Image Segmentation on Embedded Systems via Superpixel Convolutional Networks

Simone Mentasti[1] and Matteo Matteucci[1]

***Abstract*— In this paper we describe a lightweight framework for fast image segmentation on embedded systems, based on superpixels, which leverages on convolutional and graph-convolutional neural networks. In particular, we analyzed different superpixel representation looking for the best trade-off between the efficiency of the system and richness of the description. Similarly, we analyzed different network sizes, balancing the number of filters used and the prediction accuracy. We also compared two different convolutional architecture, one based on the classical encoder-decoder paradigm and one based on graphs, to guarantee a most accurate representation of the image structure. The architecture was tested on the KITTI dataset using an embedded system with CUDA capabilities.**

I. INTRODUCTION

Image segmentation is a crucial task in many fields, from robotics to medicine. Although the common goal is to classify each pixel of an image, different domains present different requirements and challenges. In the medical field, accuracy can be seen as one of the main goals, while in robotics this aspect has to be appropriately adjusted trading-off speed and precision. In small robots, with limited payload and battery powered, we are forced to use simple algorithms due to the low computational budget available. Autonomous cars do not have such limitation, but, in light of the future industrialization of the product estimated for the first years of the next decade [1], it is becoming evident the importance of reducing the cost of the additional equipment needed by these vehicles. We can achieve this goal in different ways, e.g., by reducing the number of sensors, in particular LIDARs, or by reducing the computational power needed to process data [2].

One of the most significant drawbacks of neural networks is indeed the high demand in terms of resources to process camera images in real time. Traditional image segmentation neural networks rely on high-end GPUs [3] [4] or do not consider as mandatory predicting the data at camera frame rate [5]. Nevertheless, autonomous driving cars rely heavily on these networks to accomplish the task of road segmentation to identify different elements of the vehicle surrounding, like drivable areas, road lanes, and obstacles, either cars or pedestrians (see Figure 1). With the upcoming industrialization of autonomous or semi-autonomous cars, it is not feasible to load vehicles with big and expensive GPUs to process data; then it is necessary to develop new lightweight approaches to the image segmentation problem.

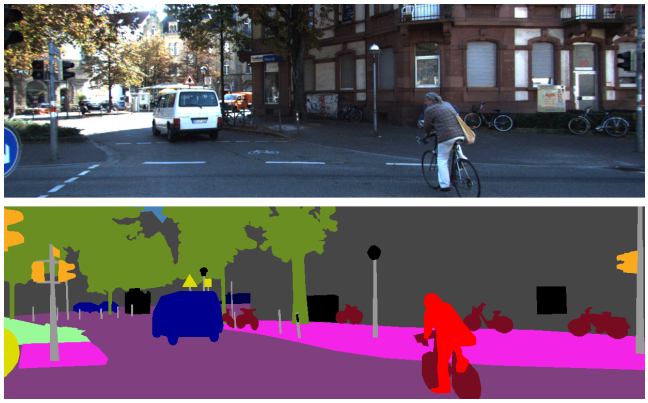

Fig. 1. Example image from the KITTI dataset. On top the original picture and on the bottom the ground truth segmentation.

The easiest way to achieve this goal is to process smaller images, but the risk with this approach is to lose crucial details, in particular elements like lines and markings can be easily lost if the picture is excessively downsampled. Our approach to this problem is to introduce a pre-processing phase to reduce the amount of data fed to the network. In particular, we segment the image using superpixels then, from each element, we extract a set of features and then feed the neural network with a tensor having size equal to the number of superpixels. Using this pre-processing step, the input of the system becomes hundreds of times smaller, making the whole prediction process extremely fast. The superpixel segmentation, unlike a simple grid subsampling, preserves edges and objects' shapes making the correct prediction of each pixel as accurate as a traditional convolutional neural network.

What we propose in this paper is then a framework composed by a pre-processing, phase based on superpixels, and two network architectures for real-time image segmentation on embedded systems. One approach uses a classical U-net style convolutional neural network while the other implements a graph convolutional neural network to achieve more accurate data representation. To test our algorithm, we use a low power device with CUDA capabilities, the Nvidia Jetson Tx2 board, which has already proved to perform ex-

Partially supported by project TEINVEIN: TEcnologie INnovative per i VEicoli Intelligenti, CUP (Codice Unico Progetto - Unique Project Code): E96D17000110009 - Call "Accordi per la Ricerca e l'Innovazione", cofunded by POR FESR 2014-2020 (Programma Operativo Regionale, Fondo Europeo di Sviluppo Regionale Regional Operational Programme, European Regional Development Fund).

[1]Simone Mentasti and Matteo Matteucci are with the Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milano, Italy simone.mentasti@polimi.it, matteo.matteucci@polimi.it

978-1-7281-3605-9/19/$31.00 ©2019 IEEE

ceptionally well in deep learning scenarios [6], in particular in time depending image segmentation task [7]. To accurately benchmark our results on the task of road segmentation, we trained our network using KITTI dataset [8].

This paper is structured as it follows; in the first section, we illustrate the general architecture of the system, then we proceed analyzing the quality of the segmentation performed using different descriptors for the processed image. Next, we analyze different versions of the neural network and number of extracted features to find the best trade-off between speed and accuracy. Lastly, we compare the results obtained with this architecture with a custom implementation of a graph convolutional neural network.

II. RELATED WORKS

In this work, we focus on real-time image segmentation in roads scenario. During past years, researchers have proposed different architectures to this purpose using mainly convolutional neural networks [9] [10] [11] [12]. They show significant results, but to achieve such precision they relay on power-hungry GPUs, creating big networks with hundreds of millions of parameters. Most of the top scores on KITTI [8] [13] are more focused on the precision in prediction than on time requirements and hardware setup.

Superpixel based classification is nowadays a consolidated approach [14], but most of its use is limited as support to classical computer vision algorithms [15] [16]. Due to the growing interest in superpixel segmentation as a pre-processing phase in different computer vision toolchains nowadays it is possible to find optimized implementations of this algorithm [17] [18] working on GPU.

Only recently we assisted to the combined use of superpixel and convolutional neural networks,in an architecture called SP-CNN, but its use has been limited. In particular the most remarkable results concern cloud classification [19] or aerial pictures classification [20]. This type of segmentation performs exceptionally well in such scenarios due to the texture in the acquired images. In particular, areal images of cites are, most of the time, segmented using road lines by the superpixel algorithm. Some studies showed how it is possible to divide an image in superpixels and than treat each element as a single object to be classified [21]. This approach loses some spatial information, but can be performed by a small network without the need to add a decoding phase like in U-net like segmentation network. Other methods stay closer to the classical image segmentation architecture, passing the whole set of superpixels to the system and then reconstructing the image [22].

Due to the structure of the superpixel algorithm, which allows each element to expand toward its neighbours, it is possible to model the connection between each superpixel using a graph instead of a matrix. Graph-based neural networks have been research a field for a while, [23], [24]. In general, this approach has been used on structured data where connections of elements of the input are highly significative, [25], [26], and the number of elements remains low. This explains why this approach is not generally used on image classification where the number of pixels to classify is considerably large and the connections between elements are really simple.

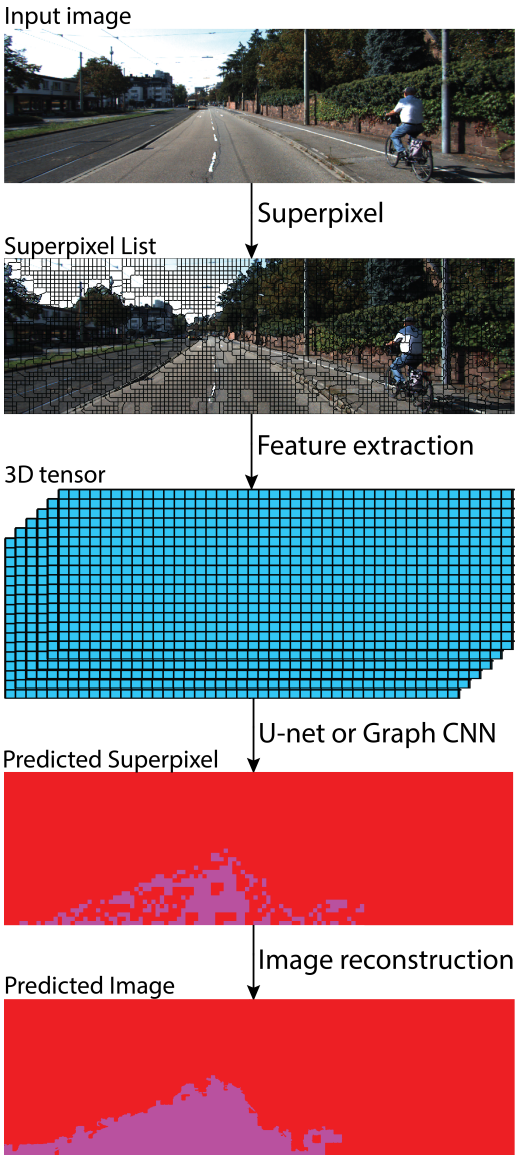

Fig. 2. Schema of our algorithm for road segmentation. As input, we have an image with size 1242x375 (Kitti images size); this picture is processed by the superpixel algorithm, which outputs a mask with shape 120x36 superpixels. For each superpixel, we extract a set of features, creating a 3D tensor that is fed to the neural network. Lastly, from the network predictions, we reconstruct the segmentation on the original input applying the predicted value to each superpixel in the mask.

III. PROPOSED METHOD

Our segmentation algorithm is divided into four blocks, shown in Figure 2; in the first block the image is acquired and processed by a uniform superpixel segmentation algorithm through which the system divides the picture into NxM superpixel areas. In the second phase, we extract for each superpixel a set of features. Then, we feed a convolutional neural network with a tensor of size NxM and depth equal to the number of features extracted. For this phase, we tested

two different approaches, one using classical U-net like network and one using a graph convolutional neural network (G-CNN). In particular, as stated in the previous section, it is possible to represent the connection between superpixels using a graph; this approach offers a better representation of the superpixel structure with respect to the 3D tensor used in the U-net. The G-CNN, differently from the U-net, takes as input a graph where each node is indeed represented by a list of features extracted from a superpixel and the arc between two nodes represents the adjacency between two superpixels in the image. Last we remap each superpixel value computed by the network to the original image.

A. *Superpixel segmentation and features extraction*

The first block of our framework consists in the superpixel segmentation. This pre-processing phase takes as input a picture and returns a segmentation mask where each pixel of the input image is linked to one of the NxM superpixels. The size in pixels of the single superpixel is not constant because it adapts around the particular texture, but the total number of superpixel in the image does not change so that it can be easily fed to a CNN. To ensure that the input of the network remains constant, the first step of the superpixel segmentation returns a uniform grid division of NxM size; then the algorithm iteratively adapts to the texture around. Using this procedure, we guarantee that, even if its surroundings completely incorporate a superpixel, we maintain a constant size input; simply some superpixel will have zero size. This operation is the most resource demanding of the whole system. For this reason, we have implemented it using CUDA, which allows reducing drastically the time needed to compute the superpixel.

To feed the neural network with enough information to correctly classify each superpixel, we proceed extracting a set of features for each of them. Previous works in SP-CNN have proposed complex sets of features, having in some cases almost one hundred of them [21] [27]. Our goal is to have a lightweight and efficient algorithm; then, we tested different feature sets looking for the best trade-off between execution time and precision, while limiting the dimensionality of the data.

For each region, we extract the mean RGB and HSV values, plus the number of pixels in the superpixel and the mean value of x and y components. Then we computed the histogram of the H component of HSV. During our analysis, we also tested different representations, adding more histograms or even LBP pattern descriptors. In the next section, we analyze in details the sets of features used and how different combinations impact on the precision of the segmentation.

B. *Network architectures*

The output of the previous step is a tensor with N rows, M columns and depth equal to the number of features. Considering the size of the single superpixel, around 100 pixels, the height and width of this tensor are one-tenth of the size of the original picture. This implies that we can not use traditional size encoding-decoding network like U-net [28], because after five steps of max-pooling the tensor size would be too small.

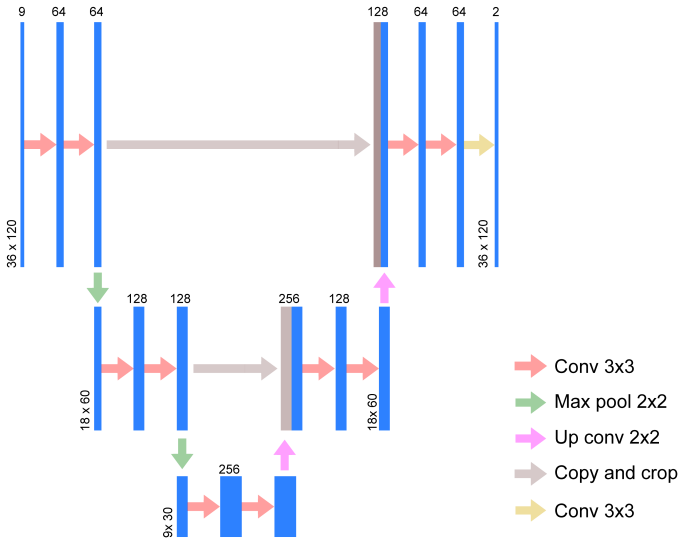

Fig. 3. Structure of the U-net. The architecture used is the classical encoder-decoder from the original paper but, due to the considerably smaller input, the depth of the network is reduced to just two max-pooling layers.

Differently from classical segmentation, our tensor has already some information, and the superpixel algorithm has already divided the image into similar areas. This means it is not necessary to have such a deep network.

Our convolutional architecture is based on a classical U-net, but we reduced the number of layers. In this way, the computational power needed to predict the superpixel label is extremely low, and the size of the tensors remains significative for the segmentation task. The structure of the network used for the kitty dataset is shown in Figure 3.

On the other hand, the graph convolutional neural network we used is based on work of [26]. The graph approach, differently from the previous, takes into consideration the connections between the superpixel removing those elements which have size zero due to the erosion process of the superpixel algorithm. In particular, the superpixel process fuses areas with similar characteristics, which are connected, into a single element, as shown in Figure 4. This means that we have an item with twice the size of a classical superpixel near one with size zero. When we train the network using the U-net style architecture the elements with zero size are still part of the input because we have fixed size inputs in convolutions.

The number of "fused" superpixels is considerably lower compared to the total amount of superpixels, but there is no reason apart from the fixed input size of the network to predict those elements. This is why we decided to test a second approach based on a graph convolutional neural network. Using a graph, we can specify the connection between the superpixels and we do not connect to the graph those elements with zero size, giving the network a more accurate representation of the picture structure.

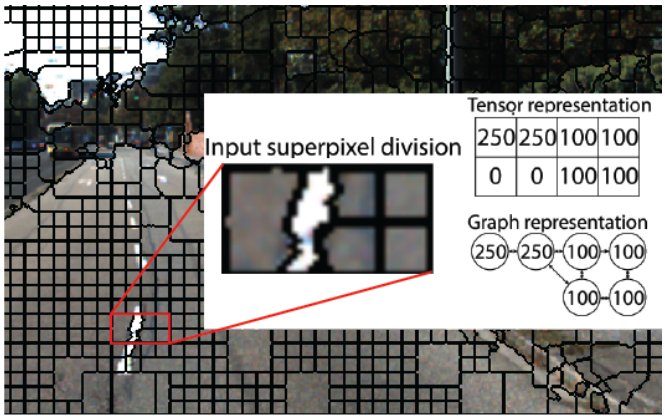

Fig. 4. Comparison between the Tensor representation (on the top), where superpixel fusion is not considered, and the graph representation (on the bottom) where zero size superpixel are not connected to other elements of the graph. The number inside the box and circle indicate the number of elements of a single superpixel.

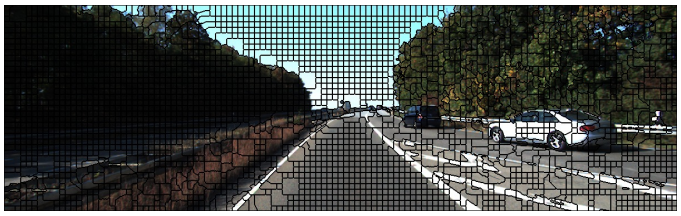

Fig. 5. KITTI picture processed with the superpixel algorithm.

The last phase of the segmentation toolchain is image reconstruction. The output of the network is indeed an NxM image where each superpixel has a predicted label. What we do next is parsing the original picture assigning to each pixel the correct label based on the superpixel mask computed in the first phase.

IV. RESULTS

To properly evaluate our algorithm we tested it on the KITTI benchmark dataset for road segmentation [8]. We first processed the entire KITTI dataset extracting for each image a set of superpixel as shown in Figure 5.

From a KITTI image with a size of 1242x375 pixels we extracted 120x36 superpixels. Then from each superpixel, we computed a set of features; in particular, we experimented with different representations. We started using only the mean RGB values of each superpixel and the number of pixels of each cluster; then we gradually added more features like different colour spaces, shape, texture descriptors and histograms. While RGB mean values take only three values, a complete histogram requires 256 different numbers for each channel. Due to the limited size of the network, it is counterproductive to use hundreds of features. For this reason, the histogram has been processed and compressed into a 16 values array taking the mean from a group of 16 consecutive elements of the original vector. A list of the tested parameter configurations is shown in Table I.

We also performed a set of tests using different superpixel sizes, but we quickly determined that a further increase of the superpixel size caused an excessive loss of accuracy, forcing also a smaller network architecture. While a reduction of superpixels size did not offer a significative improvement in the segmentation task. In particular, the chosen resolution was already able to correctly follow the shape of the object in the pictures, and smaller superpixels did not produce better superpixel segmentations while adding computational time to process a bigger tensor.

TABLE I

LIST OF FEATURES SETS USED FOR THE TESTS

num. of feature	feature
4	N. pixels, RGB mean
7	N. pixels, RGB mean, HSV mean
9	N. pixels,size x, size y, RGB mean, HSV mean
23	N. pixels, RGB mean, HSV mean, hist HSV
26	N. pixels, RGB mean, HSV mean, LBP, mean, hist H
39	index, size x,size y, N. pixels, RGB mean, H histogram, LBP histogram
42	index, size x, size y, N. pixels, RGB mean, HSV mean, H histogram, LBP histogram

This whole process of superpixel segmentation and feature extraction takes, on images of size 1242x375, at average 60 milliseconds using a GPU optimized algorithm [17] on a laptop equipped with Nvidia 840, while on the Jetson TX2 board the compute time is approximately 50 milliseconds (i.e., 20Hz.). The extraction of 3 features or a full set of 50 elements does not affect the elaboration time; the superpixel segmentation algorithm indeed requires most of the computational power.

Extracted features are converted into a tensor and fed to a convolutional neural network. As previously stated, we decided to use a classical architecture like U-net, but, due to the reduced size of the input images, we drastically reduced the number of layers as shown in Figure 3. Our architecture maintains the classical structure of the U-net, with a contracting path built using two 3X3 convolutions followed by a 2X2 max-pooling for each layer. While the expansive path is created using a 2X2 bilinear up-convolution, a concatenation with the corresponding feature of the contraction path, and two 3X3 convolutions. The main difference from the original U-net is the number of layers which is limited to three. Another important aspect is the shape of the input, which is no more an RGB tensor, which has depth 3, but is made using the extracted features and can have depth up to 42 elements. The network, thanks to its limited size, runs extremely fast and on the Nvidia Jetson TX2, requiring only four milliseconds for the forward pass.

To evaluate the different feature sets, we set up a test on the KITTI dataset; we started training the network using only the RGB descriptor, and then we gradually added more features up to 42 elements. For all the test we divided the

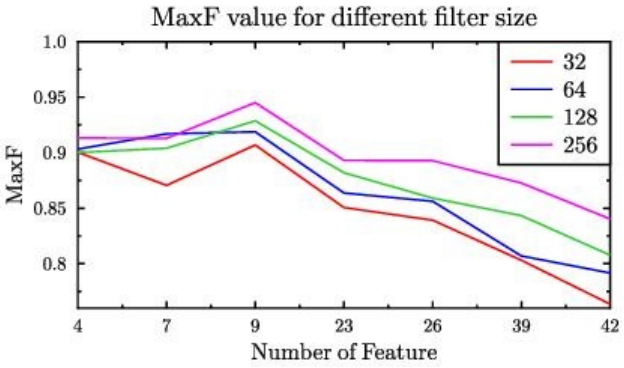

Fig. 6. Quality of the reconstruction as a function of used features. On the x-axis we have the number of features extracted; while on the y-axis we have the MaxF value, namely the harmonic average of the precision and recall. It is possible to notice how the MaxF value increases only for the first steps, decreasing with a value greater than 9. From this, we gathered that adding more feature to the representation does not guarantee better prediction.

Kitti dataset into a training set, 70% of the images, and a validation set, the remaining 30%. All the results shown refer to the MaxF value on the validation set, where MaxF stands for the F1 score, namely the harmonic average of the precision and recall. The results obtained, as a function of the extracted feature listed in Tabel I, are shown in Figure 6 for different convolutions' filter sizes. We notice that adding features increases the overall quality of the reconstruction up to a certain extent. Detailed representation of the superpixel, like his 16 element histogram can cause a loss in precision. The superpixel with 42 features produces a less accurate reconstruction than the one described only by his RGB mean values.

As reported in Figure 6, in our test we also analyzed the size of the network, experimenting a different number of filters; in particular, we analyzed the same architecture using 32, 64, 128 and 256 filters. We noticed that the architecture with 128 filters performs better in most situations, while the further increase to 256 filters does not produce a significant improvement. We can also notice that the use of simple shape descriptor, as the size in pixels on the x and y-axis, causes a significative improvement in precision, while the use of histograms generally causes a loss in quality. The image representations that use more features generally benefit from the increased complexity of the network but are not yet able to achieve the same results of the nine-features version. On the other hand, the increased number of connections significantly changes the time needed to predict an image. The best tradeoff between quality and performance we found is then the nine feature representation and a 64 filters U-Net. The complete performance of this configuration of the network on the Kitti dataset is shown in Table II divided into the three KITTI categories of road scenes: urban marked, urban marked multiline and urban unmarked. For an accurate description of the metrics used, we refer to the original Kitti paper [8]. These results place this framework within the top Kitti scores for road segmentation [29], [30], [13], despite

TABLE II

BEST ACHIEVED RESULT ON THE KITTI DATASET

metrics	um_road	umm_road	uu_road
MaxF	93.92	89.18	77.13
AvgPrec	85.55	80.65	57.80
PRE_wp	92.24	93.03	67.66
REC_wp	95.66	85.64	89.69
FPR_wp	1.85	2.13	6.67
FNR_wp	4.34	14.36	10.31

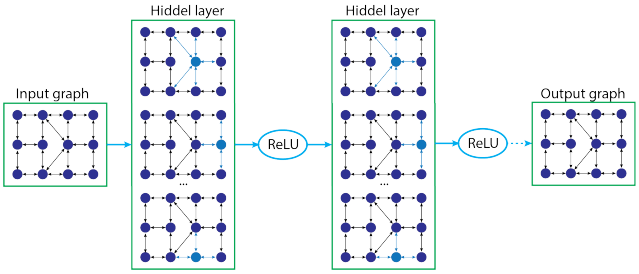

Fig. 7. Structure of the graph convolutional neural network based on [26], but with a considerably larger input due to the different application field.

TABLE III

RESULT ON THE KITTI DATASET USING GRAPH-CNN

metrics	um_road	umm_road	uu_road
MaxF	84.61	88.73	70.05
AvgPrec	70.82	79.33	49.81
PRE_wp	76.25	84.96	57.73
REC_wp	95.04	92.86	89.08
FPR_wp	5.86	4.93	10.80
FNR_wp	4.96	7.14	10.92

less performing hardware or computational time than most of the proposed network.

A. Graph convolutional neural network

Using the best performing feature set retrieved from the results on CNN we also trained a graph convolutional neural network to be consistent with the uneven topology of superpixels. The model of the system is based on the one proposed by [26], as reported in Figure 7.

The main difference from the Kipf and Welling architecture consist of the size of the input which, for us, is considerably larger. This has forced us to increase the number of hidden units from 16 in the original paper to 256. We also decreased the learning rate to achieve higher precision. The increased input size and number of parameters cause a significative change in the computational time of the network. In particular, the process of graph computing, executed only on CPU, requires 0.4 s at average; while the execution time of the network is comparable to the U-net version. This is because for these tests we did not implement the algorithm using CUDA.

We performed a similar analysis to the one proposed for U-net, using different sets of features; like in the previous tests, we found that the best performing feature sets are the one with seven and nine feature. The results of this architecture on the Kitti dataset, using the same features set as Table II are shown in Table III. Despite the better representation of

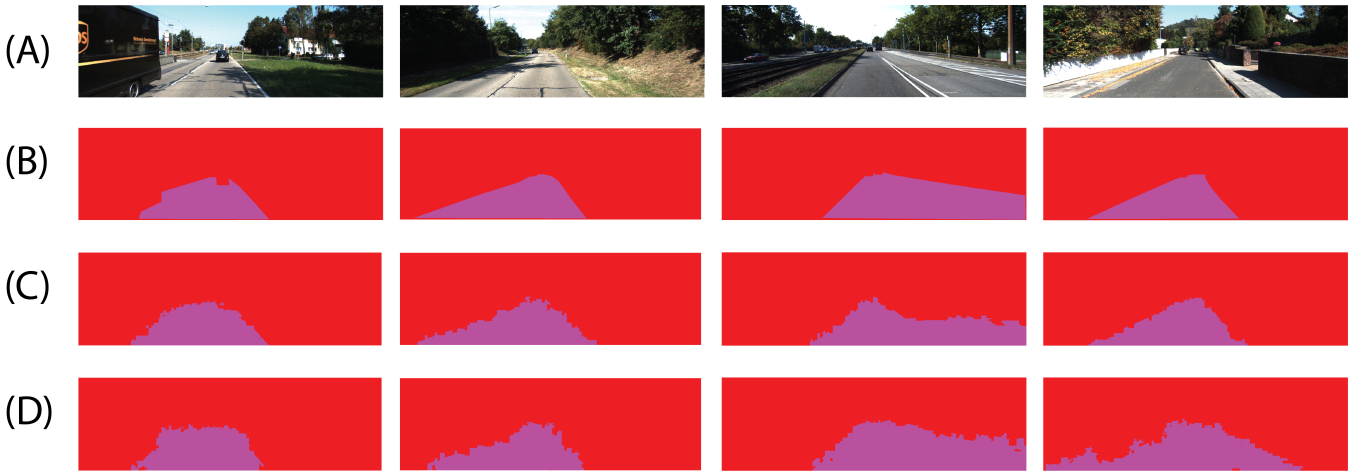

Fig. 8. Comparison between the kitti original image (A), kitti groud-truth (B), predicted image using U-net (C) and predicted image using graph CNN (D)

the input image, the network performs slightly worse than the U-net version. It also requires more time than the U-net approach to predict a single image.

V. CONCLUSIONS AND FUTURE WORKS

In this paper, we presented a framework based on convolutional networks on superpixels for fast road segmentation. The results achieved on the Kitti benchmark dataset show the potential of this approach in time-demanding embedded scenarios. The proposed solution can run in real time, i.e., 20 Hz, on a small, low-power device like a Jetson TX2. Despite this, the framework achieves good accuracy in prediction, with performance comparable to the state of the art on the Kitti benchmark dataset [13]. On the plus side, the comparison tests between multiple feature sets demonstrate how a simple representation performs considerably better on small and fast convolutional networks. For a visual example of the accuracy of our toolchain Figure 8 shows a comparison between the Kitti ground truth and the prediction from the two networks. For a more detailed analysis of the two networks performances, it is possible to compare the results of Table II and Table III.

Further improvements of this approach can lead to a generalization of the task of image segmentation on the multilabel segmentation dataset like Cityscape. Regarding the graph-based approach, a more optimized implementation of the graph creation process, using GPU parallelization could lead to a significative improvement in the prediction speed, making this architecture desirable for time-depending tasks too.

REFERENCES

[1] L. Jones, "Driverless when and cars: where? [automotive autonomous vehicles]," *Engineering Technology*, vol. 12, no. 2, pp. 36–40, March 2017.

[2] B. C. Zanchin, R. Adamshuk, M. M. Santos, and K. S. Collazos, "On the instrumentation and classification of autonomous cars," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2017, pp. 2631–2636.

[3] Z. Chen and Z. Chen, "Rbnet: A deep neural network for unified road and road boundary detection," in *Neural Information Processing*, D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, Eds. Cham: Springer International Publishing, 2017, pp. 677–687.

[4] J. Muoz-Bulnes, C. Fernandez, I. Parra, D. Fernndez-Llorca, and M. A. Sotelo, "Deep fully convolutional networks with random data augmentation for enhanced generalization in road detection," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017, pp. 366–371.

[5] R. Mohan, "Deep deconvolutional networks for scene parsing," *CoRR*, vol. abs/1411.4101, 2014.

[6] K. Rungsuptaweekoon, V. Visoottiviseth, and R. Takano, "Evaluating the power efficiency of deep learning inference on embedded gpu systems," in *2017 2nd International Conference on Information Technology (INCIT)*, Nov 2017, pp. 1–5.

[7] D. Yudin and D. Slavioglo, "Usage of fully convolutional network with clustering for traffic light detection," in *2018 7th Mediterranean Conference on Embedded Computing (MECO)*, June 2018, pp. 1–6.

[8] J. Fritsch, T. Khnl, and A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, Oct 2013, pp. 1693–1700.

[9] S. Yadav, S. Patra, C. Arora, and S. Banerjee, "Deep cnn with color lines model for unmarked road segmentation," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 585–589.

[10] I. Ardiyanto and T. B. Adji, "Deep residual coalesced convolutional network for efficient semantic road segmentation," in *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, May 2017, pp. 378–381.

[11] B. D. Brabandere, D. Neven, and L. V. Gool, "Semantic instance segmentation for autonomous driving," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, July 2017, pp. 478–480.

[12] N. Garnett, S. Silberstein, S. Oron, E. Fetaya, U. Verner, A. Ayash, V. Goldner, R. Cohen, K. Horn, and D. Levi, "Real-time category-based and general obstacle detection for autonomous driving," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, Oct 2017, pp. 198–205.

[13] K. I. of Technology. (2013) Road/lane detection evaluation. [Online]. Available: http://www.cvlibs.net/datasets/kitti/eval_road.php

[14] Ren and Malik, "Learning a classification model for segmentation," in *Proceedings Ninth IEEE International Conference on Computer Vision*, Oct 2003, pp. 10–17 vol.1.

[15] J. Ding, C. Lin, I. Lu, and Y. Cheng, "Real-time interactive image segmentation using improved superpixels," in *2015 IEEE International Conference on Digital Signal Processing (DSP)*, July 2015, pp. 740–744.

[16] D. Yang, J. Huang, J. Zhang, and R. Zhang, "Cascaded superpixel

pedestrian object segmentation algorithm," in *2018 Chinese Control And Decision Conference (CCDC)*. IEEE, 2018, pp. 5975–5978.

[17] C. Y. Ren, V. A. Prisacariu, and I. D. Reid, "gslicr: Slic superpixels at over 250hz," *arXiv preprint arXiv:1509.04232*, 2015.

[18] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Ssstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, Nov 2012.

[19] L. Wu, J. He, M. Jian, J. Zhang, and Y. Zou, "Fast cloud image segmentation with superpixel analysis based convolutional networks," *2017 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 1–5, 2017.

[20] W. Zhao, L. Jiao, W. Ma, J. Zhao, J. Zhao, H. Liu, X. Cao, and S. Yang, "Superpixel-based multiple local cnn for panchromatic and multispectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 4141–4156, July 2017.

[21] F. Zohourian, B. Antic, J. Siegemund, M. Meuter, and J. Pauli, "Superpixel-based road segmentation for real-time systems using cnn." in *VISIGRAPP (5: VISAPP)*, 2018, pp. 257–265.

[22] R. Gadde, V. Jampani, M. Kiefel, D. Kappler, and P. V. Gehler, "Superpixel convolutional networks using bilateral inceptions," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 597–613.

[23] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "Computational capabilities of graph neural networks," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 81–102, Jan 2009.

[24] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[25] A. Quek, Z. Wang, J. Zhang, and D. Feng, "Structural image classification with graph neural networks," in *2011 International Conference on Digital Image Computing: Techniques and Applications*, Dec 2011, pp. 416–421.

[26] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[27] O. H. Maghsoudi, "Superpixel based segmentation and classification of polyps in wireless capsule endoscopy," in *2017 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*. IEEE, 2017, pp. 1–4.

[28] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[29] S. Yadav, S. Patra, C. Arora, and S. Banerjee, "Deep cnn with color lines model for unmarked road segmentation," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sep. 2017, pp. 585–589.

[30] S. Gu, T. Lu, Y. Zhang, J. M. Alvarez, J. Yang, and H. Kong, "3-d lidar + monocular camera: An inverse-depth-induced fusion framework for urban road detection," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 3, pp. 351–360, Sep. 2018.