To be determined *- Evaluation of Side Outputs for Convolutional Neural Networks*

Felipe A. L. Reis[†], Raquel Almeida[†], Silvio Jamil F. Guimarães[†], Zenilton K. G. do Patrocínio Jr.[†],
Simon Malinowski[*] and Ewa Kijak[*]

[†]*Audio-Visual Information Processing Laboratory - Pontifical Catholic University of Minas Gerais*
Belo Horizonte, Minas Gerais, Brazil
{falreis, raquel.almeida.685026}@sga.pucminas.br, {sjamil, zenilton}@pucminas.br

[*]*Linkmedia, IRISA - Université de Rennes 1*
Rennes, France
{simon.malinowski, ewa.kijak}@irisa.fr

***Abstract*—**

***Index Terms*—convolutional neural network, image segmentation, mathematical morphology, CNN, region detection**

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. 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, such as gray level, color or texture; (ii) region continuity, without holes; (iii) significant difference to adjacency regions; and (iv) spacial accuracy with smooth boundaries and without raggedness.

Image segmentation is an active topic of research and in a typical approach the procedure could be divided in two stages [2]: (i) low-level analysis, which evaluate the pixel characteristics, neighboring relation and it is ideally uncommitted in terms of position, orientation, size and contrast; and (ii) high-level analysis, which maps the low-level characteristics to fulfill the task.

Recently, the deep learning approach 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]. Facing this new paradigm, researches initially replaced hand-engineered features in the low-level analysis by the features learned in deep models [4]–[6], which mostly achieve the desirable characteristics. More recently, there is many proposals exploring the learned model for the high-level analysis, creating maps from the outputs of different layers in a deep learning network [7]–[10].

One challenge on the latter strategy is how to combine the output from distinct layers, considering that they are presented with different sizes and could represent different concepts. In this work, it is presented strategies to combine the outputs from different layers, by using simple merging functions that explore useful behavior in the learning process. It is also studied the amount of combined outputs necessary to create a viable region proposition for the task of image segmentation. In addition, it is also presented a post-processing filtering step using mathematical morphology idempotent functions [11] to better cope with the fundamental characteristics of an ideal segmented image.

The remainder of this work is organized as it follows: In Section II it is characterized the hierarchy of concepts in deep models related work exploring theses models for high-level tasks. In Section III it is presented the reasoning and strategies proposed in this work. In Section VI it is presented the dataset description, the experimental setup and results for the experiments. And, finally, in Section VII the conclusions are laid out.

II. Related work

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 [12]–[15] or visualization methods [16], [17]. Those efforts provided more clarity of the deep models and characterized the learned features as complex concepts build from simpler ones. Also it demonstrate the learning progression from detailed to coarser

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.

representations as the scale and resolutions reduce through the network. When applied for object recognition task for instance, the raw pixel on the input layer is learned as segments and parts until the composition of an object concept on posterior layers, while the input scale reduces to a single feature vector on the output.

The knowledge of the concept abstraction and learning progression allowed new research endeavors to explore them in high-level tasks. For instance, three main architectures standout in recent years, namely: (i) Holistically-nested Edge Detection (HED) [18]; (ii) Convolutional Oriented Boundaries (COB) [9]; and (iii) Rich Convolutional Features (RCF) [10]. These architectures explicit explore the models of a traditional deep network to perform a certain high-level task, in which all extract side-outputs of the network and each present a different strategy to combine them .

The HED network create a side-output layer at each stage of the VGG16 network [5] as boundary maps. In HED, each side-output is associated with a classifier in a deeply supervised scheme [6] for the task of edge detection. This association insert the fusion process in the network, attributing weights for each side-output that will be learned individually and determine its contribution on the final evaluation. The evaluation is performed by a cost-sensitive function to balance the bias towards non-edge pixels. The HED network significantly improved the performance in multiple datasets and the extended version [7] also applied the network for the segmentation task.

The authors in [8] 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 mainly from HED by the attribution to candidate contours orientation information and weights representing the contour strength. The contour orientations are estimated by approximation to known polygon segments and the segments weights are computed based on the candidate contour neighboring region used as a confidence measure. To combine the side outputs it is used a non-linear function to regress both the segment weights and the orientation maps, creating region hierarchical trees by thresholding the contour strength. The network perform well in multiple tasks such object proposal, object detection, semantic contour and segmentation.

Finally, the RCF network, that not only create multiple side-outputs, but also uses multiple scales of the images in the input layer. Differently from the HED network, RCF extract one side-output at each convolutional layer of VGG, arguing that this could create more detailed representations and improve the network accuracy. The merging process is performed by a series of operations, comprising grouping by convolutions, element-wise sums, up-samplings, local loss functions and concatenation.

III. Convolutional side-outputs for image segmentation

Hierarchies are long associated with the image segmentation task [19]–[23], to a degree that it improves a coherent organization of nested regions. The main motivation for using well-defined hierarchies is that different hierarchical level contains different detail level. In this work, instead of using a well-defined hand-engineered hierarchical structure, it is proposed to explore the concept abstraction resultant of the deep network dynamics, extracting side-outputs at different layers that ideally would contain different level of details.

The idea is to combine the side-output maps into a single proposition to be evaluated in the image segmentation task, driving the learning flow towards creating adequate regions for the task. In an optimal scenario, the side-outputs would contain enough details to cope with the task, whilst creating coherent region proposals.

Amongst the many strategies for deep models, convolutional networks are well-know by the concept abstraction due the multiple stages of convolution and have been successfully used for the object recognition task. They are usually characterized by three nested functions in multiple layers, namely: (i) convolution; (ii) spatial pooling; and (iii) non-linear activation. Formally, let a convolutional network f composed by L layers be defined as:

$$f(\mathbf{X}) = \mathbf{W}_L \mathbf{H}_{L-1} \tag{1}$$

in which:

- $\mathbf{W}_l$ is the associated weights for the layer l;
- $\mathbf{H}_l$ is the output of the hidden layer l, defined as Caso f (2) nao seja uma camada de saida da rede, ela eh chamada de camada intermediaria ou escondida (hidden layer) (BENGIO-GOODFELLOW). No caso, a formula deveria ser L-2.:

$$\mathbf{H}_l = pooling(activation(\mathbf{W}_l \mathbf{H}_{l-1})) \ \forall l \in \{1, ..., L-1\} \tag{2}$$

For consistency, consider $\mathbf{H}_0 = \mathbf{X} = \{X_1, X_2, ..., X_n\}$ the set of N input images I.

The VGG network [5] is one of the first attempts to create deeper models following the convolutional scheme. The core of the layers in VGG is defined by a convolution immediately followed by a rectified linear unit, as follows:

$$C_l = ReLU(\mathbf{W}_l \mathbf{H}_{l-1}) \ \forall l \in \{1, .., L-1\} \tag{3}$$

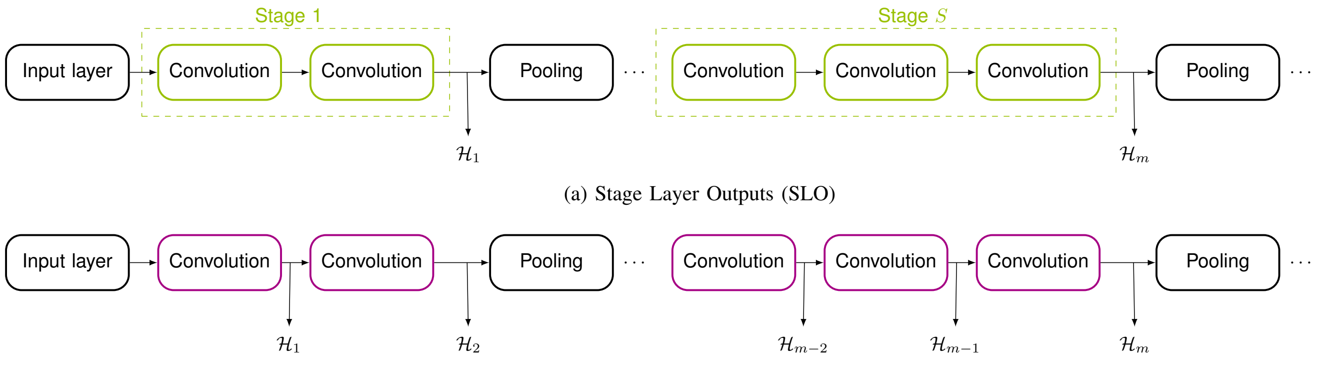
in which $ReLU(\cdot) = max(0, \cdot)$. There is also two types of stages, $S^{(1)}$ and $S^{(2)}$, that could formally defined as:

$$S^{(1)} = ReLU(\mathbf{W}_l(ReLU(\mathbf{W}_{l-1}\mathbf{H}_{l-2}))) \tag{4}$$

$$S^{(2)} = ReLU(\mathbf{W}_l(ReLU(\mathbf{W}_{l-1}(ReLU(\mathbf{W}_{l-2}\mathbf{H}_{l-3})))) \tag{5}$$

The output of a hidden layer is computed as $maxpool(S^{(1)})$ or $maxpool(S^{(2)})$ for all S stages in the network.

Questions on which and how many side-outputs would be adequate for the image segmentation task, are assessed using two different extraction strategies, both applied in the VGG



(a) Stage Layer Outputs (SLO)

(b) All Layers Outputs (ALO)

Fig. 1. Illustration for the two side-outputs extraction strategies: (a) side-outputs extracted at each stage of the network and (b) side-outputs extracted at each convolutional layer

network. Namely: (i) Stage Layer Outputs (**SLO**), inspired by the HED model, creating one side-output for each VGG stage; and (ii) All Layers Outputs (**ALO**), inspired by the RCF model, creating one side-output for each convolutional layer.

Formally, the set $\mathcal{H}$ of M side outputs maps in each strategy is defined as:

$$\mathcal{H}_{SLO} = \{\mathcal{H}_1, ..., \mathcal{H}_m | m \in [1, S] \text{ and } \mathcal{H}_m \in \{S^{(1)}, S^{(2)}\}\} \quad (6)$$

$$\mathcal{H}_{ALO} = \{\mathcal{H}_1, ..., \mathcal{H}_m | m \in [1, L-1] \text{ and } H_m = C_l \; \forall l \in \{1, .., L-1\}\} \quad (7)$$

In the case of **SLO**, the number of side-outputs amounts to the number of pooling layers in the network and for **ALO**, it is equal to the number of convolutional layers. An illustration for both strategies is presented in Figure 1.

IV. Merging strategies

The main question in dealing with side-outputs in convolutional networks is how to combine them, considering that they are presented in different scales and could represent different concepts. The goal is to produce a single proposition to be evaluated in the task, but retain the useful information presented at different layers.

In this work, the strategy to overcome those challenges is to combine the side-outputs by exploring the knowledge of the learning process. To achieve that, it is proposed to apply simple merging functions that would enhance different desirable behavior, as described in the following:

- *ADD*: Aims to balance negative and positive weights;
- *AVG*: Aims to create a proposition representing the whole network learning;
- *MAX*: Aims to represent confident values.

Formally, the single proposition Z to be evaluated in the task, under each strategy could be defined as:

$$Z_{ADD} = \sum_{i=1}^{M} (\mathcal{H}_i) \quad (8)$$

$$Z_{AVG} = \frac{\sum_{i=1}^{M} (\mathcal{H}_i)}{M} \quad (9)$$

$$Z_{MAX} = \max_{1 \leq i \leq M} (\mathcal{H}_i) \quad (10)$$

The operations are performed element-wise on each side-output after they are re-scaled for the input size, while maintaining the connectivity pattern.

Once the combined map is created, it is evaluated on the segmentation task which aims to provide partition of an image into a set of regions representing meaningful areas. This could be reduced to a binary problem aiming to distinguish each pixel of the image as belonging to a region of interest or the background. If confronted with multiple regions of interest this minimal formulation could be executed individually and paired later.

Formally, consider once again the set of N training images $\mathbf{X}$ and alike $\mathbf{Y} = \{Y_1, Y_2, ..., Y_n\}$ the set of ground-truth images in which each pixel is labeled. The ground-truth images are used to calculate the pixel accuracy measuring the rate that a pixel is correctly predicted to belong to the region of interest or the background.

A. Loss evaluation

For loss evaluation, we used two different functions: the well known categorical cross-entropy and fuse-pixel-error, proposed by [18].

Formally, categorical cross-entropy for classes p and q and a given set, is defined as:

$$H(p, q) = E_p[-\log q] \quad (11)$$

where $H(p)$ is the entropy of p. Once images contains a set of pixels, and each one is defined by Equation 11, the value all pixels set for a single image is defined as:

$$H(p, q) = -\sum E_p[\log q] \quad (12)$$

Formally, fuse-pixel-error, as defined in [18] consists in the distance $Dist(\cdot, \cdot)$ between the ground-truth label map and the neural net predictions. The loss function ζ_{fuse} is defined as:

Nao encontrei o simbolo adequado para formula - utilizei a letra zeta ζ por enquanto

$$\zeta_{fuse}(W, w, h) = Dist(Y, \hat{Y}_{fuse}) \tag{13}$$

, where $\hat{Y}_{fuse} \equiv \sigma(\sum_{m=1}^{M} h_m \hat{A}_{side}^{(m)})$ and $\mathbf{h} = (h_1, ..., h_M)$.

V. Post-processing

Mathematical morphology is consistent with the non-linear image analysis, presenting solid theoretical foundation and idempotent functions. The formulations are presented in the complete lattice geometric space, in which the functions are performed considering whole sets operating over another whole set. In mathematical morphology, the operators are known a priori and defined using the sets of structuring elements.

In this work, it is proposed to use mathematical morphology as post-processing step, meaning that this step is not inserted in the learning stage. The main goal is to better cope with the fundamental properties of a well-segmented image, particularly, region uniformity and continuity. To achieve that, it is proposed to use a function filter, called area opening, which tend to destroy the small, thin and conspicuous areas.

Formally, let $\hat{Y} \in \mathbb{R}^2$ be the output of a testing image consistent with the representation created by the parameters learned in the network. Consider B a structuring element and γ_B the morphological opening produced by it. Consider also λ the threshold parameter which will determine how small a certain area must be to be purged. In this case, $\gamma_B \subseteq \gamma_\lambda$ if and only if B is a finite union of connected components of area greater or equal to λ. It is expected that this simple additional step could reduce possible noises on the final result, and in this way improve the results.

VI. Experiments

Experiments were conducted in the KITTI Road/Lane Dataset, part of KITTI Vision Benchmarking Suite [24]. The dataset contains images for road and lane estimation for the task of image segmentation. It is consisted of 289 training and 290 test images. The ground-truth is manually annotated for two different road types: (i) road, road area composing all lanes; and (ii) lane, lane the vehicle is currently driving on. It is important to notice that the ground-truth is only available for training set and the test evaluation should be performed using KITTI Server.

In this work, it is used only the road ground-truths and the lane annotations were ignored. This dataset contains the same image with different ground-truths for lane and road estimation. Then, we prefer to use the road estimation and build the classifier on a binary problem (road and background). The road type is divided in three different categories of road scenes, namely: (i) uu_road, urban unmarked; (ii) um_road, urban marked; and (ii) umm_road, urban multiple marked lanes.

Rever lista de procedimentos de data augmentation

To increase the number of images in the training set, it is performed some data augmentation procedures. It was added images with pepper/salt noise, horizontal flipping (mirror) and changes in contrast and brightness. Procedures that would create undesired behavior, such as the road in the sky and distortions that would change the nature of the objects in the scene, such as cars and pedestrians were avoided. Augmentation procedures resulted in 2601 images, divided in 2080 samples for training and 521 samples for validation (about 20%).

A. Experimental setup

Our networks were build using using Keras [25] with Tensorflow [26]. We used a pre-trained VGG16 model to initialize the weights. Also, we use SGD optimization with learning rate set to 1e-3, decay of 5e-6 and momentum of 0.95. The default batch size contains 16 images. All training experiments were performed in GeForce GTX 1080 8GB GPU.

For simplicity, in the remaining of this work, the network using the side outputs extracted at each stage of the VGG will be called Stage Layer Outputs (**SLO**) and it is composed by $n = 5$ side outputs. Similarly, for the side outputs extracted at each convolutional layer, it will be called All Layers Outputs (**ALO**) and it is composed by $n = 13$ side outputs. For comparison, it is also defined a network similar to VGG, with only the final output, without any side outputs, called No Side Outputs (**NSO**).

The operations to combine side outputs are presented in the name of the methods. The merging operations **ADD**, **AVG** and **MAX** are available for both ALO and SLO methods. Once NSO does not contains side outputs, it does not contains merging strategies.

B. Training results - Methods Comparison

The first test set was design to identify the best neural network and its best merging methods. We train all nets with all merging methods for 100 epochs to determine which one learns faster and achieve the best result. This conduct lead us to understand how layers can be easily combined to produce outputs with good precision.

Figures 2, 3 and 4 presents the relevant curves obtained during the training phase for the proposed approaches. Figure 2, presents categorical cross-entropy loss curve for tested approaches. ALO networks appears to be more stable with a faster decay than NSO and all SLO approaches. Also, it is important to notice that NSO and SLO-MAX produces high instability in the learning course. In other hand, ALO-AVG presents the best result for the test, followed by ALO-MAX and ALO-ADD merging strategies.

Figures 3 and 4 presents metrics to identify the best values. Figure 3 contains validation accuracy and is related to categorical cross entropy method while Figure 4 shows Fuse Pixel Error method. While Figure 3 shows an expected curve, after loss evaluation in the previous image, Figure 4 indicates a new

Fig. 2. Categorical Cross-Entropy Validation Loss

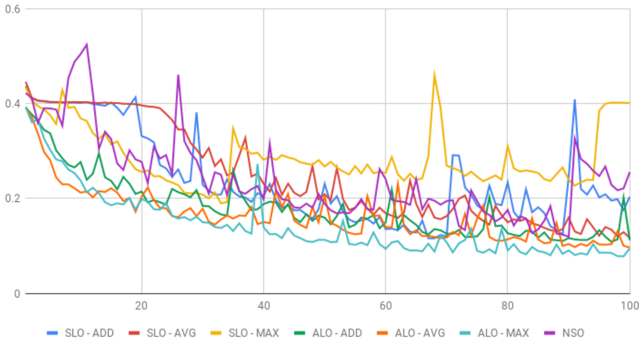
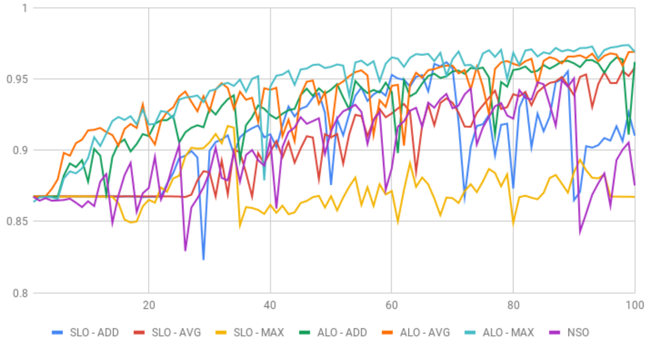


Fig. 3. Categorical Cross-Entropy Validation Accuracy



information: SLO-MAX neural network seens to overfitting around 40 epochs. After some steps with unstable pixel error values, the values persists almost unchange for the rest of training epochs.

From previous graphs, in short we have ALO networks with superior and more desirable behavior than the SLO and NSO models. It is believed that these results are consequence of the considerably larger amount of side outputs, which create more possibilities of interchangeability between confidant values.

Fig. 4. Pixel Error Validation Loss

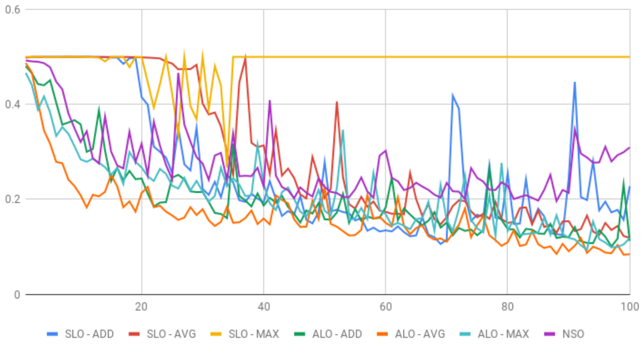
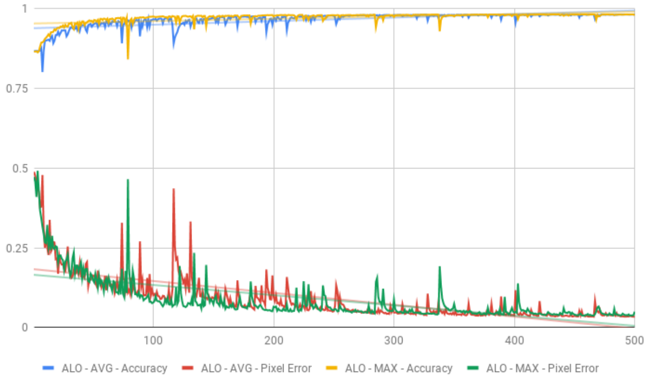


Fig. 5. Accuracy and Pixel Error for 500 epochs test set



C. *Best results*

In order to improve the results, a new set of tests were performed using 500 training epochs. As some networks had a poor performance, we evaluated only the two-best nets in the previous test. The categorical cross entropy validation accuracy and pixel error validation loss, for both ALO-AVG and ALO-MAX nets are available in Figure 5.

The best accuracy value for ALO-AVG model was **0.9821** and the best fuse pixel error was **0.0332**. For ALO-MAX model, the best accuracy and fuse pixel error were **0.9823** and **0.0372**.

Once each merging strategy contains a different behavior for side outputs, its interesting to evaluate how each side output contributes with the final output. For this, Figure 6 was created. Once **ALO** network contains 13 side outputs, we decided only to visualize outputs from each stage. In the image provide, white pixels are classified as road, while black pixels are classified as background.

Figure 6 indicates that the first two side outputs does not produce significant information. Images are almost white, indicating that all pixels were classified as road. ALO-AVG's third layer contains a clear separation between road pixel than non-road pixels. ALO-MAX's third layer, by other hand, does not clearly separates road from non-road pixels. It is possible to see get information but as not clear as ALO-AVG's.

Figure 6 also indicates that fourty layer clearly contains the best side output for both networks. A road mark is clearly visible, but with some noise. ALO-MAX contains a lot of noise, while ALO-AVG contains a few ones. The final side output contains a lot of noise, with results far away worse than the previous one. This possible indicates that these layers does not learn correctly from the previous one.

Talvez adicionar artigo do Bengio, indicando que difcil treinar as ltimas camadas - tem nos slides do Zenilton de Deep Learning

D. *Post-processing using mathematical morphology*

After the training procedure, we create a post processing step to reduce possible noises in results proposition. For this,

Fig. 6. Side outputs results from ALO-AVG and ALO-MAX networks. Acho q devo treinar a rede ALO-ADD para exibir a sada lateral, melhorando a discusso

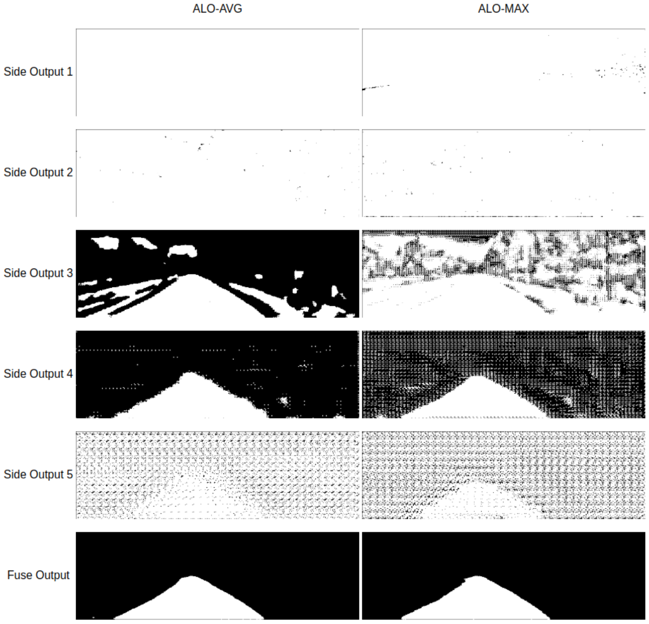
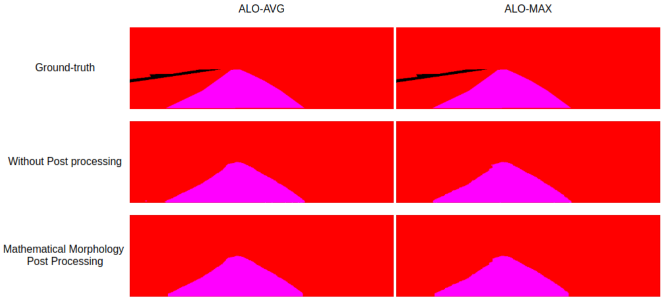


Fig. 7. Comparison between ALO-AVG without post processing and ALO-AVG with post-processing with mathematical morphology. Substituir imagem por uma com evidencia mais clara.



we used the mathematical morphology operation of Opening [11]. This procedure removes small noises created by the foreground (the road) in the background. We defined a set of kernels with the sizes of 5×5, 7×7, 9×9, 11×11 and 13×13 and applied in the images to reduce different sizes of noises.

For comparison, we developed Figure 7. In this image, we selected an output result that clearly shows the benefits of mathematical morphology post processing. It is possible to see the removal of part of the noise in the right of the image. This procedure increase the confidence, once small variations in the results can lead a potential problem, if used in a self-driving vehicle.

TABLE I
KITTI BENCHMARK EVALUATION RESULTS FOR EACH CATEGORY

um_road

Method	MaxF	AP	PRE	REC	FPR	FNR
SLO	96.92%	87.36%	94.47%	**99.49%**	1.13%	**0.51%**
SLO-mm	**97.01%**	**87.68%**	**94.83%**	99.30%	**1.05%**	0.70%
ALO	96.39%	86.81%	93.87%	99.05%	1.25%	0.95%
ALO-mm	96.65%	87.51%	94.64%	98.74%	1.08%	1.26%
State-of-the-art	**97.05%**	**93.53%**	**97.18%**	**96.92%**	**1.28%**	**3.08%**

umm_road

Method	MaxF	AP	PRE	REC	FPR	FNR
SLO	97.57%	89.44%	96.05%	**99.15%**	1.24%	**0.85%**
SLO-mm	**97.61%**	**89.67%**	**96.30%**	98.97%	**1.16%**	1.03%
ALO	97.05%	88.83%	95.37%	98.78%	1.46%	1.22%
ALO-mm	97.21%	89.31%	95.90%	98.56%	1.29%	1.44%
State-of-the-art	97.77%	**95.64%**	**97.75%**	**97.79%**	**2.48%**	**2.21%**

uu_road

Method	MaxF	AP	PRE	REC	FPR	FNR
SLO	95.16%	**85.73%**	92.94%	97.49%	1.16%	2.51%
SLO-mm	**95.42%**	86.48%	**93.77%**	97.13%	**1.01%**	2.87%
ALO	94.70%	84.87%	92.00%	**97.56%**	1.33%	**2.44%**
ALO-mm	95.20%	86.15%	93.40%	97.08%	1.08%	2.92%
State-of-the-art	**95.95%**	**95.25%**	**95.25%**	**95.65%**	**1.21%**	**4.35%**

E. Evaluation results and comparison with the state-of-the-art

Reminding that the test evaluation could only be performed using KITTI Server, the metrics provided are maximum F1-measure (MaxF), average precision (AP), precision (PRE), recall (REC), false positive rate (FPR) and false negative rate (FNR).

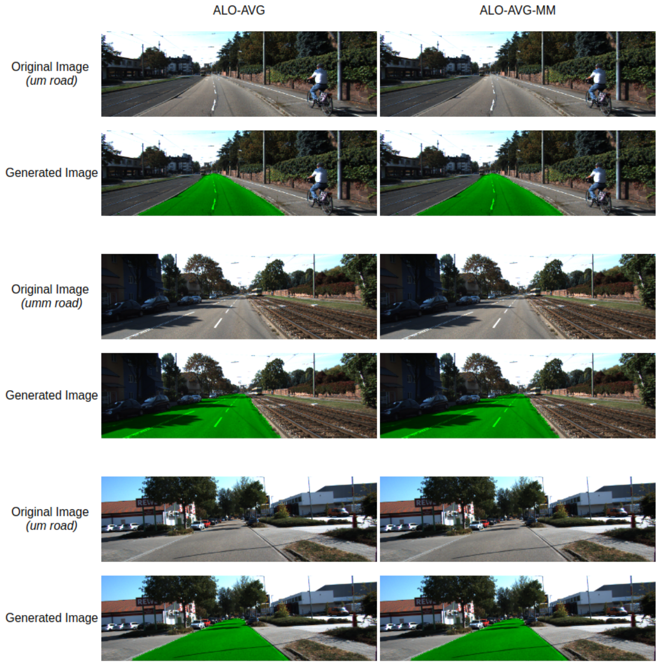
The server tests were performed using ALO-AVG method, the best one in the training process. To provide succint labels, we will use the name **ALO-AVG** for the regular approach and **ALO-AVG-MM** for the version with mathematical morphology post-processing.

The results achieved on the test set according to each category in the road scenes are presented in Table I. As expected, the ALO-AVG-MM model performed better then the ALO-AVG in all almost all of the cases. It is also possible to notice that although the post-processing slightly improved the overall performance, it also increased the number of false negatives. This could be an indications that perhaps the applied kernel sizes are not adequate and are removing more of the foreground than the desired.

If compared with the state-of-the-art (called *PLARD*, an anonymous submission on the KITTI Server platform), the proposed method is comparable and sometimes superior, regarding the maximum F1-measure and the recall metrics. This is due the fact that although the reported state-of-the-art on the dataset presents a superior average precision, it also almost always presents a higher rate of false positives an negatives. This indicates that the proposed methods are more precise in delineating the regions to be segmented.

As a final and visual representation of the results, we presents in Figure 8. This image shows the results marked (in green) over the road, to shows the performance of our model.

Fig. 8. Visual representation of the results



VII. Conclusion

This work addressed the problem of merging side outputs extracted from the convolutional layer model VGG to create region propositions for the task of image segmentation. It was evaluated the use of $add()$, $avg()$ and $max()$ merging strategies to combine the side results. The functions to enhance were evaluated using a cross-entropy and pixel-error loss functions. It was also studied the impact that the number of side outputs and a version without any side outputs in a similiar network. At last, a simple mathematical morphology operation was proposed to enhance the performance on the task and remove some noises.

Experiments demonstrated that the $avg()$ function is viable for merging maps with different sizes and connotations, and could place the proposed strategy among the state-of-the-art approaches for the task on the Kitti dataset. The use of $avg()$ merging strategy was adopted before in [10], but no explanation was given to the use of this method over other possible merging strategies. This paper helps to explain the good results achieved by those researchers.

It was also demonstrated that a large amount of side outputs increases the network capabilities during the training step and could also create jumps that could lead to better performance, in terms of accuracy. The training graphs also show that number of side outputs contributes for a faster decay in loss function and stable results. The post-processing strategy slightly improved the performance, but requires further studies.

This research opens novel opportunities for study such as: (i) exploring different merging functions, less susceptible a values fluctuations; (ii) explore regularization techniques to sustain larger amounts of side outputs consistent; and (iii) insert the mathematical morphology kernels on the learning process to search for the best kernel size.

The code and a file containing all dependencies to reproduce the experiments is public available online in https://github.com/falreis/segmentation-eval.

References

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

[2] L. Guigues, J. P. Cocquerez, and H. Le Men, “Scale-sets image analysis,” *International Journal of Computer Vision*, vol. 68, no. 3, pp. 289–317, 2006.

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

[4] 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.

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

[6] 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.

[7] S. Xie and Z. Tu, “Holistically-nested edge detection,” *International Journal of Computer Vision*, vol. 125, no. 1, pp. 3–18, Dec 2017.

[8] 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.

[9] K.-K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool, “Convolutional oriented boundaries,” in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 580–596.

[10] Y. Liu, M.-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)*, vol. 00, Jul 2017, pp. 5872–5881.

[11] L. Najman and H. Talbot, *Mathematical morphology: from theory to applications*, 2013.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

[18] S. Xie and Z. Tu, “Holistically-nested edge detection,” in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[19] R. Jones, “Component trees for image filtering and segmentation,” in *IEEE Workshop on Nonlinear Signal and Image Processing, E. Coyle, Ed., Mackinac Island*, vol. 9, 1997.

[20] J. Cardelino, G. Randall, M. Bertalmio, and V. Caselles, “Region based segmentation using the tree of shapes,” in *2006 International Conference on Image Processing*, Oct 2006, pp. 2421–2424.

[21] P. Soille and L. Najman, "On morphological hierarchical representations for image processing and spatial data clustering," in *Applications of Discrete Geometry and Mathematical Morphology*, U. Köthe, A. Montanvert, and P. Soille, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 43–67.

[22] Y. Xu, T. Graud, and L. Najman, "Hierarchical image simplification and segmentation based on mumfordshah-salient level line selection," *Pattern Recognition Letters*, vol. 83, pp. 278 – 286, 2016.

[23] J. Cousty, L. Najman, Y. Kenmochi, and S. Guimarães, "Hierarchical segmentations with graphs: Quasi-flat zones, minimum spanning trees, and saliency maps," *Journal of Mathematical Imaging and Vision*, vol. 60, no. 4, pp. 479–502, May 2018.

[24] 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.

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

[26] 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/