IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 11 March 2023, accepted 21 June 2023, date of publication 27 June 2023, date of current version 6 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3289968

RESEARCH ARTICLE

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

TANMAY SINGHA, DUC-SON PHAM, (Senior Member, IEEE), AND ANEESH KRISHNA
School of Electrical Engineering, Computing, and Mathematical Sciences, Curtin University, Perth, WA 6102, Australia
Corresponding author: Duc-Son Pham (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 Convolutional Neural Networks (DCNNs) in general classification tasks, many offline semantic segmentation models have been developed with deep architectures [10], [11], based on well-known backbone networks, usually ResNet [12], which is 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

The associate editor coordinating the review of this manuscript and approving it for publication was Eduardo Rosa-Molinar.

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.

in the feature scaling module, DeepLab ensures a wider field of view for context inclusion.

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 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, deploying large models that can achieve satisfactory real-time performance is prohibitive. This has led to a growing interest in developing lightweight semantic segmentation models [19], [20], [21] for 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 in developing real-time models for mobile devices and resource-constrained applications is the high input resolution with a large field of view, which can significantly increase memory usage. To address 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 design consists of a shallow branch for high-resolution input and a deep branch for low-resolution input. By using a shallow branch with fewer layers, this approach effectively reduces the computational cost while controlling the field of view 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 capture better contextual details for improved scene understanding.

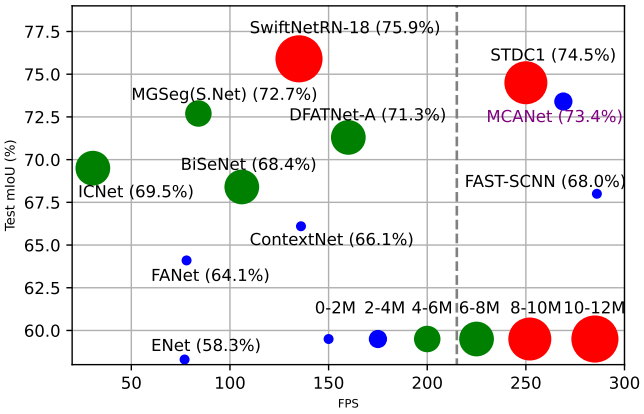

FIGURE 2. Test accuracy vs parameters among real-time models.

We introduce a completely new multi-encoder architecture in this study. Here, the number of stages in each successive sub-encoder is reduced, while the repetition of inverted residual blocks is increased in the successive sub-encoder. Consequently, each sub-encoder becomes deeper than the previous one, allowing for extraction of more semantic information from the scene. Lateral connections are also used at the same stage. After the complete encoding process, we obtain five rich global feature maps at different scales, which are then used for feature fusion at the decoder end. We demonstrate that **MCANet** enables 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 achieve state-of-the-art results, giving our model a competitive advantage in resource-constrained applications.

We make the following contributions in this research study:

- Introduce a novel backbone architecture, 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-encoder to control the number of model parameters.
- 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.
- Relying on this novel backbone architecture and an efficient multi-stage feature aggregation module, introduce an efficient semantic segmentation model named **MCANet** that achieves the optimal trade-off between model accuracy and model efficiency for resource-constrained mobile devices. This can be viewed in Figure 2.
- Finally, we provide comprehensive experiments on both structured and unstructured environments with various numbers of classes and demonstrate the model's superior performance in all circumstances among the existing real-time semantic segmentation models having fewer than 3 million (M) parameters.

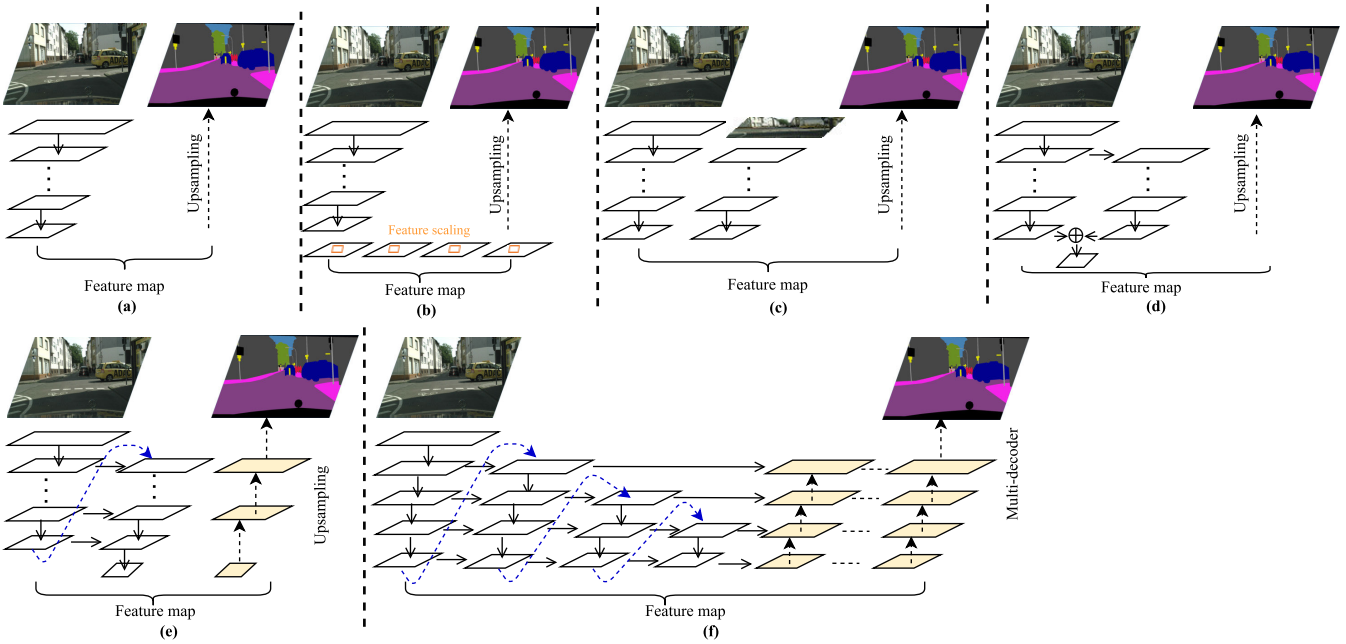

FIGURE 3. Different architectures of semantic segmentation models: (a) One-branch, (b) One-branch with feature scaling technique, (c) Multi-branch, (d) Dual-branch with down-sampling, (e) Feature re-use in sub-encoders with increasing stages, and (f) Feature re-use in multiple sub-encoders with decreasing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

On structured datasets such as CamVid [25], BDD100K [26], and KITTI [27], as well as on unstructured datasets such as IDD-lite, the proposed model produces the state-of-the-art (SOTA) performance among the existing real-time semantic segmentation models. On Cityscapes [28], the proposed model generates a test accuracy of 73.4% without using ImageNet [29] or any pre-trained weights, which is the best performance among the existing real-time semantic models with fewer than 3M 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. Figure 2 clearly demonstrates the proposed model's superiority in terms of achieving the best balance between model accuracy and model efficiency.

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), a simple encoder-decoder architecture [10] was 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 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 the 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, a few semantic segmentation models [21], [34], [35], [36] have started to use the attention mechanism to guide the feature learning process using high-level information.

Recently, a new model known as MGSeg [37] has emerged, aiming to improve model efficiency further. It achieves this by utilizing a lightweight backbone (ResNet-18) and incorporating a hybrid feature attention and feature scaling module. It achieves impressive performance, achieving 77.8% test accuracy on Cityscapes. However, it should be noted that the model has 13.3 million parameters and requires 96.5 GFLOPs (Giga Floating Point Operations) at an input resolution of 1024×1024. Although MGSeg has made strides in improving efficiency, the large number of parameters in the whole design can still pose computational challenges, particularly for high-resolution input images. As a result, these models may not be suitable for resource-constrained applications where computational resources are limited.

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. In this design, the encoder network takes a single input and employs a few convolution layers to down-sample the input image. Subsequently, two branches are formed: 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], and ESPNet [41] achieve improved accuracy and efficiency by employing this approach, which requires less data pre-processing.

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 the aforementioned designs, we propose a novel approach called the multi-encoder network, which leverages feature map reuse through dynamic sub-encoders and generates refined output through multiple decoding paths. Figure 3(f) provides an overview of the feature reuse in the multi-encoder design. The specifics of our proposed design will be discussed in the following 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.

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, -	$h/s \times w/s \times c'$

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 [43], [44], [45]. The optimized architecture of these residual 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 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. 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

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.

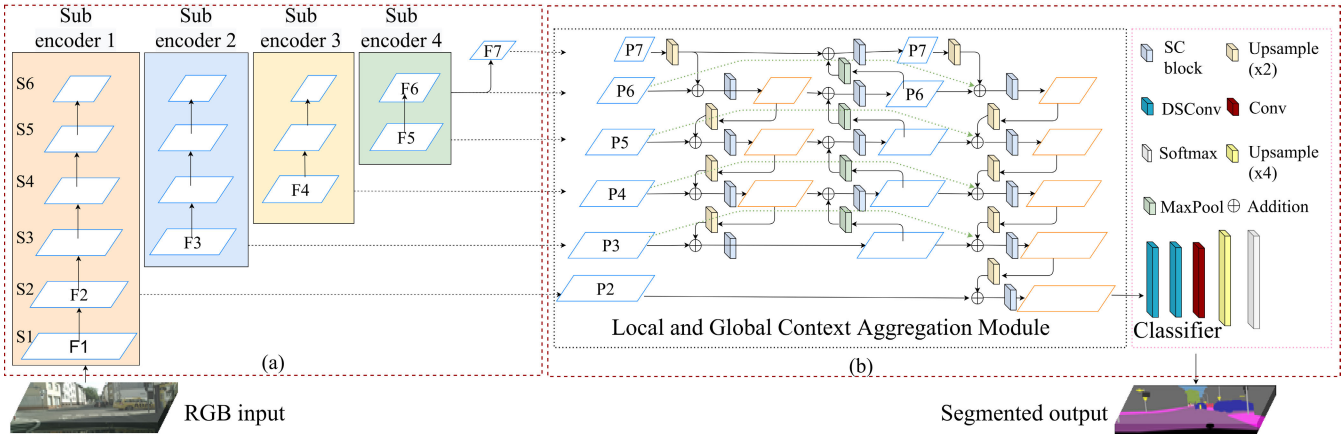

FIGURE 4. Complete pipeline of our proposed MCANet: (a) Multi-encoder design: Feature re-use in sub-encoders. Black dotted lines define the lateral connections from the previous sub-encoder at the same levels and green dotted lines show reusing of global feature map in the next sub-encoder. Feature F7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

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.

Empirical studies have shown that high-level features at lower resolutions contain rich semantic details that are conducive to attention weight learning [13], [37], [42]. Therefore, many models, such as DRANet [35] and DFANet [42], introduce an attention module in the later stages of their encoder networks. The attention mechanism utilizes 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 a fully connected (FC) attention module before being used as an input feature map for the next sub-encoder. It has been demonstrated that deploying the FC module enhances model performance 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, each sub-encoder has the same layered architecture, but the spatial dimensions of the input feature map for each sub-encoder differ. 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, in DFANet, features at the fourth, fifth,

and sixth stages are reused three, two, and one time, respectively. However, high-level features (the sixth stage) should be processed more extensively compared to intermediate and shallow features. Additionally, the uniform layered architecture of each sub-encoder is ineffective in acquiring any additional knowledge while reusing deep feature maps:

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

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

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 the feature maps $F^i_{l_j}$, where i represents the sub-encoder number and j represents the level (l) position, $F^1_{l_6}$ 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 an 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. Similarly, 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:

$$F^3_{l_4} = \text{Conv}(\text{Upsample4}(F^2_{l_6})), \tag{3}$$

$$F^3_{l_4} = F^2_{l_4} + F^3_{l_4}, \tag{4}$$

$$F^4_{l_5} = \text{Conv}(\text{Upsample2}(F^3_{l_6})), \tag{5}$$

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

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 reducing the number of MBConv blocks in subsequent encoders is that the repetition of shallow and intermediate stages does not contribute significantly to context assimilation, as these stages contain more spatial details than contextual information. While the repetition of intermediate stages is reduced, reusing deep features in the successive sub-encoder is strategically increased to enhance context assimilation.

Furthermore, Table 2 also illustrates that by increasing the value of the depth multiplier M_d, the repetition of deep MBConv6 blocks is increased in the subsequent sub-encoder. This allows us to increase the depth of the model without creating additional stages. Compared to DFANet, our proposed multi-encoder design reuses deep semantic features more effectively and eliminates the need for 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. Therefore, we introduce a novel component called the Local and Global Context Aggregation (LGCA), 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 $P_{l_i} = \text{Conv}(F_{l_i})$. 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):

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

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

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 a few layers in the classifier module supplements model performance [24], [39]. Hence, we employ two depth-wise separable convolution layers, one standard convolution layer, one upsample layer, and one softmax layer. 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. Since the input feature map in the classifier module has 64 channels, the choice of DSConv layers helps reduce the number of parameters and GFLOPs. We implement one standard convolution layer for the finest segmentation, setting the number of channels to be the same as the number of classes in the target dataset. The spatial dimensions of the feature map in the classifier module are one-fourth of the original input. To ensure equal height and width, we utilize a bilinear upsampling layer that upscales the feature map by 2^2 times. Additionally, we employ a Dropout layer to address model overfitting. Finally, the softmax activation function is used to assign a class label to each 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 conducted extensive experiments on structured and unstructured public datasets. We strictly followed 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 a resolution of 1024×2048, where objects are classified into 35 classes and grouped into 8 different categories. Following the protocols used in the literature for Cityscapes, we used 19 classes for pixel annotations. The dataset consists of around 5,000 finely annotated images, out of which 2,975 images are used for training, 500 samples are used for validation, and the remaining 1,525 images are used for testing. However, annotations for the test set are not provided with the dataset. To evaluate our model on the test set, we submitted the test results of the proposed model to the Cityscapes online evaluation server, and the results were published on the server.

CamVid [25] is a small structured dataset that 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-tuned classes out of the 32 classes in the dataset. To improve performance on CamVid, we utilized transfer learning by pre-training the model on the Cityscapes dataset with appropriate class mapping between the two datasets.

Similarly 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. Fine-grained annotations are provided only for the training and validation sets. The original input resolution of this dataset is 720×1280 pixels. On the other hand, KITTI is a small urban street scene dataset that provides only 200 training images with fine-tuned annotations and 200 test images without annotations. Each input image has a resolution of 375×1280 pixels. Similar to Cityscapes, KITTI also provides an online evaluation server for the test set. We submitted the test set results of our proposed model to the KITTI evaluation server to obtain the test results.

2) UNSTRUCTURED DATASET

The four datasets mentioned above primarily focus on urban street scenes captured in western countries, such as Europe or the USA, where the road environment is well-structured and there are fewer variations in the objects present. However, such well-defined traffic environments are not representative of the road conditions in Asian countries, such as India. To evaluate the performance of the proposed model in unstructured road conditions, we trained the model using the IDD-lite (Indian Driving Dataset lite version) [48]. This dataset consists of 1,404 urban and rural training images, 204 validation samples, and 404 test samples, each with a resolution of 227 × 320. The dataset divides the entire object space into seven classes: drivable, non-drivable, living things, vehicles, roadside objects, far objects, and sky. We reported the performance of the proposed model on each of these classes.

Furthermore, we also trained the proposed model using IDD part 1 and part 2, which contain approximately 14,027 training samples and 2,036 validation samples. Similar to IDD-lite, we reported the performance of the proposed model 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 employed 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 aim to demonstrate the importance of each component of our proposed model in achieving the best segmentation performance. Firstly, we highlight the significance of the multi-encoder design. To do this, we initially trained the proposed model with only the first sub-encoder (as described in Table 2). Subsequently, we progressively added the second, third, and fourth sub-encoders. The validation results reported in Table 3 were obtained after training the model on the Cityscapes training set for 500 epochs at the full input resolution of 1024 × 2048 px. In the results section, we reported the best validation and test mIoU achieved by our proposed model after fine-tuning the model and training it for a large number of epochs. Initially, we did not utilize multiple paths at the decoder side and only employed the first top-down path of the LGCA module to fuse features at different levels. Therefore, the first five rows of Table 3 do not include LGCA. The table clearly demonstrates that with the addition of each sub-encoder, the model's performance noticeably improves and reaches a validation mIoU of 70.7% after 500 epochs.

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

The sign '-' means ASPP/LGCA is not used and '✓' means ASPP/LGCA is used in the pipeline.

We also explored the use of ASPP (Atrous Spatial Pyramid Pooling) [13] for feature scaling on top of the fourth encoder. ASPP is known to filter the feature map with various sizes of receptive fields and has the potential to improve segmentation performance. However, we observed a slight drop in performance. This could be attributed to the fact that the model's backbone produces a low-resolution feature map that is 2^6 times smaller than the original input size. Due to this performance reduction, we did not incorporate ASPP in our model design. Furthermore, the different layered architectures of each sub-encoder already provide feature scaling capability, making ASPP unnecessary. The last row of Table 3 demonstrates that with the inclusion of LGCA on top of the multi-encoder network, our proposed model, called **MCANet**, achieves a validation mIoU of 71.8% after 500 epochs, while having only 2.72 million parameters and 31.2 GFLOPs. If we upsample the global feature map by 2^3 times at the decoder end, the GFLOPs count would be reduced to 27.5 at full input resolution. However, this may lead to boundary degeneration effects in the output. Therefore, we perform upsampling of 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 (as shown in Figure 4(b)).

From Table 3, it is evident that the model's performance improves with the addition of each sub-encoder and LGCA

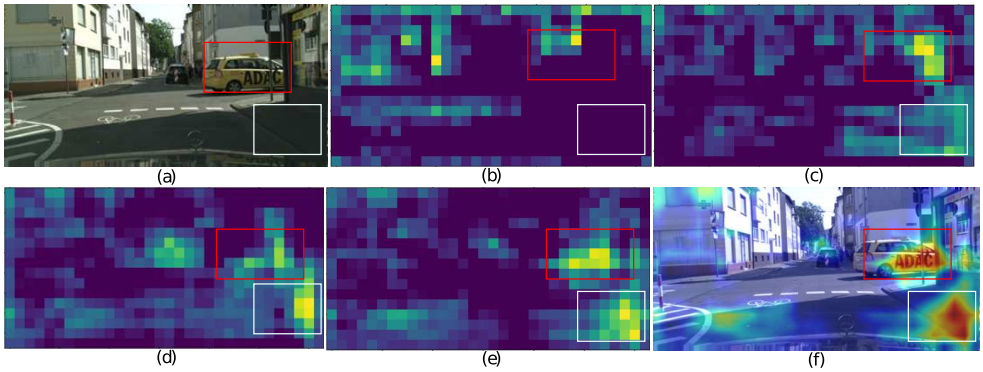

FIGURE 5. Illustration of feature similarities of a group of pixels using score map. Bright and hotter color means more similar feature among the pixels. Weights of intermediate Conv layers are used to produce score map at various levels for an (a) input image. (b) Score map after first sub-encoder,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

module. To visually demonstrate this improvement, we generated score maps (before the softmax function) at various stages using feature maps from different levels, and these score maps are presented in Figure 5. In each score map, pixels with similar features are highlighted using a hotter color. We used feature maps at 16×32 resolution after the first, second, third, fourth sub-encoders, and after LGCA (as depicted in Figure 4). In Figure 5, we highlighted two main sections (car and pedestrian) in the input image using red and white boxes. It clearly illustrates that with the successive addition of each sub-encoder and LGCA module, the model becomes capable of identifying more pixels with similar features.

Therefore, both the quantitative and qualitative studies provide clear evidence of the effectiveness of the multiple sub-encoder design for feature extraction at various levels and the use of LGCA for constructing the segmented output using the extracted features from 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 on the Cityscapes dataset for 1000 epochs with a batch size of 4 on each GPU. During the performance evaluation on the validation set, we utilized the training set. However, to improve the accuracy on the test set, we merged both the training and validation sets. Additionally, we included additional coarse images from Cityscapes for a small number of epochs, which resulted in a slight improvement of only 0.4% in test performance. Table 4 presents the class-wise performance of the model on the Cityscapes validation and test sets. It is observed that the proposed model performed exceptionally well on the top 5 classes (including road, building, vegetation, sky, and car), with accuracy above 90% on both the validation and test sets. Overall, MCANet

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†‡ [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

The '†' sign indicates that the test result is not available at the Cityscapes evaluation server. The '‡' sign indicates that the model is pre-trained using the ImageNet dataset. The '*' sign indicates that the existing models mentioned in the table were trained by the study, and their corresponding results are reported in the table.

TABLE 7. GFLOPs and FPS at different input resolution.

Input size	256 × 512		384 × 768		512 × 1024		768 × 1536		1024 × 2048	
Model	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

achieved a validation mIoU of 74.8% and a test mIoU of 73.4% on the Cityscapes dataset. The performance on the test set was independently evaluated by the Cityscapes evaluation server, and the results are available on the server.

The 19 classes of Cityscapes are categorized into 7 categories, and the corresponding category-wise mIoU is presented in Table 5. It demonstrates that MCANet performed exceptionally well in 5 out of the 7 categories. The average performance in the object and human categories is relatively lower due to the limited occurrence of classes within these categories in the entire dataset. This non-uniform distribution of classes is a common challenge across existing models. Overall, the proposed model achieved an impressive category-wise mIoU of almost 89% on both sets.

Performance comparison To illustrate the effectiveness of our proposed model, we compared its performance with existing real-time semantic segmentation models. It is generally acknowledged in the literature that offline models have a large number of parameters due to their deep network architecture, while real-time models have significantly fewer parameters. Therefore, in Table 6, we did not include the performance of existing offline models to ensure a meaningful comparison. Typically, offline semantic models have over 40 million parameters and achieve mIoU values of 80-84% on the Cityscapes test set. For example, DeepLabV3+ [13] and PSPNet [14] achieve test mIoU values of 82.1% and 81.2%, respectively, with 43 and 250.8 million parameters. In contrast, most existing real-time semantic segmentation models have less than 10 million parameters and achieve test mIoU values of 68-72% on Cityscapes. For instance, STDC1 [51], MGSeg [37], DFANet [42], ICNet [22], and BiSeNet [21] achieve test mIoU values of 75.3%, 72.7%, 71.3%, 69.5%, and 68.4%, respectively. However, these models still have moderately large parameter counts ranging from 4.5-8.4 million. SwiftNet [38], which uses a pre-trained ResNet-18 (RN18) as a backbone, achieves a test mIoU of 75.9% on Cityscapes with 11.8 million parameters. While all these models demonstrate 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 million parameters and achieve test class mIoU values of 58-68%. These models are more efficient in real-time environments but their performance lags behind moderately large real-time semantic models by 4-6%. Striking a balance between model size and performance, our proposed model, MCANet, achieves 74.8% and 73.4% class mIoU on the Cityscapes validation and test sets, respectively, with only 2.7 million parameters. This clearly demonstrates the superior performance of our model on the Cityscapes dataset.

For consistency, we decided to replicate a few existing real-time models based on publicly available implementations on GitHub to ensure a more meaningful comparison. These models are marked with an asterisk '*' sign in the following tables. We trained these models under the same system configurations with the full input resolution. The results obtained from our experiments on the Cityscapes validation set are presented in Table 6 and are marked with an asterisk

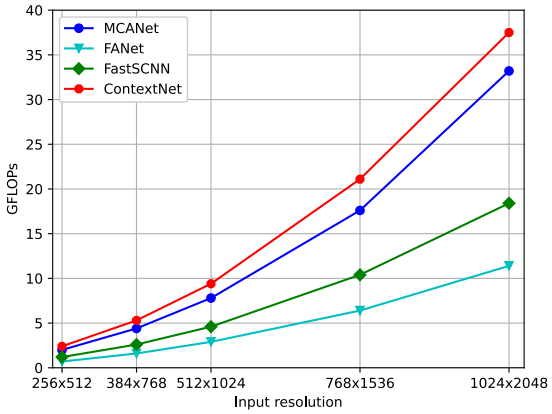

FIGURE 6. The plot Input size vs GLFOPs illustrates that with an increase in input resolution, the GFLOPs count of all models also increases.

'*' sign. Our experimental results for the existing models may differ from the actual literature, but it provides a fair comparison based on the same system settings.

Table 6 provides the class and category-based mIoU results for different existing models on the validation and test sets of Cityscapes. The mean iIoU, as provided by the Cityscapes evaluation server, is also included, along with the model parameters and GFLOPs count. However, some real-time semantic segmentation models did not publish their results on the Cityscapes evaluation server, resulting in unavailable iIoU values denoted by the sign "-". Our result on the Cityscapes test set is available on the benchmark server.

While class mIoU is the primary metric for comparison as it comes from the Cityscapes evaluation server, we also discuss GFLOPs count for completeness. However, it is worth noting that GFLOPs count is not an optimal metric as it depends on the model size and input resolution. The increase in input resolution leads to a somewhat polynomial increase in GFLOPs count, as shown in Figure 6. This has resulted in inconsistent GFLOPs counts being reported in previous work. For example, ENet has 3.8 GFLOPs at a 360 × 360 input resolution with 0.4 million parameters, while STDC1 [51] claims 0.8 GFLOPs at a 224 × 224 input resolution with 8.4 million parameters.

Since we trained three existing models (FANet [40], Fast-SCNN [39], ContextNet [24]) using the Cityscapes dataset under the same system configuration, we measured the GFLOPs of these models at different input resolutions. The results are presented in Figure 6 and Table 7. It can be observed that as the input resolution increases, the GFLOPs count of all models also increases, with ContextNet [24] having the highest GFLOPs count. Our proposed model produces 31.2 and 2.0 GFLOPs at input resolutions of 1024 × 2048 and 256 × 512, respectively.

Another commonly mentioned metric in semantic segmentation evaluation is frames per second (FPS). However, it is evident that FPS is highly dependent on hardware and input resolution. For the sake of completeness, we also discuss this metric here. To ensure a meaningful comparison, we measured the FPS of all four trained models listed in Table 7 under the same system configuration at different input resolutions and presented the experimental results. To measure FPS, we first converted the TensorFlow model to a TensorRT optimized model and then used a single Tesla T4 GPU with 16GB memory. We used a batch size of 4 and averaged the FPS value over 10 iterations. It is evident from the results that Fast-SCNN [39] achieves higher FPS compared to the other models at different input resolutions. We acknowledge that our own measurements may differ from the figures published in the original papers, possibly due to variations in hardware and measurement methods. However, what is more important is the relative performance based on the most intuitive way to measure the overall computation of the entire pipeline.

When compared to our proposed model, all three models listed in Table 7 have approximately 2-3 times fewer parameters. They employ a computationally cheaper method for upsampling the global feature map in the decoder, which results in boundary degradation in the output and overall lower segmentation performance. Our effective upsampling solution, as described earlier, introduces additional computational cost, leading to a lower FPS. However, we believe that such a trade-off is worthwhile for significantly improved segmentation quality. In Table 6, we reported the FPS of our proposed model at an input resolution of 512 × 1024.

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 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.

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

Existing Models marked with the '*' sign are trained by the study.

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

Existing Models marked with the '*' sign are trained by the study.

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

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 complex 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.

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

4) PERFORMANCE ON KITTI

We followed the same training protocol as BDD100k [26] to train our proposed model with the KITTI [27] dataset, and the results on the KITTI test set are presented in Table 10. The KITTI dataset is primarily used for stereo, visual odometry, and depth analysis. Therefore, we did not find any existing real-time semantic segmentation models that were specifically trained and tested on the KITTI fine-tune dataset and had their results submitted to the evaluation server. We came across a few works from the KITTI server, such as DeepLabV3Plus+SDCNetAug [52], SGDepth [53], SDNet [54], and PAG [55], which primarily focused on depth analysis. Although these models incorporate a semantic head to enhance the model's performance in addition to the depth analysis decoder head. Among these models, SGDepth [53] achieves relatively better results with a class mIoU of 53.0% on the KITTI test set, followed by SDNet [54] with 51.1%. The current state-of-the-art result on the KITTI test set is achieved by DeepLabV3Plus+SDCNetAug [52], which is a combination of multiple models. It utilizes a joint video prediction model to augment the training sets for robust semantic segmentation, leverages a deep semantic model (DeepLabV3Plus) for feature extraction, and applies a boundary label relaxation technique to reduce noise at the object edges. Due to the collective efforts of multiple models and the presence of large synthetic training sets, this model achieves a class mIoU of 72.8% on the KITTI test set. In comparison, our proposed lightweight single model is significantly smaller and specifically designed for scene parsing. It achieves a class mIoU of 58.5% and a category (Cat.) mIoU of 83.0% on the KITTI test set, setting the state-of-the-art performance in the real-time semantic segmentation category. The results of MCANet were independently generated by the KITTI evaluation server. For tensor dimension compatibility, we used an input resolution of 384 × 1280 to train the model on the KITTI dataset. All the models listed in Table 10 were pre-trained with the Cityscapes dataset.

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

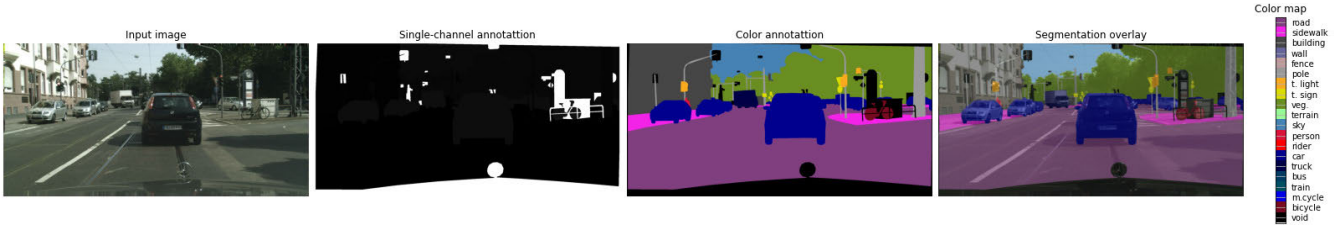

FIGURE 7. Colour mapping of Cityscapes dataset.

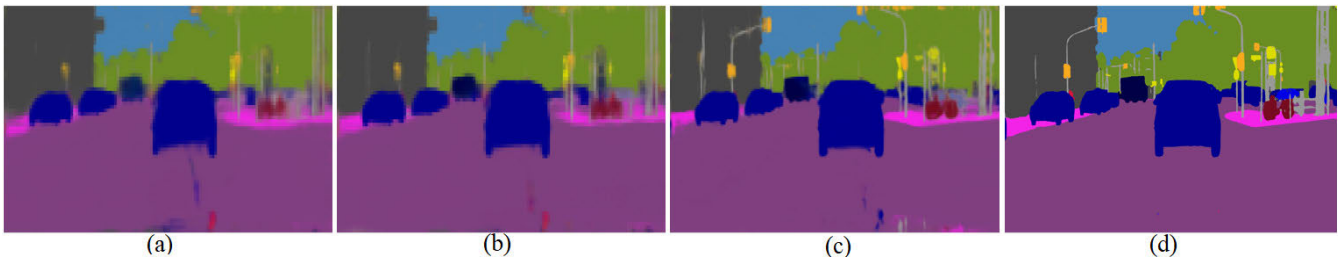

FIGURE 8. Output produced by (a) ContextNet, (b) FANet, (c) FAST-SCNN, (d) MCANet using Cityscapes validation image.

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

5) PERFORMANCE ON IDD-LITE

he IDD-lite dataset is primarily designed for resource-constrained devices that lack sufficient hardware resources to train models with large input resolutions. To ensure tensor size compatibility, we trained our model with a 256 × 384 input resolution. The performance of different existing models on the IDD-lite validation set is presented in Table 11. Among the existing models, Eff-UNet (E.Net B7) [56] achieves the state-of-the-art performance on the IDD-lite validation set and won the first prize in the IDD-lite segmentation challenge held in 2019. Eff-UNet [56] utilizes a large feature extractor called EfficientNet-B7 (E.Net B7) [57], which has 66M parameters and 37 GFLOPs at a 224 × 224 input resolution. However, despite the IDD-lite dataset targeting resource-constrained embedded devices, the evaluated existing models on this dataset are still too large and computationally inefficient for mobile devices.

Table 11 presents the results of the top-performing existing models on the IDD-lite dataset, including their parameters and GFLOPs at a 128 × 256 input resolution. It is evident that all these existing models have a large number of parameters and GFLOPs, making them impractical to run on resource-constrained embedded devices, especially at higher input resolutions. In contrast, our proposed MCANet is 10 to 58 times smaller than all the listed existing models while achieving a state-of-the-art result (73.8% mIoU)

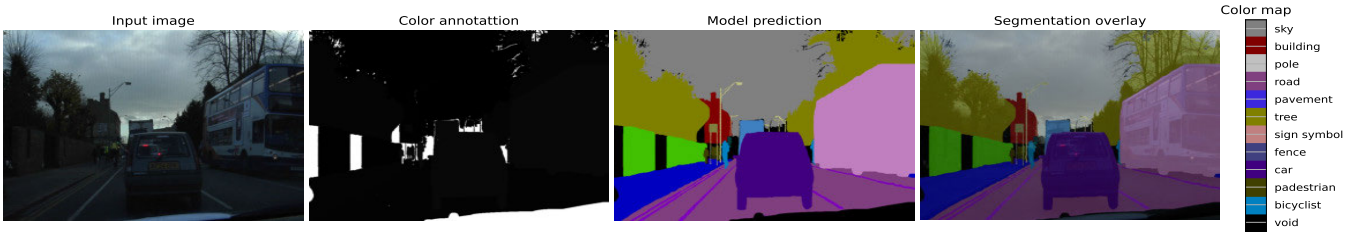

FIGURE 10. Colour mapping of CamVid dataset.

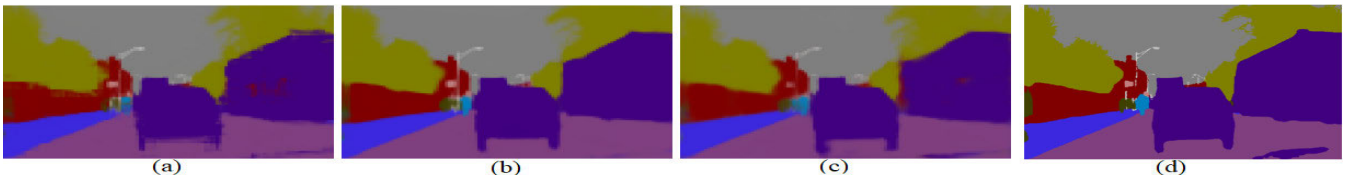

FIGURE 11. Output produced by (a) ContextNet, (b) FAST-SCNN, (c) FANet, (d) MCANet using CamVid validation image.

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

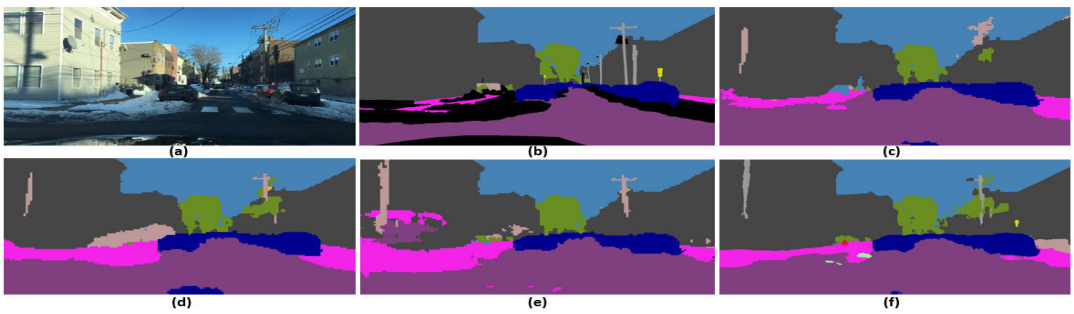

FIGURE 13. Models prediction on BDD100K validation set. (a) RGB input, (b) Coloured annotation, (c) ContextNet, (d) Fast-SCNN, (e) FANet, (f) MCANet.

on the IDD-lite validation set, similar to Eff-UNet (E.Net B7) [56]. Table 11 also provides information on the model parameters, GFLOPs, and FPS count. It is clear that our proposed MCANet is more efficient compared to all the existing models, as it processes a higher number of frames (494) per second while significantly reducing computational usage by reducing the number of parameters and GFLOPs.

Table 12 presents the class-wise mIoU performance of Eff-Unet (E.Net B7) [56] and our proposed MCANet. Additionally, we report the performance of our proposed model on seven classes of the IDD (part 1 and part 2) [58] and Cityscapes [28] datasets. Our model achieves 75.5% and 71.6% mIoU on the IDD and Cityscapes datasets, respectively.

6) QUALITATIVE RESULTS AND ANALYSIS

In this section, we demonstrate the quality of the output produced by our proposed model and compare it with other models. Figure 7 and 10 display the annotation and color map used for the Cityscapes and CamVid datasets, respectively.

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

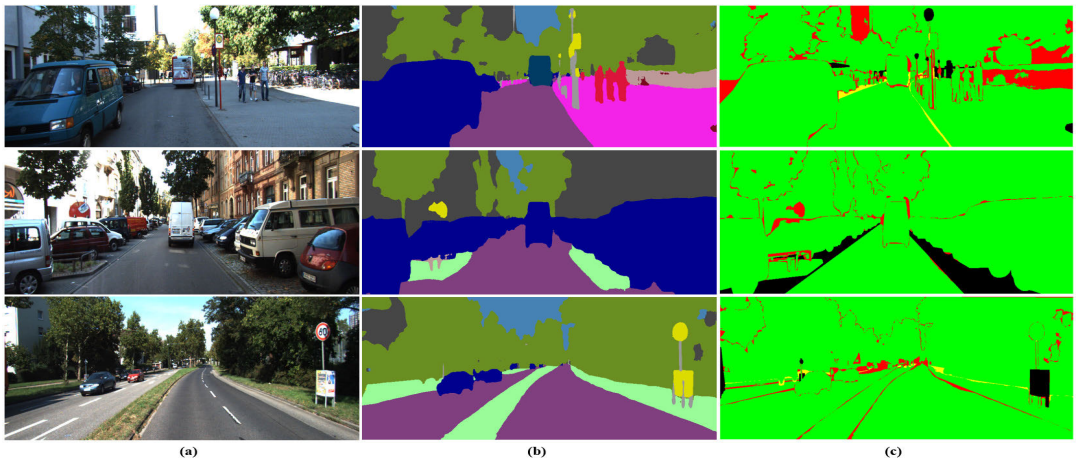

FIGURE 15. Output produced by MCANet using KITTI test set samples.

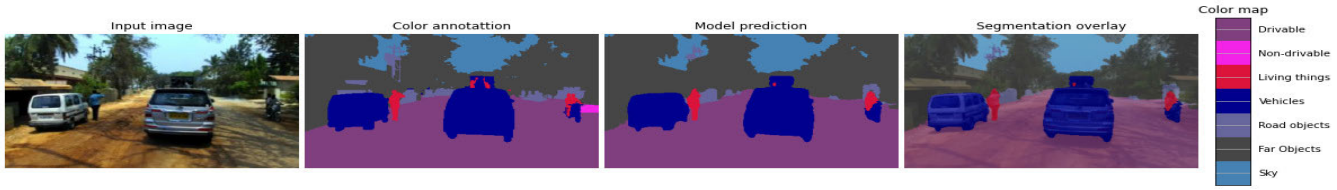

FIGURE 16. Color map of IDD lite dataset and model prediction using validation sample.

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

Nineteen color codes are used for Cityscapes, while eleven color codes are used for CamVid. The BDD100K and KITTI datasets follow the same color codes as Cityscapes. The void class is excluded for all datasets.

Figure 8 showcases the corresponding segmented output of the input images depicted in Figure 7. The outputs generated by ContextNet, Fast-SCNN, and FANet exhibit a boundary degradation effect due to the 2^3 times upsampling at the end of the decoder. On the other hand, our proposed model MCANet produces outputs with sharp and clear edges for each object in the scene. The results in Figure 9 demonstrate that our model accurately positions tiny classes such as poles,

traffic lights, and traffic signs in the test samples, without overlooking them amidst 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 make an implementation of our model available at https://github.com/tanmaysingha/MCANet for reproducing the results presented in this work.

REFERENCES

[1] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Gläser, F. Timm, W. Wiesbeck, and K. Dietmayer, ''Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges,'' *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1341–1360, Mar. 2021.

[2] X. Chang, H. Pan, W. Sun, and H. Gao, ''YolTrack: Multitask learning based real-time multiobject tracking and segmentation for autonomous vehicles,'' *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5323–5333, Dec. 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, Jan. 2021.

[4] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, ''Feature pyramid networks for object detection,'' in *Proc. CVPR*, Jul. 2017, pp. 936–944.

[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*, Nov. 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. 82031–82057, 2021.

[7] M. Z. Khan, M. K. Gajendran, Y. Lee, and M. A. Khan, ''Deep neural architectures for medical image semantic segmentation: Review,'' *IEEE Access*, vol. 9, pp. 83002–83024, 2021.

[8] P. Yin, R. Yuan, Y. Cheng, and Q. Wu, ''Deep guidance network for biomedical image segmentation,'' *IEEE Access*, vol. 8, pp. 116106–116116, 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. 153869–153884, 2019.

[10] J. Long, E. Shelhamer, and T. Darrell, ''Fully convolutional networks for semantic segmentation,'' in *Proc. CVPR*, Jun. 2015, pp. 3431–3440.

[11] L. 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 Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[12] S. Targ, D. Almeida, and K. Lyman, "ResNet in ResNet: Generalizing residual architectures," 2016, *arXiv: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*, Jul. 2017, pp. 6230–6239.

[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*, Jun. 2020, pp. 9370–9380.

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

[17] Y. Deng, "Deep learning on mobile devices: A review," *Proc. SPIE*, vol. 10993, May 2019, Art. no. 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*, Oct. 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," 2016, *arXiv:1606.02147*.

[20] M. Treml, J. Arjona-Medina, T. Unterthiner, R. Durgesh, F. Friedmann, P. Schuberth, A. Mayr, M. Heusel, M. Hofmarcher, M. Widrich, B. Nessler, and S. Hochreiter, "Speeding up semantic segmentation for autonomous driving," in *Proc. MLITS, NIPS Workshop*, 2016, vol. 2, no. 7, pp. 1–77.

[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*, Jul. 2017, pp. 5168–5177.

[24] R. P. K. Poudel, U. Bonde, S. Liwicki, and C. Zach, "ContextNet: Exploring context and detail for semantic segmentation in real-time," 2018, *arXiv:1805.04554*.

[25] 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, Jan. 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*, Jun. 2020, pp. 2633–2642.

[27] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *Int. J. Comput. Vis.*, vol. 126, no. 9, pp. 961–972, Sep. 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*, Jun. 2016, pp. 3213–3223.

[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*, Jun. 2009, pp. 248–255.

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

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

[32] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. CVPR*, Jul. 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 Recognit.*, vol. 90, pp. 119–133, Jun. 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*, Oct. 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 Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 6, pp. 2547–2560, Jun. 2021.

[36] J. Cai, Y. Liu, and P. Qin, "Attention based quick network with optical flow estimation for semantic segmentation," *IEEE Access*, vol. 11, pp. 12402–12413, 2023.

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

[38] M. Oršić and S. Šegvić, "Efficient semantic segmentation with pyramidal fusion," *Pattern Recognit.*, vol. 110, Feb. 2021, Art. no. 107611.

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

[40] T. Singha, D.-S. Pham, and A. Krishna, "FANet: Feature aggregation network for semantic segmentation," in *Proc. DICTA*, 2020, pp. 1–8.

[41] T. Singha, D.-S. Pham, A. Krishna, and J. Dunstan, "Efficient segmentation pyramid network," in *Proc. ICONIP*. Cham, Switzerland: 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*, Jun. 2019, pp. 9522–9531.

[43] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. CVPR*, Jun. 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, pp. 1–9, Apr. 2017.

[45] A. Howard, M. Sandler, B. Chen, W. Wang, L. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," in *Proc. ICCV*, Oct. 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*, Jun. 2018, pp. 8759–8768.

[47] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," 2019, *arXiv:1911.09070*.

[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*. Cham, Switzerland: 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, pp. 1–10, Feb. 2018.

[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, pp. 1–12, Jun. 2017.

[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*, Jun. 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*, Jun. 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*. Cham, Switzerland: Springer, 2020, pp. 582–600.

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

[55] S. Kong and C. Fowlkes, "Pixel-wise attentional gating for scene parsing," in *Proc. WACV*, Jan. 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*, Jun. 2020, pp. 1473–1481.

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

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

TANMAY SINGHA received the Master of Technology and Master of Computer Applications degrees in information technology from the University of Calcutta, India. He is currently a Ph.D. Research Scholar in computer science with Curtin University, Australia. Before joining Curtin University, he was a Lecturer with the Department of IT, Royal University of Bhutan, Bhutan, for nine years. His current research interests include 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.

DUC-SON PHAM (Senior Member, IEEE) received the Ph.D. degree from the Curtin University of Technology, in 2005. He is currently a Senior Lecturer with the Discipline of Computing, Curtin University, Perth, WA, 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 was a recipient of the Young Author Best Paper Award for a publication in IEEE TRANSACTIONS ON SIGNAL PROCESSING, in 2010.

ANEESH KRISHNA received the Ph.D. degree in computer science from the University of Wollongong, Australia. He was a Lecturer with the School of Computer Science and Software Engineering, University of Wollongong, from February 2006 to June 2009. He is currently a Discipline Lead of computing with the School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Australia. He has expertise in large-scale complex software development, model-driven development and evolution, service computing, software quality, requirements engineering, agent systems, formal methods, data mining, computer vision, and machine learning. His research is (or has been) funded by the Australian Research Council (ARC), Australian government agencies, such as the Department of Defence, the Australian Academy of Science, the NSW State Emergency Service, and the Department of Industry, Science, Energy and Resources, as well as various industry partners. He has supervised nine Ph.D. and two M.Phil. students to the successful completion of their degrees. He works closely with industry and undertakes practical; real-world industry-focused research projects. He has worked (and led in most instances) on several successful projects with companies, such as Woodside Energy, Amristar Solutions, Thales, Deloitte, IBM, Immersive Technologies, Gaia Resources, AFG Group, Autism West Inc., BW Solar Australia, Western Australia Dementia Training Center, and Andrew Corporation. He has widely published (with more than 140 articles) in the above areas in top journals and international conferences. He has been on the organizing committee and served as an invited technical program committee member for many conferences and workshops in the areas related to his research.