Efficient ConvNet for Real-time Semantic Segmentation

Eduardo Romera[1], José M. Álvarez[2], Luis M. Bergasa[1] and Roberto Arroyo[1]

***Abstract*— Semantic segmentation is a task that covers most of the perception needs of intelligent vehicles in an unified way. ConvNets excel at this task, as they can be trained end-to-end to accurately classify multiple object categories in an image at the pixel level. However, current approaches normally involve complex architectures that are expensive in terms of computational resources and are not feasible for ITS applications. In this paper, we propose a deep architecture that is able to run in real-time while providing accurate semantic segmentation. The core of our ConvNet is a novel layer that uses residual connections and factorized convolutions in order to remain highly efficient while still retaining remarkable performance. Our network is able to run at 83 FPS in a single Titan X, and at more than 7 FPS in a Jetson TX1 (embedded GPU). A comprehensive set of experiments demonstrates that our system, trained from scratch on the challenging Cityscapes dataset, achieves a classification performance that is among the state of the art, while being orders of magnitude faster to compute than other architectures that achieve top precision. This makes our model an ideal approach for scene understanding in intelligent vehicles applications.**

I. INTRODUCTION

Autonomous driving is a challenging topic that requires complex solutions in perception tasks such as recognition of road, traffic signs/lights, vehicles and pedestrians. Traditional vision approaches were normally focused on patch-based object detection that aimed to find these objects in the image independently with different algorithms [1]. However, these are not independent problems that should be treated separately, as they all belong to the same scene. Therefore, it is beneficial to perform scene understanding in an unified way, as it allows to solve many tasks at once in order to take their inter-relations and context into account in the classification problem. The task of full-image semantic segmentation aims at solving exactly this problem, by classifying a wide variety of object classes directly at the pixel level of an image, which supposes a very rich source of processed information for higher-level vehicle tasks like navigation.

Convolutional Neural Networks (ConvNets), initially designed for image classification tasks, have demonstrated impressive capabilities at segmentation by being able to classify several object categories pixel-wise and end-to-end on an image with very low error rates (e.g. [2] [3][4]). Recent works have achieved progressive leaps in the accuracy that is obtained by these deep architectures, until the point of making them reliable enough for real world applications. However, top performing approaches (e.g. [5][6][7][8]) have been focused on achieving minor improvements in accuracy at the expense of high increases in network complexity, making them unfeasible in terms of computational resources. On the other side, some works have focused on efficiency by proposing architectures that can reach real-time segmentation [9][10], but this is normally at the expense of accuracy (e.g. aggressively reducing network parameters).

In this paper, we propose an architecture that achieves top efficiency and accuracy without sitting on only one of these sides. Our network, based on a novel residual block that uses factorized convolutions, has been designed to maximize its performance while keeping an efficiency that is suitable for real-time operation in modern GPUs. Some examples of our network's segmentation output can be seen in Fig. 1. Our experiments demonstrate that it can learn to segment (at the pixel level) a widely varied set of object classes from a complex urban scene, with an accuracy that is as competitively high as the most complex alternative approaches in the state of the art, while having a computing

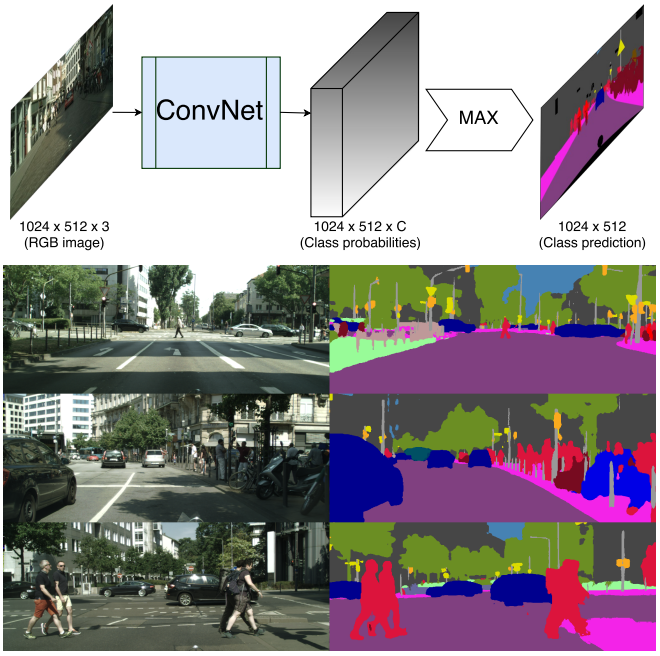


Fig. 1. Top: Simple diagram that depicts the proposed segmentation system. Bottom: Examples of segmentation produced by our ConvNet. Left: input images; Right: output segmentation of our architecture (19 classes).

*This work has been funded in part from the Spanish MINECO through the SmartElderlyCar project (TRA2015-70501-C2-1-R) and from the RoboCity2030-III-CM project (Robótica aplicada a la mejora de la calidad de vida de los ciudadanos. fase III; S2013/MIT-2748), funded by Programas de actividades I+D (CAM) and cofunded by EU Structural Funds. The authors also thank NVIDIA for generous hardware donations.

[1]Eduardo Romera, Luis M. Bergasa, Roberto Arroyo are with the Electronics Department, University of Alcalá (UAH), Alcalá de Henares, Spain {eduardo.romera, roberto.arroyo}@edu.uah.es, luism.bergasa@uah.es; [2]José M. Alvarez is with CSIRO-Data61, Canberra, Australia Jose.Alvarezlopez@data61.csiro.au

cost that is orders of magnitude lower. It can run at several FPS on a modern GPU, even on embedded devices that can be mounted on a vehicle. This makes it an ideal solution for the perception tasks of a self-driving vehicle, which aim to understand as much as possible of the driving scene while being able to operate in real-time.

This document is disposed as follows: In Section II, we discuss about some related works on the task of semantic segmentation. In Section III, we describe our architecture in depth, which can be easily reproduced to produce full-image segmentation in real-time. Finally, in Section IV, we perform a comprehensive set of experiments that aims at evaluating both the accuracy of the network and the computational resources that are required to process it.

II. RELATED WORKS

ConvNets were initially designed for image classification challenges, which consist in predicting single object categories from images. Long et al. [2] (FCN) firstly adapted known classification networks (e.g. VGG16 or GoogleNet) to perform end-to-end full-image semantic segmentation by making them fully convolutional and upsampling the output feature maps. However, directly adapting these networks results in coarse pixel outputs (and thus low pixel accuracy) due to the high downsampling that is performed in the classification task to gather more context. To refine these outputs, the authors propose to fuse them with activation maps from shallower layers using skip connections. Kendall et al. [3] (SegNet) proposed to upsample the features with a large decoder segment that performs finer unpooling by using the indices of the encoder's max-pooling blocks. Other works like [4] (Deeplab) have proposed to refine the coarse output by using CRFs, and works like [11] (CRFasRNN) proposed to integrate them inside the convolutional architecture. However, relying on algorithms like CRF to refine segmentation highly increases the network's computational overload.

Very recent works have achieved top performance by adapting Residual Networks (ResNets) [12] into the segmentation task, and combining them with dilated convolutions [13], which allow exponential expansion of the receptive field without loss of resolution or coverage. The work in [8] (DeepLab2) combines a ResNet-101 with spatial pyramid pooling (SPP) and Conditional Random Fields (CRF) to reach state-of-the-art performance. Lin et al. [6] (RefineNet) propose a multi-path refinement network that exploits all the information available along the downsampling process to enable high-resolution predictions using long-range residual connections. Pohlen et al. [7] (FRRN) propose a ResNet-like architecture that combines multi-scale context with pixel-level accuracy by using two processing streams, one that is processed at full resolution and another that performs downsampling operations. Ghiasi et al. [14] (LRR) propose a complex architecture that constructs a Laplacian pyramid to process and combine features at multiple scales. All these approaches achieve top performance but the required resources make them extremely expensive in terms of resources in modern GPUs, becoming unfeasible for ITS applications.

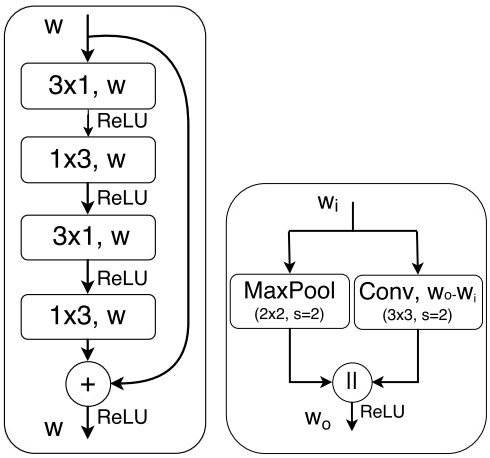
The recent ENet [9] sits on the opposite side in terms of efficiency, in which authors also adapt ResNet to the segmentation task, but make important sacrifices in the network layers to gain efficiency at the expense of a lower classification performance compared to the other approaches. In this paper, we propose an architecture that aims at obtaining the best possible trade-off between accuracy and efficiency, without neglecting any of them independently.

III. ARCHITECTURE

Our ConvNet builds upon an efficient redesign of convolutional blocks with residual connections. Residual connections [12] supposed a breakthrough because they avoid the degradation problem that is present in architectures with a large amount of stacked layers. This has allowed very deep networks (with hundreds of layers) to achieve outstanding classification performances. As described in Sec. II, various works have recently adapted large ResNets to the semantic segmentation task, achieving top performances in the state of the art. However, the trend of indiscriminately enlarging the depth of ConvNets to improve classification performance has been proven very inefficient in recent works focused in the image classification task [15][16]. Instead, we propose a "wider" architecture (as opposed to "deeper") that makes an extremely efficient use of its minimized amount of layers to achieve accurate segmentation in real time.

Our architecture is built in a sequential way by stacking layers based on our novel redesign of the residual layer. Residual layers were initially proposed in [12] and are widely used in several recent top-performing architectures. Authors in [12] proposed two designs: bottleneck and non-bottleneck. Both are similar but the bottleneck one is commonly used for efficiency reasons (e.g. ENet), as it internally reduces the computed feature maps in order to reduce computation. However, the non-bottleneck design can be beneficial in shallow versions of ResNet (like our architecture) due to their increased internal "width" (feature dimensions) compared to the bottleneck version. We believe that dense-prediction tasks like semantic segmentation can greatly benefit from this additional width, despite of their lower efficiency. In order to compensate efficiency, we repurpose the non-bottleneck design to be entirely built with convolutions with factorized (1D) kernels [17], which results in faster execution, reduced number of parameters and better regularization, without significant impact in its learning performance. We refer to this design as "Non-bottleneck-1D" (in short non-bt-1D), which is displayed in Fig. 2a. This proposed block combines the strengths of both designs: the efficiency of the bottleneck and the width of the non-bottleneck.

Our architecture is fully depicted in Table I. We follow an encoder-decoder architecture like SegNet [3] and ENet [9]. These avoid the need of using skip layers to refine the output (like FCN architectures [2]), by keeping a more sequential architecture based on an encoder segment, which is similar to original classification architectures (it produces downsampled feature maps), and a subsequent decoder segment that upsamples these features to the original input resolution.



(a) Non-bottleneck-1D (b) Downsampler block

Fig. 2. Diagram depicting each of the blocks that compose our architecture. "w": width (number of output feature maps) of the convolution. "s": stride

TABLE I

LAYER DISPOSAL OF OUR CONVNET. "OUT-F": NUMBER OF FEATURE MAPS AT LAYER'S OUTPUT."OUT-RES": OUTPUT RESOLUTION, GIVEN FOR AN EXAMPLE INPUT SIZE OF 1024X512.

Layer	Type	out-F	out-Res
1	**Downsampler block**	16	512x256
2	**Downsampler block**	64	256x128
3-7	5 x **Non-bt-1D**	64	256x128
8	**Downsampler block**	128	128x64
9	**Non-bt-1D** (dilated 2)	128	128x64
10	**Non-bt-1D** (dilated 4)	128	128x64
11	**Non-bt-1D** (dilated 8)	128	128x64
12	**Non-bt-1D** (dilated 16)	128	128x64
13	**Non-bt-1D** (dilated 2)	128	128x64
14	**Non-bt-1D** (dilated 4)	128	128x64
15	**Non-bt-1D** (dilated 8)	128	128x64
16	**Non-bt-1D** (dilated 16)	128	128x64
17	**Deconvolution** (upsampling)	64	256x128
18-19	2 x **Non-bt-1D**	64	256x128
20	**Deconvolution** (upsampling)	16	512x256
21-22	2 x **Non-bt-1D**	16	512x256
23	**Deconvolution** (upsampling)	C	1024x512

Layers from 1 to 16 in our architecture form the encoder, composed of residual blocks and downsampling blocks. Downsampling (reducing the spatial resolution) has the drawback of reducing the pixel accuracy (coarser outputs), but it also has two benefits: it lets the deeper layers gather more context (to improve classification) and it helps to reduce computation. Therefore, to keep a good balance we perform three downsamplings: at layers 1, 2 and 8. The decoder is formed by layers from 17 to 23. It upsamples the feature maps to match the input resolution. As opposed to SegNet and ENet, we do not use max-unpooling operation for the upsampling, but we use simple deconvolution layers with stride 2 (also known as full-convolutions). Fig. 3 contains a depiction of the feature maps generated by each of the blocks in our architecture, from the RGB image (encoder's input) to the pixel class probabilities (decoder's output).

Our downsampler block (Fig. 2b), inspired by the initial block of ENet [9], performs downsampling by concatenating the parallel outputs of a single 3x3 convolution with stride 2

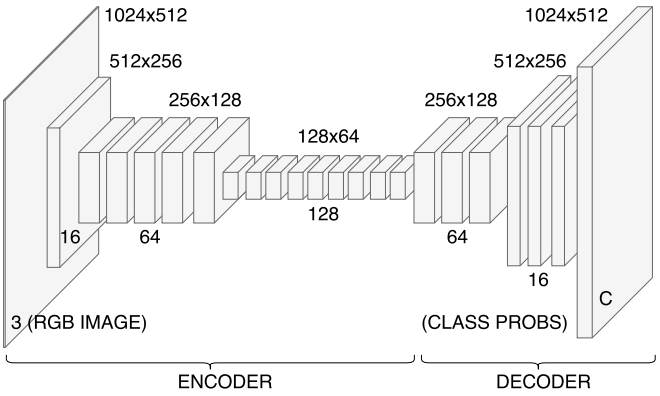


Fig. 3. Diagram depicting the feature maps produced by each of our architecture blocks, for an example input image of 1024x512. Each top value depicts the spatial resolution and each bottom value depicts the "width" or number of feature maps (C: number of classes).

and a Max-Pooling module. ENet uses it only as the initial block to perform early downsampling, but we use it in all the downsampling layers that are present in our architecture. Additionally, we also interleave some dilated convolutions [13] in our layers to improve classification by gathering more context. In Table I, in those blocks that are marked as "dilated", we change the second pair of 3x1 and 1x3 convolutions for a pair of dilated 1D convolutions. In the training phase, we also include Batch-Normalization [18] to accelerate convergence, and Dropout [19] as a regularization measure, although we triplicate its probability (0.3 in contrast to 0.1 used in ENet), as this yielded better results in our architecture. BN layers are placed after each 1D pair (after the 1x3 convolutions), and Dropout is placed at the end of each non-bt-1D layer (between the last 1x3 convolution and the add operation). Additional training details, regarding how to train our architecture end-to-end on a fully labeled dataset and our recommended hyperparameters, are described in the following section.

IV. EXPERIMENTS

We conduct an extensive set of experiments to validate the potential of our architecture. These are described below:

Dataset. We use the Cityscapes dataset [20], a recent dataset of urban scenes that has been widely adopted in semantic segmentation benchmarks due to its highly varied set of scenarios and challenging set of 19 labeled classes. It contains a train set of 2975 images, validation set of 500 images and a test set of 1525 images. The test labels are not available but it is possible to evaluate them on an online test server. We train our model on the Train set uniquely (fine annotations), without using the validation set for training. We train directly from scratch (e.g. not pre-training on larger datasets like ImageNet) to keep simplicity and to fully understand the capacity of our network. We perform simple data augmentation at training time by doing random horizontal flips and translations of 0-2 pixels in both axes. All pixel-level accuracy results are reported using the commonly used Intersection-over-Union (IoU) metric. More details can be found in [20]. The dataset resolution is 2048x1024, and

all accuracy results are reported at this resolution. We train our model to perform inference at 1024x512, but the output is rescaled (by simple interpolation) to the original dataset resolution for evaluation.

Setup. All experiments are conducted using the Torch7 framework [21]. Our model is trained using the Adam optimization [22] of stochastic gradient descent. Training is performed with a batch size of 12, momentum of 0.9, weight decay of $2e^{-4}$, and we start with a learning rate of $5e^{-4}$ that we divide by a factor of 2 every time that the training error becomes stagnant (usually every 50 epochs), in order to accelerate convergence. We used the class weighing technique proposed in [9]: $w_{class} = \frac{1}{ln(c+p_{class})}$, setting $c = 1.10$, which gave better results in our case. We firstly train the encoder segment with downsampled annotations and then attach the decoder to continue training end-to-end to produce segmentation with the same resolution as the input. With this setup, both reach convergence between 200-250 epochs. Training directly with the decoder from scratch (without training the encoder separately) is also possible, but it yielded slightly lower results in our experiments.

Comparison to the state of the art. Table II displays the results of our architecture at the Cityscapes test server compared to all other state-of-the-art approaches that are present at the Cityscapes benchmark at the date of submission, are not anonymous submissions (i.e. have an associated paper) and use comparable data (i.e. the fine annotations). Results are evaluated at a resolution of 2048x1024. "Pretrain" displays if the model has been pretrained using external data like ImageNet or Pascal. "fwt" displays the forward time in seconds evaluated on a single Titan X (Maxwell). Times are obtained from the benchmark's web and "n/a" indicates that this value was not published. Our architecture, trained from scratch (without pretraining) on the 2975 fine train images, achieves a 68% mean IoU at 19 classes and a 86.5% mean IoU at the 7 categories, while running at 24 ms. This comparison reflects that our architecture achieves a significantly better accuracy than most approaches focused on efficiency, while keeping an efficiency as competitive as the fastest one and being able to run in real-time on a single GPU. Most of the top-accuracy approaches have not published the time required to process a forward pass nor evaluated their efficiency. However, these approaches achieve top results by highly increasing the complexity and resources of their networks: RefineNet [6] and FRRN [7] employ large ResNets-like architectures in multiple pipelines that work with high resolution feature maps; Deeplabv2 [8] uses a large ResNet (101-layers) to improve their previous result in the Deeplab [23] model; And LRR-4x [14] constructs a Laplacian pyramid that processes and combines features at multiple scales. In summary, Deep ResNets demand high computational resources (more if they are computed at high resolution), and using multiple pipelines equals multiple architectures in parallel, which is also extremely demanding in resources. Therefore, it can be assumed that these approaches are not efficient and our network achieves the best available trade-off between segmentation accuracy and computational resources. Furthermore, our architecture has not required to pretrain on additional datasets like ImageNet, and it reaches its result directly trained from scratch with the fine Cityscapes annotations. This helps to keep simplicity in design and the training process, as pretraining on ImageNet would probably boost the accuracy (due to transferability of features) but would also require adapting the network and a long pretraining process of several days on that dataset.

TABLE II

LIST OF RESULTS IN THE CITYSCAPES TEST SET OF OUR ARCHITECTURE COMPARED TO OTHER APPROACHES IN THE STATE OF THE ART, AS REPORTED IN THE ONLINE BENCHMARK OF THE DATASET. "CLA"=CLASS, "CAT"=CATEGORY, "FWT"= FORWARD PASS TIME.

Network	Pretrain	Cla-IoU	Cat-IoU	fwt [s]
RefineNet [6]	ImageNet	73.6	87.9	n/a
FRRN [7]	-	71.8	88.9	n/a
Adelaide-cntxt [5]	ImageNet	71.6	87.3	35+
Deeplabv2-CRF [8]	ImageNet	70.4	86.4	n/a
LRR-4x [14]	ImageNet	69.7	88.2	n/a
Ours	-	**68.0**	**86.5**	**0.024**
Dilation10 [13]	ImageNet	67.1	86.5	4.0
DPN [24]	ImageNet	66.8	86.0	n/a
Scale inv.+CRF [25]	ImageNet	66.3	85.0	n/a
FCN-8s [2]	ImN+Pasc	65.3	85.7	0.5
Uhrig et al [26]	ImageNet	64.3	85.9	n/a
DeepLab [23]	ImageNet	63.1	81.2	4.0
CRFasRNN [11]	ImageNet	62.5	82.7	0.7
SQ [10]	ImageNet	59.8	84.3	0.06
ENet [9]	-	58.3	80.4	0.013
SegNet basic [3]	ImageNet	57.0	79.1	0.06
SegNet extended [3]	ImageNet	56.1	79.8	0.06

Per-class accuracy Table III displays the results on every one of the 19 classes evaluated on the Cityscapes test set at 2048x1024 for our architecture compared to the ones that have the fastest reported speed in the benchmark. Our architecture achieves the top accuracy by significant margins on all the 19 evaluated classes, while keeping a similar speed as the fastest ones. It achieves slight improvements on the general classes (Road, Sidewalk, Building, Vegetation, Sky, Car), while obtaining a significant accuracy improvement on all the challenging classes (Wall, Fence, Pole, Traffic Light, Traffic Sign, Terrain, Pedestrian, Rider, Truck, Bus, Train, Motorbike, Bicycle). These are challenging because the dataset contains significantly less training samples (e.g. train/truck compared to road) or they have more challenging shapes (e.g. pedestrian vs car). Our network leverages the improved design of our proposed residual layers to increase the performance on these classes, even when trained using the same amount of samples as the other approaches.

Processing time comparison Table IV displays inference time (forward pass) for different resolutions on a single Tegra TX1 (Jetson TX1) and on a single NVIDIA Titan X (Maxwell), compared to other architectures that had these results available. For comparison, we use the same set of resolutions as in their papers. At 640x360, a resolution that is enough to recognize any urban scene accurately, our network achieves around 83 FPS on a single Titan X and more than 7 FPS on a Tegra TX1, an embedded GPU that uses less than 10 Watts at full load. At 1024x512 (the ratio used in

TABLE III

PER-CLASS IOU (%) ON THE CITYSCAPES TEST SET OF OUR ARCHITECTURE COMPARED TO THE FASTEST ARCHITECTURES. LIST OF CLASSES (FROM LEFT TO RIGHT): ROAD, SIDE-WALK, BUILDING, WALL, FENCE, POLE, TRAFFIC LIGHT, TRAFFIC SIGN, VEGETATION, TERRAIN, SKY, PEDESTRIAN, RIDER, CAR, TRUCK, BUS, TRAIN, MOTORBIKE AND BICYCLE. "CLASS": MEAN IOU (19 CLASSES); "CAT" MEAN IOU (7 CATEGORIES).

Network	Roa	Sid	Bui	Wal	Fen	Pol	TLi	TSi	Veg	Ter	Sky	Ped	Rid	Car	Tru	Bus	Tra	Mot	Bic	Class	Cat
SegNet [3]	96.4	73.2	84.0	28.4	29.0	35.7	39.8	45.1	87.0	63.8	91.8	62.8	42.8	89.3	38.1	43.1	44.1	35.8	51.9	56.95	79.13
ENet [9]	96.3	74.2	75.0	32.2	33.2	43.4	34.1	44.0	88.6	61.4	90.6	65.5	38.4	90.6	36.9	50.5	48.1	38.8	55.4	58.28	80.39
SQ [10]	96.9	75.4	87.8	31.59	35.7	50.9	52.0	61.7	90.9	65.8	93.0	73.8	42.6	91.5	18.8	41.2	33.3	34.0	59.9	59.84	84.31
Ours	**97.7**	**81.0**	**89.8**	**42.5**	**48.0**	**56.3**	**59.8**	**65.3**	**91.4**	**68.2**	**94.2**	**76.8**	**57.1**	**92.8**	**50.8**	**60.1**	**51.8**	**47.3**	**61.7**	**68.02**	**86.46**

TABLE IV

INFERENCE TIMES OF FASTEST ARCHITECTURES ON TEGRA TX1 AND TITAN X AT DIFFERENT RESOLUTIONS

Model	NVIDIA TEGRA TX1 (Jetson)						NVIDIA TITAN X (Maxwell)					
	480x320		640x360		1280x720		640x360		1280x720		1920x1080	
	ms	fps	ms	fps	ms	fps	ms	fps	ms	fps	ms	fps
SegNet [3]	757	1.3	1251	0.8	-	-	69	14.6	289	3.5	637	1.6
ENet [9]	47	21.1	69	14.6	262	3.8	7	135.4	21	46.8	46	21.6
SQ [10]	60	16.7	86	11.6	389	2.6	n/a					
Ours	93	10.8	141	7.1	535	1.9	12	83.3	41	24.4	88	11.4

the Cityscapes tests), our network achieves 24 ms (41 FPS) on a Titan X. In summary, our network achieves a speed that is as competitively fast as the fastest ones (ENet and SQ), while having a significantly better accuracy. These inference times demonstrate that it is possible to run our network in real-time to provide full-image semantic segmentation, even in embedded systems that can be mounted on a vehicle.

Qualitative experiments. Fig. 4 contains various examples of segmentation produced by our architecture (d) and ENet (c), compared to the ground truth (b). While both networks can accurately segment the road that is immediately ahead of the vehicle, ENet gives much coarser predictions for objects that are more distant or that require finer accuracy at the pixel level (e.g. pedestrians, traffic signs). Our architecture yields consistent results for all classes, even at far distances in the scene. The IoU measurement used in the quantitative results is a challenging measurement that takes into account the confusion between all classes and aims to even the impact between small ones (e.g. traffic light) and large ones (e.g. road), but it does not reflect the fact that the total pixel accuracy (i.e. percentage of correct pixel predictions) is over 95%. This explains why the segmentation is qualitatively good already, even though 68% mean IoU (19 classes) and 86.5% category IoU (7 categories) might seem like low values. Despite of having lower accuracy on specific challenging classes like "train" or "wall", the network already has an excellent accuracy on important classes like "road", "pedestrians" or "vehicles". This makes the network suitable for ITS applications like self-driving cars, as it can already provide accurate and complex scene understanding to higher level algorithms like navigation.

V. CONCLUSIONS

In this paper, we have proposed an architecture that achieves accurate pixel-wise semantic segmentation in real time, even in embedded GPUs that can be mounted on a vehicle. In contrast to top-performing approaches in the state of the art, that develop complex architectures that are expensive in computational resources, and in contrast to the alternative efficient architectures, that perform significant sacrifices in the network design to gain efficiency in exchange of parameters, our design maximizes its accuracy while remaining extremely efficient. This results in an architecture that achieves a performance that is as competitive as the state of the art, while being as efficient as the fastest networks available. Such a network provides an excellent trade-off between reliability and efficiency, which makes it suitable for countless ITS applications such as scene understanding in self-driving vehicles.

Future works will involve in-depth experiments regarding the power consumption of the model, compression techniques (e.g. binarization of weights) for further reduction of the model's computational resources, and experiments on different datasets and images taken on other environments of smart vehicles (e.g. rural environments and highways).

REFERENCES

[1] E. Romera, L. M. Bergasa, and R. Arroyo, "Can we unify monocular detectors for autonomous driving by using the pixel-wise semantic segmentation of cnns?" *arXiv preprint arXiv:1607.00971*, 2016.

[2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conf. on Computer Vision and Pattern Recog. (CVPR)*, 2015, pp. 3431–3440.

[3] V. Badrinarayanan, A. Handa, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling," *arXiv preprint arXiv:1505.07293*, 2015.

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014.

[5] G. Lin, C. Shen, A. Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *IEEE Conf. on Computer Vision and Pattern Recog. (CVPR)*, 2016, pp. 3194–3203.

[6] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation," *arXiv preprint arXiv:1611.06612*, 2016.

[7] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," *arXiv preprint arXiv:1611.08323*, 2016.

[8] 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," *arXiv preprint arXiv:1606.00915*, 2016.

[9] 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.

[10] M. Treml, J. Arjona-Medina, T. Unterthiner, and et al., "Speeding up semantic segmentation for autonomous driving," in *MLITS, NIPS Workshop*, 2016.

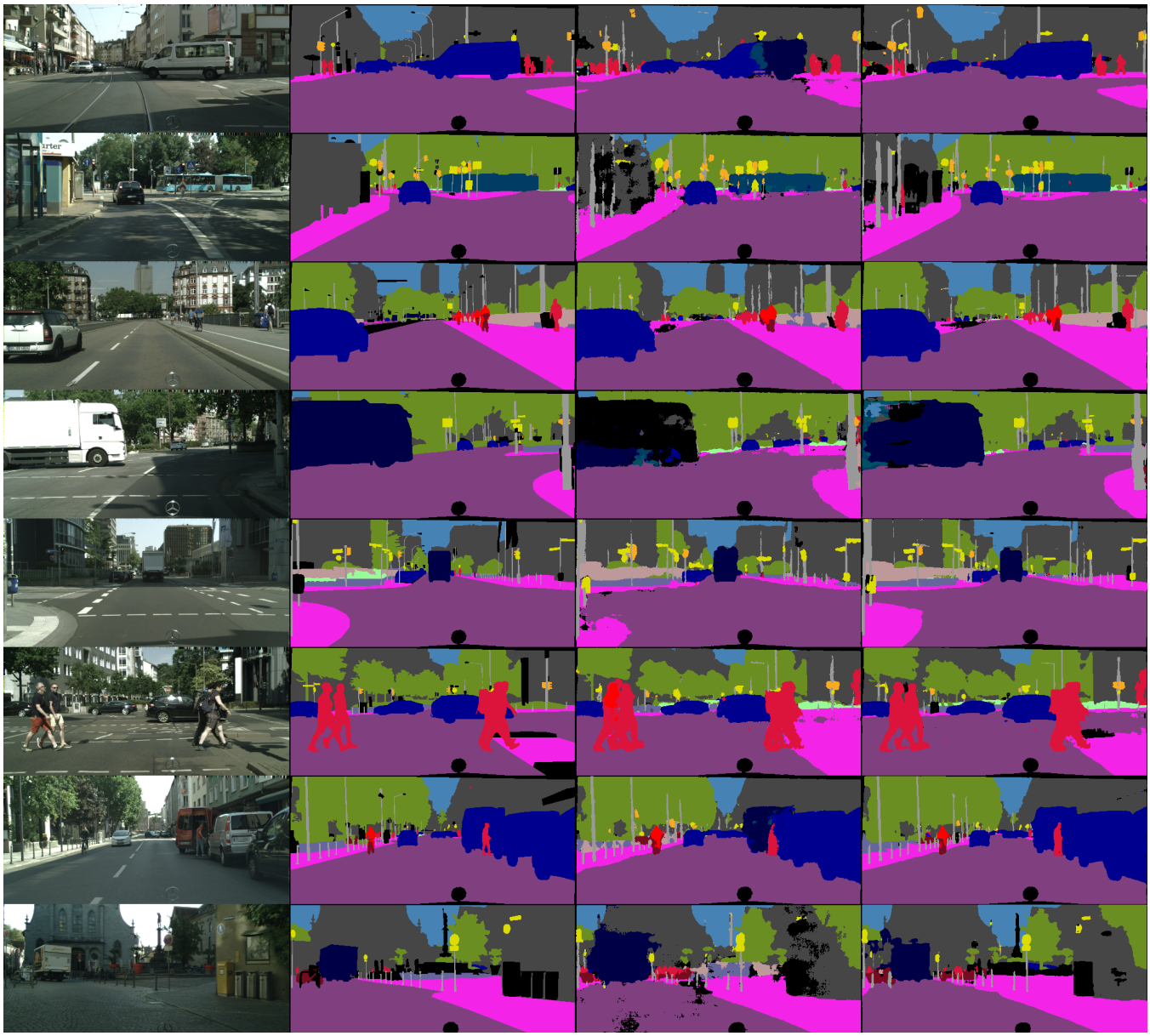


Fig. 4. Examples of segmentation produced by our ConvNet compared to ENet. From left to right: (a) Input image, (b) Ground truth, (c) ENet, (d) Ours.

[11] 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 *IEEE International Conf. on Computer Vision (ICCV)*, 2015, pp. 1529–1537.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[13] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[14] G. Ghiasi and C. C. Fowlkes, "Laplacian pyramid reconstruction and refinement for semantic segmentation," in *European Conf. on Computer Vision (ECCV)*, 2016, pp. 519–534.

[15] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[16] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," *arXiv preprint arXiv:1603.05279*, 2016.

[17] J. Alvarez and L. Petersson, "Decomposeme: Simplifying convnets for end-to-end learning," *arXiv preprint arXiv:1606.05426*, 2016.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[19] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[20] 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 *IEEE Conf. on Computer Vision and Pattern Recog. (CVPR)*, 2016, pp. 3213–3223.

[21] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS Workshop*, no. EPFL-CONF-192376, 2011.

[22] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille, "Weakly-and semi-supervised learning of a dcnn for semantic image segmentation," *arXiv preprint arXiv:1502.02734*, 2015.

[24] Z. Liu, X. Li, P. Luo, C.-C. Loy, and X. Tang, "Semantic image segmentation via deep parsing network," in *Proceedings of the IEEE International Conf. on Computer Vision*, 2015, pp. 1377–1385.

[25] I. Krešo, D. Čaušević, J. Krapac, and S. Šegvić, "Convolutional scale invariance for semantic segmentation," in *German Conf. on Pattern Recog. (GCPR)*, 2016, pp. 64–75.

[26] J. Uhrig, M. Cordts, U. Franke, and T. Brox, "Pixel-level encoding and depth layering for instance-level semantic labeling," *arXiv preprint arXiv:1604.05096*, 2016.