# Combining convolutional side-outputs for road image segmentation

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

[†]*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* – Univ Rennes, Inria, CNRS, IRISA
Rennes, France
{ewa.kijak, simon.malinowski}@irisa.fr

***Abstract*—Image segmentation consists in creating partitions within an image into meaningful areas and objects. It can be used in scene understanding and recognition, in fields like biology, medicine, robotics, satellite imaging, amongst others. In this work we take advantage of the learned model in a deep architecture, by extracting side-outputs at different layers of the network for the task of image segmentation. We study the impact of the amount of side-outputs and evaluate strategies to combine them. A post-processing filtering based on mathematical morphology idempotent functions is also used in order to remove some undesirable noises. Experiments were performed on the publicly available KITTI Road Dataset for image segmentation. Our comparison shows that the use of multiples side outputs can increase the overall performance of the network, making it easier to train and more stable when compared with a single output in the end of the network. Also, for a small number of training epochs (500), we achieved a competitive performance when compared to the best algorithm in Kitti Evaluation Server.**

***Index Terms*—convolutional neural network, CNN, image segmentation, mathematical morphology, region detection, side-outputs, merging strategies**

## I. Introduction

Image segmentation refers to the partition of an image into a set of regions representing meaningful areas, and it is considered as a challenging semantic task, aiming to determine and group uniform regions for analysis. According to [1], an adequate segmented image should present 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 between adjacent regions; and (iv) spatial accuracy with smooth boundaries and without raggedness. Image segmentation is still an active topic of research and, usually, it could be divided in two stages [2]: (i) low-level analysis, which evaluates 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, deep learning approaches have drastically changed the computational paradigm for visual tasks. The main advantage of deep learning algorithms is that they do not require an engineered model to operate, meaning that they can not only learn the features to represent the data, but also the models to describe them [3]. Facing this new paradigm, hand-crafted features used in the low-level analysis were first replaced by the features learned in deep models [4]–[6], which mostly achieved the desirable results. More recently, many proposals explore the learned model for the high-level analysis, in order to create segmentation maps from the outputs of different layers of a deep network [7]–[10]. These outputs are network samples, which do not influence the architecture and are therefore often called *side-outputs*. One challenge on the latter strategy is the combination of the side-outputs from distinct layers, considering that they have different sizes and could represent different aspects of the input. In this work, we propose some strategies to combine the side-outputs from different layers by using simple merging functions in order to explore useful behavior in the learning process. We also study the amount of combined side-outputs that is needed to create a viable region proposition. Moreover, we propose the use of a post-processing filtering based on mathematical morphology idempotent functions [11] in order to remove some undesirable small segments.

The networks are trained for a road image segmentation task. The goal in this application is to improve the performance of self-driving cars, aiming to distinguish the road from different objects such pedestrians, sidewalks and other vehicles. This road identification can also can be used in road monitoring and traffic management. In this context, providing a robust and reliable segmentation is essential.

The remainder of this work is organized as follows. In Section II, related works that characterize the hierarchy of concepts in deep models are described. In Section III, the

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.

proposed method and the merging strategies are presented. In Section IV, a quantitative and qualitative assessment are done. And, finally, in Section V, some conclusions are drawn.

## 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 effort has been made to investigate the networks operation, whether through methodical experimentation [12]–[15] or visualization methods [16], [17]. Those efforts provided more clarity on the deep models and characterized the learned features as complex concepts build from simpler ones. Also they demonstrate the learning progression from detailed to coarser representations as the scale and resolution reduce through the convolutional neural network (CNN). When applied to 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 scale reduces to a single feature vector on the output.

Recent works aim at taking advantage of the multi-scale representation naturally embedded in deep convolutional network, through the increasing of receptive fields as sub-sampling is applied, in high-level tasks. For the edge detection task in particular, three main architectures standout in recent years, namely: (i) Holistically-nested Edge Detection (HED) [7], [18]; (ii) Convolutional Oriented Boundaries (COB) [9]; and (iii) Rich Convolutional Features (RCF) [10]. These architectures all extract side-outputs from a traditional CNN model, but each present a different strategy to combine them to achieve the edge-detection task.

The HED network creates a side-output layer generating boundary maps at each last convolutional layer of each stage of the VGG16 network [5]. In HED, each side-output is associated with a classifier in a deeply supervised scheme [6] for the task of edge detection. The combination of the side-ouput predictions is achieved by a fusion layer added to the network. This fusion layer learns the weights for each side-output that 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 for multiple datasets.

The authors in [8] use the edge maps created by the HED network alongside with other features such as brightness, color, gradient and variance to describe images. The goal is to create an efficient framework for real-time segmentation, focused on a fusion strategy to update region features.

In the COB network, the authors also create edge maps from the side activations, but the method mainly differs from HED in that the candidate contours are assigned an orientation information. It also generates region hierarchies by efficiently combining multiscale oriented contour detections. The network performs well in multiple tasks such as object proposal, object detection, semantic contour and segmentation.

Finally, in contrast to the HED network, RCF uses the CNN features of all the convolutional layers, arguing that this could create more detailed representations and improve the network accuracy. The side outputs from all convolutional layers of a same stage are combined using a series of operations such as $1 \times 1$ convolutions, element-wise sums, and up-sampling. The up-sampled feature maps of each stage are then concatenated and fused by a $1 \times 1$ convolution to produce the final prediction. The RCF network, not only creates multiple side-outputs, but also uses image pyramids during testing, presenting multiple scales of an image to the detector and averaging all the resulting edge probability maps to get a final prediction map.

Motivated by the performance of these approaches, we want to understand in this work the influence of side-output for a segmentation task, that is where they should be extracted them and how to combine them to provide the more relevant information regarding the task.

## III. Side-outputs merging strategies and mathematical morphology post-processing

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 levels contain different detail levels. In this work, instead of using a well-defined hand-engineered hierarchical structure, we propose to explore the concept abstraction resulting from the deep network dynamics, by 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 an 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 and create coherent region proposals. Amongst the many strategies for deep models, convolutional networks are well-known for the concept abstraction resulting from the multiple stages of convolution and have been successfully used for the object recognition task. They are usually composed of multiple layers, each layer being characterized by three nested functions, namely: (i) convolution; (ii) non-linear activation; and (iii) spatial pooling.

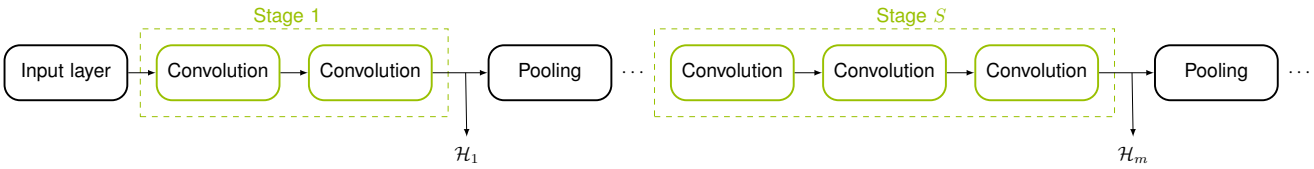
Let $\mathbf{X} = \{X_1, X_2, ..., X_N\}$ be a set of $N$ input images. Formally, a convolutional network $f$ composed of $L$ layers can be recursively defined as:

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

where $\mathbf{W}_l$ are the learned weights for the layer $l$, and $\mathbf{H}_l$ is the feature map produced by the convolutional layer $l$, defined as:

$$\mathbf{H}_l = activation(\mathbf{W}_l \mathbf{H}_{l-1}) \ \forall l \in \{1, ..., L-1\} \quad \text{and } \mathbf{H}_0 = \mathbf{X} \tag{2}$$

A convolutional layer can be followed by a pooling layer that downsamples the feature map $\mathbf{H}_l$. The set of convolutional layers before a pooling layer is called a stage.




(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 (**SLO**) and (b) side-outputs extracted at each convolutional layer (**ALO**)

A side-output layer is a layer producing a side-output map $\mathcal{H}_m$ from a convolutional layer. In our approach, the side-output layers operate a $1 \times 1$ convolution followed by a deconvolutional layer used to up-sample the side-output map to the input image size. This allows to cope with the different sizes of tensors depending on the layers. The deconvolutional layer is based on the transposed convolution [24]. In this process, the transposed weighted map is used to perform convolution between the side-output maps and an appropriate kernel to up-sample each position while maintaining the connectivity pattern. We expect side-output maps to carry information on the segmentation task at different layers of the network, thus at different scales.

The base network we use in this work is VGG16 [5] which is one of the first attempts to create deeper models, following the convolutional scheme. The network is composed of 13 convolutional layers and 5 stages that contain 2 or 3 convolutional layers. It uses the rectified linear unit defined as $ReLU(\cdot) = max(0, \cdot)$ as activation function and a max-pooling. We remove the last pooling layer as well as the fully connected layers as they are presented in a very small scale.

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 to the VGG16 network. Namely: (i) Stage Layer Outputs (**SLO**), inspired by the HED model, creating one side-output for each VGG16 stage; and (ii) All Layers Outputs (**ALO**), inspired by the RCF model, creating one side-output for each convolutional layer.

Therefore, in the case of **SLO**, the number of side-outputs corresponds to the number $S$ of pooling layers in the network and for **ALO**, it is equal to the number $L$ of convolutional layers.

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

$$\mathcal{H}_{SLO} = \{\mathcal{H}_1, ..., \mathcal{H}_m | m \in [1, S]\} \quad (3)$$

$$\mathcal{H}_{ALO} = \{\mathcal{H}_1, ..., \mathcal{H}_m | m \in [1, L-1]\} \quad (4)$$

Both strategies are illustrated in Figure 1.

### A. Merging strategies

When dealing with side-outputs in convolutional networks, the main question 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 $Z$ to be evaluated in the task, while retaining the useful information contained 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, we compare different merging functions that would enhance different desirable behaviors, 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 the most confident values.

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

$$Z_{ADD} = \sum_{i=1}^{M} (\mathcal{H}_i) \quad (5)$$

$$Z_{AVG} = \frac{\sum_{i=1}^{M} (\mathcal{H}_i)}{M} \quad (6)$$

$$Z_{MAX} = \max_{1 \leq i \leq M} (\mathcal{H}_i) \quad (7)$$

The operations are performed element-wise on each side-output. After element-wise combination, a convolutional $1 \times 1$ operation is performed again with ReLU activation, producing the final prediction $\hat{Y}$. The overview of the method is illustrated in Figure 2.

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.

Fig. 2. Method overview

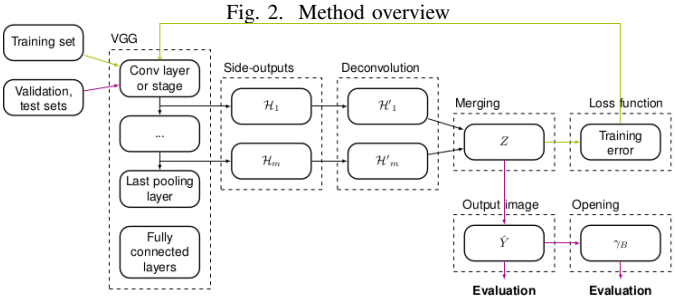
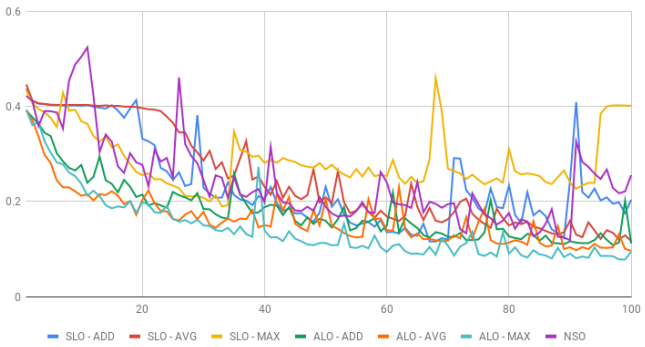



Fig. 3. Categorical Cross Entropy Validation Loss




### B. Post-processing

In this work, we propose to use mathematical morphology as post-processing step, meaning that this step is not inserted in the learning stage. Mathematical morphology is consistent with the non-linear image analysis, presenting solid theoretical foundation and idempotent functions [11]. 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 opening, which tend to destroy the small, and conspicuous areas according to the size of the structuring element. Thus, this operation removes objects that are relatively smaller than structuring element.

This additional step could reduce possible noises on the final result and improve the accuracy on distinguishing the road from other objects presented on the image.

## IV. Experiments

Experiments were conducted using the KITTI Road/Lane dataset, part of KITTI Vision Benchmarking Suite [25]. This dataset contains images for road and lane estimation for the task of image segmentation. It is composed of 289 training and 290 test RGB images, with the size of $1242 \times 375$ pixels. The ground-truth is manually annotated for two different road types: (i) **road** is the road area composing all lanes; and (ii) **lane** is the 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 the KITTI Server.

Even if roads and lanes are tagged in this dataset, we only consider in this work road tags. Lane annotations are ignored. The road type is divided in three different categories of road scenes, namely: (i) urban unmarked (**uu_road**), (ii) urban marked (**um_road**), and (ii) urban multiple marked lanes (**umm_road**).

To increase the number of images in the training set, a data augmentation procedure is performed. The following transformations were applied: pepper/salt noise, horizontal flipping (mirror), contrast change, brightness change, noise shadow and random rain/snow. 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 result in 2601 images, divided in 2080 samples for training and 521 samples for validation (about 20%).

### A. Experimental setup

Our networks were built using using Keras [26] with Tensorflow [27]. 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. To fit the network and speed up the process, all images were reduced to $624 \times 192$ pixels (about 50%). Training experiments were performed in GeForce GTX 1080 8GB GPU.

The **SLO** network is composed of $n = 5$ side-outputs, and the **ALO** network is composed of $n = 13$ side-outputs. The merging operations **ADD**, **AVG** and **MAX** are available for both ALO and SLO methods, resulting in 6 different approaches. As a baseline, we use the VGG16 network without any side-output, keeping only the final output. We call this baseline No Side Outputs (**NSO**).

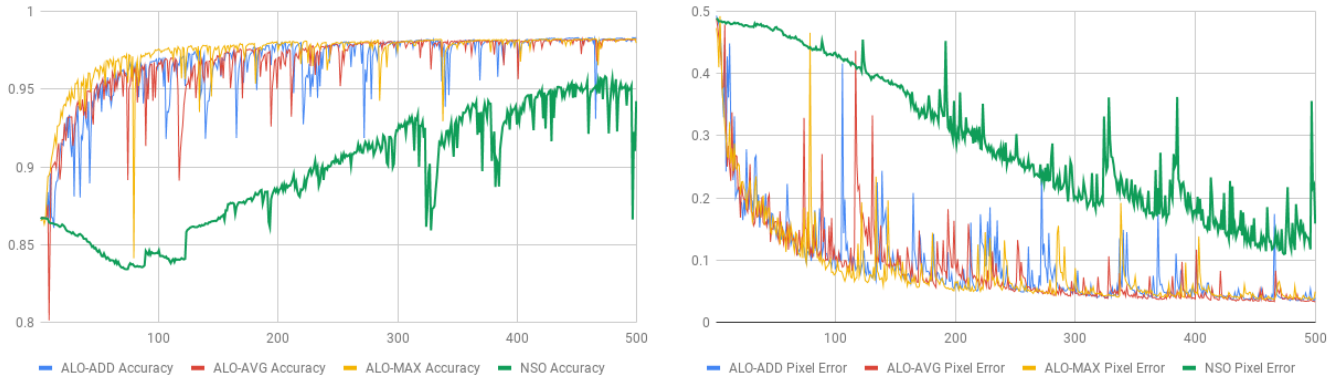
### B. Training results - Methods Comparison

The first test was designed to identify the best neural network and its best merging methods. We train all nets combined with all merging methods for 100 epochs to determine the best combination. This experiment leads us to understand how layers can be easily combined to produce outputs with good precision.

Figure 3 presents the categorical cross-entropy loss curves obtained during the training phase for the proposed approaches. ALO networks appear 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 produce high instability during the learning. On the other hand, ALO-AVG presents the best result for this experiment, followed by ALO-MAX and ALO-ADD merging strategies.

From the previous graph, it is possible to conclude that ALO networks have superior and more desirable behavior than the SLO and NSO models. These results are probably due to considerably larger amount of side-outputs, which create more possibilities of interchangeability between confident values.

Fig. 4. Categorical Cross Entropy Validation Accuracy and Pixel-Error results for 500 epochs validation set




### C. Best results

In order to improve the results, a new set of tests were performed using 500 training epochs. As some networks have a poor performance in the previous test and other tests with different parameters, we decided to evaluate all ALO networks in this new round of tests. We also trained NSO network for sake of comparison.

To evaluate the performance of the different approaches, we use two different metrics. The first one is the well known categorical cross entropy accuracy. We also use a metric called pixel-error. This measure evaluates the number of pixels incorrectly classified over the total number of the pixels. The metric was adopted when we observed high values of accuracy in results in which there were visibly many errors, mainly in the existence of numerous false positives. Performances according these two metrics are shown in Figure 4.

We can clearly see in this figure that side-outputs influence the performance of the networks. All ALO networks outperform NSO network. Also, learning curves of ALO networks show that the network learns faster when using multiple side outputs. Hence, it is possible to use bigger learning rate parameters, improving the learning performance at early stages.

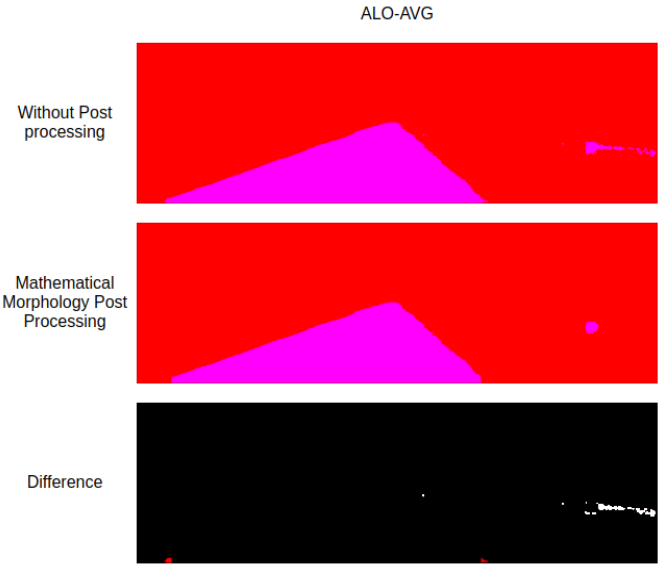
The best results of both metrics are quite similar for all merging strategies. This indicates the absence of a far better method to combine side-outputs in ALO network. The best result for cross entropy validation metric is just **0.0009** above the worst one (0.983 for ALO-ADD and 0.9821 for ALO-AVG). For pixel-error, the best value is just **0.0040** above the worst one (0.0332 for ALO-AVG and 0.0372 for ALO-MAX). According to the pixel-error metric, ALO-AVG is the best approach on the validation set.

### 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, we used the mathematical morphology operation of Opening, as defined in Section III-B. This procedure removes small noises created by the foreground (the road) in the background.

The opening operation was applied using square structuring elements full of ones. We used a kernels with the size of $13 \times 13$. It allowed sections incorrectly classified by the network to be eliminated by the sequential thinning of noises in different shapes. The results also become smoother with this procedure.

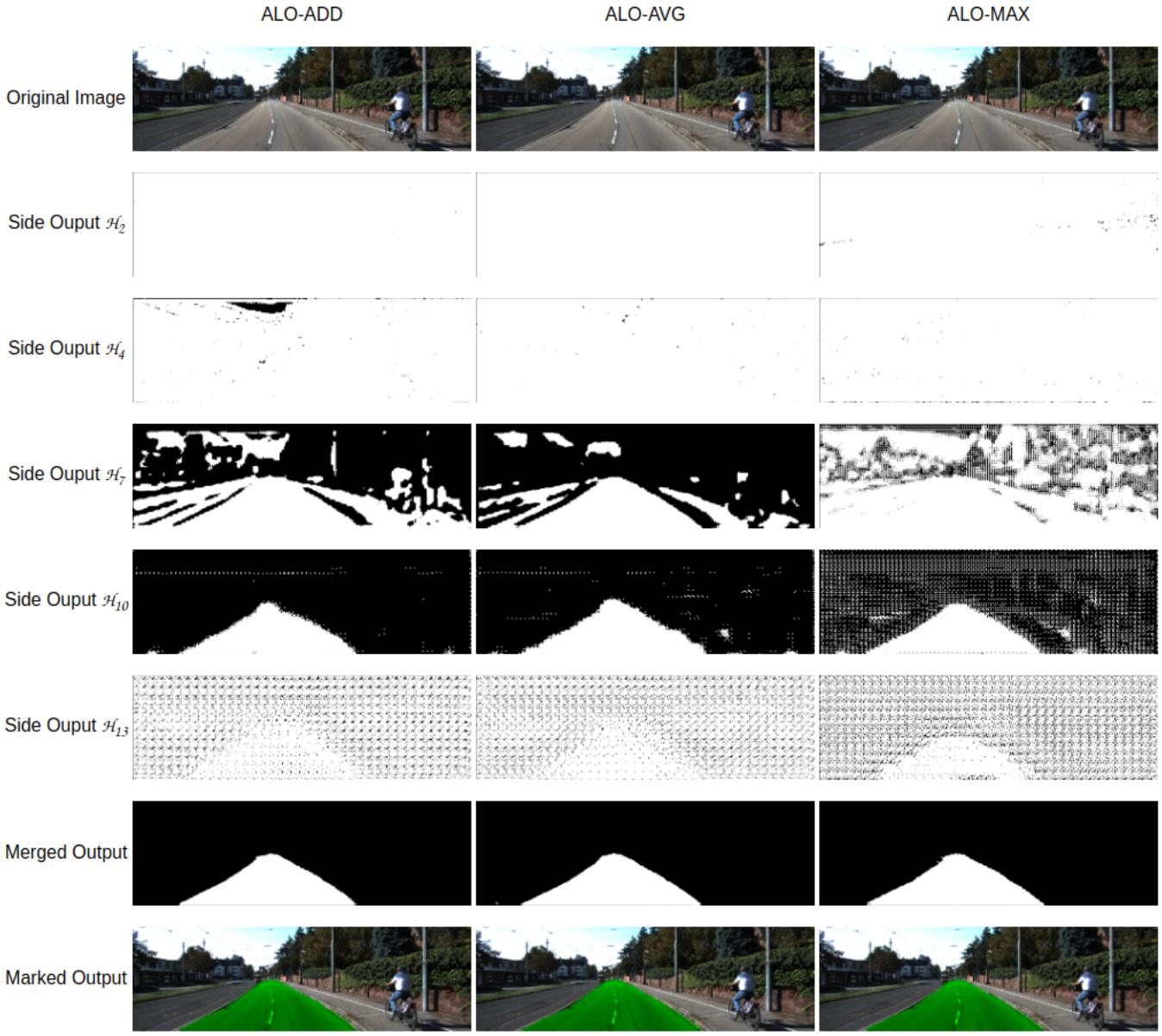
Fig. 5. Comparison between ALO-AVG without post processing and ALO-AVG with post-processing with mathematical morphology. In the last picture, *white* pixels represents desirable differences while *red* pixels represents undesirable ones.




A simple comparison of our procedure with the original network prediction is presented in Figure 5. 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 the noise in the far right of the image (*white pixels*). The noise removal increases the confidence, as small variations in the results can lead to a potential problem, if used in a self-driving vehicle.

A side effect of this method is the removal of some points that seems to fit correctly. This situation happens frequently in

Fig. 6. Side outputs for each merging strategy in ALO network.




the base of the road proposition. In Figure 5, it can be seen in the bottom left and the bottom right of the road (*red pixels*).

### *E. Side-outputs contribution in each merging strategy*

Each layer of each merging strategies learns in its particular way. The merging strategy influences how the networks learn. In Figure 6, we can see the contribution of layers to the final segmentation output. To simplify the study of side-outputs, we decided to visualize only the last output from each stage in **ALO** networks. To make the layers outputs easier to analyze, images were converted to black and white, where white pixels represent roads and black pixels background.

Figure 6 indicates that the first two stage side-outputs ($\mathcal{H}_2$ and $\mathcal{H}_4$) do not produce significant information. Images are almost white, indicating that all pixels were classified as road. ALO-AVG and ALO-ADD third layer contain a clear separation between road pixel and background pixels. ALO-MAX's third layer, on the other hand, does not clearly separate road from non-road pixels.

Figure 6 also indicates that fourth stage side-output layer ($\mathcal{H}_{10}$) clearly contains the best side-output for all ALO networks. The road marks are clearly visible, but with some noise. ALO-MAX contains a lot of noise, much more than ALO-ADD. The final stage side-output ($\mathcal{H}_{13}$) contains a lot of noise, which induces worse results than the previous layer. This possibly indicates that the layer was not able to correctly learn the information from the previous ones.

TABLE I
KITTI BENCHMARK EVALUATION RESULTS FOR ALO-AVG-MM

| Benchmark | MaxF | AP | PRE | REC | FPR | FNR |
|---|---|---|---|---|---|---|
| UM_ROAD | 91.15% | 83.82% | 89.07% | 93.33% | 5.22% | 6.67% |
| UMM_ROAD | 94.05% | 90.96% | 94.82% | 93.29% | 5.60% | 6.71% |
| UU_ROAD | 89.45% | 79.87% | 85.40% | 93.90% | 5.23% | 6.10% |
| URBAN_ROAD | 92.03% | 85.64% | 90.65% | 93.45% | 5.31% | 6.55% |

The merging layer combines all side-outputs (including the ones not shown in Figure 6) in order to make a decision. Despite poor results on some layers, the learning process adjusts itself so that even low accuracy results can be used by the model, similar to ensemble methods.

### *F. Evaluation results and comparison with the state-of-the-art*

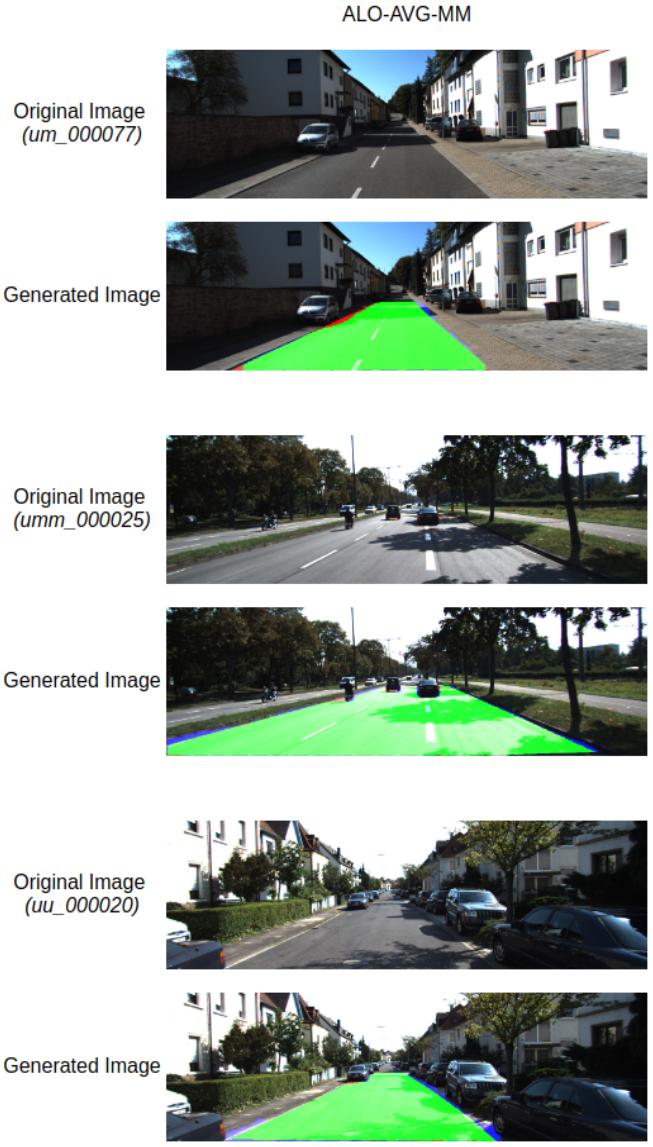
Reminding that the test evaluation could only be performed using KITTI Server, the metrics provided are maximum F1-measure (MaxF), average precision (AP), precision (PRE), recall (REC), false positive rate (FPR) and false negative rate (FNR).

The server tests were performed using ALO-AVG method, the best one in the training process. We use the name **ALO-AVG-MM** for our 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.

Compared to the best result in KITTI Server platform (called *PLARD*, an anonymous submission[1]), the results had overall performance 5.0% below. Compared with the best paper submission [28], the results were 4.0% below. Since our model was trained only for 500 times with high learning rate, it is expected that better results will be achieved with the fine-tuning of our model, which is not the main focus of this paper.

To show the performance of our model, a visual representation of ALO-AVG-MM predictions are presented in Figure 7. This image shows the true positives predictions (marked in green), false negatives (in red) and false positives (in blue). The image contains results provided by Kitti Evaluation Server, based in the generated binary map sent to the server.

Fig. 7. Visual representation of the results. The generated image was provided by Kitti Evaluation Server. Green pixels represents true positives, red pixels are false negatives and blue pixels are false positives.




## V. 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. We compare 3 different merging strategies to combine the side results: $add()$, $avg()$ and $max()$. The functions to enhance were evaluated using the cross-entropy accuracy and the pixel-error rate. The impact of the number of side-outputs was studied and compared to a version without any side-outputs for a similar network architecture. At last, a simple mathematical morphology operation was proposed to enhance the performance on the task and remove some noise.

Experiments demonstrated that the $avg()$ function is viable for merging side-output maps with different sizes and content, 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 in this paper.

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

This research opens novel opportunities for study such as:

[1]Results accessed in 2018-01-14. The method used was not described and no paper publishing information was provided.

(i) exploring different merging functions, less sensitive to fluctuations in values; (ii) exploring regularization techniques to sustain larger amounts of side-outputs consistent; and (iii) insert the mathematical morphology kernels in the learning process to search for the best kernel size.

The code and the list of dependencies to reproduce the experiments (under Anaconda environment) are publicly available online in https://github.com/falreis/segmentation-eval.

## VI. Acknowledgements

This paper acknowledges Github repositories https://github.com/lc82111/Keras_HED and https://github.com/moabitcoin/holy-edge that were helpful to provide some basic source codes used in this work.

## References

[1] D. Domínguez and R. R. Morales, *Image Segmentation: Advances*. Magnum Publishing LLC, 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*. Cambridge, MA: MIT Press, 2016, vol. 1.

[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*, 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 for Efficient Image Segmentation," in *Computer Vision – ECCV 2016*. 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*. 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)*, Jul 2017, pp. 5872–5881.

[11] L. Najman and H. Talbot, *Mathematical morphology: from theory to applications*. Hoboken, USA: John Wiley & Sons, 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*, 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*. 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] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *CoRR*, 2016.

[25] 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.

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

[27] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[28] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde, "Lidar-camera fusion for road detection using fully convolutional neural networks," *Robotics and Autonomous Systems*, 2018.