U-Net with pre-processed edge detection and learned down-sampling for semantic segmentation

K.H.W. Stolle - k.h.w.stolle@student.tue.nl

TABLE I
TARGET LABELS

Category	Labels
Flat	road, sidewalk, parking, rail track
Human	person, rider
Vehicle	car, truck, bus, on rails, motorcycle, bicycle, trailer
Construction	building, wall, fence, guard rail, bridge, tunnel
Object	pole, pole group, traffic sign, traffic light
Nature	vegetation, terrain
Sky	sky

***Abstract*—In this paper, a possible solution for the Cityscapes Pixel-Level Segmentation task is discussed. The U-Net architecture is taken as a baseline. Various improvements from literature are tested to improve the baseline score. The results are summarized in figures an images from the training process. This yields a final intersection-over-union-score of 0.84 on a downscaled version of the dataset. The results were not submitted to the testing suite, because the dataset was scaled down too much in order to be workable on the available hardware.**

I. INTRODUCTION

The Cityscapes Dataset involves a large number of pictures taken from the front of a car while driving through various cities in Germany [1]. It was published in an effort to advance research in image-recognition tasks. This paper focuses on pixel-level semantic segmentation. This involves predicting a per-pixel semantic labeling of the image without considering higher-level object instance or boundary information.

First, a baseline is set using an off-the-shelf architecture for semantic segmentation. This baseline network is then improved by data augmentation and iterative design, after which this solution is tested. The results of testing are finally interpreted and discussed.

The Python, Julia and LaTeX sources as well as the collected data of this research are published at
`https://github.com/kurt-stolle/tue-cityscapes-segmentation`

II. METHOD

A. The dataset

As with any machine learning project, first the available data has to be parsed into a practical format. The dataset consists of a collection of PNG-encoded images (the input) and a corresponding segmentation mask (the ground truth). The ground truth is a color-coded image, with RGB-channels, where every class corresponds to a single and unique color. The desired output of the network is an image with one channel per class, each represents the probability of that class being present at the image's pixel. Parsing the ground truth-image into the encoded channel representation (one-hot encoding) is done by iterating over the set of classes (Table I), comparing the mask's color at a pixel to the expected color for this class.

B. Baseline & testing environment

The baseline implementation sets a reference point to improve upon and score our method against.

The architecture U-Net is suitable for semantic segmentation of biological cells under a microscope [2]. In this paper, the same implementation is used as a baseline for the segmentation of the Cityscapes Dataset.

Our implementation of the UNet is based on [3] with some modifications to work with the Cityscapes Dataset. Because the UNet architecture requires a large amount of GPU-memory to train, all inputs of the dataset have been scaled down by a factor of 0.2 before feeding the network.

The network is optimized by the Adam optimizer [4] using Categorical Cross-Entropy as the loss function [5].

C. Data augmentation

A straightforward way to improve the accuracy of the model is to increase the size of the training set by applying a set of transforms. The following transforms were used:

- Random cropping
- Random mirroring over the horizontal axis ($p = 0.5$)

D. Measuring performance

In order to measure how well the network is performing, the Intersection-over-Union (IoU) metric is implemented, given by

$$\mathrm{IoU} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{1}$$

where TP is the amount of true-positive pixels, FP is the amount of false-positive pixels and FN is the amount of false-negative pixels. A pixel is positive if it has a class-index greater than 0, and a pixel is true if the predicted value (from the output mask) is equal to the actual value (from the ground truth mask).

E. Identifying the model's weaknesses

In addition to collecting the IoU and loss values at each step of the training process, we must also find a way to identify at which types of images the model performs worse than others. To achieve this, the sample ID from the dataset must be supplied to the training process, such that the performance

Fig. 1. The output of an edge-detection filter from the Pillow Python library when applied on a down scaled sample of the Cityscapes dataset. *We encourage the reader to zoom-in digitally to view the fine edges generated by the filter.*

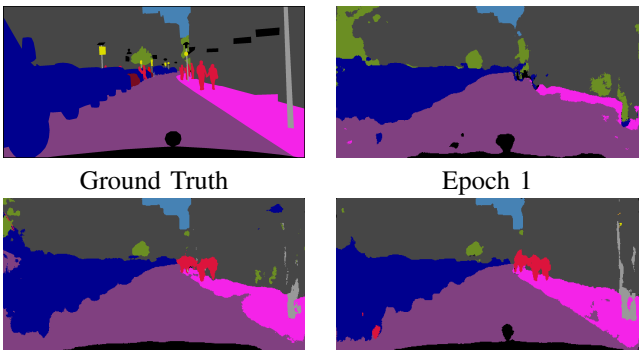


Fig. 2. Output samples of the baseline (U-net) network on a random sample of the validation set

against each sample can be logged. It is important to note that this data should not be used to affect the training process, for this will cause overfitting.

F. Decision threshold

Sometimes, the case can occur where the softmax-likelihood of a pixel corresponding to a class is very low. In this case, it would be better to predict that the class label of this pixel is void, than to predict a label with low confidence. Dropping all class-probabilities below a threshold-value addresses this issue by classifying all pixels with a likelihood less than a set value as zero [6]. Because the Cityscapes Dataset has 20 classes, the threshold was set at

$$p_{\mathrm{T}} = 2 \cdot \frac{1}{N_{\mathrm{classes}}} = 0.1 \tag{2}$$

G. Edge detection as input

A lot of semantic segmentation tasks struggle with classes bleeding into other classes. A possible was to solve this could be to use a static (non-learned) edge detection filter, and feed this into the network by replacing the alpha channel with the single-channel output of this layer [7].

Figure 1 shows the output of such a filter. The hypothesis is that this will help the edges of the semantic segmentation mask to more accurately stick to the edges of objects. This comes at a cost though: adding a another channel will increase the size of the filter kernel at the input channel, making the network consume more memory. Additionally, if the amount of channels of the first layer does not change, then this layer will not be able to extract as much features from the input as in the 3-channel (RGB) case.

H. Regularization using dropout

While the most popular fully-convolutional networks do not use dropout, implementing dropout in the convolutional layers of the U-Net could successfully regularize the network, improving the performance on unseen samples [8]. A dropout was added after each convolutional layer.

I. Strided convolutions

In order to make the network practical, the input images must be scaled to a size that the training hardware can handle. The scaling of images causes a loss of information. For example, at the scaling factor of 0.2 (the maximum that the network can handle before optimization), lamp-posts almost disappear because they are too thin. A better way to deal with this, would be to downsample the images using strided convolutions. From the course 5LSM0, it is known that the output size of a convolutional layer can be calculated via

$$O = \frac{I + P - K}{S} + 1 \tag{3}$$

where I is the input size, P the amount of padding, K the kernel size and S the stride. Using this equation, the first two convolutional layers of the network are set-up to each downsample the input image by a factor of 0.5 (for a total of 0.25). The output segmentation mask is then upscaled using the nearest-neighbours method. This should yield a mask with *rougher edges*, but higher IoU-accuracy.

Additionally, the (un)pooling operations may be replaced with a strided convolution in order to learn the up- and down-scaling operations of the network [9]. This could significantly improve the accuracy of the network.

J. Automatic learning rate adjustment

The learning rate can be automatically adjusted based on the `ReduceLROnPlateau` method, which is part of the PyTorch standard library. The starting learning rate is set to $R = 0.01$ (a value determined by trial-and-error) and decreased by a factor of 0.1 every time the IoU score does not increase for a to-be-determined amount of cycles. This should help the network to converge faster than traditional learning with a lower rate [10].

III. Results & Discussion

A. Baseline

Training the baseline U-net on the Cityscapes dataset achieves a maximum IoU-score of 0.45 and a loss of 0.55 after 10 epochs.

Figure 2 shows some output samples of this network for a random sample in the validation set. Notice how the network

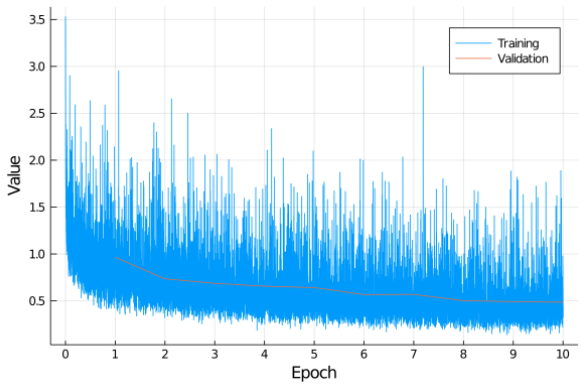


Fig. 3. The loss as a function of the training epoch for the baseline (U-net) network after data augmentation

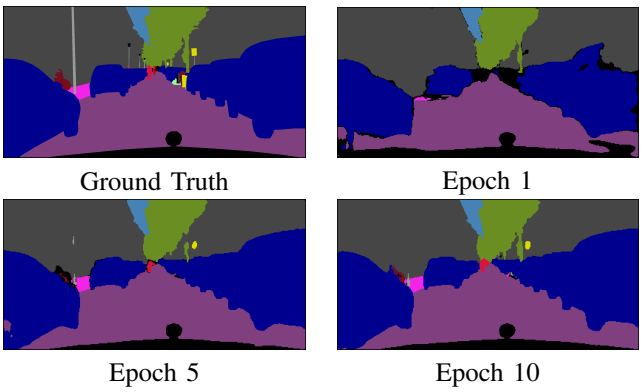


Fig. 4. Ground truth and predicted masks after training for 1, 5 and 10 epochs on a random sample of the validation set using the U-Net model with thresholds and data augmentation

fails to recognize thin objects such as poles and signs. Some parts of the image, such as the sidewalk, appear noisy even after 10 epochs.

B. Data augmentation

Figure 3 shows the loss function during the training process. Without much effort, the loss has decreased from 0.55 to 0.50

C. Threshold

Figure 4 shows the effect of a decision-threshold on the output masks. It appears that the regions with lower confidence, such as at the edges of an object, get ignored sometimes. After 10 epochs, the class masks appear to more closely match the contours of objects.

Notice from Figure 5 that the IoU now starts at 0 in the untrained network, when all predictions have a confidence less than 0.1. This is to be expected with this method.

Because the decisions of the network do not affect the cross-entropy loss, the loss curve is still exactly the same as in Figure 3. Interestingly, the loss slowly decays while the IoU remains roughly the same after 2 epochs. This can be explained by the nature of loss and decisions: after 2 epochs, predictions are reinforced with a stronger confidence but the classes that hold maximum values that lead to a decision remain roughly the same.

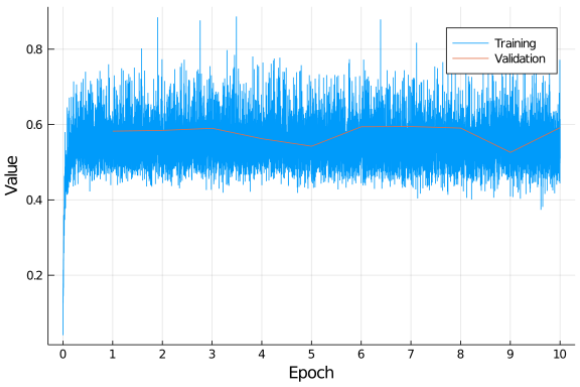
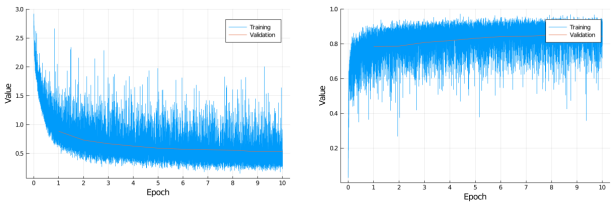


Fig. 5. The IoU as a function of the training epoch for the network with a decision threshold of 0.1.



(a) Loss (b) IoU

Fig. 6. Loss and IoU as a function of the training epoch for the network with data augmentation, decision threshold and pre-processed edge detection as the 4-th channel of the input image.

D. Edge detection as input

Figure 6 shows the effects of edge detection pre-processing being fed into the network as the 4-th channel. While the loss does not improve significantly, there is a significant improvement of 0.2 for the IoU-score.

fig. 7 shows the output of this network after 10 epochs. This is the same sample as showcased in fig. 2. By inspection, notice that more fine-details are visible on the output with edge detection, while the logo on the bottom of the car (the black circle) seems to have disappeared.

E. Regularization using dropout

Figure 8 shows the results with the smallest dropout rate applied. Values between 0.1 and 0.5 were tried, but all values

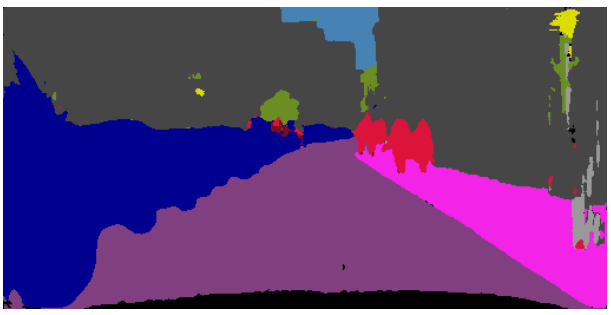
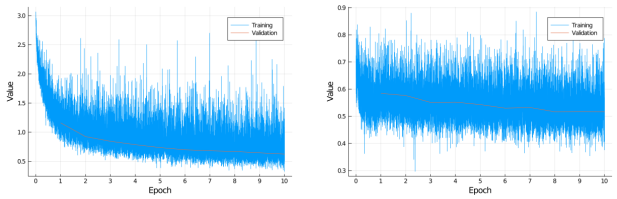
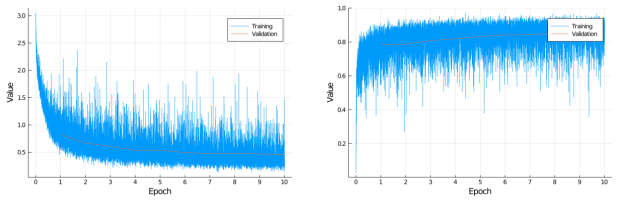


Fig. 7. Output of the network with edge detection pre-processing input after 10 epochs.



(a) Loss (b) IoU

Fig. 8. Loss and IoU as a function of the training epoch for the network with dropout ($p = 0.1$) after every convolutional layer.



(a) Loss (b) IoU

Fig. 9. Loss and IoU as a function of the training epoch with all optimizations enabled and using strided-convolutions to downsample the input

had a negative impact on the results. Dropout did thus not positively affect the network and the changes were discarded.

F. Strided convolutions

Figure 9 shows the loss and IoU-score when using strided convolutions to downsample the input. This yields an IoU of 8.4 and a loss of 0.45 after 10 epochs.

G. Automatic learning rate adjustment

Figure 10 shows the loss with automatic learning rate adjustment. At 2.5 epoch and at 8.5 epoch, the learning rate is adjusted. This is evident from the sudden drop in the validation curve. The automatic tuning of the learning rate has no effect on the result of the network, but the solution does appear to converge sooner. In order to make the adjustment work, the training process had to be adjusted to validate at more frequent intervals. This causes a net slowdown of the learning process. For this reason, this optimization is not part of the final solution.

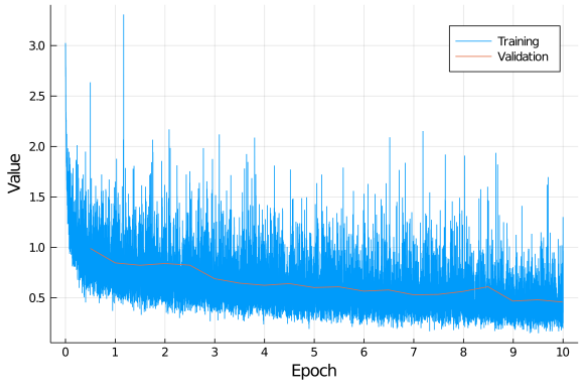


Fig. 10. Loss as a function of the training epoch with all optimizations enabled and automatic learning rate adjustment.

IV. Conclusion

The U-net architecture can be adjusted to perform pixel-level semantic segmentation on the Cityscapes Dataset with a IoU-score of 0.84. To achieve this, the following adjustments were made with respect to the baseline implementation:

- Data augmentation
- Adding a decision threshold
- Pre-processed edge detection as input
- Downsampling using strided convolutions

Note that a final score on the test-set was not achieved by the network, as the images had to be compressed too much to be accepted by the Cityscapes testing suite.

V. Future work

The most promising improvement was made by involving edge detection in the network. This was done by feeding a pre-processing step into the network's input. This may not be the optimal solution, as this reduces the amount of features that can be extracted. Another possible solution would be to feed the edge detection into the later layers of the network, after an image is reconstructed by the upsampling layers. Additionally, the edge detection could be also be a learned operation.

A limiting factor in this research was the amount of GPU-memory available. The memory usage of the network could be reduced by applying checkpointing, which trades computation time for memory usage, or by training the network with 16-bit floating point operations instead of 32-bit floating point operations. Future work will have to explore these options.

References

[1] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[3] milesial, "U-net: semantic segmentation with pytorch," https://github.com/milesial/Pytorch-UNet, 2018.

[4] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[5] S. Mannor, D. Peleg, and R. Rubinstein, "The cross entropy method for classification," in *Proceedings of the 22nd International Conference on Machine Learning*, ser. ICML '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 561–568. [Online]. Available: https://doi.org/10.1145/1102351.1102422

[6] X. Li, Z. Liu, P. Luo, C. Change Loy, and X. Tang, "Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[7] L. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille, "Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform," *CoRR*, vol. abs/1511.03328, 2015. [Online]. Available: http://arxiv.org/abs/1511.03328

[8] T. Spilsbury and P. Camps, "Don't ignore dropout in fully convolutional networks," *ArXiv*, vol. abs/1908.09162, 2019.

[9] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," 2014.

[10] L. N. Smith and N. Topin, "Super-convergence: Very fast training of residual networks using large learning rates," *CoRR*, vol. abs/1708.07120, 2017. [Online]. Available: http://arxiv.org/abs/1708.07120