A CNN Architecture for Efficient Semantic Segmentation of Street Scenes

Davide Mazzini*, Marco Buzzelli*, Danilo Pietro Pau† and Raimondo Schettini*

*DISCo - University of Milano-Bicocca, Italy
Email: davide.mazzini@unimib.it, marco.buzzelli@disco.unimib.it, schettini@unimib.it

† Advanced System Technology, STMicroelectronics, Italy
Email: danilo.pau@st.com

***Abstract*—We propose a novel modular CNN architecture that provides semantic segmentation and understanding of outdoor street environment images. Our solution processes a 512x1024 resolution image on a single Titan Xp GPU at 37.4 FPS attaining 70.4% IoU on the Cityscapes test dataset.**

***Index Terms*—semantic segmentation, efficient street scene parsing, deep convolutional neural network, multiresolution processing**

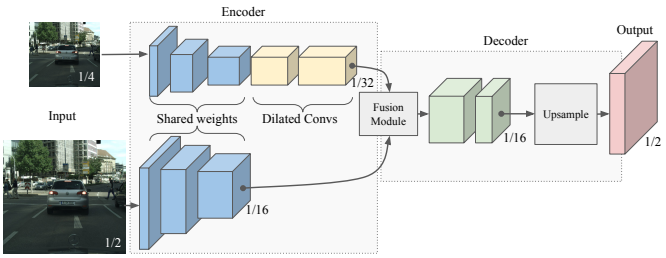


Fig. 1. Our Deep CNN architecture for semantic segmentation.

I. Introduction

Convolutional Neural Networks (CNNs) have become the standard baseline of many computer vision tasks [1]–[3]. Since the last five years almost all best performing models are based on (possibly deep) learned features as opposed to tailored-to-the-problem handcrafted features. Furthermore the increasing efficiency of novel CNN models and the introduction into the market of low-power embedded GPUs made the employment of CNN architectures computationally feasible for a wide range of practical applications. More than others, the automotive field has seen a strong expansion in recent years where a crucial role is played by perception systems for autonomous vehicles and for assisted driving.

The purpose of this work is to design a CNN architecture that can provide semantic segmentation and a global scene understanding in outdoor street environments. Our Neural Network architecture receives as input RGB images and provides a probability distribution over object classes for each pixel in the input image. The network is designed to process still images independently.

In the following paragraphs we introduce our contributions and a review of related CNN architectures for semantic segmentation. In Section II the dataset and the evaluation metrics adopted are presented. In Section III the proposed CNN architecture is described by outlining all the design choices and motivating them with experimental results. Section IV presents crucial implementation details to reproduce our results. Section V presents a comparison with state-of-the-art CNN architecture and finally Section VI presents further extensions that can be integrated into our baseline architecture.

A. Focus and contributions

We focus on semantic segmentation of street images for car-specific applications, where a model is continuously run on vehicles in order to quickly make decisions in reaction to environmental events. For this reason, our choices emerge from a compromise between precision and processing speed, building an efficient architecture that is based on a lightweight decoder. Our contributions are: we designed a new network architecture based on multi-resolution analysis, presented in Section III, which is able to achieve high quality predictions without sacrificing efficiency. The system we present can process an image with a resolution of 512×1024 pixels on a single GPU at 37.4 FPS, reaching 70.4% IoU in the Cityscapes test dataset. We have adopted an incremental approach in designing our network, as we highlight the advantages and disadvantages of each choice, and we have included the implementation details in Section IV with the aim of making the experiments easily repeatable. We designed a deep learning-based solution that is highly modular. The network architecture for semantic segmentation can be, in fact, further improved by adding an upsampling module as shown in Section V, or horizontally extended with additional tasks such as depth estimation, as shown in Section VI.

B. Related works

The great majority of current image segmentation architectures are based on the encoder-decoder structure [3] to exploit a large receptive field and, at the same time, to produce high-resolution predictions. Architectures that use

978-1-5386-6095-9/18/$31.00 ©2018 IEEE

dilated convolutions [4]–[6] to expand the receptive field, also adopt some form of downsampling to keep the computational effort low. Semantic maps are generally produced at $1/8$ or $1/16$ of the final resolution, and are subsequently upsampled using either nearest or bilinear interpolation.

A good compromise between processing speed and accuracy is difficult to achieve, and it heavily depends on the final application. For this reason, existing works can typically be categorized into two classes:

1) Accuracy-oriented architectures: The first work that successfully used CNNs for semantic segmentation is FCN [3]. Authors used a pre-trained encoder in conjunction with a simple decoding module, applying skip connections from the lower-level layers to process high-resolution activation maps. DeepLab [7] included Dilated Convolutions [8] to increase the context awareness (i.e. receptive field) of inner layers, while keeping a low number of parameters. Several methods adopted the Residual Network architecture [9] as encoder (such as DeepLabv2 [10], Resnet38 [11], FRRN [12]), further improving performance on the task of semantic segmentation. DeepLabv3 [6] and PSPNet [5] introduced the concept of context layers to further expand the theoretical receptive field of internal activations. These methods achieve high accuracy on different benchmarks but at the expense of high computational costs.

2) Efficiency-oriented architectures: ENet [13] was developed as a high-speed architecture, drastically increasing the efficiency of the model, but sacrificing accuracy. ERFNet [14] adopted a very simple encoder-decoder structure inspired by ENet. ERFNet authors designed the network structure on Residual Factorial convolutions that efficiently process the input signal with dedicated filters for each spatial dimension. SegNet [15] exploits high-resolution information by saving max-pooling indices at the encoding stage, and subsequently using them during the decoding phase. The design of ICNet [16] is based on a three-branch architecture with deep training supervision. The authors also experimented with a form of model compression to further accelerate the network.

II. Dataset and evaluation metrics

We carried out our experiments on the Cityscapes [17] dataset, which is a set of urban street images annotated with pixel-wise semantic information. It is composed of 5000 high resolution images (2048×1024) out of which 2975, 500 and 1525 images belong respectively to train, validation and test subsets. Annotations include 30 different classes of objects, although only 19 are typically used for training and evaluation. Two metrics are used for model validation: *mean of class-wise Intersection over Union (mIoU)*, defined as the mean of mIoU computed independently for each class, and *Frame Per Second (FPS)*, defined as the inverse of time necessary for our network to perform one forward pass. All the FPS performances reported in the following sections are referred to a single Titan Xp GPU.

Encoder	baseline	enc4	enc24	**enc24shared**	enc124shared
Multiresolution			✓	✓	✓
Shared Parameters				✓	✓
Subsampling factor	1	4	2 + 4	**2 + 4**	1 + 2 + 4
mIoU (%)	65.5	57.5	61.5	**63.0**	64.2
FPS	6.7	50.6	38.7	**38.7**	24.9

TABLE I

PERFORMANCE ON CITYSCAPES VALIDATION SET AND SPEED (FPS) OF FOUR ENCODER ARCHITECTURES. *baseline* IS A FULL-RESOLUTION NETWORK. *enc4* IS TRAINED AND EVALUATED WITH DOWNSAMPLED INPUT. *enc24* AND *enc124* MEANS 2 AND 3 BRANCHES WITH SUBSAMPLING FACTORS 2,4 AND 1,2,4 RESPECTIVELY. *shared* MEANS THAT WEIGHTS ARE PARTIALLY SHARED BETWEEN BRANCHES. IN BOLD THE CONFIGURATION ADOPTED IN THE FINAL MODEL.

III. Network design

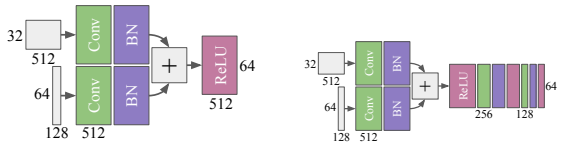
We propose a Neural Network which consists of a multiresolution architecture that jointly exploits high-resolution and large context information to produce an accurate segmentation of the input image. Our network relies on an encoder-decoder structure as depicted in Fig. 1. The encoder is composed by two branches with partially shared weights which task is to extract fine and coarse features from the input image subsampled at two different resolutions. The two multiresolution signals are merged by a *fusion module* presented in Section III-C. The decoder performs a compression in the channels dimension followed by an upsampling. The output of the network is a class-wise probability distribution for each pixel. Our network architecture, based on a fully-convolutional encoder-decoder, is presented in details in the following subsections. We designed our network through incremental steps. Each design choice is motivated by experimental results. We started from a baseline model and incrementally added single features analyzing benefits and disadvantages.

A. Input down-sampling

Input down-sampling is a simple way to speedup inference process. However aggressive down-sampling causes loss of fine details (e.g. precise borders between classes, fine textures). We setup two simple explorative experiments to investigate a trade-off between system speed and accuracy. We employed a DRN-D-22 model [4] pretrained on Imagenet as encoder and a simple bilinear upsampling as decoder. We trained and tested this base model with two different subsampling values. The first column of Table I shows the mIoU of the baseline model without subsampling. Than the same model is trained and evaluated with input images subsampled by factor 4 (second column). Model speed increases from 6.7 FPS to 50.6 but mIoU drops by 8%.

B. Multiresolution encoder

We designed a multi-resolution architecture as a compromise between model speed and discriminative power. Our multi-resolution encoder consists two branches: a low-resolution branch which is composed of all the layers of a Dilated Residual Network 22 type D (DRN-D-22) [4] with the exception of the last two. A medium-resolution branch with only the first layers of the DRN-D-22 before dilated



(a) base sum (b) postproc sum

Fig. 2. Two *fusion module* configurations. Both exploit addiction as merge strategy. *postproc sum* performs a dimensionality reduction.

Fusion Module	base sum	base concat	**postproc sum**	postproc concat
Sum	✓		✓	
Concat		✓		✓
Postprocessing Step			✓	✓
mIoU (%)	63.0	63.5	**65.8**	64.2
FPS	38.7	37.8	**37.3**	36.4

TABLE II
MIOU ON CITYSCAPES VAL SET AND FPS FOR DIFFERENT *fusion modules*. DIFFERENCES ARE: SIGNAL SUMMATION OR CONCATENATION AND PRESENCE OF A POST-PROCESSING STEP. IN BOLD THE CONFIGURATION ADOPTED IN THE FINAL MODEL.

convolutions. The goal of the first branch is to extract large context features whereas the goal of the second branch to extract more local features that will help to recover fine borders in the decoder. We experimented 3 encoder configurations. The first named *enc24* in Table I, consists of two branches that process input images with sub-sampling factors 2 and 4 with the structure defined above. The second configuration, named *enc24shared*, is similar to the first but weights are shared between the two branches. Results in Table I show that the network with shared branches achieve higher mIoU levels. Weight sharing between branches, as pointed out in [18], induces an implicit form of regularization. We chose this configuration as base encoder for the next experiments. In the third configuration named *enc124shared* in Table I, we added a further branch to elaborate full-resolution image. This brought some performance improvements but we decided not to employ this configuration in our final model because the whole system would slow down below the real-time threshold of 30FPS. The different encoder designs in Table I have a fixed the decoder architecture which is referred in Subsection III-C as *baseline*. Figure 1 depicts the second encoder design *enc24shared*. Others have been omitted but they can be intuitively deduced from the above configuration.

C. Fusion module

The purpose of the fusion module is to join information coming from low-resolution and high-resolution encoder branches. First, input from the low-resolution branch is upsampled with a differentiable bilinear filter to match the spatial size of signal coming from the medium-resolution branch. Input coming from the medium-resolution branch is expanded from 128 to 512 channels to match the number of features of the low-resolution branch. Finally the multi-resolution signals are merged. In Table II are reported experimental results of four different designs. We experimented both channel concatenation and addiction as merge strategies named *concat* and *sum* respectively. Moreover we investigated if a dimensionality reduction can be beneficial to the performance (*postproc*). This is in opposition to the baseline where the final classification layer is fed directly with the signal after the merge operation (*base*). Experimental results in Table II show that both mIoU and speed take advantage of the post-processing step. The model is benefits from the addictions of more non-linearities and the final upsampling operation is applied to a tensors with lower channel dimensions. Figure 2 depicts two different configurations: *base sum* and *postproc sum*, both with *addiction* merge strategy, without and with the post-processing step. Fusion modules with *concat* as merge strategy have a similar structure.

D. Network architecture

In Table III the complete network architecture is reported. The upper part of Table III defines the Encoder structure whereas the bottom part defines the Fusion Module and the Decoder. The network exhibits a total of 19 Millions of parameters and requires 58,7 GFLOP to perform single inference on a 512x1024 image. Most parameters are in the Encoder and in particular in the Residual Blocks.

A Block is a module composed by three layers: Convolution, Batch Normalization and ReLU. In particular, blocks with stride=2 perform input downsampling. Residual Blocks have been presented in [4]. Conv + BN is a single block with a Convolution followed by a Batch Normalization layer.

The network has two branches in the Encoder that share the same weights, for this reason in Table III there is only one column with the number of parameters for the two branches. The difference between branches relies in the signal resolution. Branch 2 is fed with a downsampled input. Branch 2 is the part of the network with the highest computational burden. Earliest layers of Branch 1 are heavier to compute, for this reason Branch 1 is shallower by design.

The Fusion Module is composed by a first part where signals are pre-processed independently followed by a second part where signals are jointly processed. The Decoder ends with a Classification layer (i.e. a 1x1 filter convolution) followed by a bilinear upsampling.

IV. EXPERIMENTAL SETUP

All the network configurations in this paper have been trained with Stochastic Gradient Descent (SGD) plus momentum. Following [4], we set the base learning rate to 0.001 and trained for 250 epochs. We adopted a *fixed step* learning rate policy. The initial value is decreased two times by a order of magnitude (at 100 and 200 epochs). We tried different base learning rates and *poly* learning rate policy from [7] but the baseline configuration gave us the best results. We found that batch size is an important parameter that affects the final model accuracy. We experimented with different values finding that the best value for our setup is 8. In contrast to what claimed in [19], increasing the batch size, in our case, negatively affects performance. We suppose that the higher stochasticity introduced by intra-batch dependencies acts as regularizer, improving the final network performance.

TABLE III

THE PROPOSED NETWORK ARCHITECTURE. FOR EACH MODULE THE OUTPUT SIZE, NUMBER OF OPERATIONS (FLOP) AND NUMBER OF PARAMETERS ARE REPORTED. *Block* IS COMPOSED BY CONVOLUTION, BATCHNORM AND RELU LAYERS. THE WORD *Dilated* MEANS THAT CONVOLUTIONS EXHIBIT A DILATION TERM.

Encoder						
Branch 1			**Branch 2**			
Type	Output size	FLOP	Type	Output size	FLOP	Parameters
			Subsample	256x512x3	393,2K	0
Block	512x1024x16	1,2G	Block	256x512x16	308,7M	2,4K
Block	512x1024x16	1,2G	Block	256x512x16	302,4M	2,3K
Block (stride=2)	256x512x32	609M	Block (stride=2)	128x256x32	302,2M	4,6K
Residual Block	128x256x64	3,9G	Residual Block	64x128x64	973,2M	131,4K
Residual Block	64x128x128	3,9G	Residual Block	32x64x128	973,1M	524,9K
			Dilated Residual Block	32x64x256	1,9G	2,1M
			Dilated Residual Block	32x64x512	7,8G	8,4M
			Dilated Block	32x64x512	4,8G	2,4M
			Block	32x64x512	4,8G	2,4M
Fusion Module						
			Upsample	64x128x512	16,8M	0
Conv + BN	64x128x512	4,8G	Conv + BN	64x128x512	19,3G	3,0M
Sum + ReLU				64x128x512	8,4M	0
Block				64x128x256	1,1G	131,3K
Block				64x128x128	268,5M	32,9K
Decoder						
Classification				64x128x19	19,9M	2,5K
Upsample				512x1024x19	39,8M	0
Total FLOP: 58,7G						Total Parameters: 19,0M

Transformation	baseline	color jitter	lighting jitter	**random scale**
mIoU (%)	65.8	62.6	64.2	**67.5**

TABLE IV

MIOU ON CITYSCAPES VALIDATION SET WITH DIFFERENT DATA AUGMENTATION TECHNIQUES USED DURING TRAINING. IN BOLD THE CONFIGURATION ADOPTED IN THE FINAL MODEL.

Furthermore we investigated some data augmentation techniques. The use of these techniques is almost cost-free in terms of computational resources at inference time. Even at train time they can be applied as a CPU pre-processing step in parallel with GPU computations. To make our model more robust to different lighting conditions we experimented the use of some light transformations. We consider this to be an important characteristic of our system since it is expected to work on real environments and outdoor scenes. We mainly applied two light transformations: Color Jitter and Lighting Jitter. Color Jitter consists in modifying image brightness, saturation and contrast in random-order. Lighting Jitter is the same jittering used in [2]. In particular $\sigma = 0.1$ is used as standard deviation to generate random noise. We also experimented a geometric transform: rescaling. Following [4] images are resized with a random scale factor between 0.5 and 2. Table IV shows the results of the application of these data augmentation techniques. Only *random scale* brought some improvements, thus we decided to include it in our baseline training procedure.

V. COMPARISON WITH THE STATE-OF-THE-ART

In Table V we reported the performance of the proposed architecture along with state of the art methods on Cityscapes test set. Information about dataset and evaluation metrics are given in Section II. We only included algorithms that declare their running time on Cityscapes leaderboard, since usually those that don't care about processing time are computationally heavy. Most of these methods, e.g. PSPNet, DeepLabv3 [5], [6] achieve very high mIoU levels (DeepLabv3+ is the best published model to date, reaching 81.2%), but they adopt very time-consuming multi-scale testing to increase accuracy, i.e. they reprocess the whole input image at different scales and average the results, in order to account for the potentially different conditions encountered during training. Our network architecture achieves 68.2% of mIoU on Cityscapes test set without any postprocessing. By replacing the bilinear upsampling in the decoder with a guided upsampling layer (see [20] and Section VI) our architecture can achieve 70.4% of mIoU with a slight speed decrease. mIoU measure has been computed by Cityscapes online evaluation server. FPS in Table V refer to a single Titan Xp GPU. Our network performs even better than some methods like Adelaide [21], Dilation10 [8] etc. that do not care about speed. Only ICNet [16] and ERFNet [14] achieve similar performances in terms of mIoU while being slightly faster. They make use of different strategies to reach high efficiency predictions which are orthogonal to ours. ICNet is based on a multiscale architecture similar to ours but with a different number of scales and different fusion modules. Authors of ICNet employed auxiliary losses to train the network while our network can be trained end-to-end with a single cross-entropy loss. In addiction to that, to lighten the computational burden of the final model, they compressed the trained model. This technique is orthogonal to the network structure design and could be employed in the exact same way to compress our model, however this goes beyond the purpose of this work. ERFNet model exhibit high efficiency thanks

Name	Subsampling	mIoU (%)	FPS
SegNet [15]	4	57.0	26.4
ENet [13]	2	58.3	121.5
SQ [22]	no	59.8	26.4
CRF-RNN [23]	2	62.5	2.2
DeepLab [7]	2	63.1	0.4
FCN-8S [3]	no	65.3	4.9
Adelaide [21]	no	66.4	0.05
Dilation10 [8]	no	67.1	0.4
ICNet [16]	no	69.5	47.9
ERFNet [14]	2	69.7	62.6
Ours (bilinear)	2	68.2	43.9
Ours (guided upsampling)	2	70.4	37.4

TABLE V
COMPARISON WITH STATE-OF-THE-ART METHODS ON CITYSCAPES TEST SET.

to the use of novel building blocks: Factorized Convolutions. Again our design choices are orthogonal to this, and our architecture could easily make use of Factorized Convolutions to improve efficiency.

VI. FURTHER EXTENSION

The proposed CNN architecture is highly modular and can be easily extended with addictional modules to improve the quality of predictions and to exploit extra functionalities. The decoder module is one of the simplest part of the architecture and is composed by a dimensionality-reduction operator followed by an upsampling operator. We chose to employ the bilinear upsampling operator as a tradeoff between computational complexity and prediction accuracy. However the bilinear operator can be substituted with a more powerful module at the cost of a slightly speed decrease. In Table V we reported two versions of our CNN architecture with bilinear and with guided upsampling module [20] in the decoder. Guided Upsampling [20] is an empowered type of upsampling operator where the regular sampling grid is warped by a differentiable CNN module.

Another extension to the current architecture involves multi-task prediction. It has been shown that jointly predicting multiple tasks can improve performance on the single objectives [25], and this is particularly true for semantic segmentation in conjunction with tasks such as depth estimation and instance recognition [26]. The current work can in fact be extended introducing estimation of the absolute distance between the subjects and the camera, as described in [24]. An example prediction is shown in Fig. 3 specifically for pixels belonging to specific semantic classes, although the same process can be generalized to the whole image.

VII. CONCLUSIONS

A new network architecture to perform real-time semantic segmentation of street scenes has been proposed. It consists of a multiresolution architecture to take full advantage of high-resolution textures and large context information. We evaluated our network on the Cityscapes test dataset showing that it is able to achieve 70.4% mIoU while running at 37.4 FPS on a single Titan Xp GPU.

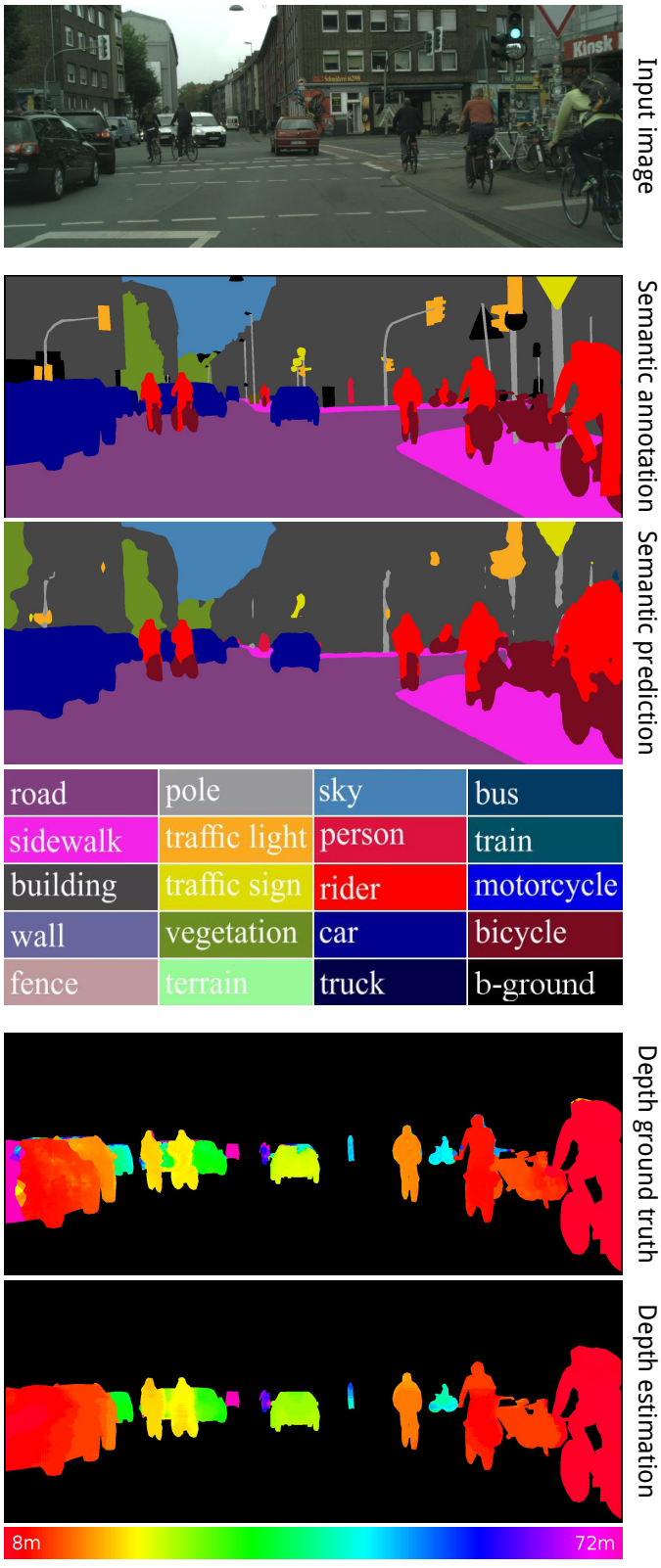


Fig. 3. Semantic segmentation results obtained with the proposed network architecture, and extension with depth estimation from [24].

ACKNOWLEDGMENTS

The research leading to these results has received funding from TEINVEIN: TEcnologie INnovative per i VEicoli Intelligenti, CUP: E96D17000110009 - Call "Accordi per la Ricerca e lInnovazione", cofunded by POR FESR 2014-2020.

We gratefully acknowledge the support of NVIDIA Corporation with the donation of a Titan Xp GPU used for this research.

REFERENCES

[1] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*. IEEE, 2014, pp. 512–519.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[4] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2881–2890.

[6] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *arXiv preprint arXiv:1802.02611*, 2018.

[7] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *CoRR*, vol. abs/1412.7062, 2014.

[8] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *International Conference on Learning Representations (ICLR)*, 2016.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[10] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[11] Z. Wu, C. Shen, and A. v. d. Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *arXiv preprint arXiv:1611.10080*, 2016.

[12] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," *arXiv preprint*, 2017.

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

[14] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2018.

[15] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[16] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," *arXiv preprint arXiv:1704.08545*, 2017.

[17] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[18] S. Bianco, D. Mazzini, and R. Schettini, "Deep multibranch neural network for painting categorization," in *International Conference on Image Analysis and Processing*. Springer, 2017, pp. 414–423.

[19] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[20] D. Mazzini, "Guided upsampling network for real-time semantic segmentation," in *BMVC*, 2018.

[21] G. Lin, C. Shen, A. Van Den Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3194–3203.

[22] M. Treml, J. Arjona-Medina, T. Unterthiner, R. Durgesh, F. Friedmann, P. Schuberth, A. Mayr, M. Heusel, M. Hofmarcher, M. Widrich *et al.*, "Speeding up semantic segmentation for autonomous driving," in *MLITS, NIPS Workshop*, 2016.

[23] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1529–1537.

[24] S. Bianco, M. Buzzelli, and R. Schettini, "A unifying representation for pixel-precise distance estimation," *Submitted to Multimedia Tools and Applications*, 2018.

[25] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.

[26] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Fast scene understanding for autonomous driving," *arXiv preprint arXiv:1708.02550*, 2017.