SegFast : A Faster SqueezeNet based Semantic Image Segmentation Technique using Depth-wise Separable Convolutions

Anisha Pal
Dept. of Information and Communication Technology
Manipal Institute of Technology
Manipal, Karnataka
anishapal006@gmail.com

Shourya Jaiswal
Dept. of Electronics and Communication Engineering
Manipal Institute of Technology
Manipal, Karnataka
shourya98@gmail.com

Swarnendu Ghosh
Dept. of Computer Science and Engineering
Jadavpur University
Kolkata, West Bengal
swarbir@gmail.com

Nibaran Das
Dept. of Computer Science and Engineering
Jadavpur University
Kolkata, West Bengal
nibaran.das@jadavpuruniversity.in

Mita Nasipuri
Dept. of Computer Science and Engineering
Jadavpur University
Kolkata, West Bengal
mita.nasipuri@jadavpuruniversity.in

ABSTRACT

Recent trends in image segmentation algorithms have shown various large scale networks with impressive performance for natural scene images. However most of the networks come with costly overheads such has large memory requirements or dependence on huge number of parallel processing units. In most cases costly graphics processing units or GPUs are used to boost computational capability. However for creating products in the real world we need to consider speed, performance as well as cost of deployment. We propose a novel "spark" module which is a combination of the "fire" module of SqueezeNet and depth-wise separable convolutions. Along with this modified SqueezeNet as an encoder we also propose the use of depth-wise separable transposed convolution for a decoder. The resultant encoder-decoder network has approximately 49 times lesser number of parameters than SegNet and almost 223 times lesser number of parameters than fully convolutional networks(FCN). Even in a CPU the network completes a forward pass for a single sample in approximately 0.39 seconds which is almost 5.1 times faster as compared to SegNet and almost 8.7 times faster compared to FCN.

CCS CONCEPTS

• **Computing methodologies → Image segmentation**; **Neural networks**;

ICVGIP 2018, December 18–22, 2018, Hyderabad, India

ACM ISBN 978-1-4503-6615-1/18/12…$15.00
https://doi.org/10.1145/3293353.3293406

KEYWORDS

Deep Learning, Image Segmentation, Squeeze Net, Depthwise separable convolution.

ACM Reference Format:
Anisha Pal, Shourya Jaiswal, Swarnendu Ghosh, Nibaran Das, and Mita Nasipuri. 2018. SegFast : A Faster SqueezeNet based Semantic Image Segmentation Technique using Depth-wise Separable Convolutions . In *11th Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP 2018), December 18–22, 2018, Hyderabad, India.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3293353.3293406

1 INTRODUCTION

The last few decades have seen great advancement in the field of computer vision. Problems such as object classification, detection, segmentation have experienced tremendous progress since the introduction of deep learning algorithms [5, 17, 20, 25]. Traditional vision based techniques relied on inaccurate hand crafted features[15, 23]. The discovery of *c*onvolutional *n*eural *n*etworks (CNN) [12] has played a pivotal role in the great success achieved in the field of computer vision. Recent developments in the field of *g*raphics *p*rocessing *u*nits(GPU) fueled the progress in terms of deeper and more complicated networks. But lately most of the state of the art networks have become heavily dependent on high-end costly GPUs. This increases the deployment cost of real life systems. The proposed work aims perform semantic image segmentation for autonomous driving systems under constrained computational environment. The proposed CNN based architecture provides good performance while being considerably smaller and faster compared to some state of the art algorithms[1, 14]. Other than route planning in self driving cars, semantic segmentation has various other applications such as traffic surveillance, obstacle detection, scene understanding and so on. Several recent semantic segmentation algorithms[1, 3, 13, 14, 18, 26] have been successful in producing remarkable results in terms of accuracy in object detection and finesse of the segmentation maps. But when it comes to speed and memory requirements the results are not so

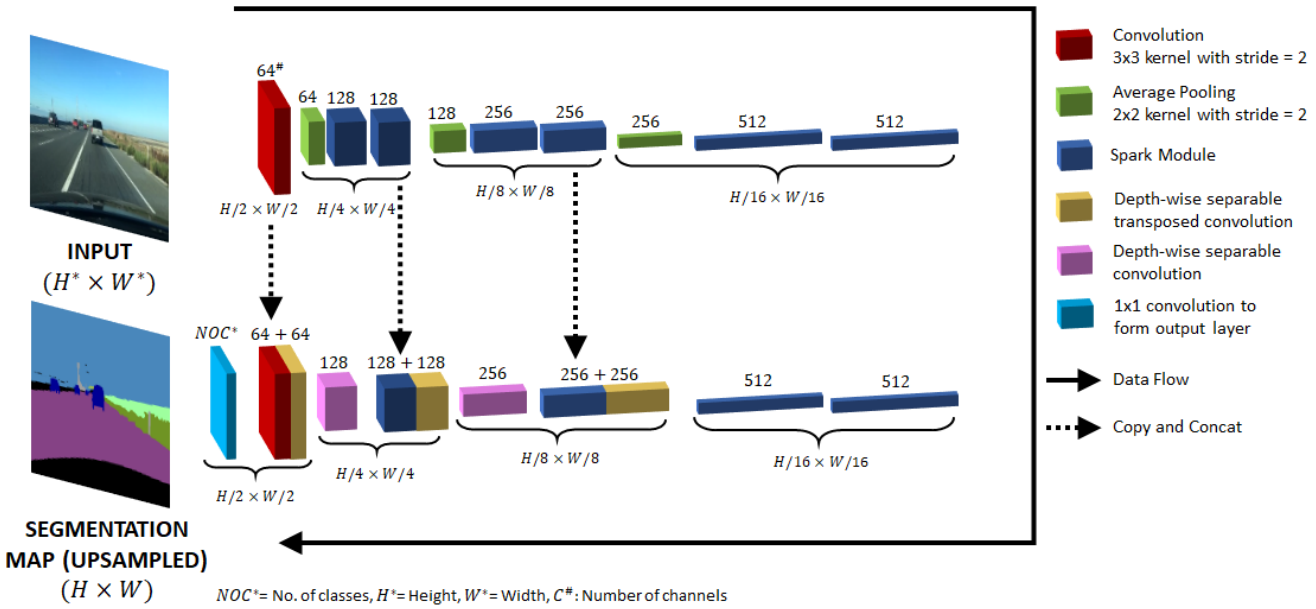

Figure 1: Network Architecture

impressive. Most of these algorithms require high end GPUs for their deployment and cannot be supported by low end devices. For development of consumer level real time applications based on image segmentation, both time of execution and cost of deployment are important factors. With time there is bound to be faster and cheaper hardware but at this point most car companies that utilize AI based driving systems come at expensive prices. Many budget cars are unable to utilize such systems and hence the mass population cannot access this technology. AI based driving systems will benefit the most if most of the cars in societies are aided by AI as this would result in much lesser traffic issues that occur due to improper driving techniques. This is especially true for countries with huge population like India where majority of vehicles belong to a low-budget category and also the traffic is quite rough. Even for human drivers, accidents occur mainly due to two factors, one is lack of recognition of an accidental situation, and the second one is a late response time when an accidental situation is recognized. Though accuracy is needed to avoid accidental situations, yet once such a situation is recognized the system needs to respond swiftly to avoid the accident. Hence a balance between speed and accuracy is needed. The proposed network aims to tackle the problems of execution time and memory requirement while achieving the similar level of performance. The system has been tested on four autonomous driving datasets namely BDD[24], CamVid[2], Synthia[19] and Mapillary[16]. Many segmentation algorithms like SegNet[1] or U-Net[18] draw inspiration from auto-encoders. They consist of an encoder or feature extractor followed by a decoder to generate pixel-level probability distributions. The encoder part of a network architecture plays a very important role in deciding the time complexity of a network. The encoder of the proposed network is based on SqueezeNet[9], a faster and less computationally expensive version of AlexNet[21]. However we have further condensed the SqueezeNet architecture by replacing regular convolutions of the fire modules with depth wise separable convolutions[4, 11]. This upgraded fire module is referred to as a "spark" module. In case of the decoder we have also proposed the use of depth-wise separable transposed convolutions to reduce the number of parameters. This results in reduction of the size and an increase in the speed of the network in comparison to other state of the art approaches without compromising much on the accuracy. In the following section we have given a brief description of the different segmentation algorithms and previous work in this field that has served as an inspiration for our network. In section 3, a detailed description of the proposed network has been provided along with descriptions of all the modules. Then experimentations and results have been discussed in section 4 and finally in section 5 we provide conclusion and pointers to future works.

2 RELATED WORK

Previously the quality of hand-crafted features were main influence behind performance of neural networks. However, using these techniques, problems like semantic segmentation for natural scene images proved to be a much more difficult task due to the complexity of underlying features. With introduction of deep learning, semantic segmentation algorithms evolved and employed various techniques for better results. The convolutional feature extractor of a VGG-Net[22] combined with a pixel level classifier resulted in a Fully Convolutional Network [14]. It resulted in more dynamic and time efficient algorithm but produced coarse segmentation maps due to the pooling layers. Later different algorithms [1, 18] employed the use of an encoder-decoder architecture while combining features from different layers of

the network to achieve better segmentation results. The encoders are essentially a series of convolutional layers followed by some downsampling operation like max-pooling or average pooling. The decoders, on the other hand, were composed upsampling layers like transposed convolutions or unpooling. This architecture improved the resolution of the segmentation map. Other algorithms used methods such as dilated convolution[3], spatial pooling pyramids[26], multi-level refinements[13] in their classifier to increase the receptive field of the neural network and improve the resolution. The different feature extractors or encoders used in the semantic segmentation algorithms also play a major role in the performance of segmentation algorithms. Some well known encoders used in semantic segmentation are based on VGGNet[22] and ResNet[6]. ResNet is better than VGGNet in terms of accuracy and computational efficiency. The main attraction of ResNet is the use of skip connections which improves the gradient flow , allowing the network to train better. A few works have used encoders based on the 152 layer DenseNet architecture [8, 10]. Although it provides better accuracy, it is computationally very expensive. The proposed network takes inspiration from the encoder-decoder architectures like of SegNet[1] and employs the concepts of depthwise seperable convolution[4, 11] on a feature extractor based on the SqueezeNet[9] architecture. Depthwise Seperable convolution has also contributed to Google's MobileNet [7]. SqueezeNet is a classification algorithm developed as a better alternative to AlexNet. It has the same accuracy level as that of AlexNet but with reduced model size and fewer parameters.

3 METHODOLOGY

The proposed architecture takes inspiration from SqueezeNet[9], Depth-wise Separable Convolution[4, 11] and the encoder-decoder technique as demonstrated in SegNet[1]. The SqueezeNet architecture is characterized by its unique fire modules. The fire modules were built to reduce the nubmer of parameters by squeezing the number of channels and expanding them by multiple parallel differently sized kernels. In the proposed approach the fire modules are replaced with an even tinier version that is aptly referred to as "spark" modules in this work. The spark modules as described in section 3.1 implement the use of depth-wise separable convolutions instead of traditional convolutions. Depth-wise separable convolutions are later explained in section 3.2. This modified SqueezeNet with spark modules is used in an encoder-decoder architecture as shown in fig. 1. The encoder consists of 4 encoding blocks, the first encoding block consists of a convolution layer followed by ReLU activation and an average pooling. The second and third blocks are similar and involve two spark modules(refer section 3.1) followed by average pooling. The last block consists of four spark modules. For decoding architectures like SegNet use max-unpooling while forwarding pooling indices. Though this reduces computation, yet it result in loss of accuracy. In this work we have proposed the use of transposed depth-wise separable convolutions. Normally depth-wise separable convolutions have only been used for normal convolutions[4, 11]. By using this technique with transposed convolutions we can easily upsample features while using much lesser amount of parameters. In the decoder, the first decoding block applies a transposed depth-wise separable convolution(refer section 3.2) to the output of the last encoder. The feature maps of the decoder and the corresponding encoder having same size are then concatenated and the result is again convolved with a 3×3 depthwise separable convolution. The output layer consists of a pixel level probability distribution obtained using a 1×1 convolution. In the output segmentation mask each pixel is a number corresponding to the class for which the probability value for that particular pixel is maximum. The final output is upsampled to fit the original input size. In [1] it was mentioned that passing indices rather than the entire feature reduced the memory requirement of the network. However, it was also mentioned that this led to a loss in the accuracy. In our network the encoded feature maps have been passed to the decoder, however observations prove that the number of parameters for our network is significantly lesser as compared to SegNet with better or comparable accuracy across multiple datasets. The proposed model achieves a fair trade off between accuracy and computational complexity. The concepts used in our network have been explained in detail in the different subsections below.

3.1 Spark Module: A Tiny Fire Module

One of the key attractions of SqueezeNet is the fire module.Fire module employs a few strategies such as decreasing the number of 3×3 filters and replacing 3×3 filters with 1×1 filters. Both of these strategies have been employed to decrease the number of parameters. The replacement of 3×3 filters with 1×1 filters is based on the fact that the number of parameters of a 3×3 filter is 9 times more than that of a 1×1 filter. A fire module consists of two separate layers, a squeeze layer and an expand layer. The main objective of the squeeze layer is to decrease the number of input channels for the 3×3 filters in the expand layer. The expand layer consists of a mix of 1×1 filters and 3×3 filters. The three tunable hyper-parameters in a fire module are the number of filters in the squeeze layer (S), number of 1×1 filters in the expand layer (E_1) and the number of 3×3 filters (E_3) in the expand layer. The relation between the three hyper-parameters is that number of filters in squeeze layer is less than the total number of filters in expand layer. In a fire module the input is first subjected to batch normalization followed by convolution operation. Then an element-wise rectified-linear non-linearity (ReLU) is applied. These operations were part of the squeeze layer. The squeeze layer is followed by the expand layer where the output of the squeeze layer is subjected to two separate convolution operations involving 1×1 kernels and 3×3 kernels. Similar to the squeeze layer each convolution operation is preceded by batch normalization and followed by ReLU operation. In our work, the fire module of SqueezeNet is further compressed by using depth-wise separable convolutions instead of the 3×3 convolutions in the expand layer E_3. Depth-wise separable convolutions are explained in the next sub-section 3.2 in details. This smaller fire module is referred to as a "spark" module henceforth. The spark model is demonstrated in fig. 2. In the diagram C is the number of channels of the input. S is the number of channels of the squeeze layer. E_1 and E_3 are the number of channels in the 1×1 and 3×3 expansion layers. Note that the number of channels of the output is only dependent on the expansion layers.

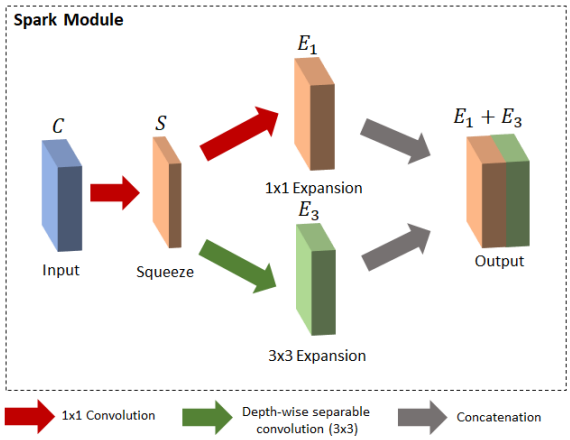

Figure 2: Spark Module

3.2 Depthwise Separable Convolution

Depthwise Separable convolution[4, 11] reduces the computational complexity and the number of parameters of the network hence increasing the efficiency. In depth-wise separable convolution two separate operations are performed, first spatial convolution operation is performed independently over each channel of the input sharing the same 1 dimensional kernel over all the channels, this ensures that the number of input channels is equal to the number of output channels after this operation. This is followed by point-wise convolution with a 1×1 kernel, projecting the channels computed by the previous stage onto a new channel space. A convolutional kernel of shape $K_h \times K_w \times C$ needs $K_h * K_w * C$ number of parameters, where K_h and K_w are the height and width of the kernel and C is the number of channels of the input. An equivalent depth-wise separable convolution will need $K_h * K_w + C$ which is definitely lesser than the former for $(K_h, K_w, C) > 1$. The output size of a depth-wise transposed convolution is same as a normal convolution which is given by,

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

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 Depthwise Separable Convolution

Transposed depth-wise separable convolution can be visualized in manner similar to depth-wise separable convolution. Similar to depthwise separable convolutions, two separate operations are performed. The first operation involves performimg spatial transposed convolution independently over each channel of the input while sharing the same single channeled kernel over all the channels of the input. This is followed by pointwise convolution with 1×1 kernel over all the channels of the output obtained after the first operation. The reduction in the number of parameters is similar to depth-wise separable convolutions. The output size is also same as a traditional transposed convolution which can be written 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 same as equation 1. The choice of strides can be adjusted to increase the scale of the input by a required factor. Transposed convolutions are hence mainly used for decoders where it is needed to upsample the activations to obtain sharper features.

4 EXPERIMENTATION AND RESULTS

The goal of the current work is to propose an approach for fast semantic image segmentation for autonomous driving problems.

4.1 Dataset Description

Since our main objective is to propose a model that can be used for real-time applications such as autonomous driving, we have trained the models on different autonomous driving datasets available and compared the results. All our training and validation images have the resized dimension of 224×224 maintain an uniformity across various experimentations. Data augmentation of any form has not been performed. We have used four different datasets namely, BDD[24], CamVid[2], Synthia [19] and Mapillary [16]. The description of the datasets have been summarized in table 1.

Table 1: Details of the used datasets

Dataset	Number of Classes	Training Samples	Validation Samples
BDD [24]	20	7,000	1,000
CamVid [2]	12	367	101
Mapillary [16]	66	18,000	2,000
Synthia [19]	13	10,786	2,621

4.2 Experimental Setup

As a control to the proposed methods we have performed the same experiments on an FCN-8s[14] and a SegNet-Basic[1] model. To establish the influence of depthwise separable convolutions we have also compared our proposed "SegFast" network along with an equivalent basic version which has normal convolution and transposed convolution instead of their depth-wise separable variant. This basic network is referred to as "SegFast-Basic". This network serves as a control for analysis of the impact of depth-wise separable convolutions and transposed convolutions. For our network we have used Adam Optimization with the initial learning rate set to 0.001, $\beta 1$ set to 0.9, $\beta 2$ set to 0.999 and ϵ set to 1e-8. For our loss function we have used negative log likelihood loss.The final output of the network has been passed through a log-softmax layer before applying the loss function. Our network was trained

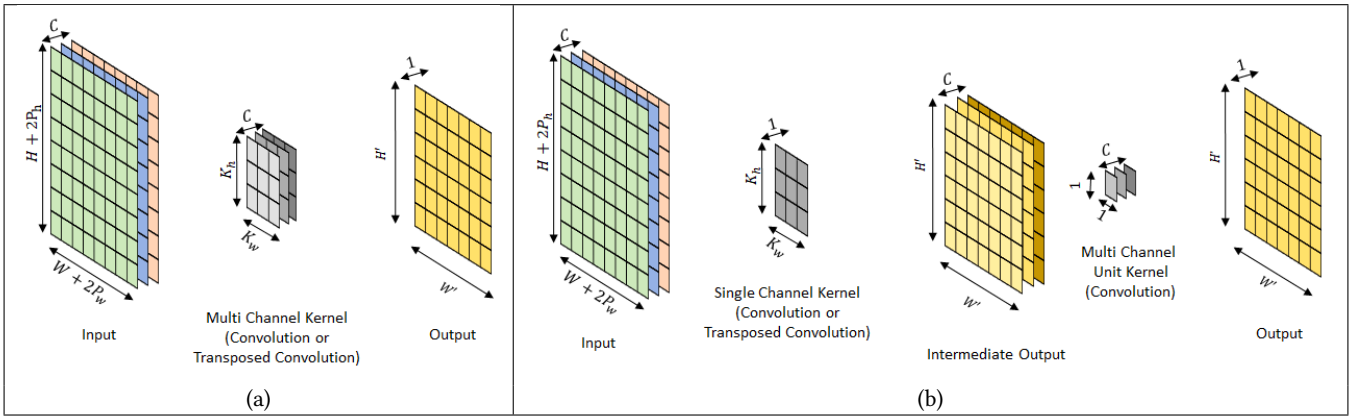

Figure 3: Depthwise separable convolutions or transposed convolutions

and tested on Nvidia Quadro P5000 with 16 GB RAM and 2560 cuda cores. The CPU of the system used was Intel(R) Xeon(R) CPU E5-2623 v4 with a 2.60 GHz processor having 8 cores, 16 threads and RAM of size 128 GB.The model was mainly trained and tested using GPU, only for time calculation the model was run on CPU to emphasize the speed of training and testing in constrained environment. The number of epochs was set to 200 and the network was trained on all the four datasets mentioned in Dataset Description. In order to find the best model we have used the method of early stopping on the validation set and the corresponding validation accuracy has been reported across all methods and datasets.

4.3 Observations

In terms of performance pixel level accuracy(table 2) and mean intersection over union (mIOU)(table 3) has been reported. Additionally, time of computation has also been expressed. The time for doing a forward and backward pass effectively indicates the testing and training time of the network. The times reported in table 4 was measured in a constrained CPU-only environment. The available GPU that is NVIDIA Quadro P5000 is too powerful to show any significant differences however when run on a CPU with only 16 threads the speed of the proposed network is quite noticable. Also the number of parameters are also computed and shown in table 5.

Table 2: Pixel Level Accuracies for the proposed network versus state of the art approaches

Dataset	FCN	SegNet	SegFast-Basic	SegFast
BDD	75.31 %	82.79 %	80.45 %	**83.48** %
CamVid	74.22 %	**88.70** %	86.31 %	85.08 %
Mapillary	79.62 %	**82.46** %	82.10 %	81.63 %
Synthia	91.81 %	92.05 %	91.65 %	**93.58** %

4.3.1 Performance. The proposed network was the best performer in case of BDD and Synthia both in terms of pixel level accuracy and mIOU. In the other datasets SegNet performed slightly better as compared to the proposed network. When compared with the SegFast-Basic variant with normal convolutions the SegFast algorithm stood out for BDD and Synthia in terms of both pixel level accuracy and mIOU and in CamVid dataset when measured in terms of only mIOU. In case of CamVid dataset the number of training images is very less whereas in case of Mapillary dataset the number of classes are significantly high at 66. Our experimental observations have led us to believe that our network performs better when provided with datasets which have a healthy balance between the number of classes and volume of training data that is the performance of the proposed approach degrades with lower volume of training data and higher number of classes. In the current system the accuracy falls primarily because of the huge decrease in the number of neurons. Hence to improve upon the current network one needs to increase its depth and width to learn more features but it will be less efficient in terms of speed. The results are shown in full details in table 2 and 3.

Table 3: Mean intersection over union(mIOU) for the proposed network versus state of the art approaches

Dataset	FCN	SegNet	SegFast-Basic	SegFast
BDD	0.3076	0.3475	0.3434	**0.3576**
CamVid	0.3342	**0.5174**	0.4431	0.5000
Mapillary	0.1910	**0.2234**	0.2206	0.2108
Synthia	0.5346	0.5430	0.5317	**0.5910**

Table 4: Forward pass and Backward pass time per sample image under CPU-only environment

Dataset	Forward Pass	Backward Pass
FCN	3.385	5.230
SegNet	1.977	3.251
SegFast-Basic	0.419	1.336
SegFast	**0.389**	**0.929**

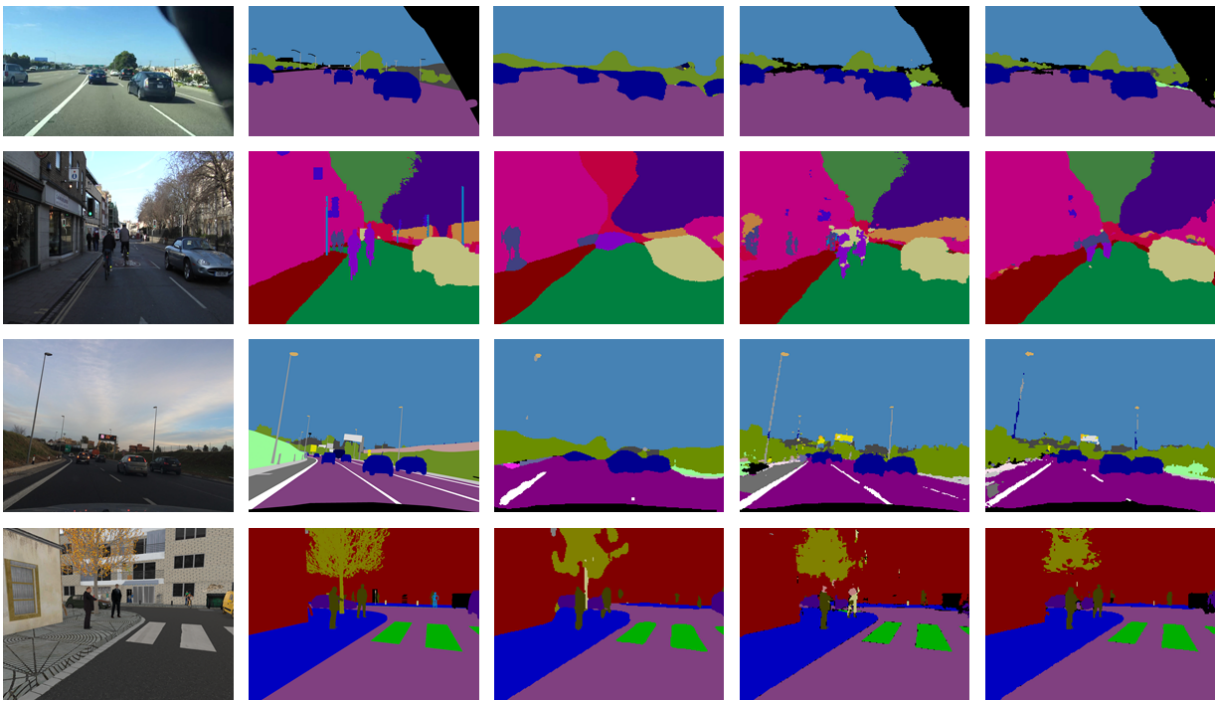

Figure 4: Output segmentation for samples from various datasets. Rows(From top to bottom):BDD, CamVid, Mapillary and Synthia datasets, Columns(From left to right): Original Image, Ground Truth, FCN-8s output, SegNet output, SegFast(proposed network) output.

4.3.2 Speed of Execution. The speed of execution denotes the time taken by a network to complete a forward or backward pass with a single sample under a CPU-only environment. The CPU used during the experiment was a Intel(R) Xeon(R) CPU E5-2623 v4 with a 2.60 GHz processor having 8 cores, 16 threads. Though networks were primarily trained in GPU, the execution time was calculated in a CPU-only environment to demonstrate the ability of the networks to perform under limited computational resource. Even with only 16 threads both the SegFast and SegFast-Basic network can analyse a sample and provide output in less than half a second. Hence it can be said that even with low-end GPUs this network can be used for real-time applications. The SegFast network is also faster than the SegFast-Basic variant which proves that the use of depth-wise separable convolutions have increased the speed of the network. During test time the SegFast network is almost 5.1 times faster than SegNet and 8.7 times faster than FCN. The main difference in architecture with respect to SegNet is in the number of convolutional layers. SegNet starts with two convolutional layers on input image where as we straight away use a strided convolution to reduce the size of the activation map. Furthermore, the SegNet encoder uses a standard VGG 16 architecture with two convolutional blocks with two layers followed by 3 convolutional blocks with 3 layers each. We in turn use only 4 convolutional blocks (as compared to a total of 5 in SegNet) with 1 layer in the first block and 2 layers in the successive 3 blocks. Table 4 shows the forward and backward pass times of all the networks.

Table 5: Number of parameters and memory requirements of various networks

Dataset	Parameters	Memory (MiB)
FCN	134,507,010	3447
SegNet	29,455,702	1515
SegFast-Basic	2,965,976	1232
SegFast	**603,288**	**703**

4.3.3 Number of parameters and memory requirements. The proposed network also contains far less number of parameters compared to its competitors. The SegFast network has almost 223 times lesser number of parameters than FCN and almost 49 times lesser number of parameters as compared to SegNet. The use of depth-wise convolutions results in almost 5 times lesser number of parameters. The proposed network occupies only around 703 MiB of space in the GPU while handling a single input of size 224×224. The number of parameters and memory requirements of all the networks are shown in table 5.

4.4 Result and Analysis

The main feature of our model as can be deduced through table4 and 5 is high speed and low memory requirement respectively. Due to the above mentioned features of the algorithm it can be deployed with ease on low end processors based devices. It is about

5.1 times faster than SegNet and about 8.7 times faster than FCN. The number of parameters as shown in table 5 is about 223 times lesser than FCN and about 49 times lesser than that of SegNet. Our model outperforms SegNet and FCN in terms of speed, number of parameters and memory usage. Segmentation results provided in table 2 and 3 in terms of pixel level accuracy and mIOU indicate that our model outperforms FCN in all of the test cases. With respect to SegNet, the proposed network provides better results for BDD and Synthia datasets and gives comparable results for CamVid and Mapillary. We have also tabulated the results of the proposed network(SegFast) against a SegFast-Basic variant which uses normal convolutions and transposed convolutions instead of depthwise seperable convolutions and transposed convolutions. The use of depth-wise separable convolutions provided comparable or better performance while reducing the number of parameters almost by a factor of 5. The outputs of the proposed SegFast network is showed in fig. 4 along with FCN and Segnet outputs. The proposed network achieves a sharper segmentation boundary as compared to FCN which is almost comparable to SegNet. In some cases the proposed network even manages to avoid some of the noisy predictions by SegNet.

5 CONCLUSION

With advances in autonomous driving technology it is very necessary to create efficient networks to promote the development of consumer level systems with good price to performance ratio. With that in mind we have proposed a encoder decoder network for semantic image segmentation which fast, small and uncompromising in terms of performance. We have proposed a novel "spark" module which is a combination of the fire module of SqueezeNet along with depth-wise separable convolutions. To make a typical encoder-decoder architecture smaller and faster we have used a SqueezeNet feature extractor with spark modules instead of fire modules as an encoder and proposed the use of depth-wise separable transposed convolutions for the decoder. The resultant network is much faster and smaller as compared to state-of-the-art approaches like SegNet or FCN while providing hardly any trade-off in terms of performance. In a hugely parallelized computing environment of a GPU the difference in time consumption would be negligible, however the goal of the current work is to promote a method that can be employed in a much more constricted environment. Hence, we have focused to demonstrate the efficiency in a CPU-only environment, so that the model can be used along with cheap hardware and be made available to a larger population under a lower budget. In future these methodologies can be further explored to compress deeper networks and achieve even better results.

ACKNOWLEDGMENT

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] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence* 39, 12 (2017), 2481–2495.

[2] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. 2009. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters* 30, 2 (2009), 88–97.

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2018. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence* 40, 4 (2018), 834–848.

[4] François Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint* (2017), 1610–02357.

[5] Li Deng. 2014. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing* 3 (2014).

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[7] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[8] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely Connected Convolutional Networks.. In *CVPR*, Vol. 1. 3.

[9] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360* (2016).

[10] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. 2017. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*. IEEE, 1175–1183.

[11] Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. 2017. Depthwise separable convolutions for neural machine translation. *arXiv preprint arXiv:1706.03059* (2017).

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[13] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. 2017. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3431–3440.

[15] David G Lowe. 1999. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, Vol. 2. Ieee, 1150–1157.

[16] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. 2017. The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes.. In *ICCV*. 5000–5009.

[17] Nikhil R Pal and Sankar K Pal. 1993. A review on image segmentation techniques. *Pattern recognition* 26, 9 (1993), 1277–1294.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

[19] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. 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*. 3234–3243.

[20] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks* 61 (2015), 85–117.

[21] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[22] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[23] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. 2013. Selective search for object recognition. *International journal of computer vision* 104, 2 (2013), 154–171.

[24] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. 2017. End-to-end learning of driving models from large-scale video datasets. *arXiv preprint* (2017).

[25] Alper Yilmaz, Omar Javed, and Mubarak Shah. 2006. Object tracking: A survey. *Acm computing surveys (CSUR)* 38, 4 (2006), 13.

[26] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. 2017. Pyramid scene parsing network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2881–2890.