Efficient Semantic Segmentation Decoder for Embedded Systems

Arindam Das, Saranya Kandan, Senthil Yogamani

Efficient Semantic Segmentation Decoder for Embedded Systems

Arindam Das[1], Saranya Kandan[2], Senthil Yogamani[3]

[1]Detection Vision Systems, Valeo India, [2]Detection Vision Systems, Valeo India, [3]Valeo Vision Systems, Valeo Ireland

{arindam.das, saranya.kandan, senthil.yogamani}@valeo.com

***Abstract**: Semantic segmentation remains a computationally intensive algorithm for embedded deployment even with the rapid growth of computation power. Thus efficient network design is a critical aspect especially for applications like automated driving which requires real-time performance. Recently, there has been a lot of research on designing efficient encoders that are mostly task agnostic. Unlike image classification and bounding box object detection tasks, decoders are computationally expensive for semantic segmentation task. In this work, we focus on efficient design of the segmentation decoder making use of an efficient encoder. We design a novel efficient non-bottleneck layer and a family of decoders which fit into a small run-time budget using VGG10 as efficient encoder. We demonstrate in our dataset that experimentation with various design choices led to an improvement of 10 % from a baseline performance. The optimal configuration of the decoder leads to a performance of 50 fps on an embedded low power SOC namely Renesas V3H.*

Keywords: *Semantic Segmentation, Visual Perception, Embedded Systems, Automated Driving.*

INTRODUCTION

Semantic segmentation provides complete semantic scene understanding wherein each pixel in an image is assigned a class label. It has applications in various fields including automated driving,augmented reality and medical image processing. The complexity of Convolution Neural Networks(CNN) architectures have been growing consistently [16] [17] [18] [19] [20] [21]. However for industrial applications, there is a computational bound because of limited resources on embedded platforms. It is essential to design efficient models which fit the run-time budget of the system. There are many papers which demonstrate large runtime improvements with minimal loss of accuracy by using various techniques. An overview of efficient CNN for semantic segmentation is provided in [1]. Majority of the work is focused on efficient encoder design and there is limited work on efficient decoder design. Wojna etal [12] recently demonstrated that decoder design plays a critical role in accuracy. In this paper, we propose the design of a novel non-bottleneck layer particularly to perform semantic segmentation task where the encoder is task independent unlike existing methods. Our non-bottleneck layer based on residual learning, has been designed to perform well for some classes that are not well represented in the dataset. Having cascaded skip connections make our non-bottleneck layer capable to handle high gradient flow and suitable for an embedded platform to run on real time with lightweight encoder model.

PROPOSED METHOD

Encoder: Efficiency of semantic segmentation is considered as seen by heavy usage of dilated convolution as used in [2] [3] [5] [8] [11] or information fusion at different resolution [7] [14] [13].However, our study is intended towards use of a more generic encoder. So we designed a task independent VGG [10] style classifier of 10 layers as encoder. As per Figure 2, convolution with stride 2 followed by max-pooling is used to reduce the problem space, thus reducing the number of hyperparameters and run-time. Obviously, this is a trade-off for segmentation accuracy, but it is not the case for other tasks like detection, classification etc. Considering this encoder to be function independent, this gap of spatial information exploration needs extensive semantic feature learning in the decoder. Convolution layers are added sequentially along with increasing width and decreasing feature space in regular interval. All the convolution layers use kernel of size 5X5 followed by Batch-Normalization to speed up the convergence and ReLU to add non-linearity. The last layer of encoder

produces total 256 feature maps.

Decoder: The proposed architecture of decoder has been obtained after redesigning the existing features of the convolutional nets, such as residual learning [6] and non-bottleneck layers [8]. In the recent past, learning through residual blocks has shown breakthrough performance in many vision related tasks and made it possible to make the network grow more deeper very easily. This advancement helped to outperform significantly in many object recognition tasks. Further the same learning strategy has also been used for semantic segmentation as in [8]. However, in [4], it has been demonstrated that residual learning for a network with lesser depth is not efficient, because the network can not handle high gradient flow during back-propagation. To circumvent this issue, in this study, the original residual learning [6] strategy has been modified with an adequate arrangement to distribute the high gradients through multiple skip connections. Our encoder design does not employ any mechanism to extract semantic features and also the current trend of decoders using few deconvolution layer for reconstruction of segmentation output is not sufficient with such encoder which has generic non-semantic knowledge. Thus it becomes a requisite for the decoder to learn semantic information from the encoded feature maps. Hence we use non-bottleneck blocks between two deconvolution layers. In [8], the authors have designed non-bottleneck layer which is 1D in nature and claimed to be a better regularizer and also faster.

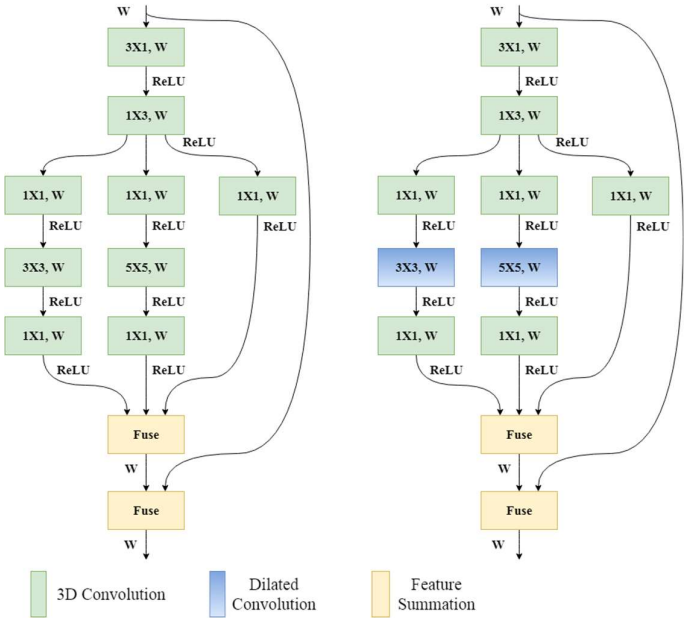


Figure 1: Illustration of non-bottleneck layers

The design of our non-bottleneck layer is shown in Figure 1. It contains both 1D and 3D kernels.1D kernel is used to extract information mainly in one direction at a time and 3D kernel is forgathering features from bigger receptive area. Later we try to look for dense information through multiple kernels with different sizes for example 3X3, 5X5 and 1X1 respectively. Following this, the features extracted using different kernels are fused to summarize the semantic features from different receptive areas and also with the input features to the same non-bottleneck layer. The multiple skip connections to the feature fusion blocks in the proposed non-bottleneck layer help to handle high gradient flow because during back-propagation the incoming gradient gets distributed among all paths. As per Figure 1, our non-bottleneck layer has two variants that are type-1 (left) and type-2(right). The only difference between two variants is the block at the right uses dilated convolution for 3X3 and 5X5 kernels. Dilated convolution helps to extract spatial information by expanding the field-of-view as per the dilation factor while maintaining the same resolution. With increased dilation

rate it is possible to achieve better accuracy owing to expanded receptive field, but this becomes computationally expensive. Thus we use only dilation rate 1 in our study to target embedded platform run-time constraints. After each convolution operation, ReLU is used as activation unit for better convergence.

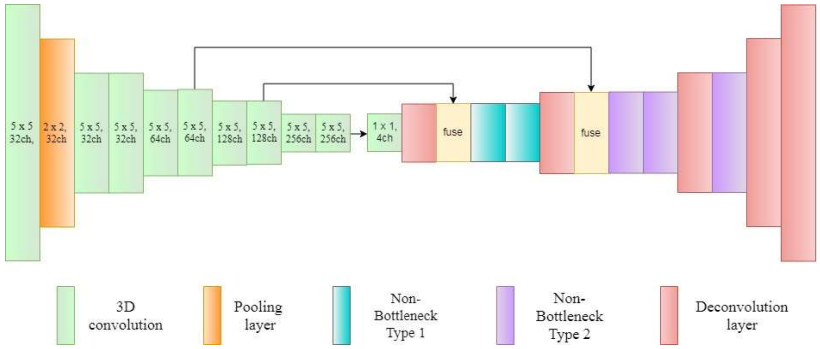
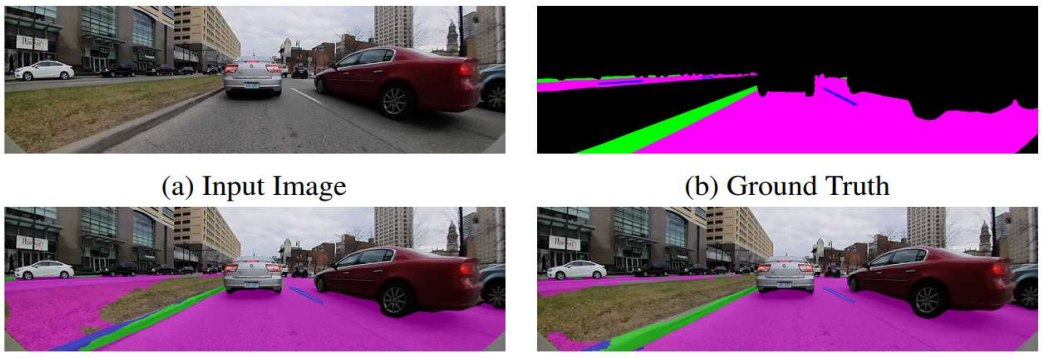


Figure 2: CNN based encoder-decoder architecture used for semantic segmentation task

Encoder-Decoder architecture: Figure 2 shows the encoder-decoder pair that is used in this work.The encoder is a very generic one and the decoder is our main proposal. The feature maps passed from the encoder are downsampled from 256 to 4 as the present study concentrates on 4 classes. One can argue that drastic change in the number of feature maps such as this can impact on accuracy but if we make the decoder wider it will shoot up the runtime significantly. Thus decreasing the number of feature maps in regular interval is not affordable and out of our budget. After first deconvolution layer, non-bottleneck layer of type-1 is used twice. Following second and third deconvolution layers,non-bottleneck layers of type-2 are used twice and once respectively. There is no non-bottleneck layer used after fourth deconvolution. This arrangement of repetitive usage of non-bottleneck layers(both type-1 and type-2) will help remove the gap of learning semantic information in the encoder side. Hence, our proposed non-bottleneck layer is generic and can be used for any segmentation task.Additionally, we pulled intermediate feature maps from the second and third last convolution layers of the encoder.



(a) Input Image (b) Ground Truth

(c) FCN Output (d) Proposed Decoder Output

Figure 3: Comparison of proposed optimal decoder against a standard FCN decoder

EXPERIMENTS

Dataset: We have used dataset that is owned by our organization. However, some samples of the dataset are shown in Figure 4, where dimension of each image is 1280X384. Samples are highly varied in nature and mostly urban scenes. Diverse lighting conditions and presence of dense shadow make this dataset challenging. In the present study, to perform semantic segmentation, we

concentrated only on 4 critical classes that are lanes, curb, road and void (everything else). The entire dataset is divided into Training, Validation and Test set each containing 3016, 981 and 1002 images respectively.Corresponding each sample image, single channel annotation was developed where each pixel has only one class label.

Training: First we train our encoder from scratch on ImageNet and then transfer the weights for the present task. The pre-training on a much larger dataset was required because our model is quite shallow and the dataset used in this work is very small in size. To make our lightweight encoder more robust we could follow the concept of layer-wise training in a supervised fashion as reported in [9].Implementation of the proposed network and all experiments are executed using Keras framework.We considered very popular Adam as optimizer. Regarding the other network configuration, weight decay and batch size were set to 0.9 and 4 respectively. Training was started with 0.0005 as initial learning rate including standard polynomial decay strategy to decrease this value over 350K iterations. Dropout is not used in our model. For all experiments, we used NVIDIA Titan X 12G GPU with 24GB RAM. No data augmentation technique has been performed during training.

Experimental results and comparison study: The hardware that we use is designed with automotive power constraints in mind, thus having restricted number of features for design of a CNN. Also we intend to utilize a generic encoder network that stays well within the budget.Thus our main objective is to design an efficient decoder that satisfies both these constraints. In the course of design of an efficient decoder, we have experimented multiple versions of decoder all of which are explained later in this section. With all these decoders VGG10 pre-trained with Imagenet is used as the encoder, to have fair comparison of the different variants of the decoder. The entire network containing the encoder along with different decoders is trained end to end with the available pixel wise ground truth label. All these variants of decoder reported in Table 1, fits all our constraints. It is quite general while capturing an urban scene, there will be very limited region occupied by lanes, curbs but it is exactly opposite for roads and void classes. So we clearly see that for effective learning there is a huge gap in problem space in these two classes while comparing with other. Without even attempting any data augmentation and class weighing technique, our non-bottleneck layers worked better for curb though there is a slight deterioration for lanes. To evaluate the segmentation performance on all the designed decoders, widely used Intersection over Union (IoU) metric is considered and details are furnished in Table 1.

As put forth earlier, we did experiments with several combinations of Non-Bottleneck layers in the network whose results are updated in Table 1. Decoder D1 uses our proposed non-bottleneck layer without 1X1 convolution after 3X3 and 5X5 convolution. Decoder D2 is same as D1 but it does not use second skip connection from encoder. Decoder D3 shares the same configuration as D2 but the batch size during training was 8 whereas it was 4 for D2. Decoder D4 is same as D3 but it does not use 1X1 convolution before 3X3 and 5X5 convolution. Decoder D5 is a bit different. After first deconvolution layer two sets of 3X1, 1X3 convolutions followed by ReLU is used. Also it uses skip connection to fuse the resultant features with the input feature maps of first 3X1 convolution.

Table 1: Results of different variants of our decoder with non-bottleneck layer and VGG10 as encoder for semantic segmentation on our dataset

Decoder configuration	Avg. class accuracy			Avg. class IoU score				Runtime(ms)
	Lanes	Curb	Road	Lanes	Curb	Road	Mean	
D1	0.5678	0.6228	0.9688	0.4943	0.4296	0.9265	0.7255	-
D2	0.4363	0.6082	**0.9694**	0.4021	0.4445	0.9198	0.6989	-
D3	0.5096	0.5682	0.9689	0.4333	0.4264	0.9254	0.7038	-
D4	0.4769	0.5932	0.9684	0.4152	0.4534	0.9248	0.7013	-
D5	0.4561	0.6059	0.9668	0.4013	0.403	0.9138	0.6866	-
D6	0.5775	0.6539	0.9566	0.4567	0.4198	0.9154	0.7106	-
D7	0.5263	0.4284	0.9464	0.4192	0.2911	0.8689	0.6458	-
D8	0.5628	**0.6755**	0.9686	0.4752	0.4717	0.9296	0.727	-
(2N1)(2N1)(2N2)(2N2)	0.6054	0.6534	0.9685	0.5294	**0.4755**	0.9299	0.742	27.36
(2N2)(2N2)(2N2)(2N2)	0.5951	0.647	0.9655	0.5061	0.5061	0.9267	0.7323	27.36
(2N2)(2N2)(2N2)	0.6	0.639	0.9661	0.5063	0.4391	0.9279	0.7315	20.37
(2N1)(2N1)(2N1)(2N1)	0.5743	0.6343	0.9691	0.4961	0.4486	0.9275	0.7284	27.36
(1N1-1N2)(1N1-1N2)(1N1-1N2)(1N1-1N2)	0.5938	0.6338	0.9681	0.5051	0.4438	0.9264	0.731	27.36
Optimal	**0.6118**	0.6588	0.9689	**0.5304**	0.4696	**0.9314**	**0.7441**	**19.37**

After second deconvolution layer, one 3X3 dilated convolution with dilation rate 1 is used and then the same non-bottleneck as used after first deconvolution layer. Only 3X3 dilated convolution with dilation rate 1 is used after third and fourth deconvolution layer. Decoder D6 is same as D5 butit uses batch size as 4 where 8 was used in D5. Decoder D7 is different in terms of kernel size in deconvolution layers. It uses kernel of size 2X2 in first and second, 3X3 in third and fourth,5X5 in fifth upsampling layer. Decoder D8 uses same non-bottleneck as D7 without 3X3 and 5X5 convolution. In Table 1, the pattern mNp, m stands for number of non-Bottleneck (N) layers, p stands for the type of non-bottleneck layer. Representation within braces (and) stands for set of non-bottleneck layers after a deconvolution layer starting from the first one. Of the different decoder variant, the best version is the one put forth in Figure 2, which is obtained after several optimization efforts and this network uses the Non-bottleneck layers detailed in Figure 1. This network also takes care of the class imbalance for Lanes and Curb and improves its class-wise IoU. We have done the runtime estimates for the top performing variants that uses the non-bottleneck layer on the target hardware whose values are put forth in Table 1. It can be seen that the best performing version is also the most cost effective one in terms of hardware budget with a runtime of 19.37ms for the entire network thus capable of handling 50fps for segmentation on V3H SOC [15].The sample segmentation outputs of our proposed optimal decoder and a standard FCN decoder are shown in Figure 3.

CONCLUSION

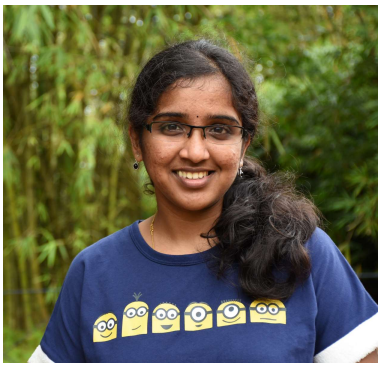
Design of efficient encoders is a growing area of research. In this work, we focused on design of efficient decoders. Firstly, we designed a novel efficient non-bottleneck layer and a family of decoders based on this layer. We experimentally show that different choice of decoder design had a large impact and the optimal configuration had 10% improvement of accuracy in terms of mean IoU overa baseline configuration. In particular, for more difficult segmentation classes like lanes and curb,higher improvements of 12% and 18% were obtained. Thus we demonstrate that the design of an efficient decoder can play a critical role for segmentation tasks as it covers a significant portion of the overall computation of the network. We hope that our preliminary study demonstrates the need for further research on efficient decoders. In future work, we build a systematic family of meta-architectures with a fixed run-time budget and learn the optimal configuration using meta-learning techniques.

REFERENCES

[1] Alexandre Briot, Prashanth Viswanath, and Senthil Yogamani. Analysis of efficient cnn design techniques for semantic segmentation. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 663–672, 2018.

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab:Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence, 40(4):834–848, 2018.
[3] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. arXiv preprint arXiv:1802.02611, 2018.
[4] Arindam Das and Senthil Yogamani. Evaluation of residual learning in lightweight deep networks for object classification. In Proceedings of the 20th Irish Machine Vision and Image Processing Conference, pages 205–208, 2018.
[5] Ryuhei Hamaguchi, Aito Fujita, Keisuke Nemoto, Tomoyuki Imaizumi, and Shuhei Hikosaka. Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery.CoRR,abs/1709.00179, 2017.
[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition.In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.
[7] Rudra PK Poudel, Ujwal Bonde, Stephan Liwicki, and Christopher Zach. Contextnet: Exploring context and detail for semantic segmentation in real-time.arXiv preprint arXiv:1805.04554, 2018.
[8] Eduardo Romera, José M Alvarez, Luis M Bergasa, and Roberto Arroyo. Efficient convnet for real-time semantic segmentation. In IEEE Intelligent Vehicles Symp.(IV), pages 1789–1794, 2017.
[9] Saikat Roy, Arindam Das, and Ujjwal Bhattacharya. Generalized stacking of layerwise-trained deep convolutional neural networks for document image classification. In Pattern Recognition (ICPR), 2016 23rd International Conference on, pages 1273–1278. IEEE, 2016.
[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition.arXiv preprint arXiv:1409.1556, 2014.
[11] Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell.Understanding convolution for semantic segmentation. In 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 1451–1460. IEEE, 2018.
[12] Zbigniew Wojna, Vittorio Ferrari, Sergio Guadarrama, Nathan Silberman, Liang-Chieh Chen, Alireza Fathi, and Jasper Uijlings. The devil is in the decoder.arXiv preprint arXiv:1707.05847, 2017.
[13] Zhenli Zhang, Xiangyu Zhang, Chao Peng, Dazhi Cheng, and Jian Sun. Exfuse: Enhancing feature fusion for semantic segmentation.arXiv preprint arXiv:1804.03821, 2018.
[14] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images.arXiv preprint arXiv:1704.08545, 2017.
[15] Renesas v3h soc specification. https://www.renesas.com/br/en/solutions/automotive/soc/r-car-v3h.html, 2018 (accessed October 27, 2018).
[16] Michal Uricár, Jan Ulicny, Ganesh Sistu, Hazem Rashed, Pavel Krizek, David Hurych, Antonin Vobecky, and Senthil Yogamani. Desoiling dataset: Restoring soiled areas on automotive fisheye cameras. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, pp. 0-0. 2019.
[17] Varun Ravi Kumar, Stefan Milz, Christian Witt, Martin Simon, Karl Amende, Johannes Petzold, Senthil Yogamani, and Timo Pech. Near-field depth estimation using monocular fisheye camera: A semi-supervised learning approach using sparse LiDAR data. In CVPR Workshop, vol. 7, p. 2. 2018.
[18] Michal Uricár, David Hurych, Pavel Krizek, and Senthil Yogamani. Challenges in designing datasets and validation for autonomous driving. arXiv preprint arXiv:1901.09270 (2019).
[19] Ciarán Eising, Jonathan Horgan, and Senthil Yogamani. Near-field perception for low-speed vehicle automation using surround-view fisheye cameras. IEEE Transactions on Intelligent Transportation Systems 23, no. 9 (2021): 13976-13993.
[20] Sambit Mohapatra, Senthil Yogamani, Heinrich Gotzig, Stefan Milz, and Patrick Mader. BEVDetNet: bird's eye view LiDAR point cloud based real-time 3D object detection for autonomous driving. In 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), pp. 2809-2815. IEEE, 2021.
[21] Eslam Mohamed, Mahmoud Ewaisha, Mennatullah Siam, Hazem Rashed, Senthil Yogamani, Waleed Hamdy, Mohamed El-Dakdouky, and Ahmad El-Sallab. Monocular instance motion segmentation for autonomous driving: Kitti instancemotseg dataset and multi-task baseline. In 2021 IEEE Intelligent Vehicles Symposium (IV), pp. 114-121. IEEE, 2021.

Arindam Das is a senior software engineer in Valeo and currently involved in deep learning based algorithm development activities in autonomous driving. He has more than 5 years of industry experience in computer vision, deep learning, pattern recognition, raster image processing and document analysis. He has authored 9 peer reviewed publications and 18 patents including 12 grants. He is a member of the Irish Pattern Recognition and Classification Society (IPRCS), a member body of the International Association for Pattern Recognition (IAPR).



Saranya Kandan is software engineer at Valeo and is involved in deep learning based algorithm and embedded software development for autonomous driving application. She has obtained her Masters degree from National University of Singapore in 2016. She has a total of three years of experience in the areas of computer vision, machine learning, deep learning and software development.

Senthil Yogamani is a computer vision architect and technical leader at Valeo Vision systems. He is currently focused on research and design of the overall computer vision algorithm architecture for surround-view camera visual perception in autonomous driving systems. He has over 12 years of experience in computer vision and machine learning including 10 years of experience in industrial automotive systems. He is an author of 46 peer reviewed publications and 31 patents. He serves in the editorial board of various IEEE automotive conferences including ITSC and ICVES and advisory board of various industry consortia including Khronos, Cognitive Vehicles and IS Auto. He is a recipient of best associate editor award at ITSC 2015 and best paper award at ITST 2012.