Semantic Segmentation - Group Project Report

Yigit Gürtunca ✉, Batu Kaan Özen ✉, Sadok Kria ✉, Di Wu ✉, and Emma Benedetti ✉

Department of Electrical and Computer Engineering, Technical University of Munich

✉ yigit.guertunca@tum.de
✉ ozenbatukaan@gmail.com
✉ sadok.kria@tum.de
✉ ge35sah@mytum.de
✉ emma.benedetti@tum.de

March 6, 2022

Abstract — This work will summarise the main findings of our exploration of semantic segmentation, carried out during the Winter Semester 2021-2022 during the *Introduction to Deep Learning* course held by Dr. Ahn. We will focus on the application of two specific semantic segmentation architectures, namely Fully Convolutional Networks and Atrous Convolution, to determine whether the depth of a convolutional network is positively correlated to the classification accuracy.
Here we show that, for semantic segmentation, this is not always the case. Clearly, the classification accuracy depends as much on the chosen architecture as other characteristics of the analysis, such as the database or the chosen evaluation metrics. To test our theory, we modeled both Fully Convolutional Networks and Atrous Convolution on the Cityscapes dataset, used to train navigation systems in self-driving cars. We employed two different metrics, namely Intersection over Union and Mean Pixel Accuracy, to evaluate the results of our model.
According to our findings, both metrics lead us to favour Atrous Convolution and an intermediate architecture (i.e. ResNet50) to achieve an optimal analysis.

1 Introduction

In Deep Learning, semantic segmentation is the process of classifying each pixel of an image belonging to a particular label, without differing across different instances of the same object. To do that, this process employs an encoder-decoder method: the encoder, usually pre-trained for the task, is composed by parallel layers that extract features from an image by using filters. The knowledge extracted by the encoder is then transferred to the decoder, which uses it to generate a segmentation mask and the final output. The decoder will also ensure that the generated mask sufficiently matches the pixel resolution of the input image.
Semantic segmentation has revealed itself to be particularly useful in multiple fields, like medical imaging or navigation in self-driving cars. In this work, we will explore semantic segmentation applied in particular to the latter field, by using the Cityscapes dataset [2].
In this work, we will first delineate the goal and expectations of the group project. Afterwards, we will proceed with a more accurate description of the dataset and the architectures employed in our work, together with the training process. In Chapter 4 we will introduce the metrics used for evaluation, i.e. the Intersection Over Union index and the Mean Pixel accuracy, as well as the results of our work. Finally, we will discuss our results and draw the conclusive statements.

2 Goals and Expectations

The goal of this group project was to use different network architectures and backbones for image segmentation purposes mainly for autonomous driving. Therefore, Cityscapes [2] dataset was used since it is a large dataset that can be used for autonomous driving purposes with its recorded street scene sequences. This dataset was used to train and to test the models. Further information regarding the dataset can be found in Chapter 3. To check the accuracies of the architectures, two different evaluation methods were used. The first method was IoU (Intersection over Union) and the second method was MPA (Mean Pixel Accuracy).

There were two expectations while implementing the project. First expectation was to get a better performance using deeper backbones. Second expectation was to acquire higher accuracies using DeepLab(v3) in comparison to the FCN (Fully Connected Layer).

The devices that were used were not powerful and efficient enough to train the models for a longer period of time. Hence, it was also expected that the results gathered from the project would not show the absolute reality, since the models would eventually need a longer training interval to reflect the truth.

3 Experiment

3.1 Dataset

In the experiment the application of semantic segmentation in the autonomous driving field was investigated. The CityScapes dataset [2] was chosen to be investigated in the project since it mainly focuses on images gathered from urban street scenes for semantic segmentation purposes. The images were recorded from 50 cities in Germany and Switzerland. There are totally 30 labeled classes. 5000 images with fine pixelwise annotations are available. An annotation example is shown in Figure 1. The image zip file we downloaded from the website is `"leftImg8bit_trainvaltest.zip (11GB) [md5]"` and its corresponding annotation zip file is `"gtFine_trainvaltest.zip (241MB) [md5]"`. Because of the limitation of computational resources, we downsampled the images from the orginal resolution 1024 * 2048 to 256 * 512 as inputs in our experiment, where 1024 images are used as the training dataset, 196 images as the validation dataset and 144 images as the test dataset.

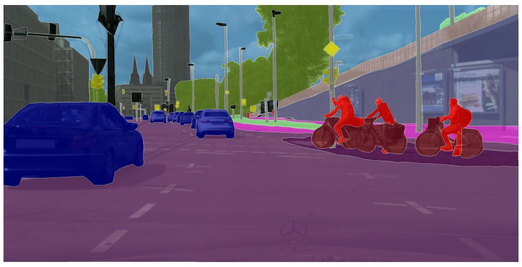

Figure 1 Cityscapes Dataset: Example Cologne

3.2 Architectures

In our work, we used two widely known architectures for semantic segmentation, namely Fully Convolutional Networks (FCN) and Atrous Convolution.

3.2.1 Fully Convolutional Networks

Convolutional networks are widely used for image recognition and tasks with a structured output. However, they have been proven to be ineffective towards semantic segmentation purposes, even in fine recognition tasks where each pixel was labeled with the target class. This is why Long, Shelhamer, and Darrel [6] developed Fully Convolutional Networks to specifically address this issue without any further elaboration on the state-of-the-art models. Both learning and inference in FCN are performed on the whole image at a time, with the application of dense feed-forward computation and backpropagation.
Fully convolutional networks are built on translation invariance, meaning that all their basic components operate only on local input regions, and depend only on relative spatial coordinates. End-to-end upsampling layers within the network allows refined prediction and learning at pixel level, followed by subsampled pooling. This method produces an output of the same size of the input, and learns more efficiently than "standard" convolutions both at an asymptotic and at an absolute level. Therefore, it avoids the complications that were claimed to be necessary in previous works.

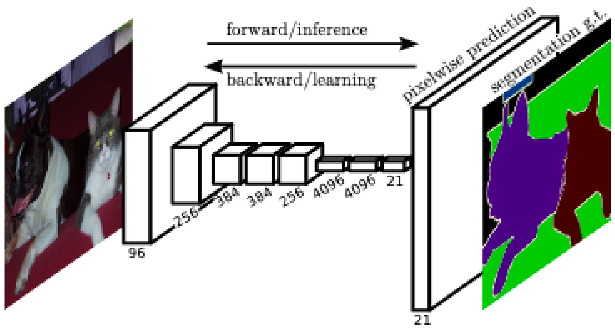

Figure 2 Architecture of Fully Convolutional Networks, which can be efficiently used for learning tasks in semantic segmentation frameworks. Source: [6]

However, the continuous convolution of their layers can be considered also a strong disadvantage of FCN. In fact, this often translates to a lower output resolution, which leads often to complex optimization problems. Therefore, in this work we also considered architectures with an additional type of convolution, namely Atrous Convolution.

3.2.2 Atrous Convolution

Atrous convolution is a dilated convolution developed by Google [1] and used in the DeepLab architecture family. This architecture is particularly useful because it tackles two specific issues commonly faced by typical convolutional networks, as well as FCN.
The first issue is the reduced feature resolution that follows a picture downsampling and subsequent up-

sampling, which was introduced in the previous section. The repeated combination of max-pooling and striding at consecutive layers usually reduces the spatial resolution of the output feature maps by a factor of 32 across each direction. Meanwhile, Atrous Convolution allows one to control the resolution at which the feature responses are computed, without learning new parameters.
Considering two-dimensional signals, for each location l on the output y and a filter w, this convolution is applied over the input feature map x as follows:

$$y[l] = \sum_k x[l + r * k]w[k] \tag{1}$$

where the atrous rate r corresponds to the stride with which we sample the input signal. This is equivalent to convolving the input x with upsampled filters, produced by inserting $r - 1$ zeros between two consecutive filter values along each spatial dimension.

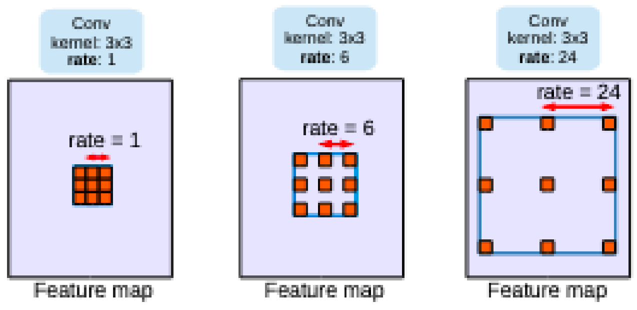

Figure 3 Atrous convolution with kernel size w 3 × 3 and different atrous rates r. Standard convolution corresponds to Atrous Convolution with $r = 1$. Large atrous rates widen the model's field-of-view, enabling object encoding at multiple scales. Source: [1]

With Atrous Convolution, one can keep the stride constant, but with a larger field-of-view, without increasing the number of parameters or the amount of computation. As a consequence, a larger output feature map can be used for classification purposes.
The second difficulty faced by common convnets is the existence of multiple-scale objects, which is quite common in many image datasets. Applying a convolutional network with shared weights usually lets smaller-scale inputs to encode the long-range contexts, while large scale inputs keep the details of the object. Although this strategy is effective for many architectures, it does not scale well with deeper CNN -as is our case- due to limited memory [1]. On the other hand, the architectures in the DeepLab family employ Atrous Spatial Pyramid Pooling (ASPP), namely parallel Atrous Convolution layers with different rates r which are able to capture multi-scale information.

3.3 Training process

Since we want to investigate the capabilities and efficiencies of different structures, all the networks are trained from scratch without transfer learning to eliminate the influence that pre-trained weights could bring. The source codes of FCN [5], ResNet backbones [3], and DeepLabV3 [4] are all from torchvision models by Pytorch Team. It needs to be mentioned that FCN with backbones ResNet18 and ResNet34 are not directly provided in source codes, so we modified the codes and built those by ourselves. In addition, the classifier layers of models in source codes are also modified to adapt for the classes defined in the Cityscapes dataset. The training process is implemented on the Colab platform. The batch size is set to be 2. The optimizer is Adam with the learning rate of 1e-4 and the weight decay of 1e-5. The loss function is determined by calculating the Mean Pixel Accuracy which is introduced in the following part. All the networks follow the same parameter configurations and are trained for 50 epochs.

4 Evaluation and Result

4.1 Intersection over Union

IoU, commonly called Jaccard index, allows to compare the "predicted" detected region with the ground truth region. It is used as a threshold to determine whether a predicted result is a true or a false positive. IoU can be represented as the overlap between the surrounding rectangle around a predicted object and the surrounding rectangle around the reference data.

The numerator describes the overlap area of the predicted mask and the ground truth's mask, whereas the denominator is the union of the predicted and the ground truth mask. The intersection over union metric measures the number of pixels in common between the target and the prediction mask, divided by the total number of pixels present across both masks.

4.2 Mean Pixel Accuracy

Mean Pixel Accuracy is a metric of evaluation for semantic segmentation that aims to report the mean

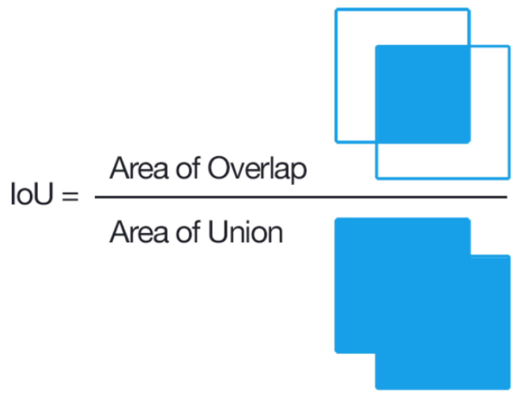

Figure 4 Definition of Intersection over Union [7]

percent value of pixels in an image that were predicted correctly.

$$\varnothing \text{ Pixel Accuracy } = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Each pixel will be compared one by one with the ground truth mask. The true positive represents a pixel that is correctly predicted to belong to the given class, whereas a true negative represents a pixel that is correctly identified as not belonging to this class. However, this metric can sometimes provide misleading results when the class representation is small within the image.

4.3 Results

In this part, we present the results of the comparison between different backbones, as well as the difference between the Fully Convolutional Network and the DeepLab model.

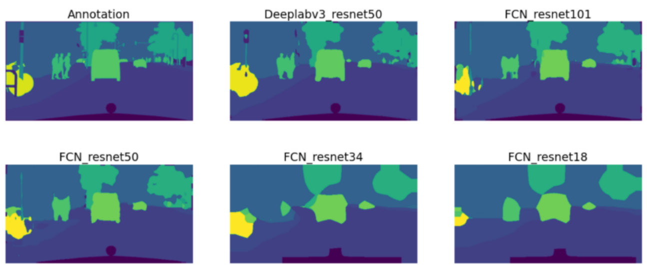

Figure 5 Comparison between the annotation mask and the predicted mask for each model

Figure 5 shows some segmentation results from the different models that were trained. The segmentation masks clearly show that DeepLab achieves the best results if the prediction and the annotation masks match between each other.Besides, the results demonstrate that the segmentation masks of the Fully Convolutional Network both with Resnet 50 and Resnet 101 as backbone are very similar.

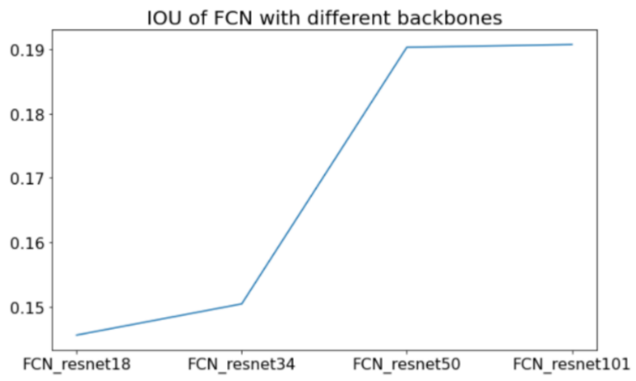

Figure 6 IoU score for Fully Convolutional Networks with Resnet 18, 34, 50 and 101 as backbone

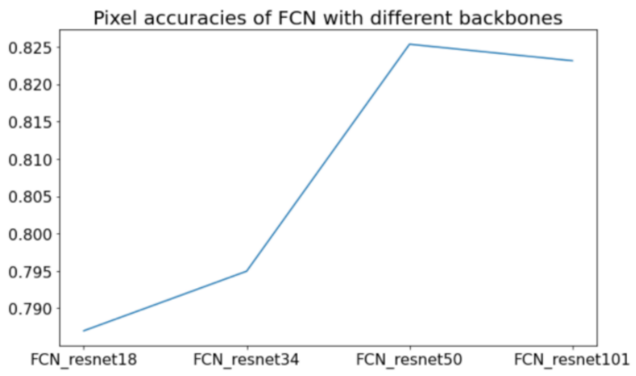

Figure 7 Mean pixel accuracy for Fully Convolutional Networks with Resnet 18, 34, 50 and 101 as backbone

Figures 6 and 7 represent respectively the intersection over union score and the mean pixel accuracy of the training process of the Fully Convolutional Network. According to the IoU score, we can clearly notice that the deeper the model is, the better it performs for the Fully Convolutional Network with resnet 18, 34 and 50 as backbone. However, the IoU scores for the FCN with Resnet 50 and 101 as backbone are very similar. Figure 7 represents the mean pixel accuracy for each model. Here too we can notice that a deeper architecture achieves better results when it comes to Resnet 18, 34 and 50. Nevertheless, the Fully Convolutional Network obtained a higher mean pixel accuracy with Resnet 50 as backbone than with a Resnet 101. These results clearly demonstrate that it is not always true that deeper backbones perform better.

The second goal of this project is, as stated above, the comparison between DeepLab and the Fully Convolutional Network. Figures 8 and 9 show respectively the IoU score and the mean pixel accuracy of FCN and DeepLab.

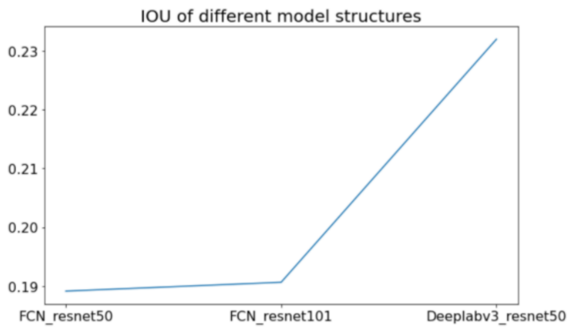

Figure 8 IoU score for FCN with both Resnet 50 and 101 and DeepLab with Resnet 50 as backbone

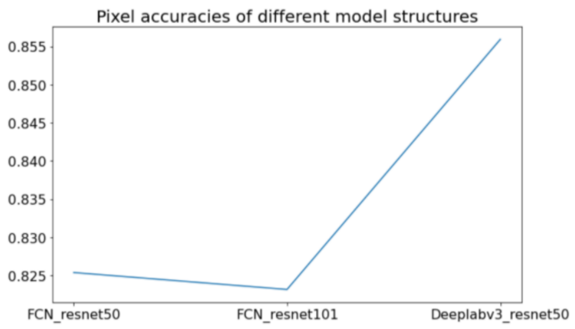

Figure 9 Mean Pixel Accuracy for FCN with both Resnet 50 and 101 and DeepLab with Resnet 50 as backbone

For the comparison we used Resnet 50 as backbone for both models. the figures above show that DeepLab is more efficient than the Fully Convolutional Network and it achieves a better intersection over union score as well as a better mean pixel accuracy.

Furthermore, Resnet 101 has also been used as backbone to FCN, in order to investigate whether it will perform better than Resnet 50. However, the results were similar and there is no major difference to notice between FCN with both Resnet 50 and Resnet 101. This can be explained by the continuous convolution and pooling layers present in the Fully Convolutional Network, which results in complex optimization problems. On the other side, Atrous Convolution enables us to effectively increase the field-of-view without adjusting for the number of parameters or for the amount on computation, which leads to an accuracy increase.

5 Conclusion

In this article, we explored Semantic Segmentation using deep neural networks under some constraints, and compared different backbones using specific comparison matrices. The biggest drawbacks of our project were not having powerful hardware, as well as a lack of datasets in the autonomous driving field which can represent a sufficient distribution of real-life scenarios. We compared between architectures with Fully Convolutional Network (FCN) and with Atrous Convolution (DeepLab), and also investigated whether using deep backbones can affect the test accuracy positively or negatively.

The result of the experiment is different than our theoretical expectation. While it was expected that using a deep and more successful backbone would have been more accurate, during our project it is examined that training a deeper neural network is a more complicated process and leads to worsened results.
During the experiment, it is also shown that architectures with Atrous Convolution performed better than Fully Convolutional Networks. One of the reasons is that Atrous Convolution allows us to effectively enlarge the field of view of filters without increasing the number of parameters or the amount of computation, which is beneficial and causes an accuracy increase. There are several problems causing an unsuccessful model, namely the small size of the dataset and the insufficient processing power. These draw-backs affected the performance of deeper neural networks negatively.

To solve the above mentioned problems, we can hypothesize to design some dataset collection algorithms. Another solution would be to train one semantic segmentation network by using all possible datasets regarding the semantic segmentation area. After that, it can be fine-tuned using the autonomous driving dataset.

Training a very large dataset is a high-performance computing problem. The development of strong graphic processing units enables us to increment in computing power using multi thread operations. Our training process is completed using GPUs via the PyTorch framework, written in CUDA. Using high-level framework can cause a lot of unnecessary calculation during the training process and slow the learning process. This problem can be minimized by completing our project using lower level Frame works and languages such as cuDNN (written in CUDA) or CUDA. Another alternative is farming multiple GPUs for increasing processing power.

In the experiment, it is realized that using deep backbones is not always the best solution. We discovered that using appropriate rather than deeper backbones gives better results. Instead of using deeper neural networks, trying to solve the problem with a better neural network design such as Atrous Convolution, Transformers, or Recurrent Neural Network, is one of the best ways to increase the accuracy of the trained models. Lastly, it is always hard to build a perfect dataset and to have sufficient hardware for real life projects, hence the training efficiency should be increased by using some alternative ways.

References

[1] Liang-Chieh Chen et al. "Rethinking Atrous Convolution for Semantic Image Segmentation". In: *CoRR* abs/1706.05587 (2017). arXiv: 1706.05587. URL: http://arxiv.org/abs/1706.05587.

[2] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[3] *Deep residual networks by Pytorch Team*. https://pytorch.org/hub/pytorch_vision_resnet/. Accessed: 2022-01-27.

[4] *DeepLabV3 by Pytorch Team*. https://pytorch.org/hub/pytorch_vision_deeplabv3_resnet101/. Accessed: 2022-01-27.

[5] *Fully-Convolutional Network by Pytorch Team*. https://pytorch.org/hub/pytorch_vision_fcn_resnet101/. Accessed: 2022-01-27.

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *CoRR* abs/1411.4038 (2014). arXiv: 1411.4038. URL: http://arxiv.org/abs/1411.4038.

[7] Ekin Tiu. *Metrics to Evaluate your Semantic Segmentation Model*. https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2. Accessed: 2022-01-25.