OBJECT SEGMENTATION

Sahil Sihag
MSc. in Embedded Systems
Universität des Saarlandes
Saarland, 66123
Matriculation Number : 7009540
sasi00004@stud.uni-saarland.de

Shashank Agarwal
MSc. in Embedded Systems
Universität des Saarlandes
Saarland, 66123
Matriculation Number : 7009562
shag00001@stud.uni-saarland.de

March 31, 2021

ABSTRACT

In an era of various devices rapidly getting dependent on the vision systems to see and interpret the world around them, detection and segmentation techniques have played an indispensable role by teaching these devices on how to decipher the world around them. In this project, we have implemented Recurrent Residual Neural Network based on U-Net model (R2-Unet) proposed by Alom et. al.[1] for semantic segmentation on Cityscapes dataset [2]. In addition to that we have proposed four different modifications to the R2U-Net architecture and compared their relative performances. Finally, the best modification of increasing model depth shows an improved detection of classes that were less frequently observed in the dataset.

1 Introduction

In computer vision, there are different possibilities in which the systems can gain a good understanding of the images. These include - Image Classification, Classification+Localisation, Object Detection, Semantic Segmentation and Instance Segmentation. Semantic segmentation (or object segmentation) is a technique to classify each pixel of the image to the object category it belongs to. Unlike instance segmentation, it only classifies different type of objects in the images as different labels and ignores the classification of each instances of each object [Figure 1].

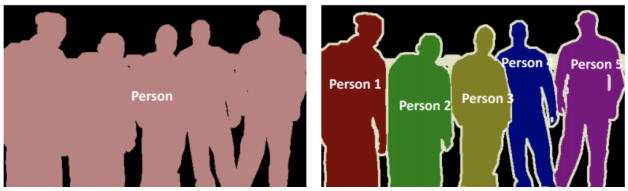

Figure 1: *Left figure:* Semantic segmentation. *Right figure:* Instance Segmentation

The Cityscapes dataset provided by Daimler AG et. al. consists of diversified urban street scenes from different cities under different weather and lighting conditions. We have used it to learn about different objects in the images and then classify another set of images (validation set), by assigning each pixel a particular label and visualise them by mapping each of those labels to different colors.

R2U-Net is the revised version of a chain of different architectures which have been proposed in recent years. It is a composition of U-Net, Residual Network and Residual Convolution Neural Network [Figure **??**].

- **U-Net :** It contains two main sides of computation - Encoder (Convolutional Network) and Decoder (Deconvolutional Network). Encoders are responsible for producing a low dimensionality dense representation of an

input image, which can already be used for semantic segmentation but lacking fine details. While, decoders recover the resolution lost by downsampling operation through a set of refinement stages.

- **ResU-Net :** It works like U-Net except that the use of residual blocks in the network help in avoiding the vanishing/exploding gradient issues.
- **R2U-Net :** It uses recurrent residual convolutional layers wherein there are recurrent convolutions before downsampling, before upsampling as well as before outputting the segmentation maps. This helps in achieving a better feature representation in the segmentation maps. [Figure 2]

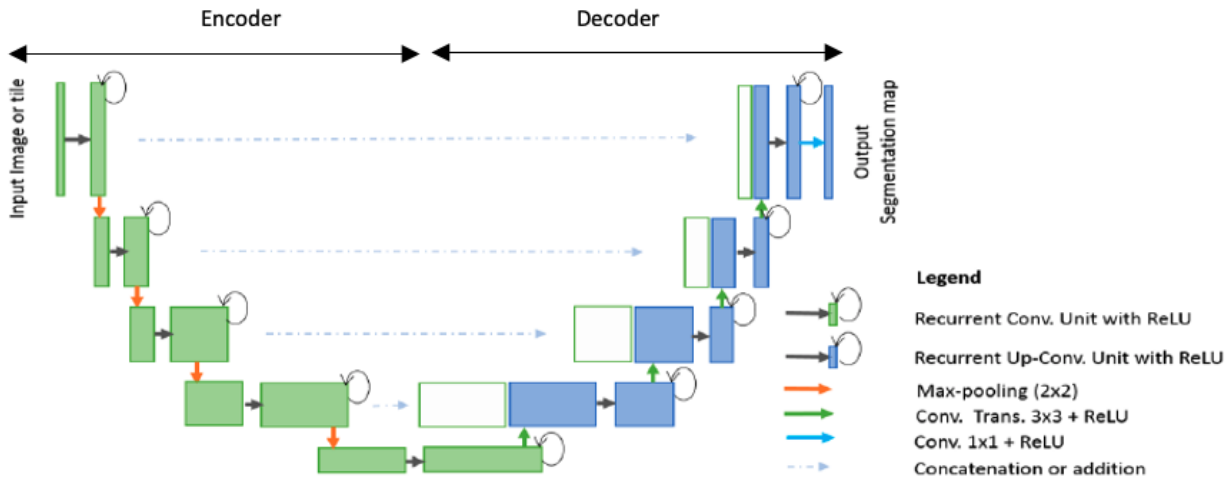

Figure 2: R2U-Net architecture with encoder and decoder blocks

2 Methodology

2.1 Dataset and Dataloader

We have used the Cityscapes dataset to train the model which contained RGB images along with their corresponding finely annotated images for semantic segmentation. There were a total of 5000 images which were further divided into training, validation and test sets. These input images and segmentation masks were resized from the original resolution of 1024*2048 to 512*1024 for decreasing the training time while causing negligible loss of information. To keep the model unbiased to the training images, we have shuffled the training images before feeding them for training.

2.2 Architecture

We used the R2U-net architecture as proposed in [1] for task 2. In order to improve the performance obtained by this base model, we implemented and compared the performance of 4 different architectures:

- **Modification 1 :** *R2U-Net with one additional downsampling and upsampling step.* This effectively increased the depth of architecture by a factor of one. So, there were a total of 2048 activation maps in the final encoding step.
- **Modification 2 :** *R2U-Net with one less downsampling and upsampling step.* This effectively decreased the depth of architecture by a factor of one. So, there were only 512 activation maps in the final encoding step.
- **Modification 3 :** *R2U-Net with decreased number of activation maps in each layer.* The number of activation maps in the first layer were decreased from 64 to 48. As the number of activation maps are doubled in each subsequent encoding step and then halved in subsequent decoding steps, the architecture had lesser activation maps in total.
- **Modification 4 :** *R2U-Net with increased number of activation maps in each layer.* The number of activation maps in the first layer were increased from 64 to 72. Effectively, the architecture had more activation maps in total.

2.3 Loss Function

We have adopted the approach of maximum likelihood estimate to seek for the probability estimates in this multi class classification problem. Cross entropy loss is independent of the individual class labels, thus making it suitable in this situation. This loss function examined each pixel individually by comparing the class predictions to the target label and then averaged this loss over all pixels in the image. Furthermore, since we have a subset of invalid classes in our dataset, the loss due to the wrong prediction of the pixels originally belonging to the invalid classes has been ignored. This was achieved by masking all the invalid classes with an ignore index and then passing this index to the loss function.

2.4 Weight Initialization

Weight initialization plays a vital role while training deep neural networks so as to prevent the gradients from exploding or vanishing. Here, we tried two approaches to initialize the weights - Xavier's Initialization and Kaiming initialization (by default in Conv2d function of PyTorch) and chose the latter one. Xavier's method failed to yield good results due to the fact that it works well for activation functions like Sigmoid and Tanh, however fails to perform well in case of ReLU. On the other hand, Kaiming's initialization performed much better due to the non-symmetric shape of ReLU.

2.5 Optimizer

While doing the optimization of dense models, adaptive learning rate methods tend to be beneficial as they learn much faster than stochastic gradient descent. Therefore, we have used Adam optimizer which helped us in yielding lower loss values than SGD (even at the very early stages of training epochs) and avoiding the saddle points.

2.6 Hyperparameters

In order to find a suitable learning rate, we compared the performance of R2U-Net on 12 different learning rates between 1E-3 to 1E-8. After comparing the results 1E-6 was found to be the most suitable. Similar approach was used to find the best learning rate for the four different models proposed above. The training of model was stopped when there was no significant decrease in the training loss.

3 Results

For the qualitative analysis of our implementations we have compared the performance across 5 different metrics: *Specificity, Sensitivity, F1 Score, Accuracy, Jaccard Score*. Along with this, we have shown the output of our model on previously unseen images.

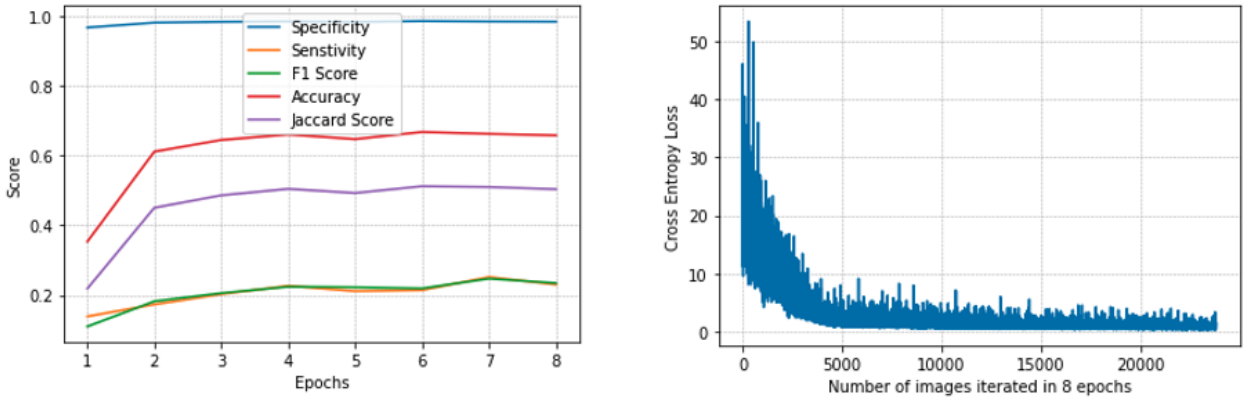

Figure 3: Quantitative analysis of R2U-Net model (Task 2). *Left figure* depicts the plot of 5 different evaluation metrics during 8 training epochs. *Right column* depicts the plot of cross entropy loss during 8 training epochs.

It is visible from the right plot in fig. 3 that loss values were initially very high but decreased very rapidly. Since there were 2975 images in the training dataset, the model reached a significantly small loss value by the end of the first epoch. As a result, the model achieved 38% accuracy after the first epoch. In subsequent epochs, the accuracy of the model increases and finally reaches 66%.

The specificity of the model is very high as it considers the value of true negatives. The values of true negatives remain high in this multi-class setting because the model finally predicts a single class and even after a wrong prediction it has correctly predicted negative classes for *number of classes* - 2 labels.

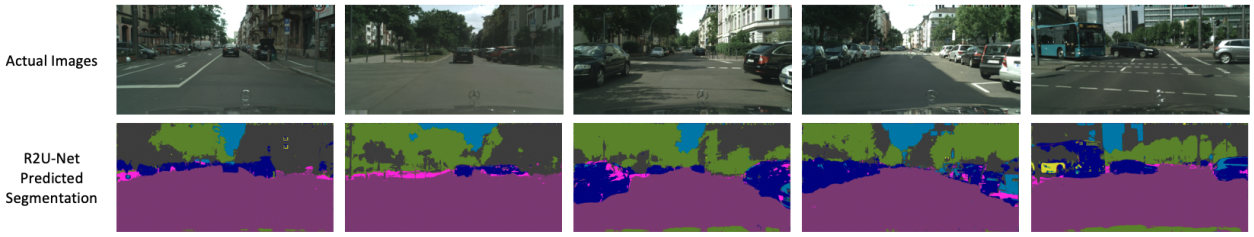

Figure 4: Model output on unseen images. *First row* shows the original images. *Second row* shows the predicted segmentation labels of R2U-Net model (Task 2).

Predictions of base R2U-Net in fig. 4 shows that it is able to correctly classify pixels belonging to sky, road, cars and vegetation. But it is not able to perform well on classes that are less frequently observed in the dataset such as train, bicycle and persons. It can also be seen that the boundary in segmentation masks is not accurate around the object edges.

Table 1: Comparison of experimental results between different models.

		Metrics Scores				
Models	Learning Rate	Accuracy	Jaccard Score	F1 Score	Specificity	Sensitivity
R2U-Net	**1E-06**	**0.66**	**0.50**	**0.23**	**0.98**	**0.23**
Modification 1 (+1 Depth)	**1E-05**	**0.73**	**0.59**	**0.40**	**0.99**	**0.46**
Modification 2 (-1 Depth)	1E-05	0.58	0.42	0.31	0.97	0.36
Modification 3 (less filters)	5E-05	0.71	0.56	0.40	0.98	0.46
Modification 4 (more filters)	1E-06	0.67	0.52	0.26	0.98	0.28

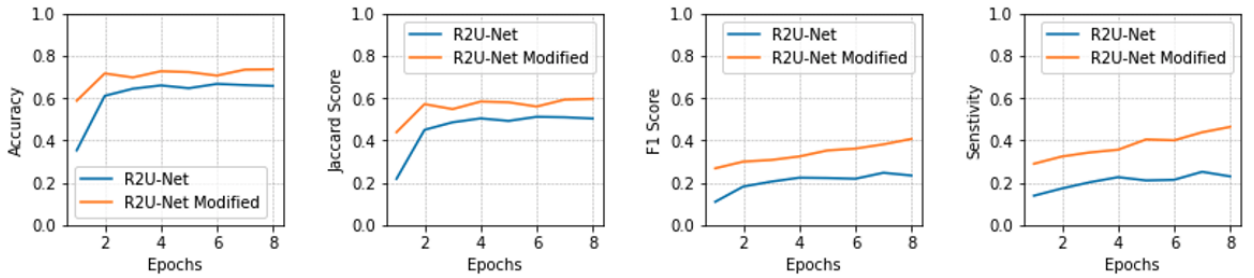

Figure 5: Comparison of the metrics scores between R2U-Net and R2U-Net with Modification 1

Our proposed modifications on R2U-Net displayed a significant increase in model performance. Modification 1, which increased the model depth by a factor of one showed 7% increase in the model accuracy. Along with this there was also improvement in Jaccard Score, F1 Score, Specificity and Sensitivity.

On the other hand, Modification 2, which decreased the model depth by a factor of one showed an 8% decrease in model accuracy in comparison to base R2U-Net model. As the metrics of the model showed an improvement when the depth of the model was increased, we can conclude that the depth of base R2U-Net could be increased for this application.

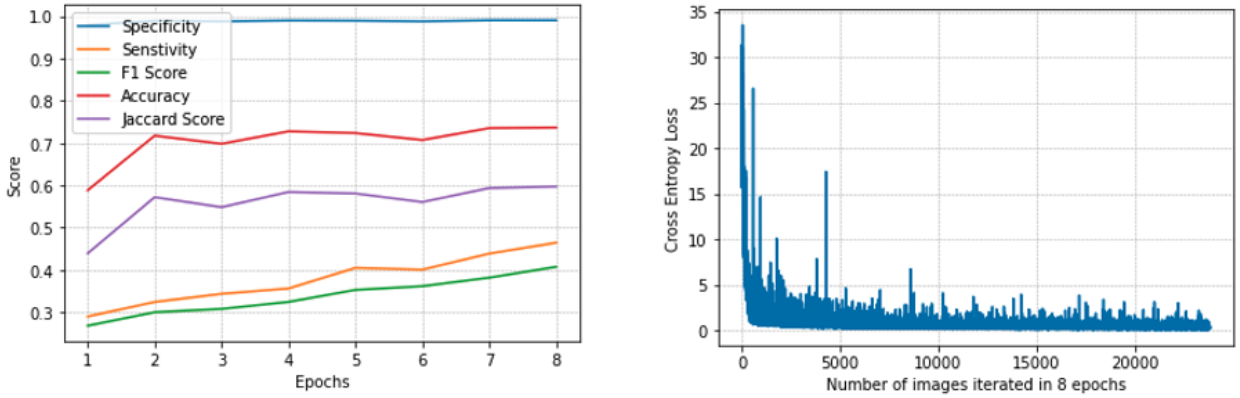

Figure 6: Quantitative analysis of R2U-Net model with modification 1 (Task 3). *Left figure* depicts the plot of 5 different evaluation metrics during 8 training epochs. *Right column* depicts the plot of cross entropy loss during 8 training epochs.

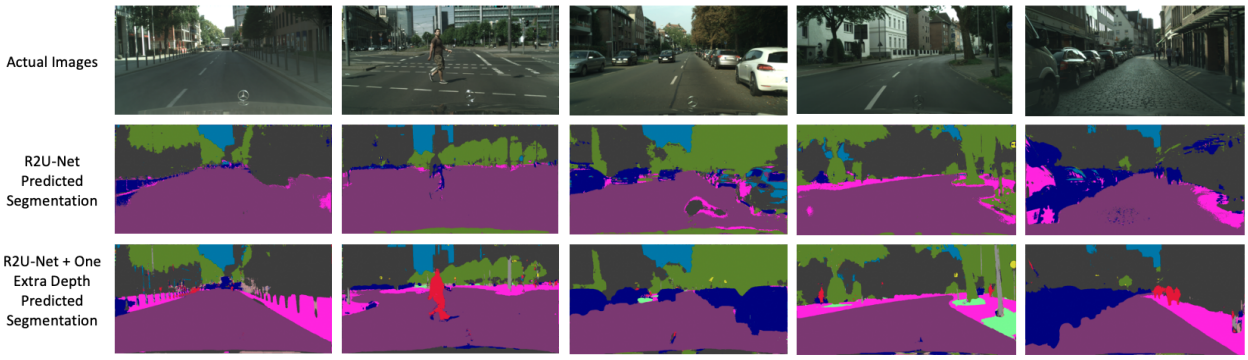

Figure 7: Comparison of model output between Task 2 and Task 3 on unseen images. *First row* shows the original images. *Second row* shows the predicted segmentation labels of R2U-Net model (task 2). *Third row* shows the predicted segmentation labels of R2U-Net with Modification 1 (task 3)

Predictions shown in fig. 7 demonstrate that the proposed modification offer a significant improvement over the base R2U-Net. The new architecture was able to identify more classes. In addition to this, it also offered better edge detection for different objects. For example, persons and sidewalks which were previously missed or not correctly identified were now correctly identified even if they occupied smaller areas in the provided images.

Table 2: Comparison of space and time consumption between different models.

	Space and Time Consumption	
Models	Forward Pass + Backward Pass (sec)	GPU Training Memory (MB)
R2U-Net	**2865**	**13455**
Modification 1	**3487**	**15091**
Modification 2	2460	12777
Modification 3	2033	10135
Modification 4	3561	15037

Even though the increase in model depth offered a significant improvement over the base R2U-Net, it also took more time in order to train and had higher space consumption on the GPU. Conversely, decrease in the model depth (or the number of activation maps in each layer) decreased the amount of time required for one training epoch of the model. This is expected as the number parameters increased and decreased in the respective categories.

3.1 Conclusion and Future Work

In this project, we proposed four different modifications to the R2U-Net architecture and compared their performance with the base R2U-Net. These models were evaluated on the Cityscapes Dataset. The experimental results demonstrate that the proposed modification of increasing the depth of architecture shows better performance in semantic segmentation task for the above dataset. In future, we would like to explore a fusion of optimizers by using Adam at the beginning to make the training faster and then shift to stochastic gradient descent with momentum in the end. This can lead the model to converge better. Another approach we would like to explore is the use of weighted cross entropy loss function by dynamically assigning weights to different classes, as proposed in [3]. This can allow the model to be more robust to class imbalances encountered in the above dataset.

References

[1] 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." In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 417–422. IEEE, 2014.

[2] Marius Cordts, et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding." In *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of*, pages 312–318. IEEE, 2014.

[3] Sheng Lu, Feng Gao, Changhao Piao, Ying Ma. "Dynamic Weighted Cross Entropy for Semantic Segmentation with Extremely Imbalanced Data" In *2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*, pages 230–233. IEEE, 2019, volume 1.

[4] Jonathan Long, Evan Shelhamer, Trevor Darrell. "Dynamic Weighted Cross Entropy for Semantic Segmentation with Extremely Imbalanced Data" In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 230–233. IEEE, 2019, volume 1.

[5] Abhishek Chaurasia, Eugenio Culurciello. "Exploiting encoder representations for efficient semantic segmentation" In *2017 IEEE Visual Communications and Image Processing (VCIP)*, 2017

[6] Meet P Shah. "Semantic Segmentation Architectures Implemented in PyTorch." In *https://github.com/meetshah1995/pytorch-semseg*, 2017