ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation

Eduardo Romera, José M. Álvarez, Luis M. Bergasa, and Roberto Arroyo

Abstract*— Semantic segmentation is a challenging task that addresses most of the perception needs of intelligent vehicles (IVs) in an unified way. Deep neural networks excel at this task, as they can be trained end-to-end to accurately classify multiple object categories in an image at pixel level. However, a good tradeoff between high quality and computational resources is yet not present in the state-of-the-art semantic segmentation approaches, limiting their application in real vehicles. 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 architecture is a novel layer that uses residual connections and factorized convolutions in order to remain efficient while retaining remarkable accuracy. Our approach is able to run at over 83 FPS in a single Titan X, and 7 FPS in a Jetson TX1 (embedded device). A comprehensive set of experiments on the publicly available Cityscapes data set demonstrates that our system achieves an accuracy that is similar to the state of the art, while being orders of magnitude faster to compute than other architectures that achieve top precision. The resulting tradeoff makes our model an ideal approach for scene understanding in IV applications. The code is publicly available at: *https://github.com/Eromera/erfnet

***Index Terms*— Intelligent vehicles, scene understanding, real-time, semantic segmentation, deep learning, residual layers.**

I. Introduction

THE perception tasks of Intelligent Vehicles (IV) suppose important challenges due to the high complexity of the environments in which they are required to operate (e.g. urban streets). While most systems rely on sensor fusion to understand as much as possible of their surrounding scene, cameras have gained significant importance in the community due to the remarkable advances in the computer vision field. Images are a rich multi-dimensional signal that is cheap to capture, but that requires complex algorithms to process. Traditional vision-based approaches were initially aimed at developing specific techniques for detecting traffic elements such as the road pavement, pedestrians, cars, signs or traffic lights independently [1]. However, recent advances in deep learning have allowed to unify all of these classification problems into one single task: semantic segmentation.

The task of semantic segmentation aims at labeling categories at the pixel-level of an image and has direct applications in the computer vision field. It is a challenging task because it requires combining dense pixel-level accuracy with multi-scale contextual reasoning [2]. Convolutional Neural Networks (ConvNets), initially designed for classification tasks [3], [4] and recently adapted to segmentation [5], have demonstrated impressive capabilities at solving these complex challenges. ConvNets are able to achieve end-to-end full-image segmentation with an accuracy that outperforms any traditional method. However, a good trade-off between high quality and computational resources is yet not present in state-of-the-art segmentation architectures.

Recently, the residual layers proposed in [6] have supposed a new trend in ConvNets design. Their reformulation of the convolutional layers to avoid the degradation problem of deep architectures has allowed recent works to achieve very high accuracies with networks that stack large amounts of layers. This strategy has been commonly adopted in new architectures that obtain top accuracy at both image classification challenges [6], [7] and semantic segmentation challenges [8]–[10]. Despite these achievements, we consider that this design strategy is not an effective way to obtain a good trade-off between accuracy and efficiency. Considering a reasonable amount of layers, enlarging the depth with more convolutions achieves only small gains in accuracy while significantly increasing the required computational resources.

Computational resources are a key limitation in IV applications. Algorithms are not only required to operate reliably, but they are required to operate fast (real-time), fit in embedded devices due to space constraints (compactness), and have low power consumption to affect as minimum as possible the vehicle autonomy. Regarding ConvNets, all this is translated into the graphics processing unit (GPU) resources that are required to process the network parameters. With this in mind, some works have aimed at developing efficient architectures that can run in real-time [11], [12]. However, these approaches usually focus on obtaining this efficiency by an aggressive reduction of parameters, which highly detriments accuracy.

Manuscript received March 10, 2017; revised June 2, 2017; accepted July 16, 2017. Date of publication October 9, 2017; date of current version December 26, 2017. This work was supported in part by the Spanish MINECO through the SmartElderlyCar Project under Grant TRA2015-70501-C2-1-R and the RoboCity2030-III-CM Project (Robótica aplicada a la mejora de la calidad de vida de los ciudadanos. Fase III; S2013/MIT-2748), in part by the Programas de actividades I+D (CAM), and in part by EU Structural Funds. The Associate Editor for this paper was Q. Wang. *(Corresponding author: Eduardo Romera.)*

E. Romera, L. M. Bergasa, and R. Arroyo are with the Department of Electronics, University of Alcalá, 28805 Alcalá de Henares, Spain (e-mail: eduardo.romera@edu.uah.es; luism.bergasa@uah.es; roberto.arroyo@edu.uah.es).

J. M. Álvarez is with CSIRO-Data61, Canberra 2601, Australia (e-mail: jose.alvarezlopez@data61.csiro.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2017.2750080

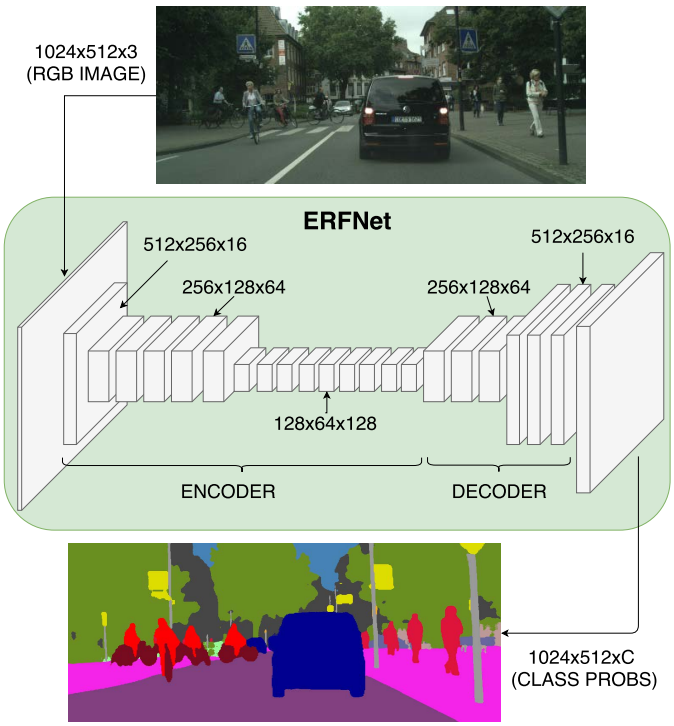


Fig. 1. Diagram that depicts the proposed segmentation system (ERFNet) for an example input image and its corresponding output (C = 19 classes). The depicted volumes correspond to the feature maps produced by each layer. All spatial resolution values are with regard to the example input (1024×512), but the network can operate with arbitrary image sizes.

In this paper, we aim at solving this trade-off as a whole, without sitting on only one of its sides. We propose ERFNet (Efficient Residual Factorized Network), a ConvNet for real-time and accurate semantic segmentation. The core element of our architecture is a novel layer design that leverages skip connections and convolutions with 1D kernels. While the skip connections allow the convolutions to learn residual functions that facilitate training, the 1D factorized convolutions allow a significant reduction of the computational costs while retaining a similar accuracy compared to the 2D ones. The proposed block is thus stacked sequentially to build our encoder-decoder architecture, which produces semantic segmentation end-to-end in the same resolution as the input (see Fig. 1 for an example). A comprehensive set of experiments on the challenging Cityscapes [13] dataset of urban scenes demonstrates the remarkable trade-off between accuracy and efficiency of our architecture, reaching an accuracy as competitive as the top networks, while also being among the fastest ones. This paper is an extension of our conference paper [14], which has been extended with a detailed description of the proposed residual block and the full architecture ERFNet, along with an extended set of experiments.

II. Related Works

ConvNets were initially designed for image classification challenges, which consist in predicting single object categories from images. Long et al. [5] (FCN) first adapted known classification networks (e.g. VGG16 [15]) to perform end-to-end 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. Badrinarayanan et al. [16] (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 [17] (Deeplab) proposed to refine the coarse output by using Conditional Random Fields (CRF), and works like [18] (CRFasRNN) proposed to integrate them inside the convolutional architecture. However, relying on algorithms like CRF to refine segmentation highly increases the computational overload of the architecture.

Recent works have achieved top segmentation accuracy by adapting ResNets [6] to the segmentation task. The core element of these networks is a residual block that incorporates identity connections from the input to output to alleviate the degradation problem present in networks with a large number of layers. Experiments based on ResNet demonstrated that a thiner (simpler layers) but deeper (using more layers) architecture with residual connections achieved state-of-the-art accuracy and can outperform more complex designs like VGG16 [15], that use regular convolutional layers without skip connections.

The work in [8] (DeepLab2) combines a ResNet-101 with spatial pyramid pooling and CRF to reach state-of-the-art segmentation accuracy. Lin et al. [9] (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. [10] (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 and Fowlkes [19] (LRR) propose a complex architecture that constructs a Laplacian pyramid to process and combine features at multiple scales. All these approaches achieve top accuracy but the required resources make them extremely expensive in terms of resources in modern GPUs, becoming unfeasible for IV applications. The recent ENet [11] 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 accuracy compared to the other approaches.

III. ERFNet: Proposed Architecture

In this section, we introduce our efficient architecture for real-time semantic segmentation. Our proposal aims at solving an efficiency limitation that is inherently present in commonly adopted versions of the residual layer, which is used in several recent ConvNets that achieve top accuracy in classification [6] and segmentation tasks [8], [11], [20]. By solving this limitation, we manage to develop a semantic segmentation architecture that makes a much more efficient use of parameters compared to existing architectures, allowing

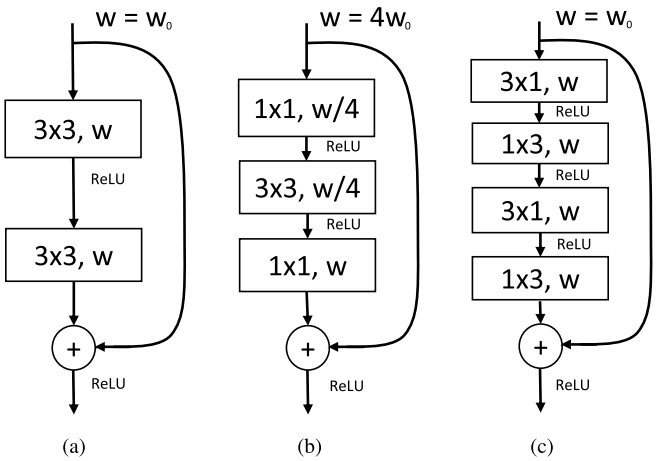


Fig. 2. Depiction of the two residual layers originally proposed in [6] (Non-bottleneck and Bottleneck), and our proposed design (Non-bottleneck-1D). w represents the number of feature maps input to the layer, internally reduced by 4 in the bottleneck design. In the convolutional blocks, "$d_1 \times d_2$, f" indicate their kernel sizes (d_1, d_2) and number of output feature maps (f). (a) Non-bottleneck. (b) Bottleneck. (c) Non-bottleneck-1D.

our network to obtain a very high segmentation accuracy while keeping top efficiency in order to satisfy the constraints present in IV applications.

A. Factorized Residual Layers

Residual layers [6] have the property of allowing convolutional layers to approximate residual functions, as the output vector y of a layer vector input x becomes:

$$y = F(x, \{W_i\}) + W_s x \tag{1}$$

where W_s is usually an identity mapping and $F(x, \{W_i\})$ represents the residual mapping to be learned. This residual formulation facilitates learning and significantly reduces the degradation problem present in architectures that stack a large amount of layers [6]. The original work proposes two instances of this residual layer: the non-bottleneck design with two 3×3 convolutions as depicted in Fig. 2 (a), or the bottleneck version as depicted in Fig. 2 (b). Both versions have similar number of parameters and almost equivalent accuracy. However, the bottleneck requires less computational resources and these scale in a more economical way as depth increases. Hence, the bottleneck design has been commonly adopted in state-of-the-art networks [6], [8], [11], [20]. However, it has been reported that non-bottleneck ResNets gain more accuracy from increased depth than the bottleneck versions, which indicates that they are not entirely equivalent and that the bottleneck design still suffers from the degradation problem [6], [7], [21].

We propose to redesign the non-bottleneck residual module in a more optimal way by entirely using convolutions with 1D filters (Fig. 2 (c)). As demonstrated in [22], any 2D filter can be represented by a combination of 1D filters in the following way. Let $\boldsymbol{W} \in \mathbb{R}^{C \times d^h \times d^v \times F}$ denote the weights of a typical 2D convolutional layer, where C is the number of input planes, F is the number of output planes (feature maps) and $d^h \times d^v$ represents the kernel size of each feature map (typically $d^h \equiv d^v \equiv d$). Let $b \in \mathbb{R}^F$ be the vector representing the bias term for each filter and $\mathbf{f^i} \in \mathbb{R}^{d^h \times d^v}$ represent the ith kernel in the layer. Common approaches first learn these filters from data and then find low-rank approximations as a post-processing step [23]. However, this approach requires additional fine tuning and the resulting filters may not be separable. Instead, [24] demonstrates that it is possible to relax the rank-1 constraint and essentially rewrite $\mathbf{f^i}$ as a linear combination of 1D filters:

$$\mathbf{f^i} = \sum_{k=1}^{K} \sigma_k^i \bar{v}_k^i \left(\bar{h}_k^i\right)^T \tag{2}$$

where $\bar{v}_k^i$ and $\left(\bar{h}_k^i\right)^T$ are vectors of length d, σ_k^i is a scalar weight, and K is the rank of $\mathbf{f^i}$. Based on this representation, Alvarez and Petersson [22] propose that each convolutional layer can be decomposed with 1D filters that can additionally include a non-linearity $\varphi(\cdot)$ in between. Thus, the i-th output of a decomposed layer, a_i^1 can be expressed as a function of its input a_*^0, in the following way:

$$a_i^1 = \varphi\left(b_i^h + \sum_{l=1}^{L} \bar{h}_{il}^T * \left[\varphi\left(b_l^v + \sum_{c=1}^{C} \bar{v}_{lc} * a_c^0\right)\right]\right) \tag{3}$$

where, L represents the number of filters in the intermediate layer, and $\varphi(\cdot)$ can be implemented with ReLU [4] or PReLU [25]. The resulting decomposed layers have intrinsically low computational cost and simplicity. Additionally, the 1D combinations improve the compactness of the model by minimizing redundancies (as the filters are shared within each 2D combination) and theoretically improve the learning capacity by inserting a non-linearity between the 1D filters [22].

Considering an equal kernel size d for simplicity, it is trivial to see that the decomposition reduces $\boldsymbol{W_{2D}} \in \mathbb{R}^{C \times d \times d \times F}$ of any 2D convolution into a pair of $\boldsymbol{W_{1D}} \in \mathbb{R}^{C \times d \times F}$, resulting the equivalent dimensions of each 1D pair in $dim = 2 \times (C \times d \times F)$. Therefore, this factorization can be applied to reduce the 3×3 convolutions on the original residual modules. While larger filters would be more benefited by this decomposition, applying it on 3×3 convolutions already yields a 33% reduction in parameters and further increases its computational efficiency.

By leveraging this decomposition, we propose a new implementation of the residual layer that makes use of the described 1D factorization to accelerate and reduce the parameters of the original non-bottleneck layer. We refer to this proposed module as "non-bottleneck-1D" (non-bt-1D), which is depicted in Fig. 2 (c). This module is faster (as in computation time) and has less parameters than the bottleneck design, while keeping a learning capacity and accuracy equivalent to the non-bottleneck one. Table I summarizes the total dimensions of the weights on the convolutions of every residual block, comparing original ones with our proposed 1D factorizations. Both non-bottleneck and bottleneck implementations can be factorized into 1D kernels. However, the non-bottleneck design is clearly more benefited, by receiving a direct 33% reduction in both convolutions and greatly accelerating its execution

TABLE I

COMPARISON OF WEIGHT SIZES BETWEEN ORIGINAL RESIDUAL BLOCKS AND OUR PROPOSED 1D DESIGN (#FM = NUMBER OF FEATURE MAPS RECEIVED AT MODULE INPUT (EXT) AND COMPUTED INTERNALLY (INT)

Residual block	#ext-fm (#int-fm)	#Weights
bottleneck	256 (64)	69K
non-bottleneck	64 (64)	73K
bottleneck-1D	256 (64)	57K
non-bottleneck-1D	64 (64)	49K

time. As demonstrated in our experiments, this acceleration even makes it faster than the bottleneck design, whose original purpose according to [6] was to accelerate training time.

Although the focus of this paper is the segmentation task, the proposed non-bottleneck-1D design is directly transferable to any existing network that makes use of residual layers, including both classification and segmentation architectures. Additionally, this design facilitates a direct increase in the "width" (seen as the number of filters or of feature maps computed), while keeping at a minimum the computational resources. Increasing width has already been proven effective in classification-aimed residual networks [7], [21]. The segmentation architecture proposed in the next section demonstrates that dense prediction tasks like semantic segmentation can also benefit from the increased width, while remaining computationally efficient due to the proposed 1D factorization.

B. Architecture Design

In this work, our main motivation is to obtain an architecture that gets the best possible trade-off between accuracy and efficiency. With this target in mind, we followed the current trend of using convolutions with residual connections as the core elements of our architecture, in order to leverage their success in classification and segmentation problems. However, as stated in the previous section, the commonly used residual layers inherited some limitations in terms of learning capacity and efficiency, which we aimed to minimize with our proposed non-bottleneck-1D (non-bt-1D) layer. This novel block, that leverages residual connections with factorized convolutions and combines the strengths of bottleneck and non-bottleneck designs, is the core of our architecture. Our network is designed by stacking sequentially the proposed non-bt-1D layers in a way that best leverages their learning performance and efficiency.

Our architecture is fully depicted in Table II. We follow an encoder-decoder architecture like SegNet [16] and ENet [11]. Contrary to architectures like FCN [5], where feature maps from different layers need to be fused to obtain a fine-grain output, our approach follows a more sequential architecture based on an encoder segment producing downsampled feature maps and a subsequent decoder segment that upsamples the feature maps to match input resolution. Long-range skip connections between the encoder and the decoder have been used to improve accuracy in other works like [26]. However, our architecture does not include these long-range skip connections as we did not obtain any empirical improvement. Fig. 1 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).

TABLE II

LAYER DISPOSAL OF OUR PROPOSED NETWORK (ERFNET). "OUT-F": NUMBER OF FEATURE MAPS AT LAYER'S OUTPUT. "OUT-RES": OUTPUT RESOLUTION FOR AN EXAMPLE INPUT SIZE OF 1024×512

	Layer	Type	out-F	out-Res
ENCODER	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
DECODER	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

The 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 precision (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. Our downsampler block, inspired by the initial block of ENet [11], performs downsampling by concatenating the parallel outputs of a single 3×3 convolution with stride 2 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 [27] in our non-bt-1D layers to gather more context, which led to an improvement in accuracy in our experiments. This technique has been proven more effective (in terms of computational cost and parameters) than using larger kernel sizes. In Table II, for those blocks that are marked as "dilated", we change the second pair of 3×1 and 1×3 convolutions for a pair of dilated 1D convolutions. We also include Dropout [28] in all our non-bt-1D layers 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.

The decoder segment is composed of the layers from 17 to 23. Its main task is to upsample the encoder's feature maps to match the input resolution. While SegNet had a relatively symmetric encoder-decoder shape (i.e. decoder of equal size to encoder), we follow a similar strategy to ENet in having a small decoder whose only purpose is to upsample the encoder's output by fine-tuning the details. In contrast to SegNet and ENet, we do not use max-unpooling operation for the upsampling. Instead, our architecture includes simple deconvolution layers with stride 2 (also known as transposed

convolutions or full-convolutions). The main advantage of using deconvolutions is not requiring to share the pooling indexes from the encoder. Therefore, deconvolutions simplify memory and computation requirements. In addition, we empirically obtained similar (or slightly better) accuracy.

IV. Experiments

We conduct a set of experiments to demonstrate the potential of our factorized residual layers and the high accuracy-efficiency trade-off of our proposed segmentation architecture.

A. General Setup

We use the Cityscapes dataset [13], 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, a 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. There are also available 20K coarsely annotated images that we did not use in our experiments. We train our models on the train set uniquely (fine annotations), without using the validation set for training. All accuracy results are reported using the commonly adopted Intersection-over-Union (IoU) metric:

$$IoU = \frac{TP}{TP + FP + FN} \tag{4}$$

where TP, FP and FN are respectively the number of true positives, false positives and false negatives at pixel level. This can be calculated for each specific class, or as an average value for all the 19 classes (Class-IoU) or the 7 categories (Cat-IoU).

All experiments are conducted using the Torch7 framework [29] with CUDA and CuDNN backends. Our model is trained using the Adam optimization [30] 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, in order to accelerate convergence. The code for training and evaluation is publicly available at **https://github.com/Eromera/erfnet**.

B. Comparison of Residual Layers

To analyze the benefits of our proposed non-bottleneck-1D block in a known architecture, we experiment by replacing the different residual layers in an existing segmentation network (ENet [11]) and training them on the same conditions on the train set. We equally evaluate the effect of using three different layer designs as described in Sec. III-A: the bottleneck (bt) [6], the non-bottleneck (non-bt) [6] and our proposed non-bottleneck-1D (non-bt-1D) designs. ENet is composed of layers with the bottleneck design (Fig. 2 (b)), which uses 1×1 convolutions to reduce the number of feature maps computed internally in the 3×3 convolution by a factor of 4 (relative to the layer input). Therefore, for a fair comparison, the non-bottleneck designs should receive 4 times less feature maps at the layer's input. We compare this by equally reducing, all feature maps used in the original network by a factor of 4 with respect to its original value in ENet (denoted with an x). As a second experiment, we directly replace the bottleneck design with the non-bottleneck one (which does not internally reduce feature maps), supposing an effective 4x increase of the computed feature maps (which is equally comparable to increasing input features in the bottleneck design by $4\times$). For these 6 cases, we train on the Cityscapes train set and evaluate on the validation set, at a resolution of 512×256.

TABLE III

Results by Replacing Layers in an Existing Segmentation Network (ENet). #FM: Number of Feature Maps at the Block's Input. (Vl-IoU, Tr-IoU): Avg. Class-IoU on the Cityscapes (Validation, Train) Sets (Max. Value After 200 Epochs). #Par: Number of Parameters. fwt: Forward Pass Time in Seconds

Module	#FM	Vl-IoU	Tr-IoU	#Par	fwt [s]
bt (orig.)	x	52.3%	77.7%	626K	0.013
non-bt	x/4	49.7%	75.1%	903K	0.009
non-bt-1D	x/4	50.1%	74.2%	454K	0.007
bt	4x	55.79%	99.3%	9.89M	0.032
non-bt	x	55.43%	98.8%	13.2M	0.027
non-bt-1D	x	58.37%	96.3%	6.87M	0.021

Results of this experiment are displayed in Table III. Both Validation (Vl) and Train (Tr) IoU measures are displayed to better analyze the network learning capacity of the evaluated models. The number of parameters (#Par) is calculated as the total number of weights and biases of each network. "fwt" is the forward-pass time in seconds for each model. Top side of the table compares the original ENet version (bt) against equivalent networks with both non-bt designs (by reducing x/4 the original network feature maps), while bottom side of the table compares the effect of replacing the bt block with non-bt versions directly (equivalent to using $4\times$ more feature maps at the bottleneck's input). Additionally, Fig. 3 shows the Training and Validation errors (Loss value used as training criterion) during training of "bt (original)" compared to replacing the residual layers with: bt-$4\times$, non-bt and non-bt-1D. For clarity, only epochs 1-120 are shown, and non-bt (x/4) and non-bt-1D (x/4) versions are not displayed, as these are not related to the presented architecture.

Analysis: As seen in Table III and Fig. 3, the higher accuracy of the architectures with wider layers (those presented in the bottom side of the table) demonstrates that computing more feature maps per layer allows networks to better approximate the loss functions that they are trying to learn. The extremely high Tr-IoU (and low train error) achieved by these wider networks indicates that they have enough capacity to approximate the complex differences between the 19 classes on the train set with almost perfect accuracy. Therefore, making a network wider is an effective way of increasing the learning capacity of a model. On this regard, our proposed non-bottleneck-1D design is the most effective choice to increase capacity of a network with the lowest impact on efficiency. Switching a bottleneck design to a non-bottleneck increases parameters by a large factor ($4\times$ more filters that results on an effective $16\times$ increase on the number of parameters). With the non-bt-1D design, this increase is equivalently done with a

TABLE IV

PER-CLASS IOU (%) RESULTS OF OUR ARCHITECTURE ON THE CITYSCAPES VALIDATION (TOP) AND TEST (BOTTOM) SETS. 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)

ERFNet	Roa	Sid	Bui	Wal	Fen	Pol	TLi	TSi	Veg	Ter	Sky	Ped	Rid	Car	Tru	Bus	Tra	Mot	Bic	Class	Cat
Scratch (VAL)	97.4	80.6	90.3	55.8	50.1	57.5	58.6	68.2	90.9	61.2	93.1	73.0	53.2	91.8	59.1	70.1	66.7	44.9	67.1	70.0	86.0
Pretrained (VAL)	97.5	81.4	90.9	54.6	54.1	59.8	62.5	71.6	91.3	62.9	93.1	75.2	55.3	92.9	67.0	77.4	59.8	41.9	68.4	71.5	86.9
Scratch (TEST)	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	68.0	86.5
Pretrained (TEST)	97.9	82.1	90.7	45.2	50.4	59.0	62.6	68.4	91.9	69.4	94.2	78.5	59.8	93.4	52.3	60.8	53.7	49.9	64.2	69.7	87.3

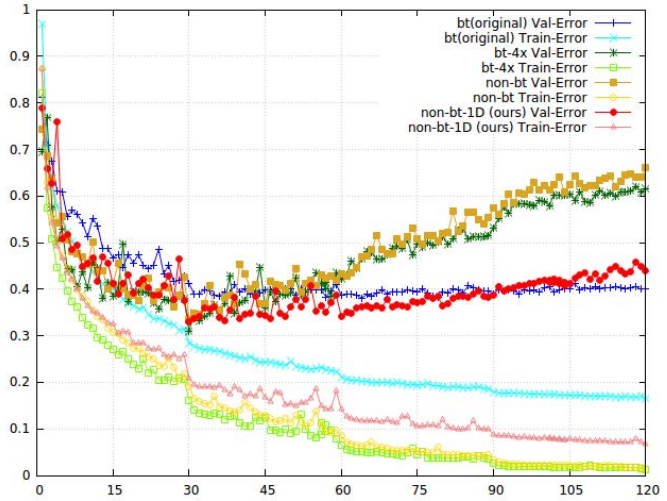


Fig. 3. Training and Validation Error during training between epochs 1 and 120, of ENet [11] "bt (original)" compared to replacing the residual layers with: bt-4x, non-bt and non-bt-1D (ours).

reduced parameter increase of 11× compared to increasing it with the bottleneck design (33% reduction). This reduction is even larger compared to the original non-bottleneck design. Despite of this high increase in parameters, the impact in computational cost is minimal (only 1.5× slower relative to the original bottleneck), and much faster than bottleneck and non-bottleneck versions with increased input features.

On the other hand, the large gap between Training and Validation errors, and the positive slope of the Val error (after a certain epoch) in Fig 3 indicates that the wider networks are using part of the increased capacity to overfit the train set. Although a similar validation accuracy between non-bt and non-bt-1D was expected, the results in Table I show a higher accuracy on the non-bt-1D case. This contrasts with their slightly lower result on the train set, which might indicate that the proposed non-bt-1D layers are in fact better at leveraging the increased learning capacity, by having better regularization. This property of the 1D factorization was already reported in [22]. Despite this better regularizing behavior of our proposed layer, the model still overfits, which indicates that the increased capacity is still not effectively translated into better accuracy. In the case of ERFNet, we leveraged this fact by designing an architecture that fully translates the increased capacity provided by our block into an accuracy improvement.

C. Evaluation of the Proposed Architecture

To evaluate the potential of our architecture, we measure its accuracy quantitatively on the widely used Cityscapes dataset.

TABLE V

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 [9]	ImageNet	73.6	87.9	n/a
FRRN [10]	-	71.8	88.9	n/a
Adelaide-cntxt [33]	ImageNet	71.6	87.3	35+
Deeplabv2-CRF [8]	ImageNet	70.4	86.4	n/a
LRR-4x [19]	ImageNet	69.7	88.2	n/a
ERFNet (pretrained)	**ImageNet**	**69.7**	**87.3**	**0.024**
ERFNet (scratch)	**-**	**68.0**	**86.5**	**0.024**
Dilation10 [27]	ImageNet	67.1	86.5	4.0
DPN [34]	ImageNet	66.8	86.0	n/a
Scale inv.+CRF [35]	ImageNet	66.3	85.0	n/a
FCN-8s [5]	ImN+Pasc	65.3	85.7	0.500
Uhrig et al [36]	ImageNet	64.3	85.9	n/a
DeepLab [37]	ImageNet	63.1	81.2	4.0
CRFasRNN [18]	ImageNet	62.5	82.7	0.700
SQ [12]	ImageNet	59.8	84.3	0.060
ENet [11]	-	58.3	80.4	0.013
SegNet basic [16]	ImageNet	57.0	79.1	0.060
SegNet extended [16]	ImageNet	56.1	79.8	0.060

We train encoder and decoder in separate steps, training first the encoder uniquely and then attaching the decoder to continue training the full architecture. To train the encoder we consider two strategies: "from scratch", where we only use Cityscapes images; and "pretrained", where the weights are initialized by training the network using a larger dataset such as ImageNet [31]. In the "from scratch" strategy, we train the encoder with downsampled (1/8 size) segmentation annotations from Cityscapes by attaching an extra convolutional layer at the end of the encoder. Once the encoder is trained, we remove this last layer and attach the decoder to train the full network (using Cityscapes). In the "pretrained" strategy, we first adapt the encoder's last layers to produce a single classification output by adding extra pooling layers and a fully connected layer, and then, train the modified encoder on ImageNet [31] (1000 classes). Once this modified encoder is trained, we remove the extra layers, attach the decoder, and train the full network using Cityscapes. On both approaches, we perform simple data augmentation at training time by doing random horizontal flips and translations of 0-2 pixels in both axes. The dataset resolution is 2048 × 1024, and all accuracy results are reported at this resolution. We train our model to perform inference at 1024 × 512, but the output is rescaled (by simple interpolation) to the original dataset resolution for evaluation. Table IV shows the inference results

TABLE VI

PER-CLASS IOU (%) ON CITYSCAPES TEST SET FOR ERFNET COMPARED TO THE FASTEST NETWORKS. *CLASSES ARE THE SAME AS IN TABLE IV

Network	Roa	Sid	Bui	Wal	Fen	Pol	TLi	TSi	Veg	Ter	Sky	Ped	Rid	Car	Tru	Bus	Tra	Mot	Bic	Class	Cat
SegNet [16]	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 [11]	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 [12]	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
ERFNet	**97.9**	**82.1**	**90.7**	**45.2**	**50.4**	**59.0**	**62.6**	**68.4**	**91.9**	**69.4**	**94.2**	**78.5**	**59.8**	**93.4**	**52.3**	**60.8**	**53.7**	**49.9**	**64.2**	**69.7**	**87.3**

TABLE VII

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

Model	NVIDIA TEGRA TX1 (Jetson)						NVIDIA TITAN X (Maxwell)					
	ms	fps	ms	fps	ms	fps	ms	fps	ms	fps	ms	fps
	480x320		640x360		1280x720		640x360		1280x720		1920x1080	
SegNet [16]	757	1.3	1251	0.8	-	-	69	14.6	289	3.5	637	1.6
ENet [11]	47	21.1	69	14.6	262	3.8	7	135.4	21	46.8	46	21.6
SQ [12]	60	16.7	86	11.6	389	2.6	n/a					
ERFNet	93	10.8	141	7.1	535	1.9	12	83.3	41	24.4	88	11.4
	512x256		**1024x512**		**2048x1024**		**512x256**		**1024x512**		**2048x1024**	
ENet* [11]	41	24.4	145	6.9	660	1.5	7	142.9	13	76.9	49	20.4
ERFNet	85	11.8	310	3.2	1240	0.8	8	125.0	24	41.7	89	11.2

on both Val and Test sets of Cityscapes for both "from scratch" and "pretrained" models. Results are shown in IoU for each of the 19 classes and the average Class and Category IoU values.

Analysis: The results displayed in Table IV show that our architecture achieves an excellent accuracy in all classes, almost perfect (>90%) on the general classes (road, building, vegetation, sky, car), while still having a remarkable accuracy in challenging classes that are easily confused (e.g. wall, fence, truck, motorcycle). Most errors in the IoU of these challenging classes are due to confusion between elements in the same category (e.g. truck vs. bus or traffic light vs. sign), which explains why the Category IoU is significantly higher than the Class IoU. Besides, the IoU is a challenging metric because the small classes greatly affect the overall IoU: on small or distant objects, a few bad pixels suppose a great part of these classes, which greatly increases false predictions that affect IoU. In terms of per-class pixel accuracy (i.e. percentage of pixels correctly segmented), our experiments report over 95% mean pixel accuracy on the validation set, which means that this percentage of pixels has correct predicted labels.

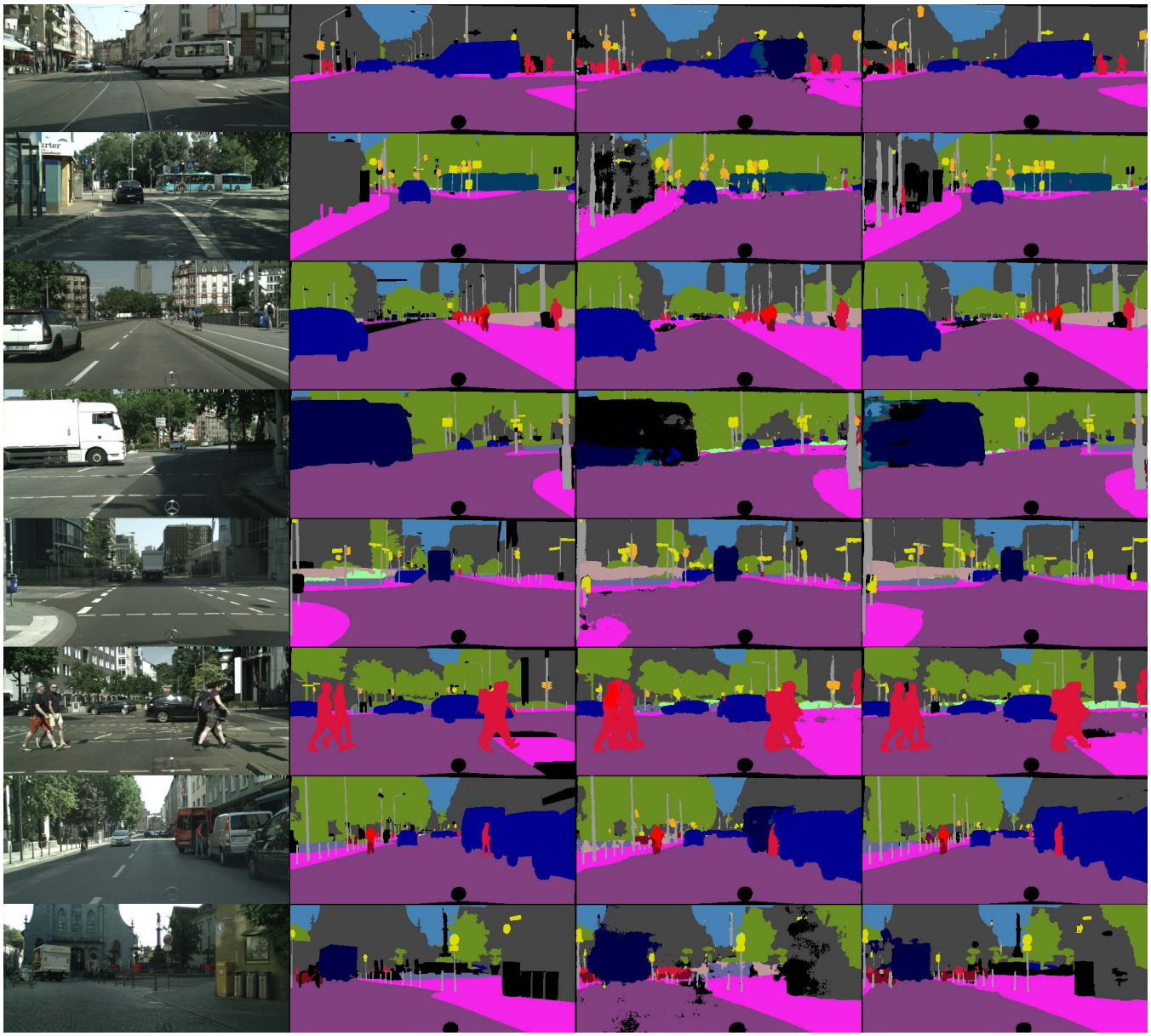
Additionally, the small difference (<2%) between Val and Test demonstrates that the model generalizes well to large amount of images like the 1525 images (test set) taken in different cities and conditions. About the results "from scratch" and "pretrained", we reported results on both due to the growing concern on industry related to models that can be only trained on a single dataset (e.g. a company's internal set of images). Although the transferability of features and the benefits of pretraining the model on a large dataset like ImageNet has been demonstrated in the literature [32] and in our experiments (with higher IoU), the provided results demonstrate that our model can also reach good accuracy trained on a single dataset ("from scratch"), without the need of pretraining on large datasets, which adds training complexity and may suppose commercial limitations.

D. Comparison to the State of the Art

Table V 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). "Pretrain" refers to the models that have 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. Table VI displays the results on every one of the 19 classes evaluated on the Cityscapes test set at 2048 × 1024 for our architecture compared to the ones that have the fastest reported speed in the benchmark.

Analysis: This comparison reflects that our architecture achieves the best available trade-off between accuracy and efficiency, obtaining 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. Our architecture ERFNet, achieves a 69.7% Class IoU and a 87.3% Category IoU on the Cityscapes Test set, which supposes a similar accuracy to the state of the art, while taking only 24 ms per image on a single GPU, which makes it one of the fastest networks available.

Compared to top-accuracy networks, ERFNet achieves a similar accuracy in both class and category IoU, while being significantly faster. 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 [9] and FRRN [10] 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 [37] model; And LRR-4× [19] 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



(a) (b) (c) (d)

Fig. 4. Qualitative examples of the segmentation produced by our architecture ERFNet (d) compared to the ground truth labels (b) and ENet [11]. (a) Input image. (b) Ground truth. (c) ENet [11]. (d) ERFNet (ours).

comparable in efficiency to the proposed network, which achieves the best available trade-off between segmentation accuracy and computational resources.

Compared to fastest architectures, our network achieves the second fastest speed while being significantly superior in accuracy. The per-class accuracy values shown in Table VI show that 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, Pedestrian, Rider, Truck, Bus, Train, Motorbike and 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). Therefore, our network highly improves on these because its improved learning capacity allows it to learn better from the same amount of samples.

E. Computational Resources

Table VII displays inference time (forward pass) for different resolutions on a single Tegra TX1 (Jetson platform) and on a single NVIDIA Titan X (Maxwell), compared to other architectures that had these results available in their papers. All times on both GPUs are considered using FP16 (half-precision floating point), although this is only leveraged by

the Tegra TX1 due to GPU-architectural reasons.[1] For a fair comparison, the values are displayed on the same set of resolutions as it was used in their papers (top side of the table). We also display results on the same aspect ratio as the Cityscapes dataset (bottom side). The exact values for SegNet, ENet and SQ are obtained from their papers, while ENet* values were obtained reproducing their model, in order to display results in the Cityscapes ratio for an easier comparison.

Analysis: At 640×360, a resolution that is enough to recognize any urban scene accurately, our network achieves over 83 FPS on a single Titan X and over 7 FPS on a Tegra TX1, an embedded GPU that uses less than 10 Watts at full load. At 1024×512 (the ratio used in the Cityscapes tests), our network achieves 24ms (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 a single GPU to provide real-time and accurate full-image semantic segmentation. On the embedded GPU, our network also achieves several FPS, which allows execution in real scenarios like intelligent vehicles. Although the common practice only considers 30 FPS as real-time, the provided values must be considered as detections per second, or times per second that the model provides full-image segmentation. This means that higher-level algorithms could already operate in real-time from a decent amount of detections per seconds, because objects could be tracked between frames with tracking techniques. Additionally, compression techniques (e.g. binarization of weights) were not considered in this work but they will be available in the near future and will allow further increase in speed.

F. Quality of the Segmentation

Fig. 4 shows various examples of segmentation produced by our architecture ERFNet (d) and ENet (c), compared to the ground truth (b). These images are from the Cityscapes validation set.

Analysis: These results demonstrate that our architecture yields consistent qualitative results for all classes, even at far distances in the scene. While both networks can accurately segment the road that is inmediately 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). As stated before, 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%, which can be well appreciated in the qualitative results. Despite the lower accuracy on specific challenging classes like "train" or "wall", the network already has an excellent accuracy on the main important categories like "road", "pedestrians" or "vehicles". This makes the network suitable for IV applications like self-driving cars, as it can already provide accurate and complex scene understanding to higher level algorithms like navigation.

[1]The Titan X (Maxwell) does not have native FP16 compute support, so the difference between FP16 and FP32 is negligible in the Titan X because Maxwell architecture is not able to translate it into an effective speed improvement. This is not the case for the new Pascal-based Titan X, which was not used in this work.

V. Conclusion and Future Works

In this paper, we have proposed an architecture that achieves accurate and fast pixel-wise semantic segmentation. In contrast to top-accuracy approaches in the state of the art, that develop complex architectures that are computationally expensive, and in contrast to the alternative efficient architectures, that perform significant sacrifices in the network design to gain efficiency in exchange of accuracy, our approach is focused on improving the core elements of our architecture: the convolutional blocks. We propose a re-design of the commonly used residual layers to make them more efficient while retaining a similar learning performance. While this design can be used in existing architectures to make them more efficient, we propose a full architecture that completely leverages its benefits to reach state-of-the-art segmentation accuracy and efficiency. Our experiments demonstrate that the resulting architecture provides an excellent trade-off between reliability and speed, which makes it suitable for countless IV applications such as scene understanding in self-driving vehicles, that require both robustness and real-time operation.

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).

Acknowledgement

The authors thank NVIDIA for generous hardware donations.

References

[1] E. Romera, L. M. Bergasa, and R. Arroyo. (Jul. 2016). "Can we unify monocular detectors for autonomous driving by using the pixel-wise semantic segmentation of CNNs?" [Online]. Available: https://arxiv.org/abs/1607.00971

[2] X. He, R. S. Zemel, and M. Á. Carreira-Perpiñán, "Multiscale conditional random fields for image labeling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2. Jun. 2004, pp. II-695–II-702.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. South Lake Tahoe, NV, USA: Curran Associates, 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

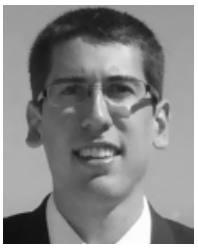
[5] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[6] K. He, X. Zhang, S. Ren, and J. Sun. (Dec. 2015). "Deep residual learning for image recognition." [Online]. Available: https://arxiv.org/abs/1512.03385

[7] S. Zagoruyko and N. Komodakis. (May 2016). "Wide residual networks." [Online]. Available: https://arxiv.org/abs/1605.07146

[8] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. (Jun. 2016). "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs." [Online]. Available: https://arxiv.org/abs/1606.00915

[9] G. Lin, A. Milan, C. Shen, and I. Reid. (Nov. 2016). "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation." [Online]. Available: https://arxiv.org/abs/1611.06612
[10] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. (Nov. 2016). "Full-resolution residual networks for semantic segmentation in street scenes." [Online]. Available: https://arxiv.org/abs/1611.08323
[11] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. (Jun. 2016). "ENet: A deep neural network architecture for real-time semantic segmentation." [Online]. Available: https://arxiv.org/abs/1606.02147
[12] M. Treml, J. Arjona-Medina, T. Unterthiner, and, "Speeding up semantic segmentation for autonomous driving," in *Proc. MLITS, NIPS Workshop*, 2016, p. 8.
[13] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3213–3223.
[14] E. Romera, J. M. Álvarez, L. M. Bergasa, and R. Arroyo, "Efficient ConvNet for real-time semantic segmentation," in *Proc. IEEE Intell. Veh. Symp. (IV)*, Jun. 2017, pp. 1789–1794.
[15] K. Simonyan and A. Zisserman. (Sep. 2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556
[16] V. Badrinarayanan, A. Handa, and R. Cipolla. (May 2015). "SegNet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling." [Online]. Available: https://arxiv.org/abs/1505.07293
[17] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. (Dec. 2014). "Semantic image segmentation with deep convolutional nets and fully connected CRFs." [Online]. Available: https://arxiv.org/abs/1412.7062
[18] S. Zheng *et al.*, "Conditional random fields as recurrent neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1529–1537.
[19] G. Ghiasi and C. C. Fowlkes, "Laplacian pyramid reconstruction and refinement for semantic segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 519–534.
[20] Z. Wu, C. Shen, and A. van den Hengel. (Apr. 2016). "High-performance semantic segmentation using very deep fully convolutional networks." [Online]. Available: https://arxiv.org/abs/1604.04339
[21] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. (Mar. 2016). "XNOR-net: ImageNet classification using binary convolutional neural networks." [Online]. Available: https://arxiv.org/abs/1603.05279
[22] J. Alvarez and L. Petersson. (Jun. 2016). "DecomposeMe: Simplifying ConvNets for end-to-end learning." [Online]. Available: https://arxiv.org/abs/1606.05426
[23] M. Jaderberg, A. Vedaldi, and A. Zisserman. (May 2014). "Speeding up convolutional neural networks with low rank expansions." [Online]. Available: https://arxiv.org/abs/1405.3866
[24] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning separable filters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2754–2761.
[25] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1026–1034.
[26] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio. (Nov. 2016). "The one hundred layers tiramisu: Fully convolutional DenseNets for semantic segmentation." [Online]. Available: https://arxiv.org/abs/1611.09326
[27] F. Yu and V. Koltun. (Nov. 2015). "Multi-scale context aggregation by dilated convolutions." [Online]. Available: https://arxiv.org/abs/1511.07122
[28] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. (Jul. 2012). "Improving neural networks by preventing co-adaptation of feature detectors." [Online]. Available: https://arxiv.org/abs/1207.0580
[29] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *Proc. BigLearn, NIPS Workshop*, 2011, pp. 1–6.
[30] D. Kingma and J. Ba. (Dec. 2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980
[31] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[32] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Adv. Neural Inf. Process. Syst.*, 2014, pp. 3320–3328.
[33] G. Lin, C. Shen, A. Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3194–3203.
[34] Z. Liu, X. Li, P. Luo, C.-C. Loy, and X. Tang, "Semantic image segmentation via deep parsing network," in *Proc. IEEE Int. Conf. Comput. Vis.*, Jun. 2015, pp. 1377–1385.
[35] I. Krešo, D. Čaušević, J. Krapac, and S. Šegvić, "Convolutional scale invariance for semantic segmentation," in *Proc. German Conf. Pattern Recognit. (GCPR)*, 2016, pp. 64–75.
[36] J. Uhrig, M. Cordts, U. Franke, and T. Brox. (Apr. 2016). "Pixel-level encoding and depth layering for instance-level semantic labeling." [Online]. Available: https://arxiv.org/abs/1604.05096
[37] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille. (Feb. 2015). "Weakly- and semi-supervised learning of a DCNN for semantic image segmentation." [Online]. Available: https://arxiv.org/abs/1502.02734
[38] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3234–3243.



Eduardo Romera received the B.S. and M.S. degrees in telecommunications engineering and the M.S. degree in electronics from the University of Alcalá (UAH), Spain, in 2014 and 2015, respectively. He is currently pursuing the Ph.D. degree, with a focus on deep learning applied to scene understanding for self-driving vehicles, with the RobeSafe Group, Department of Electronics, UAH. His first M.S. thesis was fulfilled as Erasmus at the Karlsruhe Institute of Technology, Germany, while he was a Research Assistant with the Fraunhofer IOSB Research Centre. He is also a Researcher with the RobeSafe Group, Department of Electronics, UAH. He has also done a research internship with CSIRO, Australia. His research is focused on computer vision and intelligent transport systems.

José M. Álvarez received the Ph.D. degree, with a focus on developing robust road detection algorithms for everyday driving tasks under real-world conditions, from the Autonomous University of Barcelona (UAB) in 2010. He was a Post-Doctoral Researcher with the Courant Institute of Mathematical Science, New York University. He is currently a Computer Vision Researcher with CSIRO-Data61, where he is involved in efficient methods for large-scale dynamic scene understanding and deep learning. He received the Best Ph.D. Thesis Award from UAB in 2010. Since 2014, he has been an Associate Editor of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.

Luis M. Bergasa received the Ph.D. degree in electrical engineering from the University of Alcalá (UAH), Madrid, Spain, in 1999. He is currently a Full Professor and the Director of Technology Transfer with UAH. He is the author of over 170 refereed papers in journals and international conferences and the corresponding author of seven national patents and one PCT. He has served on program/organizing committees over 20 conferences. He was a recipient of 20 prizes related to robotics and automotive fields. He is an Associate Editor of the IEEE TRANSACTIONS ON ITS.

Roberto Arroyo received the B.S. and M.S. degrees in computer science and the M.S. and Ph.D. degrees in electronics from the University of Alcalá (UAH), Madrid, Spain, in 2009, 2012, 2013, and 2017, respectively. He did two research internships at Toshiba Research Europe, Cambridge, U.K., in 2014, and at the Australian Centre for Field Robotics, Sydney, Australia, in 2016. He also collaborates as a Teacher Assistant at UAH. He has authored or co-authored several works in the main conferences and journals related to autonomous vehicles, robotics, and computer vision.