International Journal of Machine Learning and Cybernetics
https://doi.org/10.1007/s13042-019-01005-5

ORIGINAL ARTICLE

SegFast-V2: Semantic image segmentation with less parameters in deep learning for autonomous driving

Swarnendu Ghosh[1] · Anisha Pal[2] · Shourya Jaiswal[2] · K. C. Santosh[3] · Nibaran Das[1] · Mita Nasipuri[1]

Received: 8 February 2019 / Accepted: 21 August 2019

Abstract
Semantic image segmentation can be used in various driving applications, such as automatic braking, road sign alerts, park assists, and pedestrian warnings. More often, AI applications, such as autonomous modules are available in expensive vehicles. It would be appreciated if such facilities can be made available in the lower end of the price spectrum. Existing methodologies, come with a costly overhead with large number of parameters and need of costly hardware. Within this scope, the key contribution of this work is to promote the possibility of compact semantic image segmentation so that it can be extended to deploy AI based solutions to less expensive vehicles. While developing cheap and fast models one must also not compromise the factor of reliability and robustness. The proposed work is primarily based on our previous model named "SegFast", and is aimed to perform thorough analysis across a multitude of datasets. Beside "spark" modules and depth-wise separable transposed convolutions, kernel factorization is implemented to further reduce the number of parameters. The effect of MobileNet as an encoder to our model has also been analyzed. The proposed method shows a promising decrease in the number of parameters and significant gain in terms of runtime even on a single CPU environment. Despite all those speedups, the proposed approach performs at a similar level to many popular but heavier networks, such as SegNet, UNet, PSPNet, and FCN.

Keywords Compressed encoder–decoder model · Semantic image segmentation · Deep learning

1 Introduction

Over the last few decades, Artificial Intelligence (AI) has made tremendous progress in various industries. One of the major fields that have received a boost from deep learning is autonomous driving. Many top tier automobile industries have been deploying vehicles with varieties of AI modules, starting from Advanced Driver-Assistance System, such as park assist, anomaly detection and emergency braking, to fully autonomous systems, such as driver-less transportation and auto parking. While the expensive brands and models are aiming for a fully autonomous model, other brands are struggling for the cheaper AI assistive modules/technologies. These technologies must be performed in a restricted hardware configuration without a significant drop in performance. As a result, AI assistive tools integration for all automobiles can be possible. which can have an impact across the world. In many countries, traffic systems can get extremely complicated due to lack of awareness and a variety of vehicles and other objects on the streets. Cheap assistive AI can have a much more significant impact in such

✉ K. C. Santosh
santosh.kc@ieee.org

Swarnendu Ghosh
swarnendughosh.cse.rs@jadavpuruniversity.in

Anisha Pal
anishapal006@gmail.com

Shourya Jaiswal
shourya98@gmail.com

Nibaran Das
nibaran.das@jadavpuruniversity.in

Mita Nasipuri
mita.nasipuri@jadavpuruniversity.in

[1] Jadavpur University, Kolkata, India

[2] Manipal Institute of Technology, Manipal, India

[3] University of South Dakota, Vermillion, USA

Published online: 28 August 2019

scenarios, not only improving the quality of traffic but also spreading awareness about good driving practices. The reliability of a system is equally important as the cost. Therefore, it must also be ensured that the efforts to build cheaper systems do not compromise the robustness of the system. In the proposed approach, we present a semantic image segmentation in driving scenarios. In contrast to state-of-the-art tools, it has less number of parameters and it can be processed in a CPU environment. We observe that, in general, several ideas were implemented to reduce the number of parameters of convolutional neural networks. However, it does not happen in image segmentation based on its semantic values.

The current work is the extension of the initial implementation of "SegFast" [18]. The major contributions in "SegFast" was the introduction of "spark" modules in the feature extractor and transposed depth-wise separable convolutions in the decoder. In the current work, we have further analyzed by comparing against the state-of-the-art approaches, such as UNet and PSPNet. Additionally, kernel factorization can further decrease the number of parameters [22]. Besides, in this work, the implementation of MobileNet based encoder was used in place of the SqueezeNet-based encoder to learn/discuss the pros and cons of the network.

2 Related works

Previously neural networks primarily depend on the quality of hand-crafted features [7]. However, with these methods, problems like semantic image segmentation are not trivial for natural scene images. With the advent of deep learning, semantic image segmentation methods evolve and utilize various approaches (even integrating them) for optimal performance. Fully convolutional network [15] combines the convolutional feature extractor of the VGG-Net [21] with a pixel-wise classifier that can be considered as one of the examples. It results in the end-to-end and efficient algorithm but generates very coarse segmentation maps due to the sub-sampling layers. Later algorithms [1, 19] used an encoder–decoder model while forwarding features from different layers of the network to achieve better segmentation results. The encoders are a series of convolutional layers followed by sub-sampling operations, such as average pooling and/or max-pooling. The decoders, on the contrary, are made of up-sampling layers: transposed convolutions or unpooling. We find that this architecture improves the sharpness of the segmentation to a great extent. Several other algorithms use methods, such as dilated convolution [3], spatial pooling pyramids [24] and multi-level refinements [14] in their classifiers to increase the reception area of the network and to improve the sharpness. Besides, the different types of feature extractors or encoders use in the networks play an important role in the performance of segmentation algorithms. In such problems, typical encoders are in use that are commonly based on VGGNet [21] and ResNet [8]. The most important component of ResNet is the use of skip connections that improves the gradient flow while allowing the network to train better. Few other works also use encoders based on the 152 layer DenseNet architecture [10, 12]. Although it provides better accuracy, it suffers due to computational complexity. The proposed network draws inspiration from encoder–decoder architectures, such as UNet [19] and it can apply the concepts of depth-wise separable convolution [4, 13] on a feature extractor module, which is based on the SqueezeNet [11] architecture. Over the years, various strategies have been implemented to reduce the number of parameters, since fine-tuning the parameters is not straight-forward. The most prominent of them is probably XceptionNet [4] and MobileNet [9] that demonstrates the use of depth-wise separable convolutions. The MobileNet architecture has been used as an encoder in an U-Net model to perform image segmentation[1]. However, in the approach, the decoder is still based on a normal transposed convolution instead of depth-wise separable convolutions. Other approaches, such as kernel factorization comes in handy to reduce the number of parameters. One of the most important works that address this issue is the SqueezeNet [11] architecture, which has the same performance (accuracy) as that of AlexNet but with reduced model size and fewer parameters.

3 Proposed approach

The proposed architecture draws inspiration from SqueezeNet [11], depth-wise separable convolution [4, 13] and the encoder–decoder technique as reported in UNet [19]. The SqueezeNet architecture proposed the use of fire modules. The fire modules are designed to reduce the number of parameters by replacing spatial kernels with a combination of spatial and pixel level kernels. In our proposed method, the fire modules are replaced with a smaller version, which we call "spark" modules. Note that spark modules (see Sect. 3.1 for more detailed information) implement the use of depth-wise separable convolutions instead of traditional convolutions. Depth-wise separable convolutions are later described in details in Sect. 3.2. This modified SqueezeNet with spark modules is used as an encoder in the proposed architecture as shown in Fig. 1. The encoder is made of four encoding blocks, where the first one has a convolution layer, which is followed by a ReLU activation and an average pooling layer. The other two blocks are similar and use two spark modules (see Sect. 3.1 for more detailed information), where each of them is followed by an average pooling layer. The

[1] https://github.com/akirasosa/mobile-semantic-segmentation.

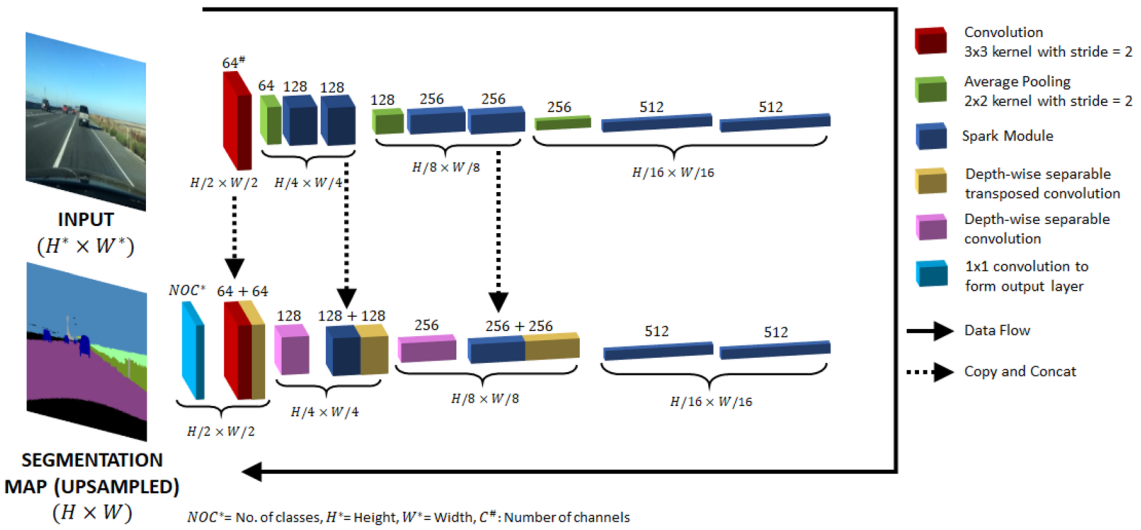


Fig. 1 Block diagram: the proposed network architecture

last block has four spark modules. Architectures like SegNet use max-unpooling while forwarding the pooling indices from the corresponding pooling layers in the encoder. Even though it reduces the computational cost, yet it results in a loss of accuracy. Therefore, in the current approach, we propose the use of transposed depth-wise separable convolutions. In general, depth-wise separable convolutions are only used for normal convolutions [4, 13]. However, transposed convolutions can help reduce the number of parameters. In the decoder, the first decoding block has a transposed depth-wise separable convolution (see Sect. 3.2 for more detailed information) after the output of the last encoder. The feature maps of the decoder and the corresponding encoder having the same size are then concatenated and the result is again convolved with a 3×3 depth-wise separable convolution. This feature concatenation technique from corresponding encoding layers was proposed in U-Net [19]. These operations are repeated three times. The output layer consists of a pixel level probability distribution obtained using a 1×1 convolution and a softmax operation. In the output segmentation mask, each pixel is represented by a specific class for which the probability value is maximum. The final output is upsampled to match the original input size. In [1], we observe that passing indices rather than the entire features can help reduce the memory requirement of the network. It also drops the performance in terms of accuracy. In our network, the encoded feature maps are passed to the decoder: U-Net [19] and the number of parameters of the proposed network is lesser than SegNet, while accuracies can be compared in multiple datasets. Besides, a further extension is explored in addition to the proposed methodology. The use of factorized kernels instead of traditional spatial kernels also helps in reducing the number of parameters. This allows using even larger 5×5 kernel with improved performance. The proposed model achieves a trade-off between accuracy and computational complexity. The proposed modules are further explained in detail in the different subsections below.

3.1 Spark module: a smaller fire module

One of the key contributions of SqueezeNet is the fire module. Fire module employs a few strategies, such as replacing spatial kernels with a combined spatial and pixel-wise kernels. The primary idea behind such an integration is to reduce the number of parameters. As an example, the replacement of 3×3 filters with 1×1 filters reduces the number of parameters by a factor of 9. A fire module consists of two layers, namely, a squeeze layer and an expand layer. The main goal of the squeeze layer is to reduce the number of input channels for the 3×3 filters in the expand layer. The expand layer consists of a combination of 1×1 filters and 3×3 filters. The three hyper-parameters in the fire module are the number of kernels in the squeeze layer (S), the number of 1×1 kernels in the expand layer (E_1) and the number of 3×3 kernels (E_3) in the expand layer. The relation between these three hyper-parameters is that the number of kernels in the squeeze layer is less than the total number of kernels in the expand layer. In a fire module, the input is first passed through a batch normalization layer followed by a convolution operation. Then an element-wise rectified-linear non-linearity (ReLU) [6, 16] is applied. ReLU has been established as one of the most efficient activation functions because of several factors such as fast differentiability, robustness against vanishing gradients (unlike sigmoid, TanH) and also lack of extra parameters (Leaky ReLU, PReLU and so on). These operations are part

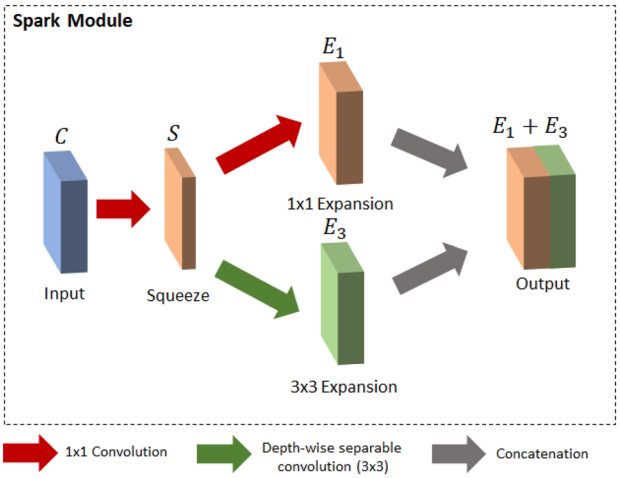


Fig. 2 Brief overview of the spark module

of the squeeze layer. The squeeze layer is followed by the expand layer, where it is subjected to two parallel convolution operations using 1×1 kernels and 3×3 kernels. Similar to the squeeze layer, each convolution operation is preceded by a batch normalization layer and is followed by a ReLU operation. In our work titled "SegFast", the fire module of SqueezeNet is further compressed by applying depth-wise separable convolutions instead of the 3×3 convolutions in the expand layer E_3. For more information about depth-wise separable convolutions, we refer to Sect. 3.2. This proposed smaller fire module is named "spark" module henceforth. The spark model is demonstrated in Fig. 2. The spark modules are further compressed in the extended version, which we call "SegFast-V2": the spatial kernel in the depth-wise separable convolution layer is replaced by its factorized version. The factorization of kernels is discussed in Sect. 3.4.

3.2 Depth-wise separable convolution

Depth-wise separable convolution [4, 13] decreases the computational complexity and the number of parameters in the network. As a result, it upgrades the overall efficiency. In depth-wise separable convolution, two separate operations are performed. Firstly, a spatial convolution operation is performed independently over each channel of the input tensor sharing the same single depth kernel over all the channels. After that, the number of input channels should be equal to the number of output channels. This is followed by pixel-wise convolution with a 1×1 kernel, projecting the features computed by the previous step onto a new feature space. A convolutional kernel of size $K_h \times K_w \times C$ requires $K_h * K_w * C$ parameters, where K_h and K_w are the height and width of the filter and C is the number of channels of the input tensor. An equivalent depth-wise separable convolution will require $K_h * K_w + C$ parameters, which is lesser than the former for $(K_h, K_w, C) > 1$. As the channel size increases in the later layers, the gain in terms of the number of parameters is more. The output size of a depth-wise separable convolution is the same as a normal convolution and can be expressed as

$$\begin{aligned} H' &= \frac{(H - K_h + 2P_h)}{S_h} + 1 \\ W' &= \frac{(W - K_w + 2P_w)}{S_w} + 1, \end{aligned} \tag{1}$$

where H' and W' are the height and width of the output, K_h and K_w are the height and width of the kernel. P_h and P_w refers to vertical and horizontal padding respectively in the input and S_h and S_w are vertical and horizontal strides. The choice of strides can be adjusted to reduce the scale of the input by a required factor.

3.3 Transposed depth-wise separable convolution

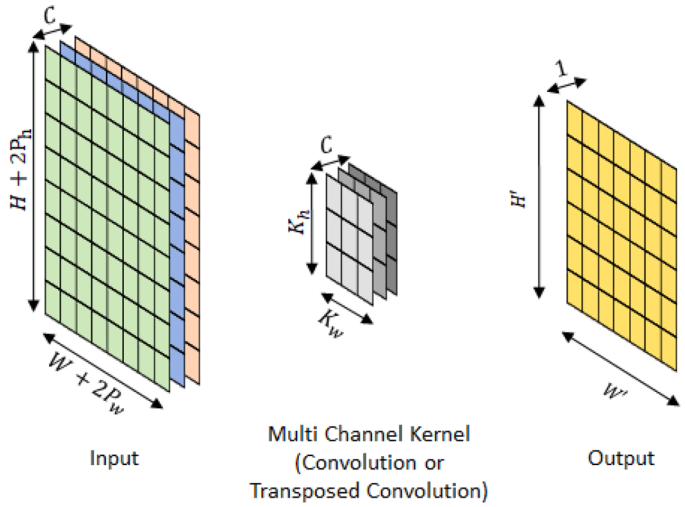
Transposed depth-wise separable convolution can be represented in a manner similar to a depth-wise separable convolution. Here also two separate operations are performed. The first operation involves performing spatial transposed convolution independently over each channel of the input while sharing the same single depth kernel over all the channels of the input. This is followed by a pixel-wise convolution with 1×1 kernel over all the channels of the output obtained after the first operation. The decrease in the number of parameters is the same as that of depth-wise separable convolutions. The size of the output tensor is also the same as a traditional transposed convolution, which can be expressed as

$$\begin{aligned} H' &= S_h(H - 1) + K_h - 2P_h \\ W' &= S_w(W - 1) + K_w - 2P_w, \end{aligned} \tag{2}$$

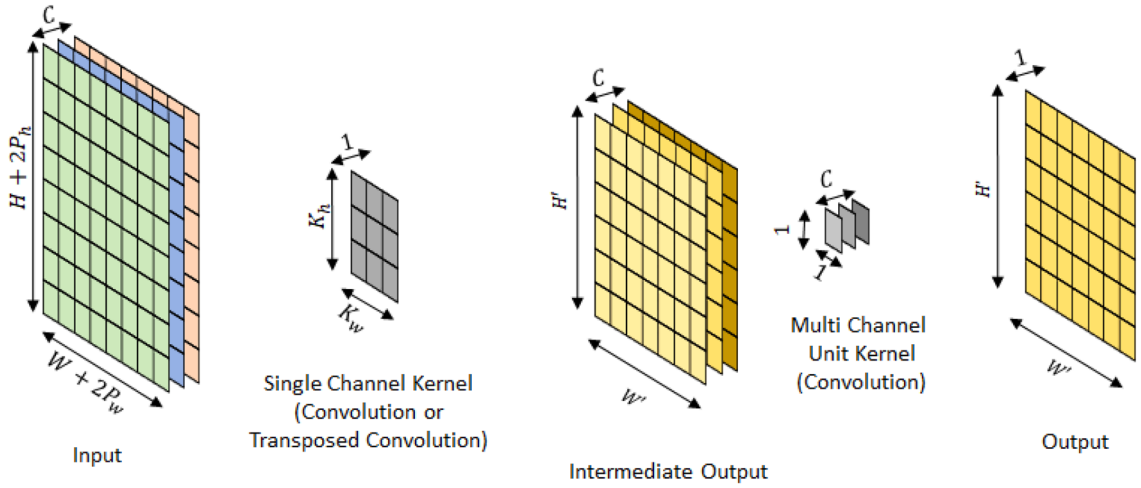
where the convention for the variables is similar to Eq. 1. The choice of strides can be adjusted to increase the size of the input by a required factor. The transposed convolutions are mainly used for decoders, where it is needed to increase the size of activations to obtain optimal features (Fig. 3).

3.4 Kernel factorization

Depth-wise separable convolutions can also be viewed as kernel factorization across the channel dimension. Additionally, the kernel can also be factorized across the spatial dimensions to further reduce the parameters. This will not only reduce the parameters but also implement higher kernel size without increasing the number of parameters by a large margin.Instead of standard spatial kernels, such as 3×3 and 5×5, factorized kernels decompose them into a



(a) Normal convolution



(b) Depth-wise separable convolution

Fig. 3 Pictorial relationship between normal and depth-wise separable convolutions or transposed convolutions

cascade of smaller kernels. For example, a 5×5 kernel can be decomposed into two successive 3×3 kernels, which in turn can be decomposed into a 1×3 kernel followed by a 3×3. Decomposing a 3×3 kernel reduces the number of parameters from 9 to 6 whereas decomposing a 5×5 kernel reduces the number of parameters from 25 to 12. Our basic network that is "SegFast" is primarily composed of 3×3 kernels. By using kernel factorization we can simulate the effect of higher size kernels like 5×5 to improve the performance while keeping the number of parameters at check. In our proposed model the all the spatial convolutions inside the spark modules, depth-wise separable convolutions and transposed convolutions. The extended variations are referred to as "SegFast-V2_3" and "SegFast-V2_5", which uses 3×3 and 5×5 factorized kernels.

3.5 MobileNet encoder

In another approach towards an efficient segmentation model, we are inspired by MobileNet classifier [9]. The MobileNet architecture is designed to create an efficient classifier in terms of speed, size, and accuracy for mobile and embedded applications. There is a standard MobileNet based segmentation approach, which uses a similar U-Net based architecture. However, we have established the use of depth-wise transposed convolution during the decoding phase. In our experiment, we used MobileNet as an encoder while the decoder was designed using the concept of transposed depth-wise separable convolution. The decoder consists of 4 blocks, and the final layer of the decoder is a single convolution layer with the number of output channels equal

to the number of classes in the data-set. Each decoding block applies a transposed depth-wise separable convolution (see Sect. 3.2 for more detailed information) to the output of the last encoder. The feature maps of the decoder and the corresponding encoder having equal size are concatenated followed by convolution using depth-wise separable convolution. After the final output is obtained by performing 1×1 convolution on the output obtained from the last decoding block, it is up-sampled to fit the original image size. In the output feature map, each pixel value corresponds to the class number for which that pixel has the maximum probability value. This model is referred to as "SegFast-Mobile". Note that an implementation of a U-Net like implementation exists with the MobileNet as an encoder,[2] however in our approach, we used a much more efficient decoder that uses the transposed depth-wise separable convolutions.

4 Experiments

4.1 Datasets

Our main goal is to create a compact model for semantic image segmentation that can be used in the domain of advanced driving assistance systems. We have trained the models on various driving-related datasets and analyzed the results. The training and test images were resized to 360×480 to simulate the effects of a camera with 480p resolution and also maintain uniformity across various experiments. No augmentation of any sort has been performed on the datasets. We have used five different datasets, such as BDD [23], CamVid [2], Cityscape [5], Synthia [20], and Mapillary [17]. Variety has been maintained across the chosen datasets in terms of the number of classes and the number of available training samples. The dataset descriptions are summarized in Table 1.

4.2 Results and discussion

We performed thorough experiments to compare the performance of the proposed set of models with various state of the art models. For comparisons, we computed the performance of fully convolutional architecture like FCN, encoder–decoder models like SegNet and UNet and networks with multi-scale pooling like PSPNet. While methods like DeepLab that implement techniques like a large field of view and conditional random fields perform quite well in problems like these, they tend to have huge overheads for all these computationally expensive modules and hence are left out of the comparative study. Depth-wise separable convolutions were effectively implemented in the MobileNet classifier [9]. We implemented a variation of our proposed network that uses the MobileNet architecture as an encoder. This can highlight the pros and cons of our proposed SqueezeNet based model as compared to another common architecture that also aims to reduce parameters. This work is an extension of our previous work where we have implemented the first version of "SegFast" [18]. As an ablation study, we replaced the depth-wise separable convolutions and transposed convolutions, with standard convolutions and included the performance of the network in the results section under the name of "SegFast-Basic". The exclusive contribution of the current approach is the kernel factorization that allows us to use larger kernels without increasing the parameters too much. We included two variations of the method, titled as "SegFast-V2_3" and "SegFast-V2_5" with 3×3 and 5×5 factorized kernels. We provided the pixel level accuracy and the mean IOU scores as a measure of performance. The pixel level accuracy has been calculated as follows:

$$Accuracy = \frac{\text{Number of pixels correctly predicted}}{\text{Total number of pixels}} \tag{3}$$

Since the main goal of the work is to create a compact architecture that performs at par with other modern approaches while working with considerable speed in a tight computational environment, we computed the forward and backward pass time for the running a single image through the network in a 16 threaded CPU (Intel Xeon E5-2623 V4). We have also shown the memory footprint of the network for single sample input. Figure 4 shows some sample outputs of the proposed set of SegFast methods with respect to some other commonly used methods for semantic image segmentation.

The primary objective of the work is to propose a model with a fewer number of parameters that can be compared with various state-of-the-art techniques. The fewer parameters result in lower memory consumption and faster forward and backward pass, thus providing the scope of building real-time systems with low-end hardware. Therefore, in our experiments, we considered the following terms: (a) number of parameters; (b) processing time; and (c) performance.

1. *Number of parameters*

   In terms of the number of parameters (see Table 2), the proposed family of SegFast networks use approximately 600K parameters. The model "SegFast-V2_3" with kernel factorization has a lesser number of parameters as compared to the normal SegFast Network (with 3×3 kernels). This allows us to increase the kernel size to 5×5 without increasing the number of parameters. However, due to the increased number of activation maps, they consume comparatively more amount of

[2] https://github.com/akirasosa/mobile-semantic-segmentation.

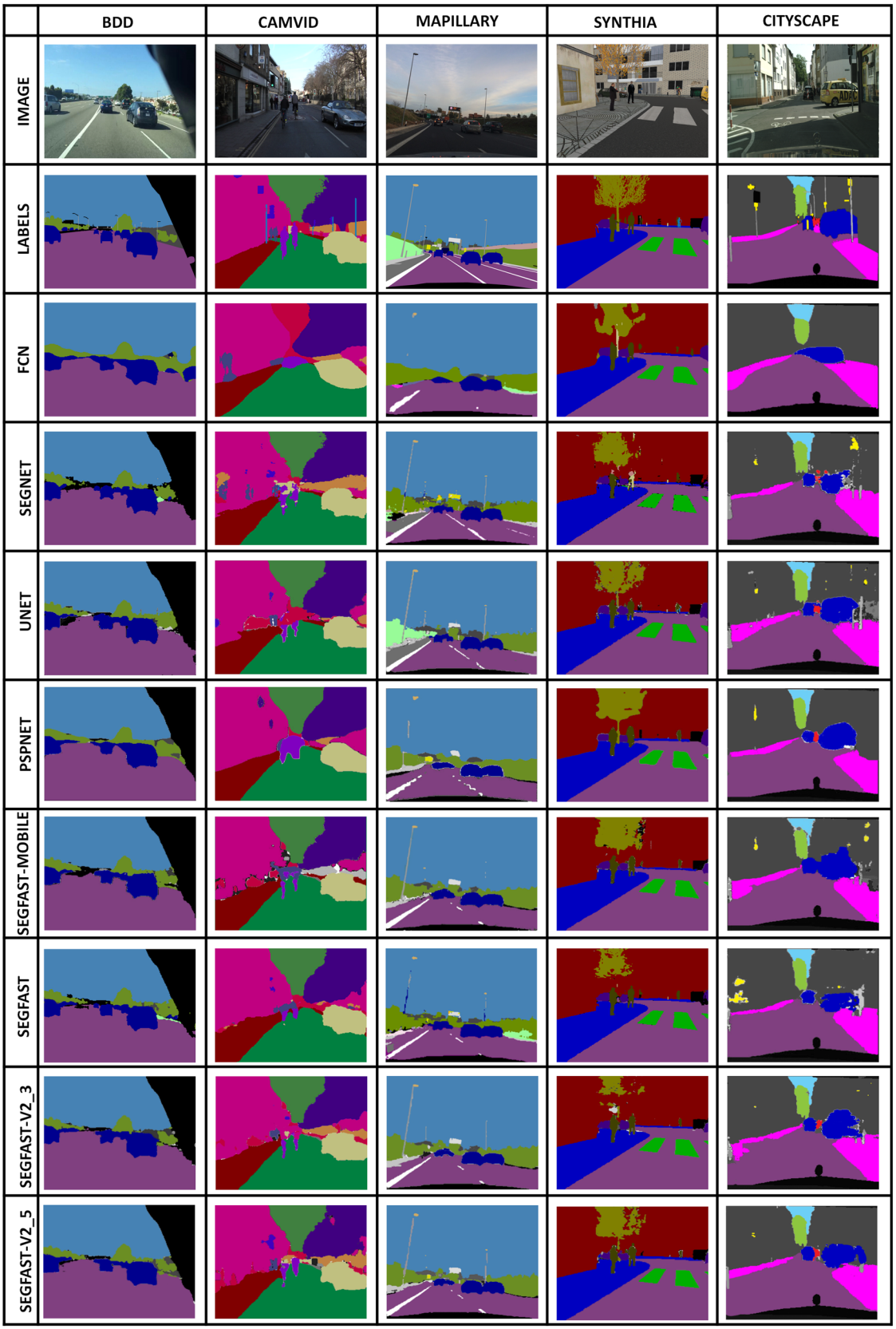


Fig. 4 Sample outputs across all proposed networks along with other modern networks across five datasets on randomly selected data

Table 1 Details of the used datasets

Dataset	Number of classes	Training samples	Test samples
BDD [23]	20	7000	1000
CamVid [2]	12	367	101
CityScape [5]	30	2975	500
Mapillary [17]	66	18,000	2000
Synthia [20]	13	10,786	2621

Table 2 Comparison of number of parameters and memory consumption of proposed networks against other popular networks

Architecture	Parameters	Memory consumption (MB)
FCN	134,507,010	3447
SegNet	29,455,702	1515
UNet	31,033,110	1471
PSPNet	65,606,764	2357
SegFast-Mobile	4,637,526	829
SegFast-Basic	2,965,976	1232
SegFast	603,288	**703**
SegFast-V2_3	**599,380**	1101
SegFast-V2_5	606,676	1199

Values in bold refer to the smallest possible units

Table 3 Forward pass and Backward Pass times (in seconds) of different networks while running the networks on a CPU

Architecture	Forward pass	Backward pass
FCN	3.39	5.23
SegNet	1.98	3.25
UNet	2.27	3.42
PSPNet	2.06	3.25
SegFast-Mobile	0.66	1.87
SegFast-Basic	0.419	1.336
SegFast	0.389	0.929
SegFast-V2_3	**0.360**	**0.858**
SegFast-V2_5	0.409	0.992

Values in bold refer to the smallest possible units

memory. The "SegFast" implementation consumes the least amount of memory and therefore, it is the network of choice if memory is of prime importance.

2. *Processing time*

   To test the speed of the networks in a constrained hardware environment, we chose to run the networks in a CPU-only mode. The numbers provided in table 3 shows the time taken for the forward pass and backward pass. The total training time is signified by the backward pass time multiplied by the number of samples in the dataset and the number of epochs trained. Whereas the forward pass time indicates the evaluation time for each sample. The fastest time was achieved by the versions with kernel factorization, which is primarily due to the reduction in the number of floating point operations. For the experiment, we used the Intel Xeon E5-2623 V4 CPU with 16 threads. This is considerably faster than many commonly used networks as well as the MobileNet based implementation. Due to extreme parallelization capabilities, the differences in times are much less significant in strong GPUs but that also defeats the purpose of the network, that is to perform well on low-end hardware. The "SegFast-V2_3" is the network of choice where speed is the primary focus.

3. *Performance*

   In terms of performance (see Tables 4 and 5), the proposed network may not be able to perform better than most of the state of the art network. However, given the increase in speed and memory utilization, the trade-off maybe acceptable in certain low-risk scenarios. The SegFast family of networks perform better than fully convolutional networks and compete quite closely with other bulky networks, such as SegNet, UNet, and PSPNet. In the ablation study where depth-wise separable convolutions are replaced with standard convolutions, a drop in performance was noted. In scenarios where performance cannot be compromised the "SegFast-Mobile" implementation can be used as it provides the best performance with a significantly lesser number of parameters. Initially a minimum sized kernel (3×3) was used to limit the number of parameters. In the extended version to accommodate higher size kernels(5×5) spatial kernel factorization have been implemented to limit the overall number of parameters. However, that does not improve the results significantly hence it can be said that the minimum resolution performs significantly enough considering the reduction in number of parameters.

5 Conclusion

The proposed set of models can be compared with state-of-the-art models, such as FCN, SegNet, UNet, and PSPNet. In contrast to traditional SqueezeNet based encoder or the MobileNet based encoders, our modified SqueezeNet based implementation uses less number of parameters. This results in a much lower number of FLOPs thus increasing the speed of execution. While using kernel factorization reduces the number of parameters, yet increases the memory footprint because of the extra number of activation maps that are needed to be stored. Considering the accuracy into account, SegFast-Mobile seems to be the strongest while

Table 4 Pixel level accuracies (in %)

Architectures	BDD	CamVid	CityScape	Synthia	Mapillary	Mean (± SD)
FCN	75.31	74.22	84.28	91.81	79.62	81.05 (± 7.21)
SegNet	82.79	88.70	86.08	92.05	82.46	86.42 (± 4.06)
UNet	81.77	87.70	87.39	95.43	84.11	87.28 (± 5.17)
PSPNet	85.17	90.79	87.84	93.64	85.98	88.68 (± 3.51)
SegFast-Mobile	85.12	89.24	86.51	93.48	83.81	87.63 (± 3.84)
SegFast-Basic	80.45	86.31	82.69	91.65	82.10	84.64 (± 4.46)
SegFast	83.48	85.08	83.71	93.58	81.63	85.50 (± 4.68)
SegFast-V2_3	81.46	85.95	84.13	93.48	82.10	85.42 (± 4.84)
SegFast-V2_5	80.92	86.70	83.86	93.63	82.70	85.56 (± 4.97)

Table 5 Mean intersection over union

Architectures	BDD	CamVid	CityScape	Synthia	Mapillary	Mean (± SD)
FCN	0.3076	0.3342	0.3503	0.5346	0.1910	0.3435 (± 0.1237)
SegNet	0.3475	0.5174	0.3702	0.5430	0.2234	0.4003 (± 0.1314)
UNet	0.3473	0.5003	0.3970	0.6767	0.2527	0.4348 (± 0.1621)
PSPNet	0.4099	0.5632	0.4190	0.6040	0.2739	0.4540 (± 0.1323)
SegFast-Mobile	0.3773	0.5030	0.3810	0.5810	0.2420	0.4169 (± 0.1302)
SegFast-Basic	0.3434	0.4431	0.3210	0.5317	0.2206	0.3720 (± 0.1193)
SegFast	0.3576	0.5000	0.3400	0.5910	0.2108	0.3999 (± 0.1481)
SegFast-V2_3	0.3360	0.4236	0.3500	0.5730	0.2160	0.3797 (± 0.1312)
SegFast-V2_5	0.3390	0.5012	0.3471	0.5919	0.2218	0.4002 (± 0.1461)

the "SegFast-V2_3" is faster. Moreover, in scenarios where memory consumption is of prime importance, the "SegFast" implementation is the network of choice. Even with the lowest quality GPUs, real-time performance can be achieved as the memory footprint of the smallest model is only about 700 MBs. This idea of network compression can also be carried out with other state of the art networks, such as SegNet, UNet, and PSPNet.

Acknowledgements This work is sponsored by SERB (Government of India, order no. SB/S3/EECE/054/2016) (dated 25/11/2016), and carried out at the Centre for Microprocessor Application for Training Education and Research, CSE Department, Jadavpur University.

References

1. Badrinarayanan V, Kendall A, Cipolla R (2017) Segnet: a deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans Pattern Anal Machine Intell 39(12):2481–2495
2. Brostow GJ, Fauqueur J, Cipolla R (2009) Semantic object classes in video: a high-definition ground truth database. Pattern Recognit Lett 30(2):88–97
3. Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL (2018) Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans Pattern Anal Mach Intell 40(4):834–848
4. Chollet F (2017) Xception: deep learning with depthwise separable convolutions, pp 1610–02357. arXiv preprint
5. Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S, Schiele B (2016) The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3213–3223
6. Dahl GE, Sainath TN, Hinton GE (2013) Improving deep neural networks for lvcsr using rectified linear units and dropout. In: 2013 IEEE international conference on acoustics, speech and signal processing, pp 8609–8613. IEEE
7. Haykin S (1994) Neural networks: a comprehensive foundation. Prentice Hall PTR, Upper Saddle River
8. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
9. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861
10. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: CVPR, vol. 1, p 3
11. Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K (2016) Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 0.5mb model size. arXiv preprint arXiv:1602.07360
12. Jégou S, Drozdzal M, Vazquez D, Romero A, Bengio Y (2017) The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In: 2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW), pp. 1175–1183. IEEE
13. Kaiser L, Gomez AN, Chollet F (2017) Depthwise separable convolutions for neural machine translation. arXiv preprint arXiv:1706.03059

14. Lin G, Milan A, Shen C, Reid I (2017) Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In: IEEE conference on computer vision and pattern recognition (CVPR)
15. Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3431–3440
16. Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp 807–814
17. Neuhold G, Ollmann T, Bulò SR, Kontschieder P (2017) The mapillary vistas dataset for semantic understanding of street scenes. In: ICCV, pp 5000–5009
18. Pal A, Jaiswal S, Ghosh S, Das N, Nasipuri M (2018) Segfast : a faster squeezenet based semantic image segmentation technique using depth-wise separable convolutions. In: Proceedings of the 11th Indian conference on computer vision, graphics and image processing (ICVGIP 2018), p 7. ACM
19. Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention. Springer, New York, pp 234–241
20. Ros G, Sellart L, Materzynska J, Vazquez D, Lopez AM (2016) The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3234–3243
21. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv :1409.1556
22. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2818–2826
23. Xu H, Gao Y, Yu F, Darrell T (2017) End-to-end learning of driving models from large-scale video datasets. arXiv preprint
24. Zhao H, Shi J, Qi X, Wang X, Jia J (2017) Pyramid scene parsing network. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 2881–2890