Computer Vision Project - Image Segmentation

Tomás Amado
M.Sc. Data Science and A.I. student
7001572
toam00001@stud.uni-saarland.de

Pablo Valdunciel
M.Sc. Data Science and A.I. student
7010186
pava00001@stud.uni-saarland.de

Abstract

Image segmentation has important applications in different fields such as medical imaging (diagnosis, surgery planning,...), recognition tasks (face recognition) and computer vision. Segmentation methods based on deep neural networks have been evolving and achieving higher performance in different tasks over the last years. In this report, we summarize our work within the Computer Vision project of the *Neural Networks: Theory and Implementation* (WiSem2020/21) course at Saarland University, which focuses on understanding and implementing different deep neural network models for the mentioned methods. During this project, we familiarised ourselves with two image segmentation datasets: the well known PASCAL Visual Object Classes (PASCAL VOC) and the Cityscapes datasets. We successfully implemented a version of the R2U-Net model used in medical image segmentation, named R2U-Net64, that can be used in multi-class pixel-level segmentation tasks with the Cityscapes dataset and we improved this model's performance by implementing and incorporating a module based on height-driven attention networks (HANet) into the architecture.

1 Introduction

Image segmentation is the process of partitioning a digital image into multiple segments. There exist two major types of image segmentation: semantic segmentation, where each pixel is marked as belonging to a class of object, and instance segmentation, where every pixel is marked as belonging to an instance of an object. The focus of this project has been on semantic segmentation, also called pixel-level segmentation.

In the first part of the project, we worked with the PASCAL Visual Object Classes Challenge 2012 (VOC2012) dataset [Everingham u. a.], which consists of a variety of images containing objects that belong to up to 20 different classes (person, bird, bicycle, sofa, ...). Given an image, the objective was to generate a pixel-wise segmentation given the class of the object visible at each pixel, or *background* otherwise (see *Figure 1*). We approached this pixel-segmentation task by implementing a simple segmentation network named SimpleSegNet, powered with a learning transfer from a VGG-16 network [Simonyan und Zisserman (2015)] for image classification already trained on the ImageNet dataset [Deng u. a. (2009)].

For the second part of the project we implemented a version of a variant of the popular image segmentation architecture U-Net [Ronneberger u. a. (2015)] : R2U-Net [Alom u. a. (2018)] which combines deep residual [He u. a. (2015)] and recurrent convolutional networks [Liang und Hu (2015)] and adds them to the down-sampling and up-sampling convolutional layers in the original U-Net architecture with the purpose of achieving better feature representations. We trained this model on the Cityscapes dataset [Cordts u. a. (2016)], which consists of annotated images for the task of semantic urban scene understanding with 30 classes belonging to the commonly observed objects in streets (vehicles, pedestrians, traffic signs, ...) (see *Figure 2*).

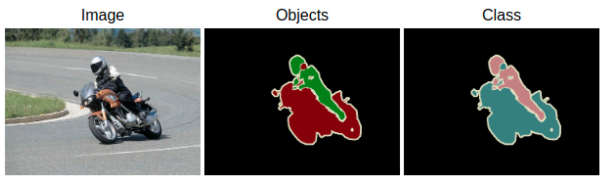


Figure 1: Pixel-level segmentation example from the VOC2012 challenge [Everingham u. a.]

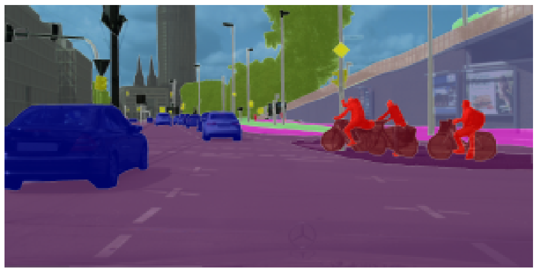


Figure 2: Pixel annotations example from the Cityscapes dataset [Cordts u. a. (2016)]

In order to improve our results obtained with our implementation of R2U-Net on the Cityscapes dataset [Cordts u. a. (2016)], we dag into the list of models that are currently at the top of the Pixel-Level Semantic Labeling Task of the Cityscapes Benchmark [Cordts u. a.]. In the process we studied different interesting ideas: the creation of a computational unit, the *Split-Attention block* [Zhang u. a. (2020)], which applies the channel-wise attention on different network branches and combines its diverse representations; an improved UNet architecture, called *UNet++* [Zhou u. a. (2018)] where the encoder and decoder sub-networks are connected through a series of nested, dense skip pathways; a hierarchical multi-scale attention pipeline, named HRNet [Tao u. a. (2020)], where the inferences based in the same image in several scales are combined; or, the inclusion of an *affinity loss* that supervises the feature aggregation to distinguish the intra-class (pixels of the same class) and inter-class context (pixels of different class) context [Yu u. a. (2020)].

In the end, we decided on using a general add-on module, called height-driven attention networks (HANet) [Choi u. a. (2020)], which considers the vertical position of a pixel to emphasize informative features or classes. The authors of this network studied the class distributions of the Cityscapes dataset on three different horizontal sections of the image (upper, middle and lower), showing that they are completely different. Usually, the upper part of the image is mostly composed of buildings, vegetation and sky, while the middle part contains different shorts of objects. The lower part is mainly composed of road. For instance, the probability of the *car* class p_{car} is 7.0% on average given an entire image, but this probability drops to 0.6% for the upper region (because cars cannot fly up in the sky), peaks on 17.4% in the middle region, and decreases again until 2.2% in the lower region of the image.

The report is organized as follows: in Section II, we describe the architecture of all the models that we have implemented; in Section III, we describe the experimental setup and analyse the obtained results; in Section IV, we explain our conclusions after working in this project.

2 Models Architectures

2.1 SimpleSegNet

The SimpleSegNet model is an extension of the VGG 16-layer model (configuration "D") [Simonyan und Zisserman (2015)] for image classification; the extension of the VGG-16 architecture consists in adding a convolution layer followed by a transpose-convolution layer to obtain the desired output dimensions of the segmentation task. As opposed to other models that extend the VGG-16 architecture for segmentation tasks, the final 3 dense layers have not been removed in this case. In order to implement SimpleSegNet, we used the already implemented PyTorch VGG-16 model [PyTorch] and, to obtain more satisfactory results and reduce the necessary training, we initialized the corresponding part of the network with the weights obtained after pre-training the VGG-16 model on a subset of the ImageNet dataset [Deng u. a. (2009)].

In parallel to the mentioned model, we also implemented the SegNet architecture [Badrinarayanan u. a. (2016)], which is also an extension of the VGG-16 architecture. However, some yet undetected error in this implementation leads to poor predictions that are very far from the ground-truth segmentation. Our implementation of SegNet has been included in our repository, although we haven't analysed its results in detail.

2.2 R2U-Net64

The R2U-Net model is based on U-net [Ronneberger u. a. (2015)]. The U-net model performs a number convolutions on each layer along with max pooling to obtain a larger amount of channels and reduce the size of the input, and then performs transpose convolutions while reducing channels until the number of channels is equal to the number of classes to predict. This gives the model U-shape. In addition to this, the output of each down sampling layer is cropped and concatenated to the corresponding up-sampling layer.

R2U-Net [Alom u. a. (2018)] changes this architecture by replacing the convolutional layers in both the down-sampling and up-sampling section of the model with recurrent residual convolutions units. In these units, convolutions are performed in a recurrent manner: the output of a convolution is fed into another convolution with shared weights and, at the same time, the original input is added to this result. After two of these recurrent convolutions, a residual operation is performed: the original input that is fed to the recurrent residual unit is added to the result of the recurrent operations. R2U-Net also changes the cropping in U-Net, concatenating instead the complete output of each down-sampling layer with its corresponding up-sampling layer. In order to achieve this concatenation, same sizes are ensured with padding in the transpose-convolutions of each layer.

Our implementation of R2U-Net, which we call R2U-Net64, increases the number of channels from 3 to 64 on its first layer and consequently doubles it until it reaches 1024 channels after the last max-pool or at the "lowest" point of the U. Then the number of channels is reduced in the same manner on each convolutional layer after each transpose-convolution and is finally reduced from 64 channels to 21 (the number of classes to predict). Each of these 21 channels is of the same size of the sample that was fed to the model and in order to predict a label for each pixel, the largest value along the 21 channels is used.

2.3 Height-driven Attention Networks (HANet)

In order to get a better model based on R2U-Net, we chose to incorporate the height-driven attention networks (HANet) to the original architecture. Let $\mathbf{X}_l \in R^{C_l \times H_l \times W_l}$ be a lower-level feature map and $\mathbf{X}_h \in R^{C_h \times H_h \times W_h}$ a high-level feature map that can be the input and output of one of the decoder blocks in R2U-Net respectively. Given the lower-level feature map $\mathbf{X}_l$ as input, HANet will generate a channel-wise attention map $\mathbf{A} \in R^{C_h \times H_h}$ containing per-channel scaling factors which depend on the height. Combining the high-level feature map $\mathbf{X}_h$ with the channel-wise attention map $\mathbf{A}$ a transformed new feature map

$$\tilde{\mathbf{X}}_h = F_{HANet}(X_l) \odot \mathbf{X_h} = \mathbf{A} \odot \mathbf{X_h} \tag{1}$$

can be obtained.

In order to generate $\mathbf{A}$, several steps are applied to the input: width-wise pooling, interpolation for coarse attention, and the addition of positional encoding to the intermediate feature maps that ultimately generate the height-driven attention map, following the strategy proposed in Transformer [Vaswani u. a. (2017)]. For a better understanding of how HANet works, refer to [Choi u. a. (2020)], being section 3.1 and figure 2 particularly clarifying.

The HANet module can be easily added once or several times at any layer of the segmentation network that receives a lower-feature map as input and returns a higher-feature map as output. A HANet layer has different hyperparameters that can significantly modify its functioning:

1. Positional encoding: it can be injected or not.

2. Reduction ratio r: it determines the number of convolutional layers that are included and thus the number intermediate feature maps that are generated.

3. Pooling method: there are two possible typical methods, average pooling and max pooling, to squeeze the spatial dimensions.

3 Experimental setup and Results

3.1 Database summary

3.1.1 VOC2012

The PASCAL Visual Object Classes Challenge 2012 (VOC2012) dataset [Everingham u. a.] contains images of twenty different visual object classes in realistic scenes. The twenty object classes are: person, bird, cat, cow, dog, horse, sheep, aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, tv/monitor. The VOC2012 challenge included three main object recognition competitions: classification, detection, and segmentation. Only a subset of the images were annotated with pixel-wise segmentation of each object present: 2913 images in total, consisting of 1464 images for training and 1449 images for validation. Both the images and the label masks are in RGB format. The size of each of the original image is different, with a height which varies from 246 to 500 pixels, and a width that varies from 112 to 500 pixels.

To load and preprocess the VOC212 dataset we made used of the dataloader provided to us by Dikshant Gupta. To develop a square dataset, the images, as well as the label masks, are resized to 512x512 pixels using the *nearest-neighbour* interpolation [Wikipedia] method to keep the label mask's original values. The images are also normalized considering the mean and standard deviation of the images forming the training set.

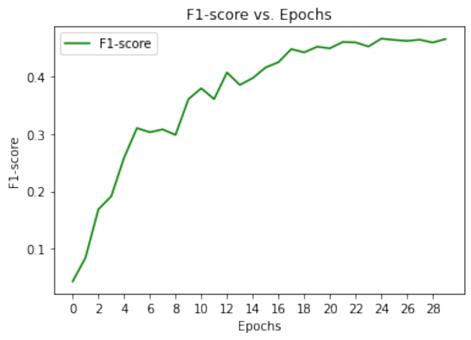
3.1.2 Cityscapes

The Cityscapes dataset [Cordts u. a. (2016)] focuses on pixel-level segmentation of urban street scenes. The images contain up to 30 classes, although only 19 of those classes were considered by us. The dataset contains a total of 5000 images, divided in 2975 images for training, 500 for validation and 1225 for test; however, annotations are only provided for the training and validation sets. The size of each original images and label masks have a size of 2048x1024 pixels and are in RGB format.

We created our own dataloader to load and preprocess the Cityscapes dataset, using some of the scripts from the *cityscapesScripts* repository [Cordts]. The images and the label mask s are resized to 512x256 pixels, using the *nearest-neighbour* interpolation method and normalized using the mean and the standard deviation of the images forming the training set. Random cropping is also applied to each image and its corresponding label mask.

3.2 Performance metrics

For the evaluation of the different models, several performance metrics are considered, including the Accuracy (AC), Sensitivity (SE), Specificity (SP), Dice's coefficient (DC) and Jaccard's similarity (JC). The computation of these metrics is based on the variables True Posisitive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) of a confusion matrix. The formula for the Accuracy (AC) is described in Eq. (2). The Sensitivity (SE), also named *recall*, *hit rate* or *true positive rate (TPR)*, can be calculated as shown in Eq. (3). The formula for the Specitivity (SP), also named *selectivity* or *true negative rate (TNR)* in other contexts, is given in Eq. (4). The Dice's coefficient (DC), equivalent to the F1-score (F1) , is described in Eq. (5). Finally, Jaccard's similarity (JC), commonly known as the PASCAL VOC intersection-over-union (IoU) metric [Eigen und Fergus (2015)], is calculated as described in Eq. (6).



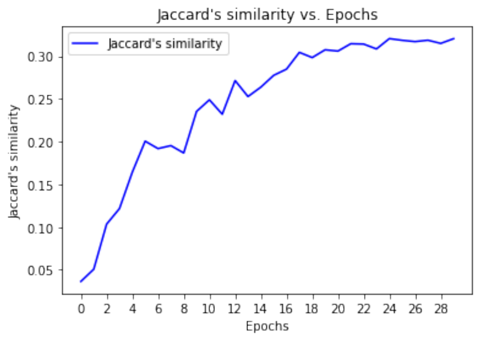


Figure 3: Evolution of selected metrics on the VOC2012 training dataset

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

$$SE = \frac{TP}{TP + FN} \quad (3)$$

$$SP = \frac{TN}{TN + FP} \quad (4)$$

$$DC/F1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \quad (5)$$

$$JC/IoU = \frac{TP}{TP + FP + FN} \quad (6)$$

These metrics can be computed when the ground truths and the predictions are binary, with each pixel taking only two possible classes. In order to compute these metrics for the pixel-level segmentation task, where there are ca. 20 different classes, we computed this metrics for each class, in a *one vs. rest* approach, and obtained an unweighted mean (*macro*) or a weighted mean (*weighted*) of all the classes.

3.3 Results

3.3.1 VOC2012

Training protocol. We train the SimpleSegNet model during 30 epochs, using the SGD optimizer with a fixed learning rate of 1e-2 and momentum of 0.9. We account for the class-imbalance problem by re-weighting each class in the cross-entropy loss computing the weight of each class using the *median-frequency balancing* proposed in [Eigen und Fergus (2015)].

The evolution of the Dice's coefficient and the Jaccard's similarity over the VOC2012 training set along the epochs of training is shown in *Figure 3*. In this figure, it can be observed how the model's performance increases along the epochs, with some important variations in the first half of the training.

In *Figure 4*, two example images, along with their ground-truth segmentation and the segmentation predicted by the SimpleSegNet network, are shown. These predictions were computed using the weights obtained after the twenty fourth epoch of training, in which the highest Jaccard's similarity, with a value of *0.3205*, was obtained. These performance results are far from those obtained by the state-of-the-art models such as EfficientNet-L2+NAS-FPN [Zoph u. a. (2020)], which achieves a Jaccard's similarity (or mIoU) value of 0.9 on the VOC2012 validation set. However, looking at the example predictions, it can be seen how our model has indeed learned the segmentation task and its predictions approximate to the ground-truth segmentation.

3.4 Cityscapes

Baseline: R2U-Net64 The authors of HANet adopted DeepLabv3+ [Chen u. a. (2017)] as baseline and added HANet to the segmentation networks at five different layers. Trying to imitate this use of HANet, we decided to use our R2U-Net64 model as baseline.

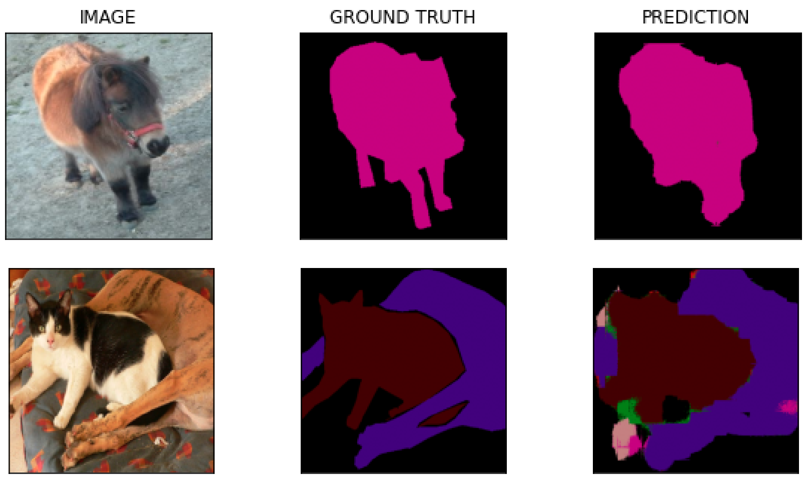


Figure 4: Example results of our SimpleSegNet model

HANet layers We extended the R2U-Net64 model by incorporating between one and three HANet layers. Considering that R2U-Net64 has five encoder blocks and four decoder blocks, the three different layers are added as follows:

- HANet layer L1: it is placed in parallel with the the decoder's second most inner block. It receives the same input as this block and its output is combined with this block's output.
- HANet layer L2: it it is placed in parallel with the the decoder's third most inner block.
- HANet layer L3: it is placed in parallel with the the decoder's last block.

Taking into account the studies made in [Choi u. a. (2020)] comparing the impact of different hyperparameter configurations, the HANet layers that we included do inject *positional encoding*, have a reduction ratio $r = 32$ and use *average pooling* as their pooling method. Although we implemented HANet ourselves, we made use of a PyTorch implementation of 1D, 2D and 3D positional encoding modules [Peter (2021)]

L1	**L2**	**L3**	**Total parameters**
✓			78,490,051
✓	✓		78,494,427
✓	✓	✓	78,495,559
R2U-Net64			78,472,851

Table 1: Comparison of parameters count between the baseline R2U-Net and R2U-Net + HANet

Training protocol For the Cityscapes dataset we perform data augmentation with Pytorch's transforms library, with which we randomly zoomed cropped each image of the training set. We did this to gain more training data and to try to obtain a better feature extraction of objects such as cars, since we noticed that sometimes these were correctly classified but did not have a very accurate shape in the predicted segmentation.

We train the R2U-Net64 model during 15 epochs, using the Adam optimizer with a fixed learning rate of 2e-4. We use cross entropy as our loss function ignoring the 255 label which belongs to the classes that are too rare, as described in [Cordts u. a. (2016)]. In our various attempts at obtaining a good model, we noticed that our validation loss started to stale in decrease after the first few epochs while the training loss continued decreasing. Usually we would attribute this behaviour to the model overfitting over the training data; however, after visualizing the model's predictions over the validation set, and seeing that these did not worsen over each epoch where the validation loss staled, we continued training for our fixed number of epochs instead of relying on early stopping.

For the three versions of R2U-Net64 + HANet model we train with the same hyperparameters as the R2U-net64 model f.

A comparison of the average selected performance metrics between the baseline R2U-Net64 model and the extended models that include one, two or three HANet layers over the Cityscapes validation set is shown in *Table 2*. For each model and each of the metrics two values corresponding to the *macro* average (left) and the *weighted* average (right) are shown. It can be observed that the model that extends R2U-Net64 with two HANet layers (L1 + L2) obtains the highest value for all but one metric.

It is worth mentioning the huge variation that exists for the Sensitivity (SE), Dice's coefficient (DC) and Jaccard's similarity (JC) between the *macro* and the *weighted* average; this is due to really low values of these metrics for some classes such as *traffic sign*, *bus*, *train* or *motorcycle*, which correspond with the classes that have the lowest representation. The *Table 3* shows the value of Jaccard's similarity per class for both the baseline R2U-Net and the extended model with two HANet layers (L1 + L2). Apart from the huge variations that exist from one class to another, it is worth mentioning that incorporating HANet has considerably improved Jaccard's similarity for those classes that are easily determined by the position in the image, such as *road*, *vegetation* or *sky*; on the other hand, the incorporation of HANet has worsened the values of other classes.

Two example images from the Cityscapes dataset, along with their ground-truth segmentation and the segmentation predicted by R2U-Net64 and R2U-Net64 + 2 HANet respectively, are shown in *Figure 5*.

L1	L2	L3	AC	SE	SP	DC	JC
✓			93.36 / 83.25	30.32/36.89	96.54 / **97.43**	25.38 / 47.79	17.10 / 32.72
✓	✓		**96.63 / 91.34**	**31.54 / 67.99**	**97.97** / 93.41	**26.50 / 69.26**	**20.02 / 57.43**
✓	✓	✓	92.25 / 78.68	13.97 / 26.33	95.92 / 96.09	12.8 / 31.74	8.23 / 20.09
R2U-Net64			95.13 / 88.12	29.14 / 53.76	97.39 / 96.61	25.35 / 61.11	17.17 / 46.23

Table 2: Comparison of performance metrics (*macro* / *weighted*) between the baseline R2U-Net and R2U-Net + HANet over the Cityscapes validation set

Class	**road**	**sidewalk**	**building**	**wall**	**fence**	**pole**	**traffic light**	**traffic sign**	**vegetation**
L1 + L2	0.6323	**0.151**	0.4167	**0.0258**	0.0742	**0.1305**	**0.0294**	**0.1372**	0.4713
R2U-Net	**0.7445**	0.022	**0.6433**	0.0184	**0.0985**	0.0903	0.0178	0.0415	**0.5593**

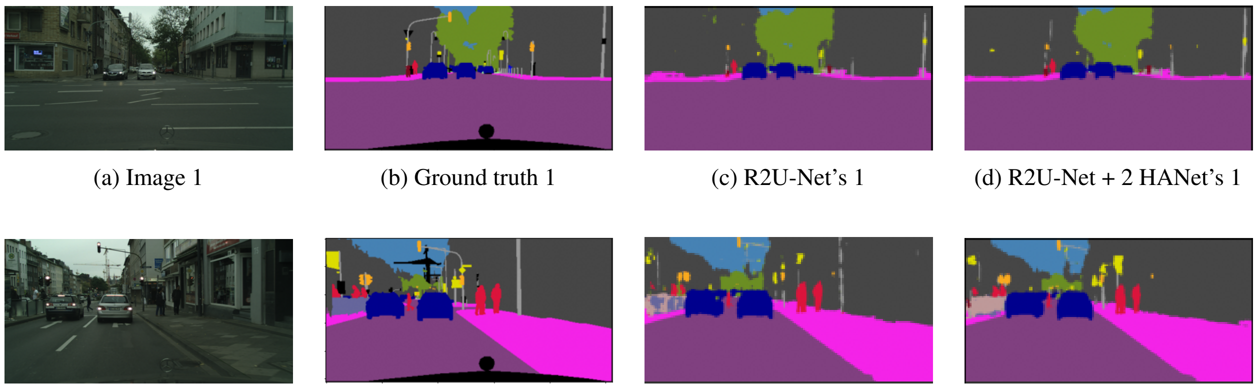
Class	**terrain**	**sky**	**person**	**rider**	**car**	**truck**	**bus**	**train**	**motorcycle**	**bicycle**
L1 + L2	**0.1657**	0.4653	**0.2037**	0.0081	0.3009	0.0	**0.0216**	0.0032	0.0	0.0247
R2U-Net	0.1042	**0.8369**	0.1268	**0.0127**	**0.3195**	0.0	0.0	**0.0319**	**0.018**	**0.1175**

Table 3: Comparison of Jaccard's simmilarity per class: R2U-Net vs. R2U-Net + (L1 + L2) HANet

4 Conclusions

In this paper, we proposed the SimpleSegNet model for the VOC2012 semantic segmentation task, we implemented the R2U-Net network and adapted it for multi-class semantic segmentation, and we latter model by incorporating different HANet layers into its architecture, proved by the comparison of performance metrics of all models over the Cityscapes dataset. By doing all this, during the course of this project we have deepened our knowledge in deep convolutional and recurrent neural networks, as well as image segmentation, being able to summarize our new acquired knowledge in the following points:

- Medical image segmentation is one of the most important areas of application of deep learning techniques and significant efforts are been made to solve the challenges of this area in particular, such as the limited annotated data or the class imbalance, and the challenges


(a) Image 1 (b) Ground truth 1 (c) R2U-Net's 1 (d) R2U-Net + 2 HANet's 1

(e) Image 2 (f) Ground truth 2 (g) R2U-Net's 2 (h) R2U-Net + 2 HANet's 2

Figure 5: Example results of R2U-Net and R2U-Net + 2 HANet

of training deep learning models in general, such as the overfitting, the training time or the gradient vanishing.

- Semantic urban scene segmentation is a complex task and even a minimal improvement of the performance requires a huge research effort and a significant improvement in the available hardware.
- Whereas the implementation of complex neural networks is enormously simplified and accelerated by the use of complex libraries such as PyTorch or Tensorflow, obtaining the best performance possible is still a difficult task which requires great knowledge of the hyperparameter tuning and considerable computing resources and time.
- Applying changes to a network architecture based on the specific characteristics of a dataset, specifically with regards to its class distribution over different pixel positions can be greatly useful for improving results, as shown with the results we obtained from adding the HANet modules into our R2U-Net model.

References

[Alom u. a. 2018] ALOM, Md Z. ; HASAN, Mahmudul ; YAKOPCIC, Chris ; TAHA, Tarek M. ; ASARI, Vijayan K.: *Recurrent Residual Convolutional Neural Network based on U-Net (R2U-Net) for Medical Image Segmentation.* 2018

[Badrinarayanan u. a. 2016] BADRINARAYANAN, Vijay ; KENDALL, Alex ; CIPOLLA, Roberto: *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation.* 2016

[Chen u. a. 2017] CHEN, Liang-Chieh ; PAPANDREOU, George ; KOKKINOS, Iasonas ; MURPHY, Kevin ; YUILLE, Alan L.: *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs.* 2017

[Choi u. a. 2020] CHOI, Sungha ; KIM, Joanne T. ; CHOO, Jaegul: *Cars Can't Fly up in the Sky: Improving Urban-Scene Segmentation via Height-driven Attention Networks.* 2020

[Cordts] CORDTS, Marius: *cityscapesScripts - scripts for inspection, preparation, and evaluation of the Cityscapes dataset.* `https://github.com/mcordts/cityscapesScripts`. – (Accessed on 03/26/2021)

[Cordts u. a.] CORDTS, Marius ; OMRAN, Mohamed ; RAMOS, Sebastian ; REHFELD, Timo ; ENZWEILER, Markus ; BENENSON, Rodrigo ; FRANKE, Uwe ; ROTH, Stefan ; SCHIELE, Bernt: *Benchmark Suite – Cityscapes Dataset*

[Cordts u. a. 2016] CORDTS, Marius ; OMRAN, Mohamed ; RAMOS, Sebastian ; REHFELD, Timo ; ENZWEILER, Markus ; BENENSON, Rodrigo ; FRANKE, Uwe ; ROTH, Stefan ; SCHIELE, Bernt: *The Cityscapes Dataset for Semantic Urban Scene Understanding.* 2016

[Deng u. a. 2009] DENG, J. ; DONG, W. ; SOCHER, R. ; LI, L. ; KAI LI ; LI FEI-FEI: ImageNet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, S. 248–255

[Eigen und Fergus 2015] EIGEN, David ; FERGUS, Rob: Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture. In: *arXiv:1411.4734 [cs]* (2015), Dezember. – URL `http://arxiv.org/abs/1411.4734`. – Zugriffsdatum: 2021-03-04. – arXiv: 1411.4734

[Everingham u. a.] EVERINGHAM, M. ; VAN GOOL, L. ; WILLIAMS, C. K. I. ; WINN, J. ; ZISSERMAN, A.: *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results.* http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html

[He u. a. 2015] HE, Kaiming ; ZHANG, Xiangyu ; REN, Shaoqing ; SUN, Jian: *Deep Residual Learning for Image Recognition.* 2015. – URL `http://arxiv.org/abs/1512.03385`

[Liang und Hu 2015] LIANG, Ming ; HU, Xiaolin: *Recurrent convolutional neural network for object recognition.* 2015

[Peter 2021] PETER: *tatp22/multidim-positional-encoding.* März 2021. – URL `https://github.com/tatp22/multidim-positional-encoding`. – Zugriffsdatum: 2021-03-21. – original-date: 2020-07-26T11:06:17Z

[PyTorch] PYTORCH: *torchvision.models — Torchvision master documentation.* `https://pytorch.org/vision/stable/models.html#torchvision.models.vgg16`. – (Accessed on 03/22/2021)

[Ronneberger u. a. 2015] RONNEBERGER, Olaf ; FISCHER, Philipp ; BROX, Thomas: *U-Net: Convolutional Networks for Biomedical Image Segmentation.* 2015

[Simonyan und Zisserman 2015] SIMONYAN, Karen ; ZISSERMAN, Andrew: *Very Deep Convolutional Networks for Large-Scale Image Recognition.* 2015

[Tao u. a. 2020] TAO, Andrew ; SAPRA, Karan ; CATANZARO, Bryan: *Hierarchical Multi-Scale Attention for Semantic Segmentation.* 2020

[Vaswani u. a. 2017] VASWANI, Ashish ; SHAZEER, Noam ; PARMAR, Niki ; USZKOREIT, Jakob ; JONES, Llion ; GOMEZ, Aidan N. ; KAISER, Lukasz ; POLOSUKHIN, Illia: *Attention Is All You Need*. 2017

[Wikipedia] WIKIPEDIA: *Nearest-neighbor interpolation*. `https://en.wikipedia.org/wiki/Nearest-neighbor_interpolation`. – (Accessed on 03/26/2021)

[Yu u. a. 2020] YU, Changqian ; WANG, Jingbo ; GAO, Changxin ; YU, Gang ; SHEN, Chunhua ; SANG, Nong: *Context Prior for Scene Segmentation*. 2020

[Zhang u. a. 2020] ZHANG, Hang ; WU, Chongruo ; ZHANG, Zhongyue ; ZHU, Yi ; LIN, Haibin ; ZHANG, Zhi ; SUN, Yue ; HE, Tong ; MUELLER, Jonas ; MANMATHA, R. ; LI, Mu ; SMOLA, Alexander: *ResNeSt: Split-Attention Networks*. 2020

[Zhou u. a. 2018] ZHOU, Zongwei ; SIDDIQUEE, Md Mahfuzur R. ; TAJBAKHSH, Nima ; LIANG, Jianming: *UNet++: A Nested U-Net Architecture for Medical Image Segmentation*. 2018

[Zoph u. a. 2020] ZOPH, Barret ; GHIASI, Golnaz ; LIN, Tsung-Yi ; CUI, Yin ; LIU, Hanxiao ; CUBUK, Ekin D. ; LE, Quoc V.: *Rethinking Pre-training and Self-training*. 2020

A Repository

In the following appendix we provide a description of what are the contents of our GitHub repository.

```
Vision-Project-Image-Segmentation
├─ datasets
│  ├─ pascalvoc.py
│  └─ cityscapes.py
├─ docs
├─ networks
│  ├─ r2unet
│  ├─ hanet.py
│  ├─ r2unet_hanet.py
│  ├─ segnet.py
│  └─ utils.py
├─ .gitignore.py
├─ README.md
├─ Vision_task_1.ipynb
├─ Vision_task_2.ipynb
├─ Vision_task_3.ipynb
├─ cityscapes_dataset.ipynb
├─ evaluation.py
├─ evaluation_metrics.ipynb
├─ training.py
└─ utils.py
```

datasets: contains the dataloaders for the PASCAL Visual Object Classes Challenge 2012 (VOC2012) dataset and the Cityscapes dataset.

docs: general documentation.

evaluation_metrics.ipynb: notebook in which shows how to compute performance metrics with the *EvaluationReport* class.

networks: contains our implementations of R2U-Net16, R2U-Net64, HANet, R2U-Net16 + HANet, R2U-Net64 + HANet and SegNet.

evaluation: contains the *EvaluationReport* class, which allows to compute several metrics (accuracy, precision, recall, sensitivity, F1-score, Jaccard's similarity, ...) for an individual class or as an average of all classes. The metrics can be computed providing the ground truths and the predictions, or a dataset and the model that should be used to generate the predictions.

cityscapes_dataset.ipynb: notebook in which the functionality of the *cityscapesDataset* class is explored.

training: contains methods for training, including training with *early stopping* based on the validation set's loss.

utils: contains different utilities including the method to visualize the segmentation results in a (image, ground truth, prediction) grid or the evolution of some a metric vs. the epochs.