CM3: Convolutional-Max and Mathematical Morphology to Image Segmentation

Felipe Augusto Lima Reis, Raquel Almeida, Silvio Jamil F. Guimarães, Zenilton K. G. do Patrocínio Jr
Audio-Visual Information Processing Laboratory (VIPLAB)
Pontifical Catholic University of Minas Gerais (PUC Minas)
Belo Horizonte, Minas Gerais, Brazil
{falreis, raquel.almeida.685026}@sga.pucminas.br, {sjamil, zenilton}@pucminas.br

***Abstract*—**
***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Image segmentation refers to the partition of an image into a set of regions representing meaningful areas. It is considered a challenging semantic task aiming to determine and group uniform regions for analysis. reescrito Felipe - mantido mesma estrutura DOMINGUEZ According to [1], to create an adequate segmented image it is necessary that the output presents some fundamental characteristics, such as: (i) region uniformity and homogeneity in its features, aka. gray level, color, or texture; (ii) region continuity, without holes; (iii) significant difference to adjacency regions; and (iv) spacially accurationess with smoothness, without raggedness. Segmentation is an active topic of research and in a traditional approach the task is performed using hand-engineered features [2].

Insert scale-set theory reference to characterize hand-eng. Talvez retirar o paragrafo do deep para introduzir o related

Recently, deep learning architectures drastically changed the computational paradigm for visual tasks. The main advantage of deep learning algorithms is that it does not require an engineered model to operate, meaning that they are capable of learning not only the features to represent the data but also the models to describe it [3]. The success of these approaches depends on a set of standards learned by the network, in which more complex concepts are built from simpler ones. In the deep learning approach applied in images, the raw pixel on the input layer is learned as segments and parts until the composition of multiple object concepts later in the network.

Unsurprisingly, many approaches have been proposed in the recent years, as [4] [5] [6], to explore information from the outputs of different layers of a deep learning network, producing different maps. One challenge in this strategy is how to combine these maps, to produce an unique output that could represent different concepts and fits the goal of the network. In this work it is presented a simple strategy to combine maps from the layer outputs and create region proposition for the task of binary image segmentation.

The remainder of this work is organized as it follows: Section II contains the related works; Section III describes the method developed to produce segmentation; Section IV shows the dataset description, the experimental setup and results the experiments; and, finally, Section V concludes this paper.

II. Related work

Acho q a idia de multiescala no est diretamente relacionada a pegar partes da rede neural. Tanto q o RCF pega sadas laterais e ainda usa multiescala do lado de fora da rede para completar a tarefa. Se for falar de redes neurais multiescala, necessrio falar do MCG.

In the earlier years of the deep learning resurgence, an strategy in [7] (extended version in [8]), tackles the task of scene parsing—segmentation task applied for each pixel of the image, aiming to group pixels composing all the identifiable objects in the scene—using hierarchical trees and deep features alongside. Images are used as input for a convolutional network to extract deep features from multiple scales of the images, and in parallel to construct a segmentation tree, to represent in its nodes dissimilarities of neighboring pixels. The tree nodes are used to pool the correspondent deep features to be processed by a classifier. The classifier scores are used to create histogram of object classes for each node of the segmentation tree, and the final parsing proposal is built using the class entropy distribution for selecting the nodes that cover the entire image.

The proposal in [7] with an auxiliary hierarchical structure was one of the first strategies to extend the use of deep features to a complex task. It is important to bear in mind that deep learning approaches were initially described as black-box methods, meaning that not much were known about the reasoning and decisions of the created models. Much exertion have been applied to investigate the networks operation, whether by methodical experimentation [9]–[12] or visualization methods [13]–[15]. Those efforts provided more clarity of the hierarchical aspects of the deep features, which allowed researches to explore these aspects in their endeavors.

In exploring the hierarchies of deep features, three main architectures stand out in recent years, namely: (i) Holistically-nested Edge Detection (HED); (ii) Convolutional

The authors are grateful to FAPEMIG (PPM 00006-16), CNPq (Universal 421521/2016-3 and PQ 307062/2016-3), CAPES (MAXIMUM STIC-AmSUD 048/14) and PUC Minas for the financial support to this work.

Oriented Boundaries (COB); and (iii) Rich Convolutional Features (RCF). Those networks explicit explore the hierarchies by extracting side outputs of traditional convolutional networks to create boundary maps which are also learned in the network.

To the best of our knowledge, the first network exploring this strategy was HED (extended version in [4]), which applied the boundary maps for the boundary detection task, aiming to identify the limits separating uniform regions. The HED network create an side-output layer at each stage of the VGG16 network [16], in which the stages are composed by two Convolution+ReLU layers followed by a Max Pooling layer. In HED, each side-output layer is associated with a classifier in a deeply supervised scheme [17]. The layers create edge maps, which are scaled and fused at the end, to be evaluate by a cost-sensitive function to balance the bias towards not-boundary pixels. The HED network significantly improved the performance in multiple datasets. The extended version also applied the network for the segmentation task. The authors in [18] use the edge maps created by the HED network alongside with other features such as brightness, colors, gradient and variance to describe images. The goal of their proposal was to create an efficient framework to be used as real-time segmentation system, focused on a fusion strategy to update region features.

In the COB network, the authors also create edge maps from side activations, differing maily from HED by the attribution to candidate contours the orientation information and weights representing the contour strength. The contour orientations are estimated by approximation to known polygon segments and are used to create segmentations hierarchies. The segments weights are computed based on the candidate contour neighboring region to measure the confidence that the candidate is a boundary line. The weights are thresholded to determine the granularity of the segment when creating the segmentation hierarchy. The network perform well in multiple tasks such object proposal, object detection, semantic contour and segmentation.

Finally, the RCF network applied in the boundary detection task, which differ from HED by three main modifications. The first regards the input layer, in which it is used pyramids to create multiple scales of the images. The scaled images are later interpolated in the output layer, similar to [7]. The second modification regards the number of side output maps. RCF creates a side output at each Convolutial+ReLU layer of the VGG16 network, which is believed to create more detailed representations and improve the network accuracy. The last modification is in the loss function and the ground-truth of the datasets. In the ground-truth images, pixels are weighted based on a vote among multiple human-annotated values. Any pixel that not achieve a confidence vote value is disregarded by the loss function in the network. The goal is to reduce inconsistencies in the fallible human annotations and mitigate the network confusion in controversial pixels.

In [19] the authors pursued a similar direction of the afore mentioned networks. The proposal consists of joint strategies for the recognition task in large scale, specifically: NAO ENTENDI.

III. Hierarchical maps in convolutional neural networks

This work present strategies to merge hierarchical maps created from outputs of different layers of a convolutional network.verificar se eh plagio. In a convolutional network each layer is a three-dimensional array of size $h \times w \times d$, where h and w are spatial dimensions and d is the feature, channel or stride dimension. The first layer is the input image, with pixel size $h \times w$ and d color channels. Locations in higher layers correspond to the locations in the image they are path-connected to, which are called their receptive fields. Convolutional networks are built on translation invariance and their basic components (convolution, pooling, and activation functions) operate on local input regions and depend only on relative spatial coordinates.

The convolutional network model used in this work is the VGG network [16],proposed in 2014 as one of the first attempts to create deeper models for the task of object recognition. The architecture is a composition of multiple stacked convolutional layers, in which the receptive fields and stride have a fixed $3 \times 3 \times 1$ size. Following each two or three layers of convolution is placed a max-pooling layer. Also, all hidden layers are supplied with a ReLU non-linear rectification.

HED network [4] and, later, RCF [5] were based in VGG to provide edge detection. Both projects removed the final output of the network and create side outputs, that were combined in new fused output. Inspired by both works, we used changed VGG to provide image segmentation, where each pixel is binary classified into 2 categories available in the dataset.

Formally, let $S = \{(X_n, Y_n), n = 1, ..., N\}$ be the training input set for the network, in which X_n is a set of N images with three color channels and Y_n the set of N labels associated with each image with values belonging to $\{0, 1\}$. Consider also $\mathbf{W}$ the layer set of parameters in which $\mathbf{w} = \{\mathbf{w}_1, ..., \mathbf{w}_M\}$ is the associated weights for each one of the M side output maps. The objective function for training the weights for the ℓ_{side} image map could be defined as:

$$\mathcal{L}(\mathbf{W}, \mathbf{w}) = \sum_{m=1}^{M} \alpha_m \ell_{side}^{(m)}(\mathbf{W}, \mathbf{w}_m) \quad (1)$$

Inspired by HED project, we created side outputs in each VGG stage, as Figure 1. HED and RCF projects provided custom functions to balance the number of pixels of edges from the non-edges pixels. Once our problem is not as unbalanced as edge detection, we decided to use strongly known *categorical crossentropy* loss function. To merge the side outputs, we evaluated 4 different techniques:

- *Add* - Sums all predictions provided in each layer to produces only one output;
- *Average* - Makes the average value of all side outputs;
- *Maximum* - Takes the result of the most confident output;
- *Majority* - Takes the prediction values by all outputs and decided for the class that contains the more votes.

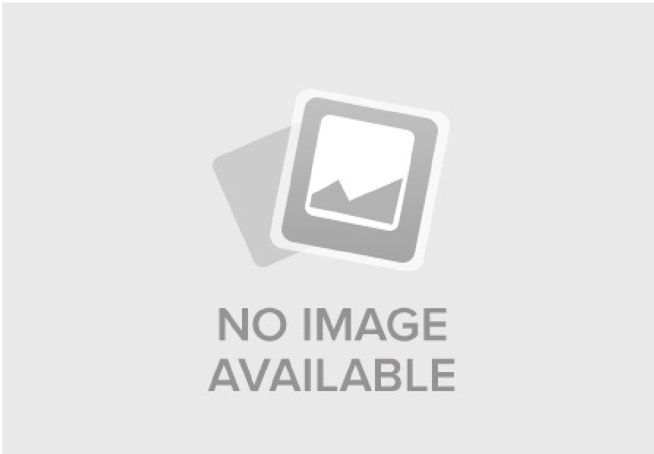

Fig. 1: Side outputs in each VGG stage

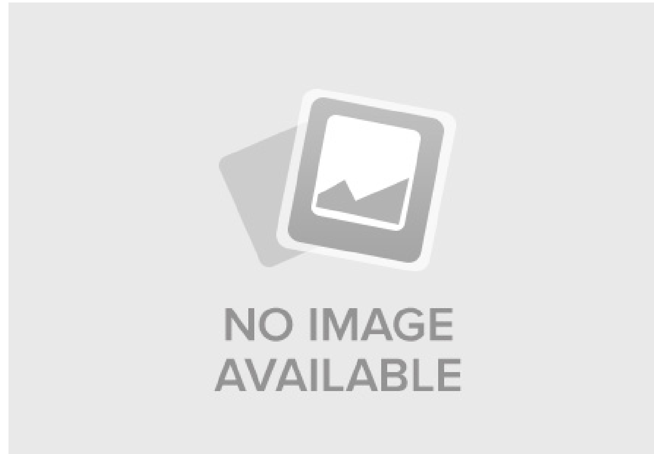

Fig. 2: Side outputs in each VGG layer

The fuction *Add* was looking for combine low and high confidence neurons into one single output. *Average* functions aims to combine low confidence neurons with high confidence ones. Also evaluates if all the network is learning the information instead of only part of it. *Max*, for other way, trust only in the most confident value, ignoring low values. This operation does not impply that all network is learning a task, but means that at least one neuron learned. Finally, the *Majority* prediction expecting a kind of consensus of the side outputs.

Since the first tests we clearly saw that *Max* operation was the far best. It was easier to train and produces better results, even without data augmentation. Other operations were difficult to train and requests a carefully set of parameters. The results often are trapped in saddle points and overfitting.

Formally max operation for side outputs can be defined as it follows. Let a set of side outputs $so = so_1, so_2, so_3, ..., so_n$. *Max* operation can be defined as the Equation 2

$$m = \max_{1 \leq j \leq n} (so_j) \tag{2}$$

Once the results were good for *Max* operation using stage outputs a simple question emerged: “What happens if we output all layers of the network and combined them?”. Then we use output layers and simple combine them with *Max* operation. A similiar approach was adopted by 5 with side outputs in all layers but combined with a convolution of 1×1 in every stage of the network. Our paper, otherwise, uses the side output without other combination in each stage. The *Max* operation is proceeded with the raw data from each layer, as Figure 2.

IV. Experiments

A. Experimental setup

Our network were build using using Keras [24] with Tensorflow [25]. We used a pre-trained VGG16 model to initialize the weights. Also, we use SGD optimization with learning rate set to 1e-4, decay of 1e-6 and momentum of 0.95. Other experiments with different values will be discussed in next sections. All training experiments were performed in GeForce GTX 1080 8GB GPU.

To increase the number of images in the training set, we made some data augmentation procedures. The techniques we used was pepper noise, horizontal flipping (mirror), changes in contrast and brightness. These procedures resulted in 1445 images, splited in 1228 samples for training and 217 validate samples (about 15%).

B. The Kitti Road/Lane dataset

KITTI Road/Lane Dataset, part of KITTI Vision Benchmarking Suite, contains images for road and lane estimation. Consists of 289 training and 290 test images . Ground truth is manually annotated for two different road types: road - road area (the composition of all lanes), and lane (the ego-lane; lane the vehicle is currently driving on) [23].

In this paper, we use only the road ground-truths and ignore lane annotations. Some images contains two different ground-truths, one for lane and other for road. Then, we prefer to use road estimation and build only one classifier. Also, it is important to say that ground truth is only available for training set. The test evaluation should be performed using KITTI Server [23].

The results in this paper were performed using KITTI Road Evaluation Benchmark, provided with KITTI Road Dataset.

C. Results

In training evaluation, it was noticed that the network was more stable, and we can set bigger learning rates parameters (1×10^3). Also, we can use different optimizers and Nesterov Optimization [22].

D. Qualitative analysis

V. Conclusion

The setup under Anaconda environment is public available online in https://github.com/falreis/segmentation-eval.

[1]Table abbreviations: *MaxF*: Maximum F1-measure, *AP*: Average precision, *PRE*: Precision, *REC*: Recall, *FPR*: False Positive Rate, *FNR*: False Negative Rate

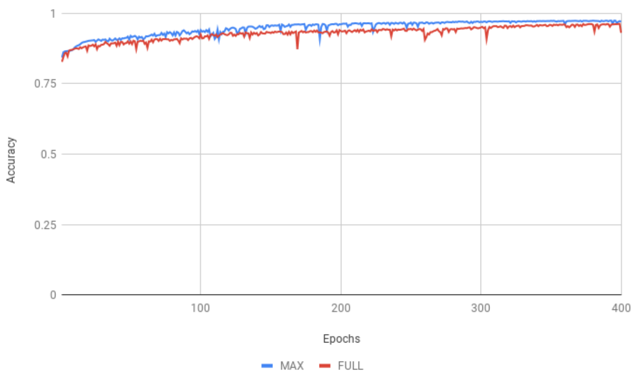

Fig. 3: Validation accuracy

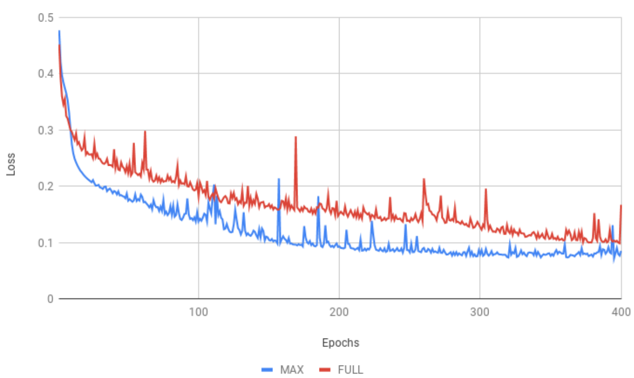

Fig. 4: Validation loss

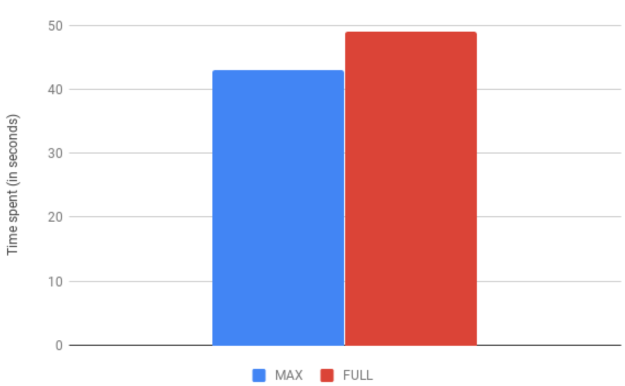

Fig. 5: Average training time

Stage Layer Outputs - Without Matematical Morfology

Category	MaxF	AP	PRE	REC	FPR	FNR
um_road	96.92%	87.36%	94.47%	99.49%	1.13%	0.51%
umm_road	97.57%	89.44%	96.05%	99.15%	1.24%	0.85%
uu_road	95.16%	85.73%	92.94%	97.49%	1.16%	2.51%

Stage Layer Outputs - With Matematical Morfology

Category	MaxF	AP	PRE	REC	FPR	FNR
um_road	97.01%	87.68%	94.83%	99.30%	1.05%	0.70%
umm_road	97.61%	89.67%	96.30%	98.97%	1.16%	1.03%
uu_road	95.42%	86.48%	93.77%	97.13%	1.01%	2.87%

All Layers Outputs - Without Matematical Morfology

Category	MaxF	AP	PRE	REC	FPR	FNR
um_road	96.39%	86.81%	93.87%	99.05%	1.25%	0.95%
umm_road	97.05%	88.83%	95.37%	98.78%	1.46%	1.22%
uu_road	94.70%	84.87%	92.00%	97.56%	1.33%	2.44%

All Layers Outputs - With Matematical Morfology

Category	MaxF	AP	PRE	REC	FPR	FNR
um_road	96.65%	87.51%	94.64%	98.74%	1.08%	1.26%
umm_road	97.21%	89.31%	95.90%	98.56%	1.29%	1.44%
uu_road	95.20%	86.15%	93.40%	97.08%	1.08%	2.92%

TABLE I: KITTI benchmark evaluation results for in each category [1]

References

[1] D. Domnguez and R. R. Morales, *Image Segmentation: Advances*, 2016, vol. 1, no. 1.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, Dec 2017. [Online]. Available: http://arxiv.org/abs/1511.00561

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, vol. 1, 2016.

[4] S. Xie and Z. Tu, "Holistically-nested edge detection," *Int. J. Comput. Vision*, vol. 125, no. 1-3, pp. 3–18, 12 2017. [Online]. Available: https://doi.org/10.1007/s11263-017-1004-z

[5] Y. Liu, M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 5872–5881.

[6] K. Maninis, J. Pont-Tuset, P. Arbelez, and L. V. Gool, "Convolutional oriented boundaries: From image segmentation to high-level tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 819–833, April 2018.

[7] C. Farabet, C. Couprie, L. Najman, and Y. Lecun, "Scene Parsing with Multiscale Feature Learning, Purity Trees, and Optimal Covers," in *29th International Conference on Machine Learning (ICML 2012)*, A. McCallum, Ed., Edinburgh, United Kingdom, Jun 2012, pp. 1–8.

[8] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning Hierarchical Features for Scene Labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915–1929, Aug 2013.

[9] R. Ilin, T. Watson, and R. Kozma, "Abstraction hierarchy in deep learning neural networks," in *International Joint Conference on Neural Networks, 30.* Anchorage, Alaska: IEEE Computer Society, 2017, pp. 768–774.

[10] C.-C. J. Kuo, "Understanding convolutional neural networks with a mathematical model," *Journal of Visual Communication and Image Representation*, vol. 41, pp. 406–413, 2016.

[11] D. Eigen, J. Rolfe, R. Fergus, and Y. Lecun, "Understanding deep architectures using a recursive convolutional network," in *International Conference on Learning Representations*. Banff, Canada: Computational and Biological Learning Society, 2014.

[12] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *International Conference on Learning Representations*. Toulon, France: Computational and Biological Learning Society, 2017.

[13] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *CoRR*, 2013.

[14] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision*, vol. 8689. ZURICH, Switzerland: Springer International Publishing, 2014, pp. 818–833.

[15] B. Alsallakh, A. Jourabloo, M. Ye, X. Liu, and L. Ren, "Do convolutional neural networks learn class hierarchy?" *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 152–165, 2018.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[17] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-Supervised Nets," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Lebanon and S. V. N. Vishwanathan, Eds., vol. 38. San Diego, California, USA: PMLR, 09–12 May 2015, pp. 562–570.

[18] M.-M. Cheng, Y. Liu, Q. Hou, J. Bian, P. Torr, S.-M. Hu, and Z. Tu, "HFS: Hierarchical Feature Selection forEfficient Image Segmentation," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 867–882.

[19] J. Fan, T. Zhao, Z. Kuang, Y. Zheng, J. Zhang, J. Yu, and J. Peng, "HD-MTL: Hierarchical Deep Multi-Task Learning for Large-Scale Visual Recognition," *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 1923–1938, Apr 2017.

[20] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012. [Online]. Available: http://arxiv.org/abs/1212.5701

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[22] Y. Nesterov, "A method of solving a convex programming problem with convergence rate O(1/sqr(k))," *Soviet Mathematics Doklady*, vol. 27, pp. 372–376, 1983. [Online]. Available: http://www.core.ucl.ac.be/$\sim\{\}$nesterov/Research/Papers/DAN83.pdf

[23] J. Fritsch, T. Kuehnl, and A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms," in *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[24] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[25] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/