (51.1%). The current SOTA result on KITTI test set is generated by a deep model, called $DeepLabV3Plus + SDCNetAug$ [52]. It is a combination of multiple models in which a joint video prediction model is deployed to scale up the training sets for robust semantic segmentation, a deep semantic model (DeepLabV3Plus) is exploited for semantic feature extraction and a boundary label relaxation technique is utilized to reduce the noise at the edges of each object in the scene. Due to this joint efforts by multiple models and the presence of large synthetic training sets, this model generates 72.8% class mIoU on KITTI test set. In comparison, the proposed light-weighted single model is much smaller and mainly designed for scene parsing. It generates 58.5% class and 83.0% category (Cat.) mIoU on KITTI test set which sets the SOTA performance in real-time semantic category. The result of MCANet is independently generated by KITTI evaluation server. For tensor dimension compatibility, we used 384×1280 input resolution for training the model on KITTI set. All the models listed in Table 10 are pre-trained with Cityscapes dataset.

Table 11. Model performance on IDD-lite validation set

Model	Val mIoU (%)	Param. (M) (%)	GFLOPs	FPS
$DeepLabV3+$ (ResNet50)	64.3	26.7	28.2	470
UNet (ResNet50)	68.6	157.3	50.8	434
Eff-UNet (E.Net B5)	70.7	34.0	5.3	323
Eff-UNet (E.Net B7) [56]	73.8	72.8	10.5	365
MCANet	**73.8**	**2.7**	0.7	494

5) Performance on IDD-lite

The IDD-lite dataset is primarily designed for resource-constrained devices that lack adequate hardware facilities to train models with large input resolutions. For better tensor size compatibility, we trained the model with a 256x384 input resolution. Table 11 displays the models' performance on the IDD-lite validation set. Among the existing models, Eff-UNet (E.Net B7) [56] produces the SOTA performance on the IDD-lite validation set and won first prize in the IDD-lite segmentation challenge held in 2019. Eff-UNet [56] employs a large feature extractor called EfficientNet-B7 (E.Net B7) [57] that has 66M parameters and 37 GFLOPs at 224×224 input resolution. Although the IDD-lite dataset targets resource-constrained embedded devices, the existing evaluated models on this dataset are too large and computationally inefficient for mobile devices. Table 11 shows the results of the existing top-performing models, along with each model's parameters and GFLOPs at 128×256 input resolution. It is evident that all these existing models contain a large number of parameters and GFLOPs, making them infeasible to run on resource-constrained embedded devices at higher input resolutions. On the contrary, the proposed MCANet is 10 to 58 times smaller than all the listed existing models, although it produces the SOTA result (73.8% mIoU) on the IDD-lite validation set, similar to Eff-UNet (E.Net B7) [56]. Table 11 also displays the model parameters, GFLOPs, and FPS count. It is clear that among all existing models, the proposed MCANet is more efficient, processing a higher number of frames (494) per second, and reducing computational usage by reducing the number of parameters and GFLOPs.

Table 12 exhibits the class-wise mIoU performance of Eff-Unet (E.Net B7) [56] and the proposed MCANet. We also reported the proposed model's performance on seven classes of the IDD (part 1 and part 2) [58] and Cityscapes [28] datasets in Table 12. It produces 75.5% and 71.6% mIoU on IDD and Cityscapes datasets.

6) Qualitative results and analysis

This section illustrates the quality of the output produced by the proposed model and compares it with other models. Figure 7 and 10 display the annotation and colour map used for Cityscapes and CamVid datasets respectively. Nineteen and eleven colour codes are used for Cityscapes and CamVid respectively. BDD100K and KITTI follow the same colour

Table 12. Class-wise model performance on IDD-lite validation set

Model	Dataset	Drivable	Non-drivable	Living things	Vehicles	Roadside objects	Far Objects	Sky	mIoU
Eff-Unet	IDD-lite	94.9	50.1	62.0	81.3	55.0	77.5	95.6	73.8
MCANet	IDD-lite	94.5	48.1	59.7	81.5	56.8	80.0	96.0	73.8
MCANet	IDD	95.4	50.3	63.7	84.2	58.5	82.0	96.2	75.6
MCANet	Cityscapes	91.4	57.5	54.6	80.7	40.0	88.8	89.6	71.8

codes like Cityscapes. For all datasets, we excluded the void class.

The corresponding segmented output of the input image shown in Figure 7 are displayed in Figure 8. The boundary degeneration effect can be clearly seen in the output produced by ContextNet, Fast-SCNN and FANet due to the 2^3 times upsampling at the end of decoder, whereas the output produced by our proposed model MCANet has sharp and clear edges of every object in the scene. Similar observation can be drawn from the Figure 9. All tiny classes such as poles, traffic lights, and traffic signs are accurately positioned in all test samples in Figure 9 without being overlooked by the larger classes.

Likewise, Figure 11 displays the output produced by different models on selected CamVid images. In contrast to the original annotation of CamVid, we formed some super classes by merging related classes. For instance, we grouped car, truck, bus, and caravan together and formed a single class called "car." Thus, the bus is represented by the same color as the car, as can be observed in Figure 11. Figure 12 displays the output generated by the proposed model using selected test samples from the CamVid dataset. In line with the quantitative results presented in Table 8, Figure 11 also confirms the model's superiority over other models.

Figure 13 shows the segmented output produced by different models using the BDD100K validation set. Classes such as ice and car hood, which are defined by the black color in the colored annotation (Figure 8(b)), are ignored during training of the model. As a result, pixels that belong to ignored classes are assigned the color of the neighboring classes. This does not affect the model's performance, as these pixels are completely disregarded when calculating mIoU. By inspecting all the output, it can be clearly seen that the quality of the output produced by the proposed model is much better than other three models in Figure 13. In order to provide a better view of different scenes that contain tiny objects, we also present the output produced by the proposed model using BDD100K test set in Figure 14. All of these figures clearly demonstrate the excellent performance of MCANet in the field of semantic segmentation.

Figure 15 shows the predictions of the proposed model on KITTI test set samples, as generated by the KITTI evaluation server. Along with the colored predictions, it also provides an error image for each sample. The second column of Figure 15 displays the proposed model's predictions, and the third column shows the corresponding error images. The color red in the error images indicates wrongly classified pixels. It is clear that pixels mostly at the boundaries of each object in the scene are incorrectly classified. However, the proposed model's object identification and overall segmentation demonstrate its excellent performance on the KITTI dataset.

Figure 16 displays the colour map of IDD-lite dataset and the output produced by the proposed model MCANet, using IDD-lite validation sample. Like the other datasets, the quality of the predicted output of the IDD-lite sample is good, and it justifies the quantitative result produced by the proposed model. In Figure 17, we also shows the model's predictions using Cityscapes and IDD samples. Like IDD-lite, seven classes are used.

V. CONCLUSION

To improve the performance of existing models in real-time semantic segmentation for resource-constrained applications and to reduce the performance gap between offline and real-time models, we introduced an efficient multi-encoder network that can handle high-resolution input images and produce competitive semantic segmentation results. The key innovative steps in our design are: a novel multi-encoder network with a dynamic layered structure for better capturing semantic information and sharing information more effectively across different scales; a new local and global context aggregation module for better semantic fusion in the output. Compared to existing real-time semantic segmentation models, our proposed model MCANet produces competitive performance in both structured and unstructured environments and sets a new benchmark on all the tested datasets while having only 2.7 M parameters. The effective design of our proposed multi-encoder fulfills the needs of feature scaling techniques and produces rich feature maps at different scales. By exploiting these feature maps, our proposed decoder assimilates contextual details in multiple paths and produces output with accurate object positioning in the scene. Although the addition of the LGCA module improves the localization of each object in the scene, it also slightly increases the processing time of each frame. Hence, in the future, we will try to optimize the design of the LGCA module to improve the model's FPS without sacrificing the model's performance. We will also exploit the design of the multi-encoder for instance and panoptic segmentation. We will make our implementation available at the official GitHub repository https://github.com/tanmaysingha/MCANet for reproducing the results presented in this work.

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3289968

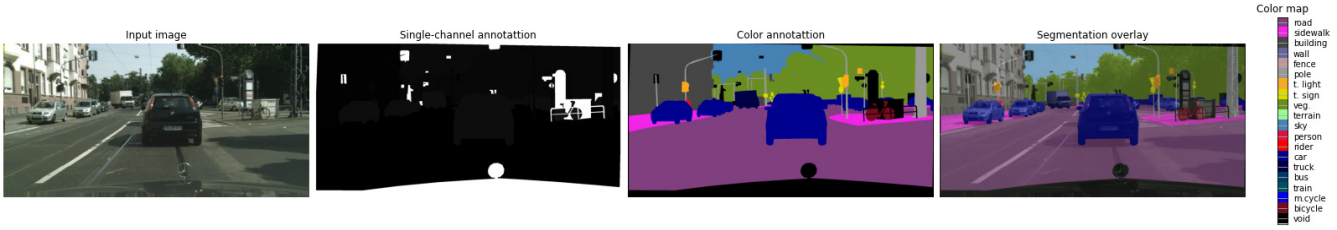

Figure 7. Colour mapping of Cityscapes dataset

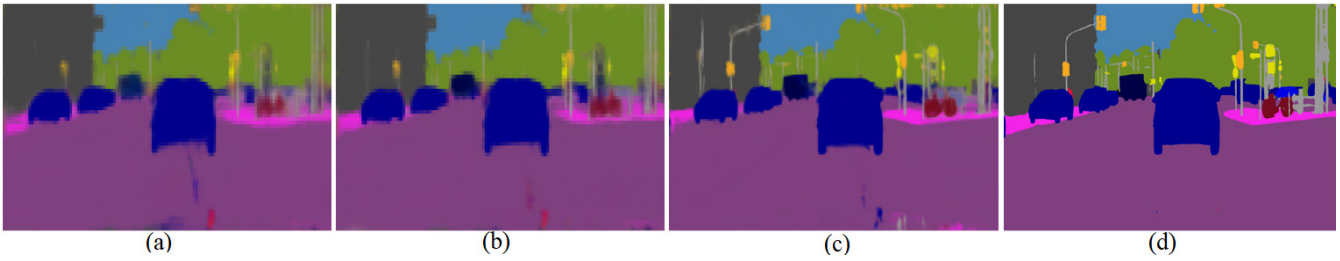

Figure 8. Output produced by (a) ContextNet, (b) FANet, (c) FAST-SCNN, (d) MCANet using Cityscapes validation image

Figure 9. Output produced by MCANet using Cityscapes test set samples

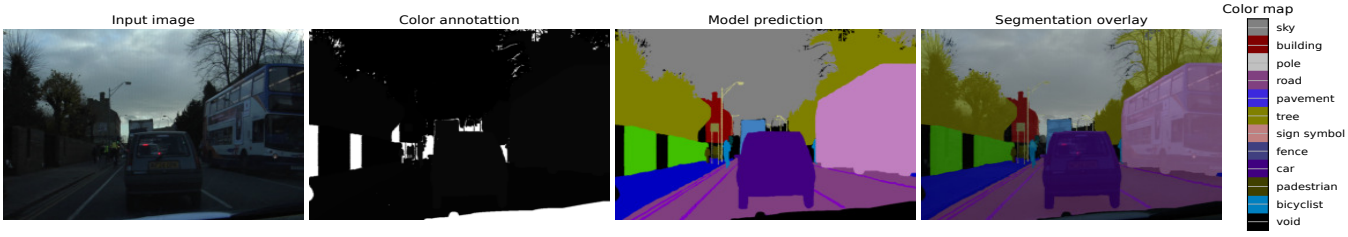

Figure 10. Colour mapping of CamVid dataset

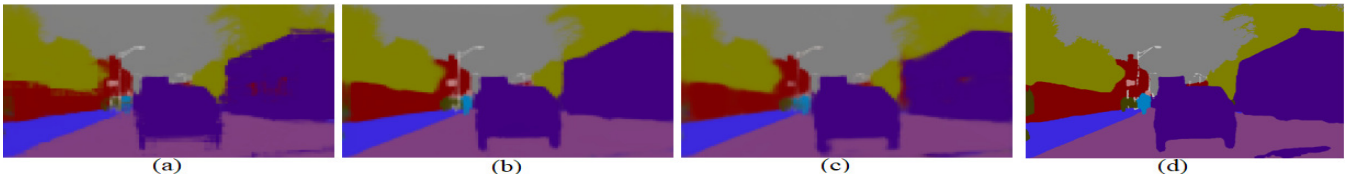

Figure 11. Output produced by (a) ContextNet, (b) FAST-SCNN, (c) FANet, (d) MCANet using CamVid validation image

Figure 12. Output produced by MCANet using CamVid test set samples

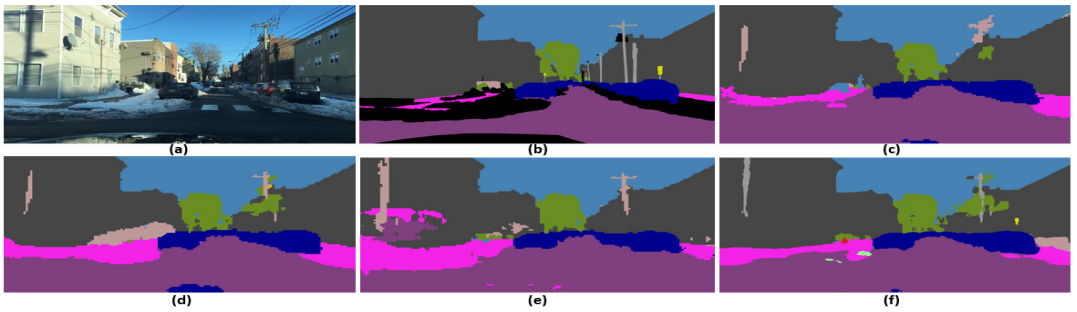

Figure 13. Models prediction on BDD100K validation set. (a) RGB input, (b) Coloured annotation, (c) ContextNet, (d) Fast-SCNN, (e) FANet, (f) MCANet

Figure 14. Output produced by MCANet using BDD100K test set samples

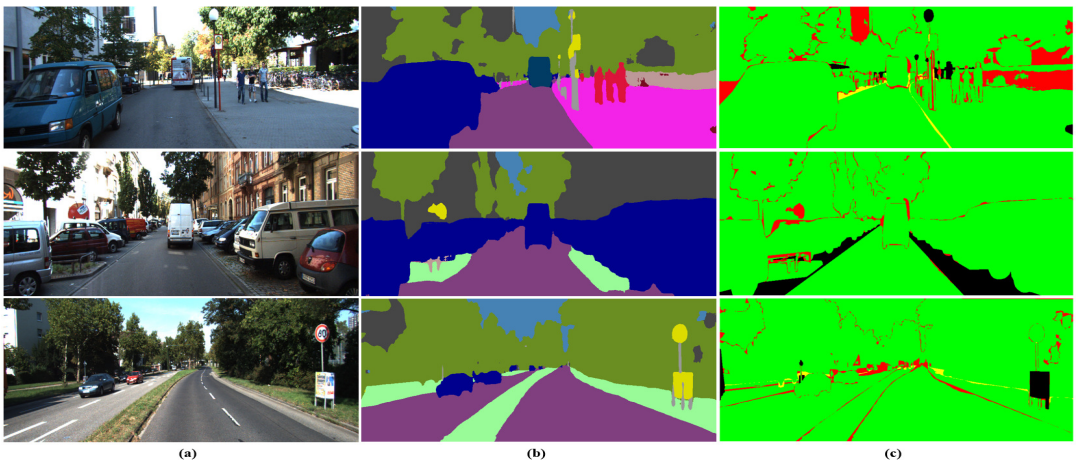

Figure 15. Output produced by MCANet using KITTI test set samples

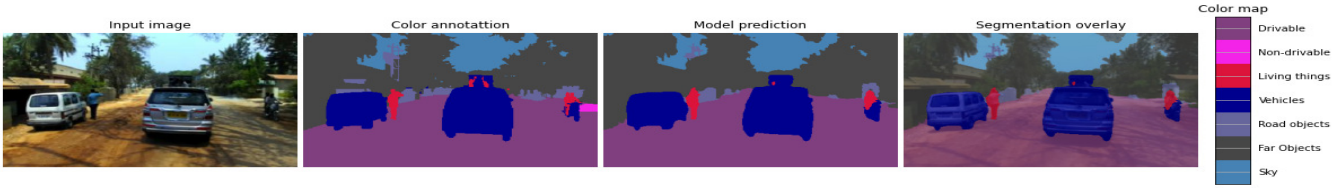

Figure 16. Color map of IDD lite dataset and model prediction using validation sample

Figure 17. Models predictions using Cityscapes and IDD samples on 7 classes. (a) Cityscapes input, (b) Cityscapes prediction, (c) IDD input, (d) IDD prediction

References

[1] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Glaeser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE TITS*, vol. 22, no. 3, pp. 1341–1360, 2020.

[2] X. Chang, H. Pan, W. Sun, and H. Gao, "Yoltrack: Multitask learning based real-time multiobject tracking and segmentation for autonomous vehicles," *IEEE TNNLS*, vol. 32, no. 12, pp. 5323–5333, 2021.

[3] G. Benitez-Garcia, L. Prudente-Tixteco, L. C. Castro-Madrid, R. Toscano-Medina, J. Olivares-Mercado, G. Sanchez-Perez, and L. J. G. Villalba, "Improving real-time hand gesture recognition with semantic segmentation," *Sensors*, vol. 21, no. 2, p. 356, 2021.

[4] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. CVPR*, 2017, pp. 2117–2125.

[5] M. K. Noman, S. M. S. Islam, J. Abu-Khalaf, and P. Lavery, "Seagrass detection from underwater digital images using faster r-cnn with nasnet," in *Proc. DICTA*. IEEE, 2021, pp. 1–6.

[6] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni, "U-net and its variants for medical image segmentation: A review of theory and applications," *IEEE Access*, vol. 9, pp. 82 031–82 057, 2021.

[7] M. Z. Khan, M. K. Gajendran, Y. Lee, and M. A. Khan, "Deep neural architectures for medical image semantic segmentation," *IEEE Access*, vol. 9, pp. 83 002–83 024, 2021.

[8] P. Yin, R. Yuan, Y. Cheng, and Q. Wu, "Deep guidance network for biomedical image segmentation," *IEEE access*, vol. 8, pp. 116 106–116 116, 2020.

[9] D. Zeng, X. Chen, M. Zhu, M. Goesele, and A. Kuijper, "Background subtraction with real-time semantic segmentation," *IEEE Access*, vol. 7, pp. 153 869–153 884, 2019.

[10] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. CVPR*, 2015, pp. 3431–3440.

[11] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE TPAMI*, vol. 40, no. 4, pp. 834–848, 2017.

[12] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *CoRR*, vol. abs/1603.08029, 2016. [Online]. Available: http://arxiv.org/abs/1603.08029

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

[14] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. CVPR*, 2017, pp. 2881–2890.

[15] S. Choi, J. T. Kim, and J. Choo, "Cars can't fly up in the sky: Improving urban-scene segmentation via height-driven attention networks," in *Proc. CVPR*, 2020, pp. 9373–9383.

[16] Y. Yuan, X. Chen, and J. Wang, "Object-contextual representations for semantic segmentation," in *Proc. ECCV*. Springer, 2020, pp. 173–190.

[17] Y. Deng, "Deep learning on mobile devices: a review," in *Mobile Multimedia/Image Processing, Security, and Applications 2019*, vol. 10993. International Society for Optics and Photonics, 2019, p. 109930A.

[18] A. Ignatov, R. Timofte, A. Kulik, S. Yang, K. Wang, F. Baum, M. Wu, L. Xu, and L. Van Gool, "Ai benchmark: All about deep learning on smartphones in 2019," in *Proc. ICCVW*. IEEE, 2019, pp. 3617–3635.

[19] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation," *CoRR*, vol. abs/1606.02147, 2016. [Online]. Available: http://arxiv.org/abs/1606.02147

[20] M. Treml, J. Arjona-Medina, T. Unterthiner, R. Durgesh, F. Friedmann, P. Schuberth, A. Mayr, M. Heusel, M. Hofmarcher, M. Widrich *et al.*, "Speeding up semantic segmentation for autonomous driving," in *MLITS, NIPS Workshop*, vol. 2, no. 7, 2016.

[21] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiseNet: Bilateral segmentation network for real-time semantic segmentation," in *Proc. ECCV*, 2018, pp. 325–341.

[22] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," in *Proc. ECCV*, 2018, pp. 405–420.

[23] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proc. CVPR*, 2017, pp. 1925–1934.

[24] R. P. K. Poudel, U. Bonde, S. Liwicki, and C. Zach, "Contextnet: Exploring context and detail for semantic segmentation in real-time," *CoRR*, vol. abs/1805.04554, 2018. [Online]. Available: http://arxiv.org/abs/1805.04554

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

[26] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100k: A diverse driving dataset for heterogeneous multitask learning," in *Proc. CVPR*, 2020, pp. 2636–2645.

[27] H. Abu Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 961–972, 2018.

[28] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. CVPR*, June 2016.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. CVPR*. Ieee, 2009, pp. 248–255.

[30] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. MICCAI*. Springer, 2015, pp. 234–241.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[32] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. CVPR*, 2017, pp. 1251–1258.

[33] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *Pattern Recognition*, vol. 90, pp. 119–133, 2019.

[34] S. Li, Q. Zhou, J. Liu, J. Wang, Y. Fan, X. Wu, and L. J. Latecki, "DCM: A Dense-Attention Context Module For Semantic Segmentation," in *Proc. IEEE ICIP*. IEEE, 2020, pp. 1431–1435.

[35] J. Fu, J. Liu, J. Jiang, Y. Li, Y. Bao, and H. Lu, "Scene segmentation with dual relation-aware attention network," *IEEE TNNLS*, vol. 32, no. 6, pp. 2547–2560, 2020.

[36] J. Cai, Y. Liu, and P. Qin, "Attention based quick network with optical flow estimation for semantic segmentation," *IEEE Access*, vol. 11, pp. 12 402–12 413, 2023.

[37] J.-Y. He, S.-H. Liang, X. Wu, B. Zhao, and L. Zhang, "MGSeg: Multiple granularity-based real-time semantic segmentation network," *IEEE TIP*, vol. 30, pp. 7200–7214, 2021.

[38] M. Oršić and S. Šegvić, "Efficient semantic segmentation with pyramidal fusion," *Pattern Recognition*, vol. 110, p. 107611, 2021.

[39] R. P. K. Poudel, S. Liwicki, and R. Cipolla, "Fast-SCNN: Fast Semantic Segmentation Network," *CoRR*, vol. abs/1902.04502, 2019. [Online]. Available: http://arxiv.org/abs/1902.04502

[40] T. Singha, D.-S. Pham, and A. Krishna, "FANet: Feature Aggregation Network for Semantic Segmentation," in *Proc. DICTA*. IEEE, 2020, pp. 1–8.

[41] T. Singha, D.-S. Pham, A. Krishna, and J. Dunstan, "Efficient segmentation pyramid network," in *Proc. ICONIP*. Springer, 2020, pp. 386–393.

[42] H. Li, P. Xiong, H. Fan, and J. Sun, "DFANet: Deep feature aggregation for real-time semantic segmentation," in *Proc. CVPR*, 2019, pp. 9522–9531.

[43] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. CVPR*, 2018, pp. 4510–4520.

[44] 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/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[45] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proc. ICCV*, 2019, pp. 1314–1324.

[46] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. CVPR*, 2018, pp. 8759–8768.

[47] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," *ArXiv*, vol. abs/1911.09070, 2019.

[48] A. Mishra, S. Kumar, T. Kalluri, G. Varma, A. Subramaian, M. Chandraker, and C. Jawahar, "Semantic segmentation datasets for resource constrained training," in *Proc. NCVPRIPG*. Springer, 2019, pp. 450–459.

[49] A. Sergeev and M. D. Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *CoRR*, vol. abs/1802.05799, 2018. [Online]. Available: http://arxiv.org/abs/1802.05799

[50] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: training imagenet in 1 hour," *CoRR*, vol. abs/1706.02677, 2017. [Online]. Available: http://arxiv.org/abs/1706.02677

[51] M. Fan, S. Lai, J. Huang, X. Wei, Z. Chai, J. Luo, and X. Wei, "Rethinking BiSeNet for real-time semantic segmentation," in *Proc. CVPR*, 2021, pp. 9716–9725.

[52] Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. Newsam, A. Tao, and B. Catanzaro, "Improving semantic segmentation via video propagation and label relaxation," in *Proc. CVPR*, 2019, pp. 8856–8865.

[53] M. Klingner, J.-A. Termöhlen, J. Mikolajczyk, and T. Fingscheidt, "Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance," in *Proc. ECCV*. Springer, 2020, pp. 582–600.

[54] M. Ochs, A. Kretz, and R. Mester, "SDNet: Semantically guided depth estimation network," in *Proc. German Conf. on Pattern Recognition*. Springer, 2019, pp. 288–302.

[55] S. Kong and C. Fowlkes, "Pixel-wise attentional gating for scene parsing," in *Proc. WACV*. IEEE, 2019, pp. 1024–1033.

[56] B. Baheti, S. Innani, S. Gajre, and S. Talbar, "Eff-UNet: A novel architecture for semantic segmentation in unstructured environment," in *Proc. CVPR Workshops*, 2020, pp. 358–359.

[57] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. ICML*. PMLR, 2019, pp. 6105–6114.

[58] G. Varma, A. Subramanian, A. Namboodiri, M. Chandraker, and C. Jawahar, "IDD: A dataset for exploring problems of autonomous navigation in unconstrained environments," in *Proc. WACV*. IEEE, 2019, pp. 1743–1751.

MR. TANMAY SINGHA is currently a PhD research scholar in Computer Science at Curtin University, Australia. He has double Master's Degrees- Master of Technology in Information Technology and Master of Computer Applications from University of Calcutta, India. Before joining Curtin University, he served nine years as a lecturer in the department of IT at Royal University of Bhutan, Bhutan. His current research interest focuses on computer vision tasks such as semantic and instance segmentation, object detection, scene graph generation, indoor and outdoor scene analysis, human pose estimation, facial landmark detection and medical image processing using deep neural networks.

DR. DUC-SON PHAM (M'02) received the PhD degree from Curtin University of Technology in 2005. He is currently a Senior Lecturer with the Discipline of Computing, Curtin University, Perth, Western Australia. His current research interests include sparse learning theory, large-scale data mining, convex optimization, and advanced deep learning with applications to computer vision and image processing. He is a Senior Member of the IEEE. He is a recipient of the Young Author Best Paper Award 2010 for a publication in IEEE Transactions on Signal Processing.

DR. ANEESH KRISHNA is currently an Associate Professor with the School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Australia. He holds a PhD in computer science from the University of Wollongong, Australia. He was a lecturer in software engineering at the School of Computer Science and Software Engineering, University of Wollongong, Australia (from February 2006 - June 2009). His research interests include AI for software engineering, model-driven development/evolution, requirements engineering, agent systems, formal methods, data mining, computer vision, machine learning, bio-informative and renewable energy systems. He has published more than 130 articles in reputed journals and international conferences. His research is (or has been) funded by the Australian Research Council (ARC), and various Australian government agencies (like NSW State Emergency Service) as well as companies such as Woodside Energy, Amristar Solutions, Autism West Incorporated, BW Solar Australia, Western Australia Dementia Training Center and Andrew Corporation. He serves as an assessor (Ozreader) for the ARC. He has been on the organising committee, served as invited technical program committee member of many conferences and workshops in the areas related to his research.