Neural Networks final project 2021

Tom Fischer
Saarland University
2563286
s8tmfisc@stud.uni-saarland.de

Mayank Malhotra
Saarland University
2577726
s8mamalh@stud.uni-saarland.de

Abstract

Deep learning techniques have been successful in image recognition as well as image generation, and are currently at a stage where they perform better than humans on multiple use cases. In this report, our focus is on semantic segmentation, a technique for classifying pixels of an image, with the goal of recognizing different types of objects in the given image. In this work we compare two deep learning architectures for semantic segmentation. The measure of comparison is a Recurrent Residual Convolutional Networks based on the U-net architecture (R2U net), which we compare to a novel architecture that combines the SegNet model with the Atrous Spatial Pyramide Pooling (ASPP) module of the DeeplabV3 model.

1 Introduction

Today deep learning networks can perform a variety of tasks from image classification, segmentation to generation of images based on a learned distribution. In this work, we investigated the field of semantic segmentation, which is widely used in medical sciences and the more recent field of autonomous driving.

When dealing with images, convolutional neural networks (CNN) have been the most successful choice for a wide array of tasks and are usually the best choice for anyone that wants good and reliable results, However, there is still a platitude of different convolution techniques that offer a wide variety of architectures that all fall into the same category of convolutional neural networks. Most commonly when trying to achieve semantic segmentation, models will be based on an encoder and a decoder unit. The encoder can be though of as a simple classifier that reduces the resolution of the original image at each step and extract more and more features from it, while the decoder's task is to make sense of the extracted features and transform them back into a classification map, that assigns each pixel to a certain class. Both architectures that we investigate in this work will follow this general architecture, but take vastly different approaches in realizing them. In this work we consider the architectures of the Recurrent Residual Convolutional Network as proposed by Alom et al. [2018] and implemented the R2UNet model, which is based on residual recurrent blocks. Since the R2UNet provides impressive results, we used this model as a baseline comparison for our experiment with the second model, that represents a novel architecture in which we combine a SegNet by Badrinarayanan et al. [2017] with the Atrous Spatial Pooling Pyramid (ASPP) which was first proposed by Chen et al. [2017].

We will structure this report by first going over both the models and show how they can be implemented and what techniques are used. Then we will showcase the types of experiments we did and will go over the results and discuss the performance of both the models and what we can learn from them. Finally, we draw the conclusion which model might work better or worse given the situation at hand.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

2 Methodology and Motivation

We have two tasks in our hand, the first task is to implement a recurrent residual neural network based on the U-Net architecture for semantic segmentation, and in our second task we implement a novel model that combines a SegNet together with an ASPP module.

Task 1: R2U-Net is a state of the art model, with three key benefits, First - a residual unit helps when training a deep architecture, second - feature accumulation with recurrent residual convolutional layers ensures better feature representation for segmentation tasks, third - it allows us to design better U-Net architecture with the same number of network parameters thus providing efficient performance.

We chose the U-Net has some other advantages to it as well, the model allows for the use of global location and context at the same time, it can work with few training samples and still provide good segmentation output, it can process the entire image in the single forward pass and generate segmentation masks, unlike patch-based approaches which do not preserve the full context of the image.

We describe the encoder and decoder operations now. The encoder part in the U-Net does the convolution operations and down-sampling using avg. or max. pooling. Using convolution we extract different set of features from our input image. As the number of layers in the encoder increases, the dimensions of the image keep on decreasing while the number of features extracted get higher. Ideally we would not want to reduce the resolution of the input image because the final output image in case of semantic segmentation needs to have the same dimensions as the input image. We do dimensionality reduction using pooling so as to reduce the number of parameters and ease the computations. When we do down-sampling we always lose information and up-sampling is simply not that accurate. Hence our prediction suffers. This is what happens in simple fully connected convolutional networks such as FCN's.

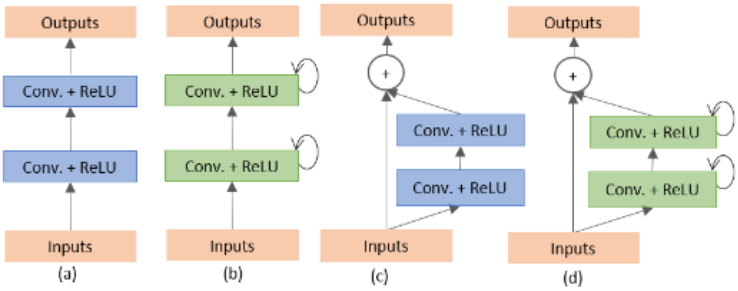
In order to improve this,the up-sampling operation in the decoder uses the final feature map from the encoders last layer and also information from the last pooling layer to make up for the lost information, this is implemented in FCN-16, if we use another pooling layer instead of using just one, then this implementation is called FCN-8. We mention these FCN architectures as we will use them to compare the U-Net. The U-net was initially designed for medical purposes and builds on top of the FCN network as stated above.

U-Net - uses the so called **"shortcut connections (copy and crop)"** to make up for the lost information. It proposes to send information to every up sampling layer in decoder from the corresponding down sampling layer in the encoder thus capturing finer information whilst also keeping the computation low. Since the layers at the beginning of the encoder would have more information they would bolster the up sampling operation of decoder by providing fine details corresponding to the input images thus improving the results a lot. Thus here we are able to answer, why the U-Net model performs better than simple FCN's. Now we look at further improvements in the U-Net design - the recurrent U-net and the recurrent residual U-net.

The RU and R2U nets use a method called as **feature concatenation or feature addition** between the encoder and the decoder layers at each level instead of the crop and copy method used in basic U-Net. Moreover the simple forward convolutional layers are replaced by recurrent convolutional layers in RU net and recurrent residual convolutional layers in the R2U net.

Recurrent NNs use a **feedback connection** at each neuron, a CNN is a simple feed forward network where a neuron has an input and an output state, but in a RNN , each neuron has an input, and output and a feedback to itself (which is the recurrent connection). This helps build a deeper model keeping the number of network parameters the same (as weights are shared in RNN's), this further helps in better understanding the context of an image and thus improves feature extraction. Though RNN's are more successful in NLP tasks, they also find their application in Imaging. Thus a better understanding of the context of the input image gives RNN's an edge over simple feed forward CNN's.

Residual NNs were introduced because the performance of feed forward NNs worsened after we increased the depth beyond a particular threshold. Usually a dept of 16-30 layers for simple feed forward CNN's is considered to be optimal, beyond which there training errors start to increase as well! Here RNN's come to the rescue.



Res-Nets or residual neural networks use the **"skip connection"** mechanism to build deeper layers. Basically the output of a convolution layer is combined with the input, and a combination of these two are passed to the next layer. Skip connections have been proven to be very helpful, especially during up-sampling process, where we need to deal with lost information. The performance of residual neural networks has been better than simple feed forward CNN's but a bit less than recurrent NN's, this statement is supported by the research paper (* first link in reference).

Hence when we combine the power of both recurrent and residual networks with CNN's, we get an even better performance. This is what the R2U-net has proved.

Task 3: SegNet with Atrous Spatial Pyramid Pooling (ASPP)
To improve upon the performance of the R2UNet model we decided to choose a more experimental path. Our idea was to merge two of the most prevalent model architectures into one model and observe how well the performance will hold up to the standards set by the R2Unet model. The two architectures that we chose are SegNet, as proposed by Badrinarayanan et al. [2017]. in 2017 and the ASPP module, which was proposed in by Chen et al. [2017] and is used in the DeeplabV3 model from the Deeplab family developed by Chen et al. at Google. DeeplabV3 was particularly interesting to us, since the nature of the ASPP module, is that it can be applied to any network. In its original form, the authors applied the ASPP module to the Resnet-50 and Resnet-101 networks which lead to very strong results. However, Resnet-101 contains 101 layers which makes it very memory and time intensive to train it in combination with Deeplabs ASPP module. This is where we decided to test how well the ASPP module works when combining it with our SegNet implementation. SegNet uses a deep convolutional Encoder-Decoder architecture that combines convolution, batch normalization and ReLU at each convolutional layer and pooling at the up-/ and downsampling layers. The encoder is composed of 13 convolutional layers and 5 max-pooling layers where the pooling indices are stored for later use. The decoder is composed of 13 analogous transposed convolutional layers and 5 upsampling layers that make use of the stored indices in the encoding pass for more accurate upsampling. This architecture can be seen in Figure 1, which is the graphic from the original paper by Badrinarayanan et al. [2017]. While the SegNet in our implementation is mostly the same as the original, we made a slight modification to fit our task at hand. In our SegNet implementation, we omitted the final convolutional layer and the softmax layer in the end, since we are not interested in the semantic segmentation prediction of the SegNet, but only the extracted feature maps of the model. The actual prediction will then be made by the ASPP module, which is built on top of this modified SegNet.

The ASPP module uses atrous (also called diluted) convolutions run in parallel using different dilation rates and kernel sizes together. The outputs of these atrous convolutions are then concatenated together with an Image Pooling layer, which was also used in Parsenet by Liu et al. [2015] to retain image-level features, and fed into two final convolutional layers to receive the final output. The reason why we think this seemingly arbitrary combination of SegNet and the ASPP module from the DeeplabV3 architecture might play well together, is that after using the SegNet inspired architecture for task 1, we noticed that it performs well in detecting structures from the training data, but it struggled to correctly identify whole objects and often found different classes within the same object. This is where the ASPP module is supposed to help. The ASPP re samples features at different atrous scales to gain a better concept of the context around a pixel which should help mitigate miss-classifying pixels inside of objects as done by the raw SegNet from task 1. The architecture of the ASPP module can be seen in Figure 2, but again we made slight adjustments to this structure. In our implementation, the first blocks between the input image and the ASPP module will be our

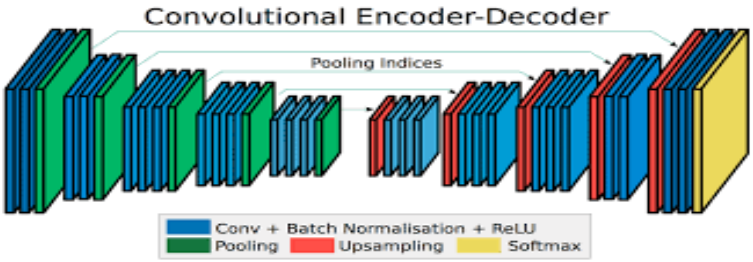


Figure 1: SegNet architecture

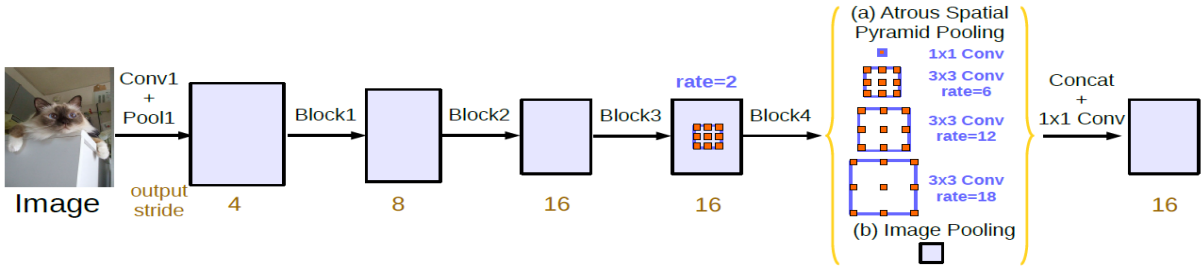


Figure 2: ASPP module

SegNet model, and its output will feed directly into the ASPP module, which is marked with the yellow brackets in Figure 1. The outputs of all the individual parts of the ASPP module will then be concatenated and fed into a convolutional layer that uses batch normalization, ReLU and dropout and finally the last layer which will yield the final desired output.

3 Experiments

For the experimental setup, we evaluated the performance of our models on the Cityscapes dataset Cordts et al. [2016]. Both models were trained for 100 epochs on the finely annotated training data (2975 images) and evaluated on the validation set consisting of 500 finely annotated images. To allow for a reasonable runtime, we rescale the input images and normalize them to get a faster convergence of the optimizer.

The R2U-Net Architecture - For our R2U net architecture, we have 5 recurrent convolutional layers in the encoder, we use a convolutional $kernel_size = 3, stride = 1, padding = 1$. After the 2D convolution we do a batch normalisation followed by a ReLU activation function. The maximum number of features we extract are 1024 , the input image had 3 channels (RGB). Max pooling is done as to reduce the dimensions / resolution of the image. The kernel size for the max-pool is 2 and the stride is also 2 units.

The decoder uses 4 recurrent layers with up-convolution, we do the up-convolution to restore the original resolution. We do the up-convolution first then pass the output to the recurrent layer , which again performs 2d convolution with kernel size 3 , followed by batch normalization and ReLU activation unit.

Table 1: Evaluation

Evaluation metrics for R2UNet and SegNet-ASPP					
Model	Accuracy	Sensitivity	Specificity	F1-Score	Jaccard Similarity
R2UNet	0.9892	0.8912	0.9943	0.8912	0.8377
SegNet-ASPP	0.9888	0.8644	0.9942	0.8644	0.8046

The concatenation operations at each level provide the residual mechanism, from each layer of encoder to each layer of the decoder. This is how we achieve 'recurrent-residual connections'. Finally we do a 1x1 (kernel size=1, stride=1, padding=0) convolution with ReLU activation. The output of this final layer is equal to the number of classes we have in our dataset, which is 34.

We have initialised weights and biases randomly for the R2U-net task and the images were down-scaled to size 512x512 through random clipping.

Now coming to the Segnet-ASPP.

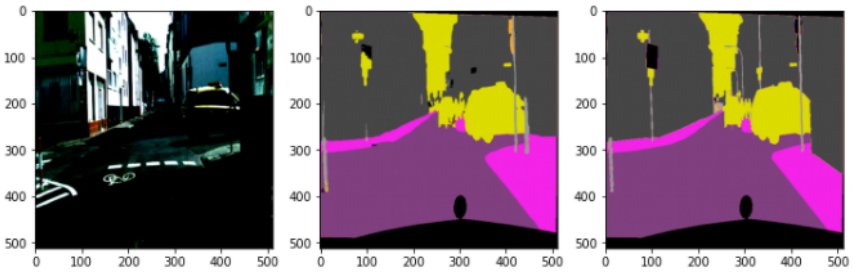
For our SegNet+ASPP architecture, we down scaled the images to size 512x256 with bilinear interpolation and we down scaled the labels to 512x256 using the Nearest Neighbor interpolation.

We used batch-normalization after each convolutional layer and thus omitted the initial normalization of the raw images. We initialized the weights using the distribution as proposed by **?** and initialized the biases as 0. The weights of the encoder part of our SegNet however are initialized by taking the weights of a pre-trained VGG16 by Simonyan and Zisserman [2015] trained on ImageNet published by Deng et al. [2009] as provided in the Pytorch library for more efficient training. This trick of using the weights of a pre-trained VGG16 model was also utilized in the original publication of SegNet Badrinarayanan et al. [2017] and helps the feature detection to converge faster, which in result leads to a more stable convergence of the decoder.
In the SegNet encoder and decoder we used a convolution kernel of size 3, stride of 1 and a padding of 1. Pooling is done with a kernel size of 2 and stride 2 in both the encoder and decoder modules of the SegNet. In the ASPP module we chose a kernel size of 3 and as padding and dilation rates 6, 12 and 18.

Evaluation of our models will be done with 5 common metrics for Image Segmentation tasks. These metrics will be Accuracy, Sensitivity, Specificity, F1-Score and Jaccard similarity. The metrics were evaluated on the validation set and can be found in Table 1. As we can see, the R2UNet model outperformed the SegNet-ASPP model by a slight margin. We think that this is most likely due to the case, that the ASPP module on top of the SegNet achieved what we wanted to, in that it gives the SegNet more context for each pixel, which should eliminate the cases where pixels were miss-classified inside of other objects. However, it seems that it did this too well and eliminated complete objects, which results in a big drop in the evaluated metrics. This is visualized in the sample predictions which we provide later. We can see that the SegNet-ASPP model sometimes fails to find whole objects in the background of the images.

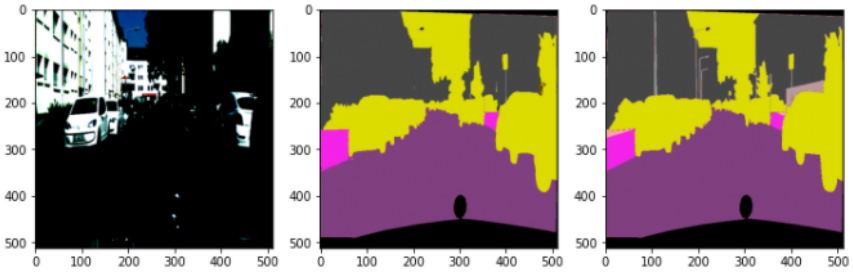
The results of the R2U-Net and the SegNet-ASPP on the CityScapes dataset are visualized below, where the image, predicted mask and ground truth are displayed in that respective order. The first set of images are from the R2U-Net and the second set of images are outputs of the SegNet-ASPP. Both have been evaluated on images from the validation set.

Image | Prediction | Label



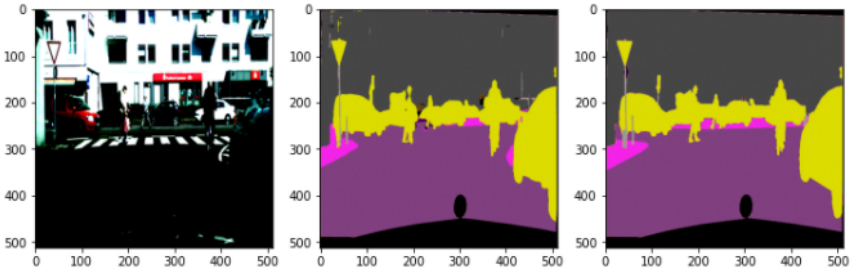
```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

Image | Prediction | Label



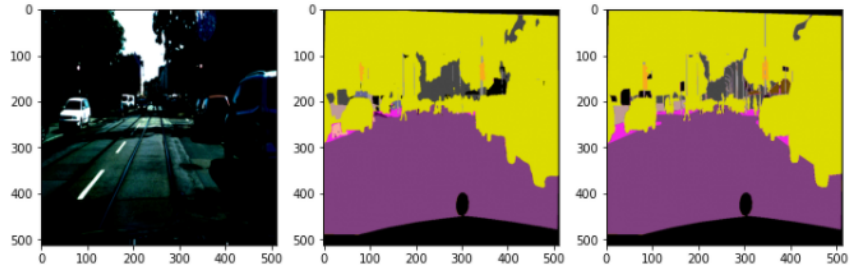
```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

Image | Prediction | Label



```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

Image | Prediction | Label



```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

Image | Prediction | Label

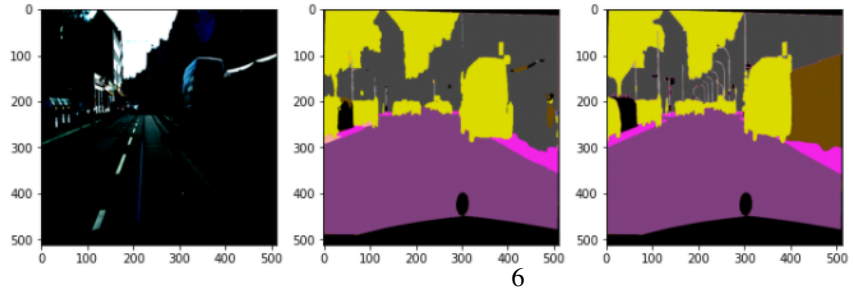


Figure 3: Examples from the SegNet-ASPP model

4 Conclusion

In this paper we implemented two distinctly different models for Image Segmentation using different techniques and compared their accuracy with each other. The R2UNet model was an already existing architecture that was initially designed for the purpose of medical image segmentation and the SegNet-ASPP model was our creation with which we tried to surpass the accuracy of the R2UNet. While we got a working model with very solid results from our SegNet-ASPP architecture, it performed slightly worse than the R2UNet, which is most likely due to the ASPP module and the basic SegNet model not harmonizing as well as we hoped to. However we are still content with the results and learned a lot from this project

References

Md Zahangir Alom, Mahmudul Hasan, Chris Yakopcic, Tarek M. Taha, and Vijayan K. Asari. Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation, 2018.

V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017. doi: 10.1109/TPAMI.2016.2644615.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Wei Liu, Andrew Rabinovich, and Alexander C. Berg. Parsenet: Looking wider to see better, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.