FANet: Feature Aggregation Network for Semantic Segmentation

Tanmay Singha
School of EECMS
Curtin University
Perth, Australia
tanmay.singha@postgrad.curtin.edu.au

Duc-Son Pham
School of EECMS
Curtin University
Perth, Australia
dspham@ieee.org

Aneesh Krishna
School of EECMS
Curtin University
Perth, Australia
a.krishna@curtin.edu.au

***Abstract*—Due to the rapid development in robotics and autonomous industries, optimization and accuracy have become an important factor in the field of computer vision. It becomes a challenging task for the researchers to design an efficient, optimized model with high accuracy in the field of object detection and semantic segmentation. Some existing off-line scene segmentation methods have shown an outstanding result on different datasets at the cost of a large number of parameters and operations, whereas some well-known real-time semantic segmentation techniques have reduced the number of parameters and operations in demand for resource-constrained applications, but model accuracy is compromised. We propose a novel approach for scene segmentation suitable for resource-constrained embedded devices by keeping a right balance between model architecture and model performance. Exploiting the multi-scale feature fusion technique with accurate localization augmentation, we introduce a fast feature aggregation network, a real-time scene segmentation model capable of handling high-resolution input image (1024 $\times$ 2048 px). Relying on an efficient embedded vision backbone network, our feature pyramid network outperforms many existing off-line and real-time pixel-wise deep convolution neural networks (CNNs) and produces 89.7% pixel accuracy and 65.9% mean intersection over union (mIoU) on the Cityscapes benchmark validation dataset whilst having only 1.1M parameters and 5.8B FLOPS.**

***Index Terms*—Semantic segmentation, BiFPN, MobileNet.**

I. Introduction

Semantic segmentation is one of the most challenging tasks in computer vision. It aims to assign a class/label to each pixel of an input image. These classes/labels are defined by the training set and could be anything such as car, building, people, bicycle, train, and many more. By clustering parts of the image together based on same object of interest, it identifies regions of different objects in the scene. Thus, it opens the door for developing numerous real-time applications specially in the field of autonomous driving, robotics, virtual reality and video surveillance.

Over the decade, innumerable CNN models [1], [2], [3] have been proposed to generate a segmentation map for an entire image in a single forward pass. Due to the high robustness to variance in scale and handling capability of rich semantics, CNNs are widely used for object detection, instance and scene segmentation. To meet the growing demand of multi-scale testing, it becomes essential to improve prediction accuracy of these models [4], [5]. To address this, feature pyramid network (FPN) was introduced in [6]. It utilizes an in-network feature hierarchy architecture and introduces a top-down pathway to achieve multi-scale feature fusion. Many object detection and instance segmentation models adopt FPN to achieve high accuracy. Later on, PANet [7] was introduced to enhance the entire feature hierarchy of FPN. It resolves the issue of localizing the signals in the lower layers of FPN by introducing a bottom-up path augmentation. Inspired by these two approaches, recently Google Brain has come up a new technique, called Bi-directional Feature Pyramid Network (Bi-FPN) for object detection [8]. It optimizes PANet feature fusion approach and introduces few skip connections to enhance feature maps' quality at different levels. These multi-scale techniques are vastly used for object detection, but not for semantic segmentation. Inspired by these multi-scale fusion approaches, we introduce a modified Bi-FPN technique in our proposed scene segmentation model. This modified design also eliminates the need for global contextual prior, thus optimizing the overall model architecture.

To generate rich semantic features for multi-scale feature fu-

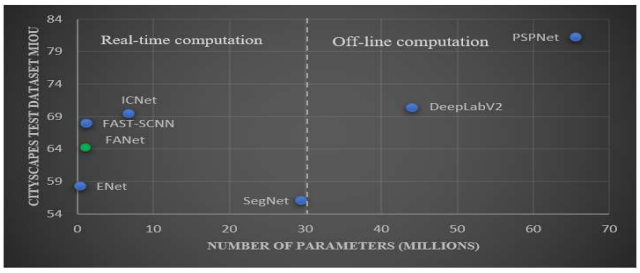

Fig. 1. **Number of parameters vs cityscapes mIoU**- Our model generates less parameters compare to others and produces better accuracy than ENet and SegNet. It also generates lesser FLOPs than all models (see Table VI)

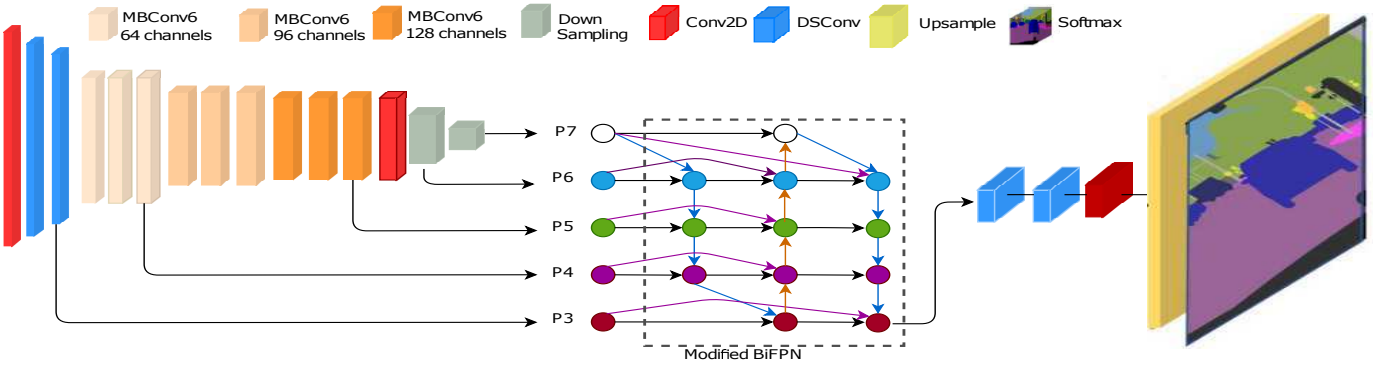

Fig. 2. Complete architecture of FANet

sion in scene segmentation, we need an efficient and optimized feature extractor. Many segmentation models use ResNet as a backbone due to its high scalable nature. It can be scaled up from ResNet-18 to ResNet-200 [9] to improve model performance, but the number of parameters and FLOPS also increase with the model size. Various segmentation models such as DeepLabV3+ [2], EfficientNets [10], Fast-SCNN [11] use MobileNetV2 [12] as feature extractor due to its high scalabilty and optimized features. Inspired by MobileNetV2 and FAST-SCNN, we use mobile inverted residual bottleneck blocks to design the backbone network of our model. We also use depth-wise convolution (DsConv) to optimize the number of operations and parameters of convolution layers, thus we strike a right balance between the model architecture and model performance. Hence, we design and propose an optimized scene segmentation model for resource-constrained computer vision devices.

II. Related Work

Semantic segmentation models typically follow an encoder-decoder architecture. In the encoder stage, a deep CNN typically computes a feature hierarchy layer by layer and develops an inherent multi-scale pyramid shape, whereas at the decoder end, high-semantic feature map is up-sampled and fused with the previous layer feature map through lateral connections to recover higher spatial dimensions. After extracting spatial details, model predicts the class for each pixel to complete the segmentation process. Different networks such as VGG [13], ResNet [9], Xception [14], or MobileNet [15] are often used as the encoder.

The Fully Convolutional Network (FCN) [16] has shown a revolutionary approach for all modern CNNs by adopting an encoder-decoder architecture. It replaces the fully connected layers from the top of the encoder by convolution one to generate a spatial map instead of classification scores. Based on this foundation, several models such as UNet [17], RefineNet [18], DeepLabV3+ [19] are then developed by exploiting the lateral connection between the low-level feature maps across resolutions and semantic levels. Inheriting the benefits of multi-scale features fusion, several other approaches (PSP-Net [20], DeepLab [2], ParseNet [21]) are also developed by utilizing a pyramid pooling module (PPM) [20] or an atrous spatial pyramid module [22] as a global contextual prior. All these techniques have shown that multi-scale feature fusion plays an important role in improving the prediction. However, fusing different feature maps of different spatial dimensions often introduces a large semantic gap caused by different depths of layers. To address this issue, FPN [6] and PANet [7] were introduced for fusing semantic features of all semantic labels. In contrast to these top-down and bottom-up approaches, the Google Brain team introduced another multi-scale feature fusion technique, called NAS-FPN [23] by leveraging the neural architecture search for automatic design of feature network topology. But due to long search time and unpredictable network architecture, this technique is not suitable for embedded devices. Recently, another efficient, scalable object detection model, named EfficientDet [8] has been introduced by the Google Brain team, based on a new feature-fusion technique, called Bi-directional Feature Pyramid Network (Bi-FPN). Bi-FPN is an optimized version of PANet and it produces better results compared to existing multi-scale feature fusion techniques.

Since, the demand for high-performing real-time methods is increasing rapidly, so several research works have been conducted in the field of semantic segmentation to target resource-constrained embedded devices. SegNet [1] is one of the pioneering models targeting real-time computation. Later on, ENet [24], ICNet [25], ContextNet [26], BiSeNet [27] were introduced to improve the performance in a real-time environment. All these models generate moderate results but still could not process high-resolution images quickly enough due to their large number of FLOPS and parameters. By keeping a balance between model size and model performance, FAST-SCNN [11] attempted to address this issue. It can process high-resolution input images quickly in real time and produces better segmentation results. Later on, another model, called DFANet [28] is introduced for real-time scene segmentation. Though the model claims to achieve 70.3% mIoU on Cityscapes test set, but it has more than 6 times parameters and FLOPS compared to FAST-SCNN and cannot process full resolution of Cityscapes images.

Therefore, inspired by the simple design of FAST-SCNN,

we develop a new model by exploiting optimized architecture of the residual block of MobileNetV2. To reduce the large semantic gap between the spatial dimensions of low-feature map and global feature map, we also incorporate a new multi-scale feature fusion approach. We detail our approach in the next section.

III. Proposed Methods

In this section, we propose an optimized and efficient network for semantic segmentation, call FANet, capable of handling high-resolution images and producing a quality output at a lower computational cost than many existing alternatives .

A. Network Architecture

The overall architecture of the proposed model is shown in Fig.2. In the following subsections, we discuss the backbone network, the modified Bi-FPN and classifier module in detail.

1) Backbone Network: Since our main focus is to design an optimized model capable of handling full-resolution images with higher accuracy in real time, we decide to employ the MobileNetV2 architecture [12]. Specifically, we use three bottleneck blocks, each repeating three times. Similar to the residual block, each bottleneck block contains an input followed by several bottlenecks, then followed by expansion. The expansion ratio, ratio between the size of the input bottleneck and the inner size, is 6. The basic implementation structure of MBConv6 is demonstrated in Table I. Here, h, w, c and c' denote the spatial dimensions of tensors, t defines the expansion ratio and s defines the stride. Note that we use ReLU non-linearity in the first two layers because of its robustness. However, we do not use it after the last layer to prevent non-linearities from destroying meaningful information. We provide the similar layered architecture to each MBConv6 block to make our design uniform and simple. Based on the size of the filter in each block of MobileNetV2, we set the filter size of each MBConv6 block. Table II shows that we use 64 channels for first bottleneck block, 96 for second and 128 for third block.

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

Inspired by FAST-SCNN, we introduce three layers at the beginning of the residual blocks to down sample the input. The first layer is a standard convolution layer (Conv) and remaining two layers are depth-wise separable convolution layers (DSConv). Although, DSConv optimizes number of operations and parameters, we employ Conv at the first stage due to less channels (only three) of the input image which makes the use of DSConv insignificant. As the low-dimensional input subspace produces rich semantic features, we introduce two max-pooling layers on top of residual blocks to reduce the spatial dimensions of feature map and create two additional stages for multi-scale feature fusion. Note that adding these down-sampling layers does not increase computational cost.

TABLE II
Layer architecture of backbone network

Stage (i)	Input	Operators	Layers (n)	Output
1	1024×2048×3	Conv, k3×3	1	512×1024×32
2	512×1024×32	DSConv, k3×3	1	256×512×48
3	256×512×48	DSConv, k3×3	1	128×256×64
4	128×256×64	MBConv6, k3×3	3	64×128×64
-	64×128×64	MBConv6, k3×3	3	32×64×96
-	32×64×96	MBConv6, k3×3	3	32×64×128
5	32×64×128	Conv, k1×1	1	32×64×64
6	32×64×64	MaxPooling	1	16×32×64
7	16×32×64	MaxPooling	1	8×16×64

The feature maps of the same spatial sizes produced by different layers fall under the same stage. Thus, we generate seven stages in backbone network and it reflects in Fig.2. Note that the original input size should be divisible by 2^7. We use semantic features of P3, P4, P5, P6, P7 levels for feature fusion. A complete description of our multi-scale feature fusion technique is given in the next sub-section.

2) Modified Bi-FPN: Fig. 3 shows the design of different features scaling techniques. FPN (a) introduces the concept of multi-scale feature fusion by setting a top-down pathway, whereas PANet (b) suggests an additional bottom-up path augmentation for preserving the local context. By exploiting neural architecture search, recently a new model, called NAS-FPN (c), is introduced for object detection. Although it optimizes a large number of operations, it is difficult to interpret the feature network due to long neural network search time. More recently, EfficientDet introduces Bi-FPN (d) technique for object detection which optimizes several cross-connections of PANet and introduces lateral connections between the input to output node if they are at the same level. Thus, it improves network performance. Inspired by this approach, we design a modified version of Bi-FPN technique (e). We introduce a new top-down path augmentation for feature aggregation and produce final semantic features at the finest level. We also employ few lateral connections between the input to output node at same label to enhance the model performance.

Formally, given a list of semantic features F^{in} = ($F^{in}_{l_1}$, $F^{in}_{l_2}$,..., $F^{in}_{l_i}$) generated by different layer stages(P3 to P7), our aim is to find a transformation f that can effectively map different features at different semantic levels and produce a rich multi-scale semantic features F^{out} = ($F^{out}_{l_1}$, $F^{out}_{l_2}$,..., $F^{out}_{l_i}$) at all labels. Finally, all contextual feature maps at different labels will be aggregated to generate final global feature map at the finest level: $FF_{out} = F^{in}_{l_1} + \sum f(F^{out})$. Fig. 2 shows a graphical representation of our multi-scale feature fusion technique. Features from P3 to P7 stages are taken for feature fusion. First, a top-down approach is employed to fuse global features with local features. Secondly, a bottom-up path augmentation technique is implemented to localize

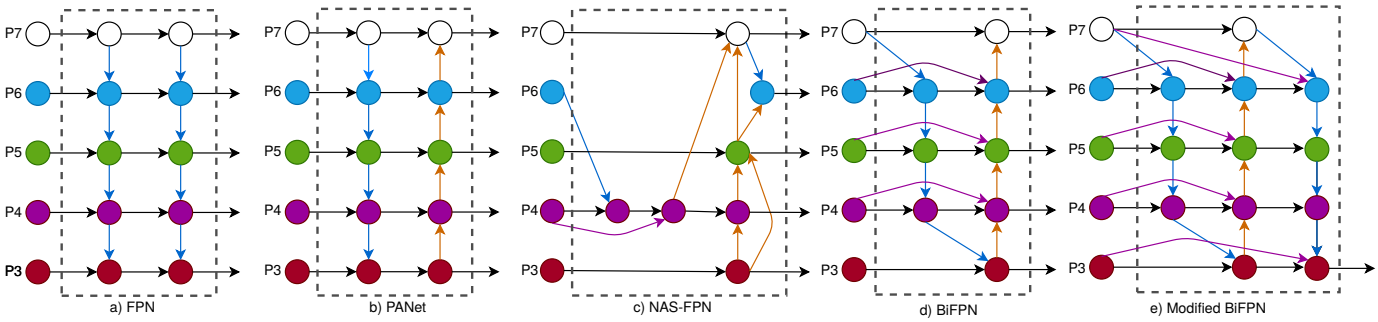

Fig. 3. Design of different feature networks. a) Feature Pyramid network - introduces a top-down pathway for multiscale feature fusion; b)Path Aggregation Network introduces a bottom-up pathway for better localisation; c) Neural Architecture Search FPN uses neural search to find out irregular feature network topology; d) Bidirectional FPN optimises PANet and introduces skip connection; d) Modified BFPN introduces an additional top-down pathway and two lateral connections for feature aggregation and better prediction

each objects of the scene and generate a list of contextual feature maps at different labels($F_{l_3}^{out}$, $F_{l_4}^{out}$, $F_{l_5}^{out}$, $F_{l_6}^{out}$ and $F_{l_7}^{out}$). The size of each feature map in each successive stage is reduced to 1/2 of its previous stage size. For example, at P3, the feature size is 128 $\times$ 256px whereas, at stage P4, the size is 64 $\times$ 128px. Once the rich contextual features are generated, we generate the final aggregated feature map at the finest level through a top-down path augmentation.

$$F_{l_7}^{out} = Conv(F_{l_7}^{out}) \tag{1}$$

$$F_{l_6}^{out} = Conv(F_{l_6}^{out} + Upsample(F_{l_7}^{out}) + Upsample(F_{l_7}^{in})) \tag{2}$$

$$F_{l_5}^{out} = Conv(F_{l_5}^{out} + Upsample(F_{l_6}^{out})) \tag{3}$$

$$F_{l_4}^{out} = Conv(F_{l_4}^{out} + Upsample(F_{l_5}^{out})) \tag{4}$$

$$F_{l_3}^{out} = Conv(F_{l_3}^{in} + F_{l_3}^{out} + Upsample(F_{l_4}^{out})) \tag{5}$$

3) Classifier module: The purpose of adding this module on top of the model is to assign a class to each pixel of feature map. To keep the model design simple, we use only two depth-wise separable convolutions, one standard convolution, one Upsample and one softmax layer. Depth-wise separable convolution layer convolves the feature map along all dimensions by optimizing the number of parameters and FLOPs, whereas Upsample layer recovers the spatial dimensions of feature map without increasing computational cost. Finally, by using softmax activation, a multinomial logistic regression function, our model assigns a class to individual pixel. Thus, segmented output is predicted by the model. In this research, we use 19 classes (excluding background) out of 30 classes of the Cityscapes dataset.

IV. Experiment

We trained the proposed FAST-FANet model with the Cityscapes training set and evaluated its performance on the validation and test sets. We also compared model performance with few off-line (DeepLabV3+, Bayesian SegNet) and real-time (FAST-SCNN) segmentation models under same configuration. Details are given in the following.

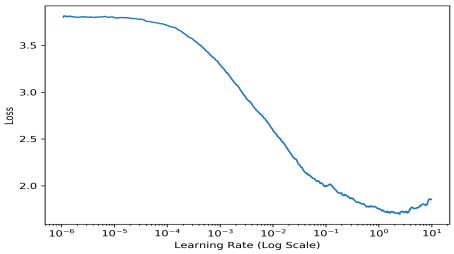

Fig. 4. Finding optimal learning rate

A. Implementation Details

To conduct this experiment, we use a dual Nvidia GeForce RTX 2080Ti GPUs system, each GPU has 11GB of memory. To exploit the parallel processing power of GPUs, we use CUDA 10.2. Our proposed model is developed using `tensorflow` 2.1.0 and `keras` 2.3.1. To utilize both GPUs in data-parallel distributed training environment, we employ the `horovod` framework [29]. It takes a single-GPU `tensorflow` program and trains it on multiple GPUs. For instance, in this research, `horovod` divides the whole training set into two sets and runs the training script on individual set in each GPU. Thus, it boosts the run-time performance by effectively utilizing all resources. We use stochastic gradient decent (SGD) as the model optimizer with 0.9 momentum.

Inspired by [2], [15], [20], we use 'poly' learning rate by setting 0.045 as base value and 0.9 as power. To find out the optimal learning rate in each epoch during training the model, we train our model for 5 epochs using polynomial scheduler and plot model loss against learning rate. Thus, we set upper and lower bound of learning rate for training. Fig. 3 illustrates the plot of learning rate vs model loss as an example. To calculate model loss, we use categorical cross-entropy.

Following the suggestion by MobileNetV2, we use 0.00004 as l_2 regularization for top layers of the model except depth-wise convolution. To avoid overfitting due to limited data, we apply various data augmentation techniques such as random horizontal flip, vertical flips, random crop, resizing and adjusting brightness, saturation and contrast of images. We also use a dropout layer just before the softmax layer. It also can

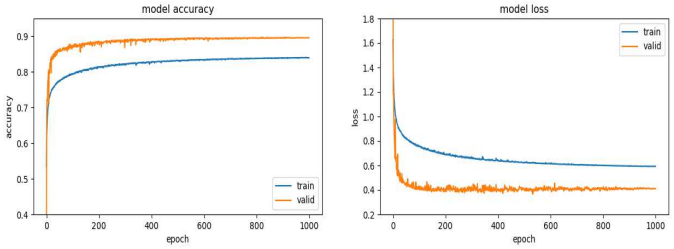

Fig. 5. Plot of model accuracy and loss against epoch

regulate model over-fitting issue and can improve validation accuracy. After training the model several times with different dropout rates, we noticed that a dropout rate less than 0.2 cannot address the overfitting issue whilst a value more than 0.6 value causes underfitting. Therefore, we set the average of these two values as dropout rate.

B. Dataset

We used Cityscapes [30] for evaluation. It is a large-scale dataset, mainly used for object detection, instance and semantic segmentation. It provides annotated data for 30 classes grouped into 8 categories. The dataset consists of around 5000 fine annotated images and 20000 coarse annotated ones.

In the experiments, we used only fine annotated images and considered only 19 classes for pixel annotations. The whole dataset is divided into three parts- training set (2,975 images), validation set (500 images) and test set 1,525 images). The labels for the training set and validation set are given by Cityscapes whereas the test labels are not provided by the benchmark. However, the prediction on the test set can be submitted to the Cityscapes server for evaluation.

We also used the CamVid dataset [31] for performance evaluation. This dataset is mainly designed for object detection in automated driving vehicle. The images of this dataset were generated from a recorded video, then annotated frames were created by assigning a class colour to each object of the frame by human operators. This dataset contains 267 images for training, 101 for validation and 233 for testing. Out of 32 classes, we used 12 classes (including void) for performance evaluation. Due to the small size of the dataset, models were under-learned. We used data augmentation to address this issue.

C. Model Evaluation

In this section, we illustrate the performance of FANet on the Cityscapes dataset for semantic segmentation. Pixel accuracy (Pi. Acc.), Class mean Intersection Over Union (mIoU) and category mIoU on validation and test sets are measured. We train our proposed model with different input resolutions for 1000 epochs. Fig. 5 shows the model accuracy and loss against the number of epochs on training and validation sets. It demonstrates that the model reached its saturation point after 800-900 epochs and obtained 89.7% pixel accuracy on the validation dataset which is much better than many existing models. Model also achieves 65.9% mIoU on validation set. We also performed an ablation study to verify our model's performance. We used FPN, BiFPN and modified BiFPN with our light-weighted backbone network and documented model performance in Table V. It can be observed that additional feature aggregation path and few lateral connections of the modified BiFPN technique enhance model's performance by 0.9%. Proposed FANet with modified BiFPN has a total of 1.1M parameters and 5.8B FLOPs which are comparably lesser than many existing real-time scene segmentation models. We also evaluated our model performance in terms of class-wise prediction accuracy at different input resolutions and presented the results in Table III as ready reference. From Tables III and IV, it is observed that the model prediction accuracy is almost 90% in most of the object categories such as flat, construction, nature, sky and vehicle, whereas in object and human categories, the model performance is lower due to the tiny size of the objects in these categories. For motorcycle and truck, performance is less than 50% due to the lack of training data available in these two classes. The same phenomenon is also observed with other models. The results for category-wise mIoU on the validation set are exhibited in Table IV: our model achieves 83.6% accuracy which is better than many real-time scene segmentation models such as SegNet, ENet and ContextNet.

D. Performance Comparison

To compare FANet performance with other existing models, we train few off-line and real-time segmentation models under same configuration with full input resolution and 4 batch size. Due to the hardware limitation, we could not set higher batch size for the experiment. Results of comparison on validation set are displayed in Table V. Due to large size of FLOPS, separable UNet, deepLab and Bayes-segnet are used as off-line segmentation model, where as FAST-SCNN is used as real-time computational model. As per the literature, FAST-SCNN produces 68.6% validation accuracy, but our experiment has achieved at most 63.3% accuracy on cityscaapes validation set and 63% on test set. We could not verify our FAST-SCNN implementation due to the unavailability of official GitHub implementation. But to our understanding, we have achieved best performance of FAST-SCNN compared to all available unofficial GitHub implementations. Therefore, our experiment shows that proposed model has achieved better accuracy (65.9%) than the FAST-SCNN. Based on other parameters such as FLOPS, number of parameters, model size, our proposed model out performs other models. We have also calculated inference time for all the models for 500 validation images and Table V shows that FANet takes less time to predict all the images. Hence, we can conclude that overall FANet performs well in real-time environment.

We have also compared the prediction on the test set as evaluated by the Cityscapes server. The result of comparison is displayed in Table VI. To compare overall performance of all models, we calculated FLOPS and number of parameters of each model listed in the Table VI. From this table, it can be observed that DeepLab and PSPNet produce best

TABLE III
FANET PERFORMANCE ON VALIDATION SET AT DIFFERENT INPUT RESOLUTIONS

Input Size	Road	S.walk	Build.	Wall	Fence	Pole	T.light	T.sign	Veg.	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	M.cycle	Bicycle	mIoU
1024×2048	96.2	75.1	89.3	52.7	47.3	47.0	53.5	64.6	89.8	55.2	92.5	70.2	45.6	90.9	47.0	70.4	61.5	38.8	65.2	65.9
768×1536	95.5	73.6	88.2	42.5	41.1	45.6	49.8	61.3	89.7	53.3	90.8	68.5	41.2	89.6	43.5	63.9	48.2	35.5	62.8	62.3
512×1024	95.4	71.8	87.1	34.9	36.8	44.2	43.6	58.0	89.4	56.5	90.8	66.7	38.3	88.1	41.5	55.8	41.4	32.1	61.9	59.7

TABLE IV
CATEGORY-WISE FANET PERFORMANCE ON CITYSCAPES VALIDATION DATASET

Input size	Flat	Construction	Object	Nature	Sky	Human	Vehicle	mIoU
1024×2048	96.7	89.4	55.6	90.1	92.5	71.5	89.2	83.6
768×1536	96.1	88.5	52.8	89.7	90.8	70.4	87.6	82.3
512×1024	96.1	87.7	50.6	89.3	90.8	68.9	86.5	81.4

TABLE V
PERFORMANCE EVALUATION OF DIFFERENT ON CITYSCAPES VALIDATION SET

Model	Class mIoU	Category mIoU	Number of parameters (in Million)	Number of FLOPS (in Billion)	Run-time per epoch	Inference time for validation set (sec.)	Model Size (MB)
Separable UNet	29.6%	62.3%	0.35	27	754	129	3.1
Bayes-SegNet	56.8%	77.2%	29.5	2720	672	214	225.4
DeepLab	58.2%	79.3%	37.9	1575	728	123	90.2
FAST-SCNN	63.3%	82.2%	1.2	7.7	258	120	9.4
FANet (with FPN)	**62.6%**	**81.3%**	**1.2**	**5.9**	**247**	**117**	**9.5**
FANet (with BiFPN)	**65.0%**	**82.5%**	**1.1**	**5.8**	**241**	**113**	**9.4**
FANet (with modified BiFPN)	**65.9%**	**83.6%**	**1.1**	**5.8**	**241**	**113**	**9.4**

*All models are trained with input of size 1024×2048×3 and batch size 4.

TABLE VI
PERFORMANCE EVALUATION OF DIFFERENT MODELS ON CITYSCAPES TEST SET

Model	Class mIoU	Category mIoU	Number of parameters (in Million)	Number of FLOPS (in Billion)
DeepLabV3+ [19]	82%	91.6%	37.7	267
DeepLabV2 [2]	70.4%	86.4%	37.9	1575
PSPNet [20]	81.2%	90.6%	65.5	516
SegNet extended [1]	56.1%	79.8%	29.5	1365
ENet [24]	58.3%	80.4%	0.4	19
ICNet [25]	69.5%	–	6.7	30
FAST-SCNN [11]	68%	84.7%	1.2	7.7
FANet	**64.1%**	**83.1%**	**1.1**	**5.8**

*empty cell means that data is missing on evaluation server.

accuracy on test set, but at the cost of large FLOPS and parameters. Due to the large size of backbone network, these models are suitable for off-line segmentation. For real-time scene segmentation, we need an optimized and efficient model which can work fast in real-time environment and requires less memory footprint. Among all the models, ENet has much less parameters. However, it has 19B FLOPS due to its multi-branch approach which slows down model performance and it also produces low prediction accuracy. On the other hand, ICNet generates a better accuracy (69.5%) among all the real-time segmentation models, but it has 30B FLOPS and 6.7M parameters. Compared to all other models, our proposed FANet has less parameters (1.1M) and FLOPS (5.8B) and also produces 64.1% class mIoU on the test set.

We also evaluated our proposed model on the CamVid dataset. For a comprehensive analysis, we compared our model performance with some existing real-time segmentation models. The results are readily shown in Table VII. Models with * sign are not trained by us. The results for these models are extracted from the literature. It is shown that FANet produces better class mIoU and Pi. accuracy on the CamVid validation set. It also demonstrates that FAST-SCNN performance is almost similar to FANet on CamVid dataset. A same observation is also made made after analyzing the predicted images by these two models. For qualitative assessment, we compare the prediction on the Cityscapes validation set and present it in Fig. 6. The second column shows the ground-truth of the

TABLE VII
EVALUATION RESULTS ON VALIDATION SET OF CAMVID DATASET

Model	input size	Class mean IoU	Pi. Acc.
SegNet* [1]	360×480	55.6%	–
Enet* [24]	360×480	51.3%	–
FAST-SCNN [11]	512×1024	57.5%	86.9%
FANet	**512×1024**	**57.8%**	**87.1%**

*empty cell means missing value in literature.

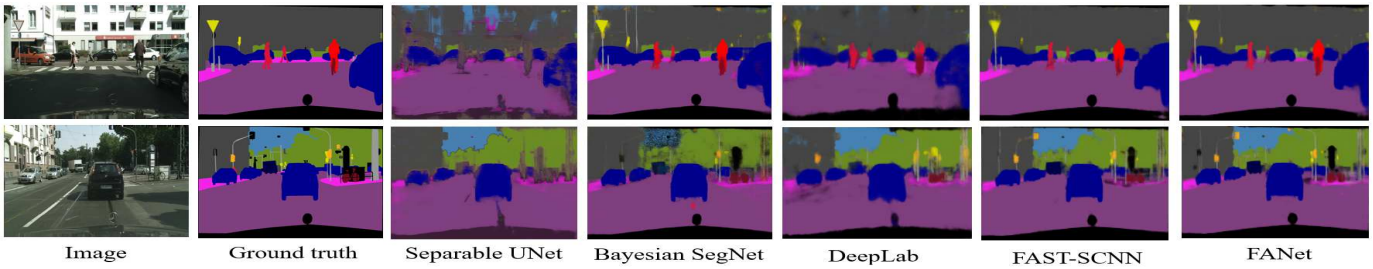

Fig. 6. All models (mentioned in table III) prediction on Cityscapes validation set

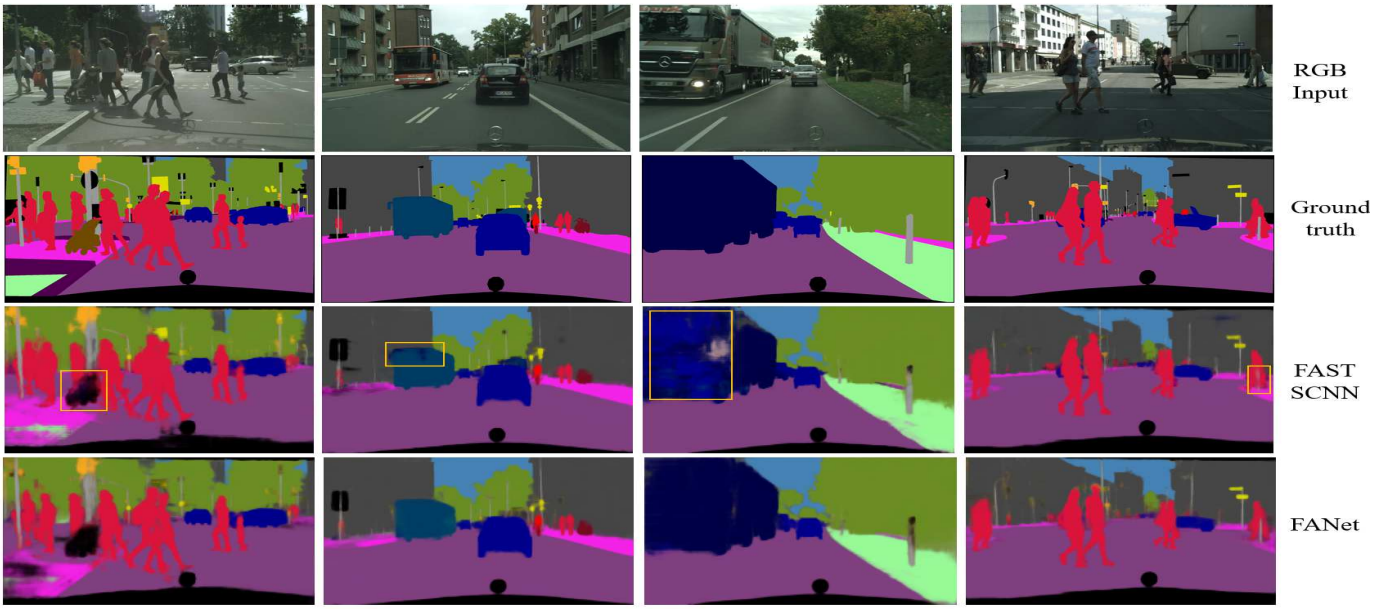

Fig. 7. Prediction samples by FAST-SCNN and FANet on Cityscapes validation set

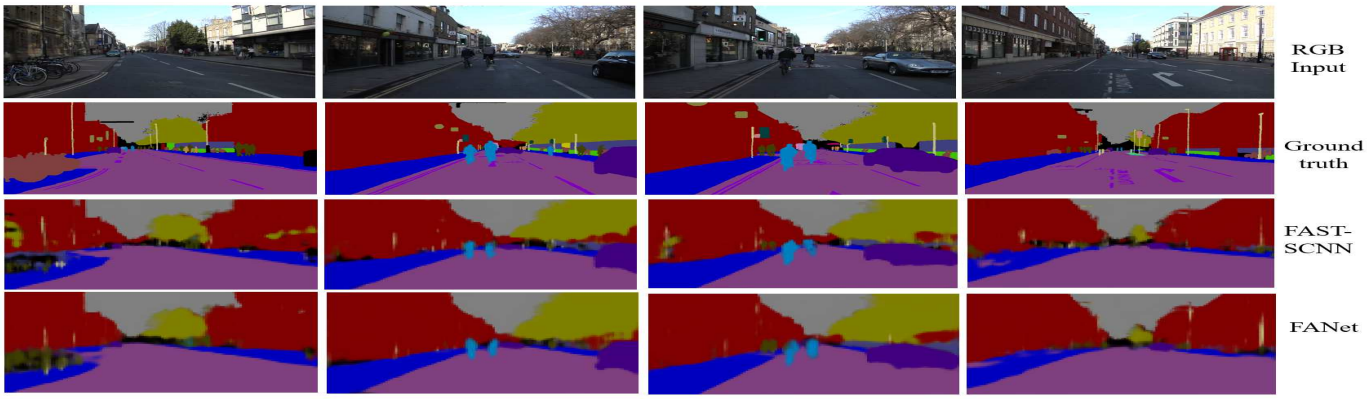

Fig. 8. FANet predictions on CamVid Validation Set

original images. It can be clearly identified that quality of the predicted images by FANet is better than other models. Especially, the edges of objects are more properly segmented and most of the tinny objects (traffic sign, traffic lights) are identified more clearly. We also observed that the quality of predicted images of FAST-SCNN is better compared to other models. We separately compared FANet's performance with FAST-SCNN and exhibit their prediction in Fig. 7. Though the quality of images by both models are almost similar but it can be observed that FAST-SCNN occasionally assigned incorrect labels to some pixels. This could be due to a large semantic gap between the local feature and global feature maps. In FANet, this gap is reduced by the multi-scale feature fusion technique which produces better predicted images overall.

Fig. 8 also shows predicted images by FAST-SCNN and FANet on CamVid dataset which appear almost same. However, the prediction quality by both models are not satisfactory due to the lack of training images which is an inherent issue with this dataset.

V. Conclusion

We have proposed an efficient and optimized semantic segmentation model which can handle high-resolution input images and can quickly produce output in real time with low computational cost. Due to its optimized structure and the ability to capture contextual information by a new feature scaling technique, it outperforms many existing real-time semantic segmentation models. We also demonstrate that our multi-scale feature fusion technique reduces the semantic gap between global feature and local feature and also substitutes the need for global contextual prior. In the future, we plan to evaluate FANet performance on COCO dataset. Our implementation of FANet is available at https://github.com/tanmaysingha/FANet.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *TPAMI*, vol. 39, no. 12, pp. 2481–2495, 2017.

[2] 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 transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. CVPR*, June 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.

[5] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. CVPR*, 2016, pp. 761–769.

[6] 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.

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

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

[9] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv preprint arXiv:1603.08029*, 2016.

[10] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[11] R. P. Poudel, S. Liwicki, and R. Cipolla, "Fast-scnn: fast semantic segmentation network," *arXiv preprint arXiv:1902.04502*, 2019.

[12] 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.

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

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

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[16] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Annals of the History of Computing*, no. 04, pp. 640–651, 2017.

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

[18] 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.

[19] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. ICCV*, September 2018.

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

[21] W. Liu, A. Rabinovich, and A. C. Berg, "Parsenet: Looking wider to see better," *arXiv preprint arXiv:1506.04579*, 2015.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[23] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Nas-fpn: Learning scalable feature pyramid architecture for object detection," in *Proc. CVPR*, 2019, pp. 7036–7045.

[24] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[25] 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.

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

[27] 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.

[28] 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.

[29] A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *arXiv preprint arXiv:1802.05799*, 2018.

[30] 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.

[31] 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.