Combining convolutional side-outputs for road image segmentation

Felipe A. L. Reis[†], Raquel Almeida[†], Simon Malinowski[*], Ewa Kijak[*]
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
{simon.malinowski, ewa.kijak}@irisa.fr

***Abstract*—Image segmentation consists in subdivide an image into meaningful areas and objects. It can be use in scene understanding and recognition, in fields like biology, medicine, robotics, satellite imaging, amongs others. Some recent approaches uses convolutional neural networks to achieve this goal. In this work it is proposed to explore the learned model in a deep architecture, study the impact of the amount of side-outputs and evaluate strategies to combine side-outputs extracted at different layers of the network. It is also proposed the use of a post-processing filtering based on mathematical morphology idempotent functions in order to remove some undesirable small segments. Experiments were performed in the public available KITTI Road Dataset for image segmentation, and our proposed approach achieve results comparable to the state-of-the-art.**

***Index Terms*—convolutional neural network, image segmentation, mathematical morphology, CNN, region detection, side-outputs, side output, 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 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, deep learning approaches have 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, proposals initially replaced hand-craft features in the low-level analysis 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 from a deep network [7]–[10].

One challenge on the latter strategy is the combination of the output from distinct layers, considering that they have different sizes and could represent different aspects of the input. These outputs are network samples, which do not influence the architecture and are therefore often called side-outputs. In this work, we propose some strategies to combine the 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 which are necessary to create a viable region proposition.

[E] side-outputs should be defined [F] Done

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 cars. 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 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 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 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.

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 demonstrates the learning progression from detailed to coarser representations as the scale and resolution reduce through the network. 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 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 explicitly 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 creates 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 inserts 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 applies 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, color, gradient and variance to describe images. The goal of their proposal was 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 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 a non-linear function is used 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 as object proposal, object detection, semantic contour and segmentation.

Finally, the RCF network, not only creates 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 concatenations.

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 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. [E] what mean whilst here? task=image segmentation = creation of coherent regions ?

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) spatial pooling; and (iii) non-linear activation.

Let's consider $\mathbf{X}$ set of N input images I. Formally, let a convolutional network f composed by L layers be defined as: [E] X should be defined [F] Done

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

in which:

- $\mathbf{W}_l$ is the associated weights for the layer l;
- $\mathbf{H}_l$ is the output of the layer l, defined as

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

For consistency, consider $\mathbf{H}_0 = \mathbf{X} = \{X_1, X_2, ..., X_N\}$.

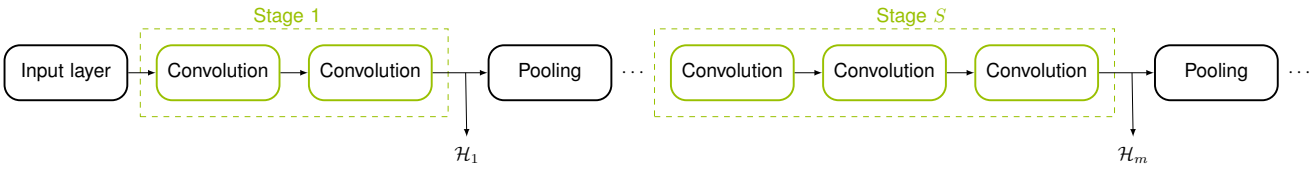
The VGG network [5] is one of the first attempts to create deeper models following the convolutional scheme. The core of the layers in VGG is defined by a convolution C immediately followed by a rectified linear unit, as follows: [E] define C_l [F] Done

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

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

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

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



(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

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 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 corresponds 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.

A. Merging strategies

[E] a subsection cannot live alone :) [F] Done

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 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, 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. To cope with the different sizes presented at different layers, the side-output maps are first re-scaled to the input image size using a transposed convolution layer [24], also called as "deconvolutional layer". In this process, the transposed weighted map is used to convolve the side-output maps with an appropriate kernel to up-sample each position 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.

[F] Maybe we should put formulation for every pixel evaluation here

After pixel-wise evaluation for a single image, it is necessary to evaluate a set of images. 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.

[F] A formulation should be here or we should remove formulations

B. Post-processing

[E] Maybe to short to be a section [F] Now subsection

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 (also know as kernel).

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. 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.

[11]

IV. Experiments

Experiments were conducted in the KITTI Road/Lane dataset, part of KITTI Vision Benchmarking Suite [25]. The dataset contains images for road and lane estimation for the task of image segmentation. It consists of 289 training and 290 test RGB images, with the size of 1242 pixels width and 375 pixels height. 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. Some images contains also sidewalks, that was not evaluated in this paper - sidewalks were classified as background. 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.

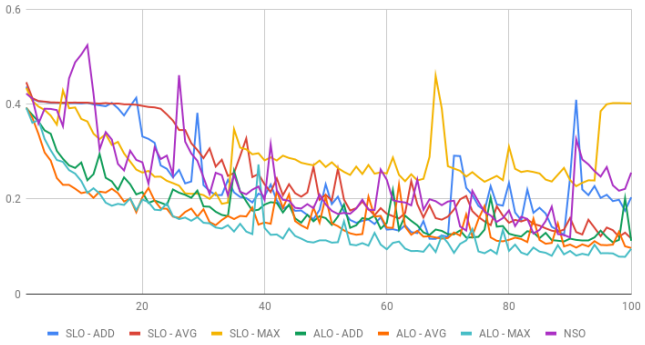
[E] What is the resolution of the images? [F] Done

In this work, only the road ground-truths is used and the lane annotations are 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.

[E] Should be good to show examples of images and groundtruths from the dataset [F] I think we won't have enought space in this paper

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 resulted in 2601 images, divided in 2080 samples for training and 521 samples for validation (about 20%).

Fig. 2. Categorical Cross Entropy Validation Loss [E] Only Pixel Error according to the text [F] Changed



A. Experimental setup

Our networks were build 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 reduce to 624 pixels width and 192 pixels height (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 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. As a baseline, we use the VGG16 network without any side-output but only the final output, called No Side Outputs (**NSO**).

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 achieves the best results. This conduct lead us to understand how layers can be easily combined to produce outputs with good precision.

Figure 2 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 in the learning course. On the other hand, ALO-AVG presents the best result for the test, followed by ALO-MAX and ALO-ADD merging strategies.

From previous graph, it is possible to conclude that ALO networks had 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 confident values.

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 in the previous test and other tests with different parameters, we decided to evaluate only ALO network in this new round of tests.

To measure the performance in our tests we use two different metrics. The first one is the well known categorical cross entropy accuracy. Also, we use an metric called pixel-error. This measure evaluates the number of pixels incorrectly classified over the total number of the pixels (number of wrong pixels). These two metrics are available in Figure 3.

The best results of both metrics are quite similar for all networks. This indicates absence of a far better method to combine side-outputs. 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).

Due to the similarity of the results, we will indicate the best method using the value of validation pixel-error metric. Also, AVG fusion method was also previous used in [18] and [10] to combine the results. For this criteria, ALO-AVG was defined as the best method of our training set.

D. Post-processing using mathematical morphology

[E] Insert reference to section IV that describes that post-processing. [11] should go in section IV. [F] Partial Done

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 [11], 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 decided to use a set of kernels with sizes of 5×5, 7×7, 9×9, 11×11 and 13×13, sequentially applied into the image. This type of operation, although unusual, achieved the best results. It allowed that sections incorrectly classified by the network could be eliminated by the sequential thinning of noises in different shapes. The results also become more smooth with this procedure.

[E] I don't understand here. All the structuring elements are applied sequentially? which makes no sense since the opening by the largest structuring element is included in the opening by the smallest structuring element if I'm right. Or are they all tested independently? In this latter case, which size is finally retained? [F] Yes, sequencially. Although unusual, it removed incorrect classification better than small kernels alones. (Text changed for better explanation)

A simple comparison of our procedure with the original network prediction is presented in Figure 4. 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 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 the base of the road proposition. In Figure 4, 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. It is possible to see how each side-output learns and contributes to the final output in Figure 5. To simplify the study of side-outputs, we decided to visualize only the last output from each stage in **ALO** network, for our trained networks. To make the layers outputs easier to analyse, the images were converted to black and white, where white pixels were classified as road and black pixels were classified as background.

[E] Not clear relatively to Fig. 2 [F] Changed section. Removed Fig. 2 information.

[E] Fig. 5 is not so clear. Notation from the text should be reused: the side-output i what is denoted $\mathcal{H}_i$ (if I'm correctly understanding) [F] Changed. I don't know if it was clear enough. I'll think in how to improve it

Figure 5 indicates that the first two stage side-outputs (outputs $\mathcal{H}_2$ and $\mathcal{H}_4$) does not produce significant information. Images are almost white, indicating that all pixels were classified as road. ALO-AVG and ALO-ADD third layer contains a clear separation between road pixel than non-road pixels. ALO-MAX's third layer, on the other hand, does not clearly separate road from non-road pixels. The results are pixelated and blur when compared with the original image.

Figure 5 also indicates that fourth stage side-output layer (output $\mathcal{H}_{10}$) 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, while ALO-ADD contains a few ones. The final stage side-output ($\mathcal{H}_{13}$) contains a lot of noise, with results far away worse than the previous layer. This possibly indicates that the layer was not able to correctly learn the information from the previous ones.

The fuse layer combine all side-outputs (including the ones not shown in Figure 5) in order to predict. Despite poor results on some layers, the learning process adjusts itself so that even very few accurate results can be used by the model, similar to ensemble methods.

[E] = the proposition Z [E] ? Not clear [E] merging layers = fuse layer? [F] Fixed. Text improved but I don't know if it is good enough. I don't know if I can cite 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. 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.

Fig. 3. Categorical Cross Entropy Validation Accuracy and Pixel-Error results for 500 epochs test set

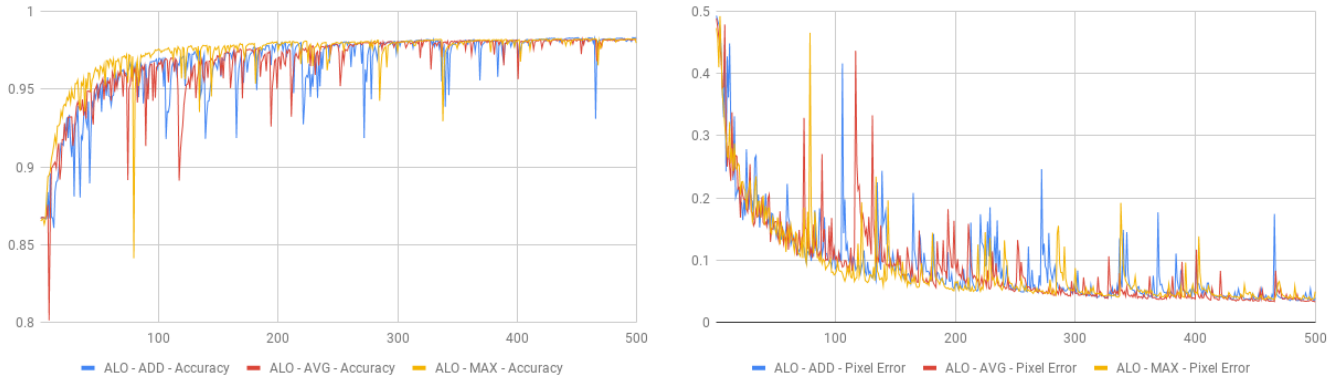
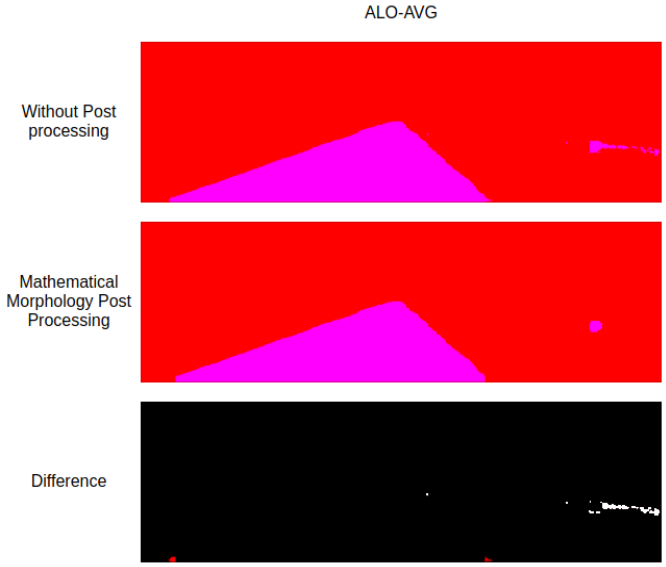


Fig. 4. Comparison between ALO-AVG without post processing and ALO-AVG with post-processing with mathematical morphology. In the last picture, *white* pixels represents desirable differences while *red* pixels represents undesirable ones.



If compared with the state-of-the-art (called *PLARD*, an anonymous submission on the KITTI Server platform [1]), the proposed method is comparable and sometimes superior, regarding the maximum F1-measure and the recall metrics. This is due to 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 and negatives. This indicates that the proposed methods are more precise in delineating the regions to be segmented.

A visual representation of ALO-AVG-MM predictions are presented in Figure 6. This image shows the predictions marked (in green) over the road, to show the performance of our model.

[1]Results accessed in 2018-01-13

TABLE I
KITTI BENCHMARK EVALUATION RESULTS FOR EACH CATEGORY

um_road

	ALO-AVG	ALO-AVG-MM	State-of-the-art
F-measure (MaxF)	%	%	**97.05%**
Avg. Precision (AP)	%	%	**93.53%**
Precision (PRE)	%	%	**97.18%**
Recall (REC)	%	%	**96.92%**
False Positive (FPR)	%	%	**1.28%**
False Negative (FNR)	%	%	**3.08%**

umm_road

	ALO-AVG	ALO-AVG-MM	State-of-the-art
F-measure (MaxF)	%	%	**97.77%**
Avg. Precision (AP)	%	%	**95.64%**
Precision (PRE)	%	%	**97.75%**
Recall (REC)	%	%	**97.79%**
False Positive (FPR)	%	%	**2.48%**
False Negative (FNR)	%	%	**2.21%**

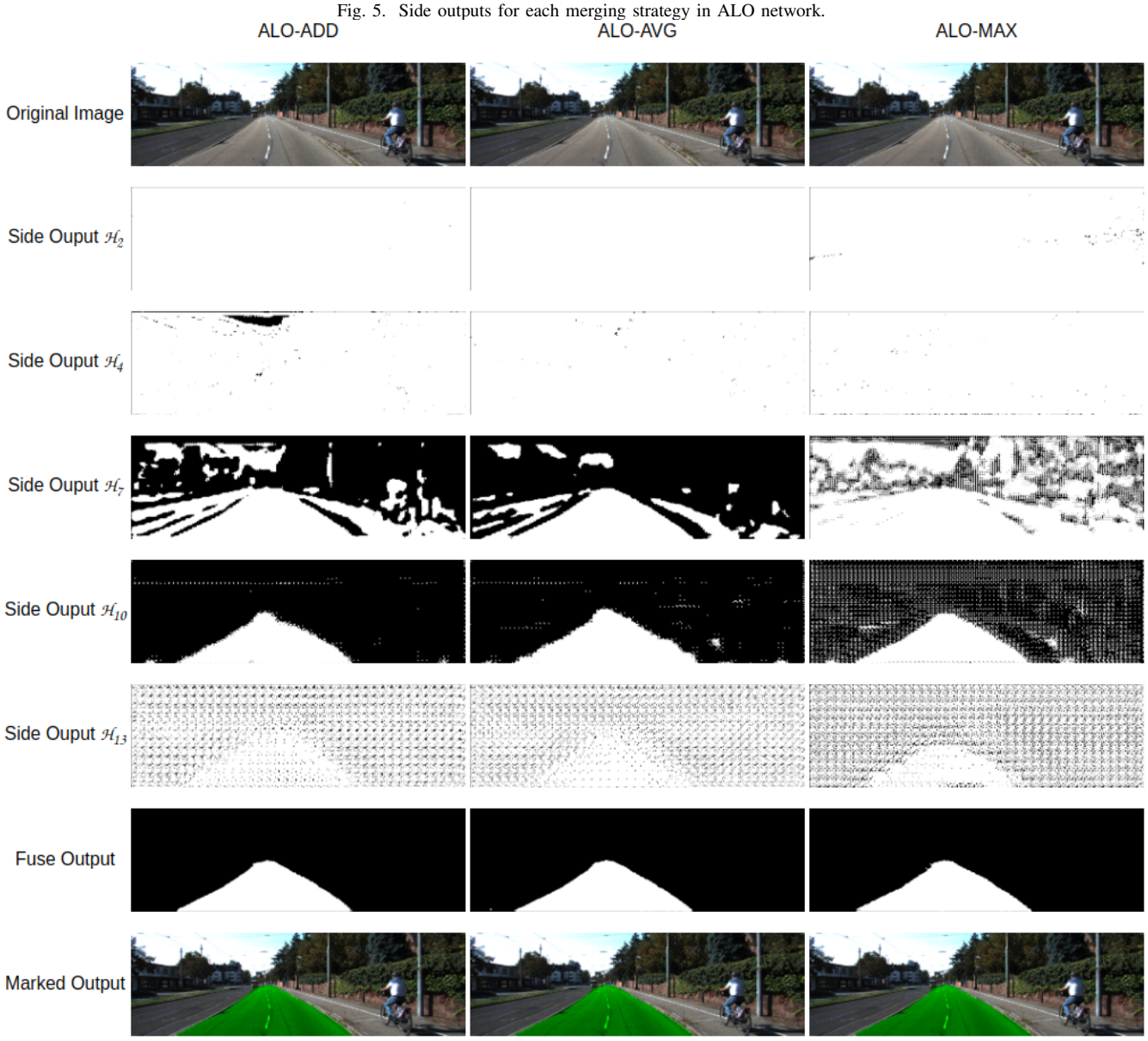
uu_road

	ALO-AVG	ALO-AVG-MM	State-of-the-art
F-measure (MaxF)	%	%	**95.95%**
Avg. Precision (AP)	%	%	**95.95%**
Precision (PRE)	%	%	**96.25%**
Recall (REC)	%	%	**95.65%**
False Positive (FPR)	%	%	**1.21%**
False Negative (FNR)	%	%	**4.35%**

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 a cross-entropy and pixel-error loss functions. 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

Fig. 5. Side outputs for each merging strategy in ALO network.



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 creates jumps that could lead to better performance, in terms of accuracy. The training graphs also shows that the number of side-outputs contributes to 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) exploring 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.

Fig. 6. Visual representation of the results

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. 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] 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," https://keras.io, 2015.

[27] 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/