# NNTI Project: Semantic Segmentation using Deep Learning

**Sohom Mukherjee**
Student Number:7010515
somu00003@stud.uni-saarland.de

**Shayari Bhattacharjee**
Student Number:7009998
shbh00002@stud.uni-saarland.de

## Abstract

In this project, we address the computer vision task of semantic segmentation using deep learning-based approaches. In the first task, we train ENet architecture on the Pascal VOC 2012 dataset and obtain mean IoU of 27.75 % on the validation set. For the second task and third tasks, we train R2Unet and PSPNet architectures on Cityscapes dataset respectively, and obtain mean IoU of 33.65 % and 75.15 % respectively, on the validation set. The code for our project is available at our GitHub repository `https://github.com/mukherjeesohom/NNTI-SemSeg`.

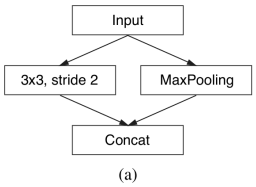
## 1 Introduction

Semantic segmentation refers to the multi-label classification task of assigning a class label to each pixel in an image. It is one of the key topics in high-level computer vision, with applications in autonomous driving and video surveillance. Over the past few years, deep learning (DL) architectures, primarily Convolutional Neural Networks (CNNs) based architectures, have surpassed traditional computer vision techniques such as graph cuts Boykov et al. [2001] and conditional random fields (CRFs) Plath et al. [2009] by large margins in terms of accuracy as well as efficiency. DL-based semantic segmentation models can be classified into the following major groups: (a) Fully convolutional networks (b) CNNs with graphical models (c) Encoder-decoder based models (d) Multi-scale and pyramid network based models (e) Recurrent neural network and attention based models.

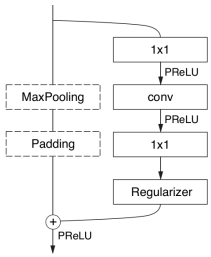
One of the first successful DL-based approaches for semantic segmentation was the fully convolutional network (FCN) proposed in Long et al. [2015]. In this work, the fully connected layers of existing well-known classification models, such as AlexNet Krizhevsky et al. [2012], VGG Simonyan and Zisserman [2014], ResNet Szegedy et al. [2015], and GoogLeNet He et al. [2016] were replaced by fully convolutional layers, thereby enabling FCN to handle inputs images of arbitrary sizes and output spatial segmentation maps. In spite of being a milestone in semantic segmentation, the FCN architecture suffered from some drawbacks - the most important being that it was not able incorporate to global context information efficiently. Some of these limitations were attempted to be solved by encoder-decoder style architectures. Two of the most popular architectures in this category are SegNet Badrinarayanan et al. [2017] and UNet Ronneberger et al. [2015]. SegNet consists of a convolutional encoder-decoder architecture: the encoder consists of 13 convolutional layers and the corresponding decoder network is followed by a pixel-wise classification layer. UNet uses a contracting encoder and symmetric expanding decoder, along with skip connections between the encoder and decoder to incorporate low-level features efficiently. DeepLab Chen et al. [2014] uses graphical models like dense CRFs to improve the segmentation map produced by FCN along boundaries. Works like Feature Pyramid Network (FPN) and Pyramid Scene Parsing Network (PSPNet) utilize the inherent multi-scale, pyramidal hierarchy of CNNs to better incorporate global context information from a scene.

We have made the following contributions in this project:

Submitted to Neural Networks: Theory and Implementation 2020 project. Do not distribute.




(a)




(b)

(a)

Table 1: ENet architecture. Output sizes are given for an example input of $512 \times 512$.

| Name | Type | Output size |
|---|---|---|
| initial | | $16 \times 256 \times 256$ |
| bottleneck1.0 | downsampling | $64 \times 128 \times 128$ |
| $4\times$ bottleneck1.x | | $64 \times 128 \times 128$ |
| bottleneck2.0 | downsampling | $128 \times 64 \times 64$ |
| bottleneck2.1 | | $128 \times 64 \times 64$ |
| bottleneck2.2 | dilated 2 | $128 \times 64 \times 64$ |
| bottleneck2.3 | asymmetric 5 | $128 \times 64 \times 64$ |
| bottleneck2.4 | dilated 4 | $128 \times 64 \times 64$ |
| bottleneck2.5 | | $128 \times 64 \times 64$ |
| bottleneck2.6 | dilated 8 | $128 \times 64 \times 64$ |
| bottleneck2.7 | asymmetric 5 | $128 \times 64 \times 64$ |
| bottleneck2.8 | dilated 16 | $128 \times 64 \times 64$ |
| *Repeat section 2, without bottleneck2.0* | | |
| bottleneck4.0 | upsampling | $64 \times 128 \times 128$ |
| bottleneck4.1 | | $64 \times 128 \times 128$ |
| bottleneck4.2 | | $64 \times 128 \times 128$ |
| bottleneck5.0 | upsampling | $16 \times 256 \times 256$ |
| bottleneck5.1 | | $16 \times 256 \times 256$ |
| fullconv | | $C \times 512 \times 512$ |

(b)

Figure 1: ENet Architecture. From Paszke et al. [2016]

- Task I: Implemented (Efficient Neural Network) ENet architecture on PASCAL VOC 2012 semantic segmentation dataset and obtained results with and without data augmentation.
- Task II: Implemented (Recurrent Residual Convolutional Neural Network) R2UNet architecture on urban scene understanding dataset Cityscapes.
- Task III: Implemented (Pyramid Scene Parsing Network) PSPNet architecture on Cityscapes dataset and compared the results with that of Task II.

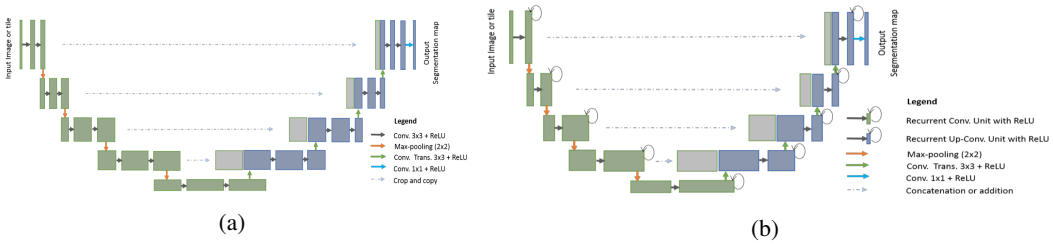
## 2 Proposed Approach

In this section we discuss the various architectures we have used for the project: (a) ENet, (b) R2Unet, and, (c) PSPNet.

### 2.1 Efficient Neural Network: ENet

Efficient Neural Networks (ENet) was proposed by Paszke et al. [2016] which was originally targeted for low latency operations. This network surpasses many other models as it requires less parameters and is computationally faster.

There are 7 stages in this architecture. The first 4 stages are part of the encoding process, whereas the rest 3 stages together accounts for the decoding. Each stage consists of multiple blocks known as bottlenecks which comprises of three convolutional layers: the objective of the first layer is dimensionality reduction, the second layer is the main convolutional(conv) layer and the third layer is for expansion. Between each layer, Batch Normalisation and PReLu is placed. For a downsampling bottleneck, a max-pool layer is concatenated to the convolutional branch. Moreover, the $1 \times 1$ projection layer is replaced by a convolution layer of $2 \times 2$ with stride 2 and zero paddings are also added to equalise the number of feature maps. The second layer of the bottleneck , the conv layer consists of $3 \times 3$ filters and can be of regular, dilated, asymmetric or full convolution type. Spatial dropout is used for the third layer of the bottleneck. For the decoding stages, the bottlenecks are transformed such that the max-pool layer is replaced by the max unpooling and padding is replaced with spatial convolution without bias. For the encoding stages, The Initial stage consists of a single block which has a $3 \times 3$ projection with a stride 2 and a max-pooling layer is concatenated as the input. The first stage consists of 5 bottlenecks. The second and third stage are almost identical except the first downsampling layer in the second stage. The fourth and fifth stage belongs to the decoding stages, where upsampling is performed and the last stage is the full convolutional layer.




(a) (b)

Figure 2: (a) UNet Architecture. (b) R2Unet Architecture. From Alom et al. [2018]

### 2.2 Recurrent Residual Convolutional Neural Network: R2UNet

U-Net model is considered very suitable for segmentation tasks because of simultaneous use of global location and context, less training samples requirements and direct production of segmentation maps due to end-to-end pipeline processing of an image. However, this network was more appropriate for medical image segmentation rather than general image segmentation tasks.. This drawback led to the development of Recurrent Residual Convolutional Neural Network(R2UNet) by Alom et al. [2018] which was superior then UNet with same number of parameters and improved training and testing performance.

In UNet (Fig.2a), the entire structure comprises of two units: encoder unit and decoder unit. ReLu activation follows the convolution in both the encoder and decoder units. For encoding, downsampling is performed with $2\times2$ max-pool operations, whereas in decoding transpose operations are used for upsampling. In R2UNet(Fig.2b), the output of the last input and present input are multiplied with their weights and are added along with the bias. This output is then fed to a ReLu activation function to obtain the result of the RCNN block. The RCNN output is then fed to the residual unit to obtain the final output for the R2UNet architecture.

### 2.3 Pyramid Scene Parsing Network: PSPNet

Many deep networks are deemed insufficient because of their inability to learn contextual relationship. Thus to improve performance of semantic segmentation, a deep network with a effective global prior representation was necessary. This drawback of deep networks was circumvented by Pyramid Scene Parsing Network (PSPNet) by Zhao et al. [2017]. In this network, the global level prior is constructed by the *Pyramid Pooling Module*.

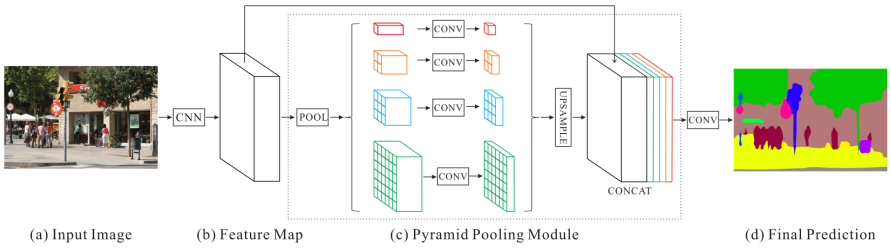
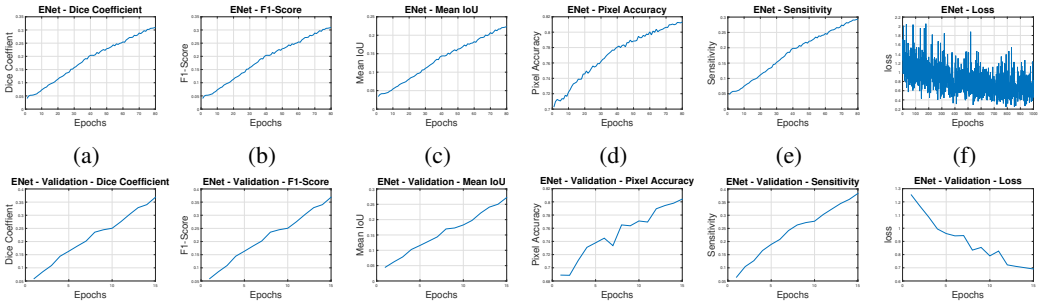



Figure 3: PSPNet Architecture. From Zhao et al. [2017]

In the architecture depicted by Figure. 3, The feature maps which is $\frac{1}{8}$ of the input image are first extracted using a ResNet with a dilated network strategy. Following the extraction of the feature maps, we obtain context information from the pyramid pooling module. The pyramid pooling module comprises of four layers called pyramidswhere the first layer(n Red) is the coarsest and has the minimum bin size of $1 \times 1$. The other layers have the bin sizes of $2\times2$, $3\times3$ and $6\times6$. This layers forms contextual representations for different locations. Then we use a $1\times1$ convolution layer after each pyramid to maintain the weight of global feature. After convolution, we upsample all the reduced size feature maps to the size of original feature maps via bilinear interpolation. After upsampling, all

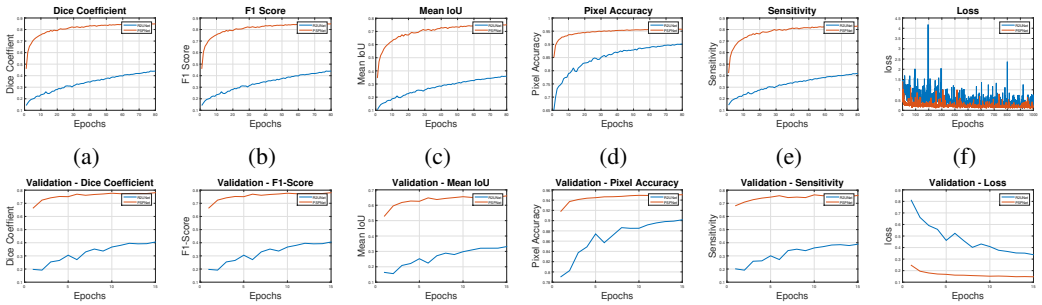



(a) (b) (c) (d) (e) (f)

(g) (h) (i) (j) (k) (l)

Figure 4: Evaluation Metric for ENet. (a)-(f) Training Metrics. (g)-(l) Validation Metrics.

the feature maps are concatenated to produce the pyramid pooling global feature. After receiving the global prior, it undergoes another set of convolution to produce the final prediction.




(a) (b) (c) (d) (e) (f)

(g) (h) (i) (j) (k) (l)

Figure 5: Comparison of evaluation metrics for R2UNet and PSPNet. (a)-(f) Training Metrics. (g)-(l) Validation Metrics.

## 3 Experiments and Results

This section is divided into six parts. The first subsection discusses the various evaluation metrics that is used to check the competency of the neural network models on the semantic segmentation task. The second subsection focuses on the datasets that were used for training and inference. The third, fourth and fifth subsection demonstrates the implementation of the semantic segmentation models ENet, R2UNet and PSPNet respectively. Lastly, the sixth subsection provides a comparative analysis between semantic segmentation results using R2UNet and PSPNet on Cityscapes dataset. All experiments have been performed on a system equipped with a single NVIDIA GeForce GTX 1080 and an Intel Core i7-4790K Processor with 32 GB RAM. The training, testing and performance evaluation scripts have been written in PyTorch and graphical plots have been obtained using `MATLAB`. All parameters used for training and validation can be found in the corresponding `json` file of the repository.

### 3.1 Evaluation Metrics

In this section, we discuss the five evaluation metrics that we have used to check the performance of the neural network arcitectures, which are F1-Score, Dice Coefficient, Mean Intersection Over Union (MIoU), Mean Pixel Accuracy (MPA) and Sensitivity (SE).

1. **F1-Score** is a popular metric for computing the accuracy of the semantic segmentation models. It is defined as the ratio between the True Positive Rate and the sum of True positive Rate and the mean of False Positive rate and False Negative Rate. The mathematical formula of F1-Score is as follows:

$$\text{F1} - \text{Score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{1}$$

where TP refers to the True Positive rate, FP and FN refers to the False Positive rate and False Negative Rate respectively.

2. **Dice Coefficient** is another popular metric to compute the accuracy of the Semantic Segmentation model. It is defined as the ratio between twice the intersection of the Ground truth (GT) and Predicted masks (Pred) divided by the total pixels of both the ground truth and predicted masks.

$$\text{Dice} = \frac{2|GT \cap Pred|}{|GT| + |Pred|} \tag{2}$$

It can also be calculated as:

$$\text{Dice} = \frac{2TP}{2TP + FP + FN} \tag{3}$$

3. **Mean Intersection Over Union (MIoU)** is the average Intersection Over Union(IoU) over all classes present in the Semantic Segmentation. The Intersection Over Union is the ratio between the intersection of the Ground Truth(GT) and the Prediction Mask(Pred) to the Union of Ground Truth(GT) and the Prediction Mask(Pred).

$$\text{MIoU} = \frac{1}{k+1} \sum_{i=0}^{i=k} \frac{|GT_i \cap Pred_i|}{|GT_i \cup Pred_i|} \tag{4}$$

where k is the total number of classes.

4. **Mean Pixel Accuracy (MPA)** is ratio between the number of correct pixel per class is computed which is then further averaged over the total number of classes.

$$\text{MPA} = \frac{1}{k+1} \sum_{i=0}^{i=k} \frac{p_{ii}}{\sum_{j=0}^{j=k} p_{ij}} \tag{5}$$

where $p_{ii}$ refers to the correct pixel in class i.

5. **Sensitivity (SE)** refers to the ratio between True Positive rate(TP) to the sum of True Positive rate(TP) and False Negative Rate(FN).

$$\text{SE} = \frac{TP}{TP + FN} \tag{6}$$

### 3.2 Datasets

In this project we have used two datasets namely, PASCAL VOC 2012 semantic segmentation dataset and Cityscapes. The brief description of the datasets are as follows:

1. **Pascal VOC**: Pascal Visual Object Classes (VOC) Everingham et al. [2010] is one of the most popular datasets when it comes to semantic segmentation. The training set contains 1464 images and the validation set contains 1449 images. There are 21 classes in this dataset which are categorised as vehicles, household, animals, aeroplane, bicycle, boat, bus, car, motorbike, train,bottle, chair, dining table, potted plant, sofa, TV/monitor, bird, cat, cow, dog, horse, sheep, and person. If a pixel doesn't belong to any of the classes, then it is labelled as the background.
2. **Cityscapes**: Cityscapes Cordts et al. [2016] is one of the large scale datasets with 5000 fine annotated images and 20000 course annotated images. This dataset consists of 30 classes that can be assembled into 8 categories as (flat surfaces, humans, vehicles, constructions, objects, nature, sky, and void. This dataset originally recorded as video consists of urban street scenes which was collected during various weather conditions and in different cities.

Table 1: Comparison of Model Complexities

| Architecture | Training Time/Epoch (min) | # Parameters | Model Size |
|---|---|---|---|
| **ENet** | 4:05 | 00.36M | 3.1 MB |
| **R2UNet** | 20:36 | 30.00M | 240.3 MB |
| **PSPNet** | 15:52 | 51.44M | 411.9 MB |

### 3.3 Semantic Segmentation using ENet

We have performed two sets of experiments using ENet. In the first experiment we followed the steps outlined in the `ipynb` notebook for Task I, and obtained modest results on the Pascal VOC 2012 dataset. In this report, we provide results for the second experiment where we used a different dataloader (analogous to the ones we use in Tasks II and III), along with data augmentation to obtain better results (mIoU of 27.75 % on the validation set). The data augmentation is performed according to the additional annotations provided by Hariharan et al. [2011]. The model is trained for 80 epochs and validation is performed at every 5 epochs. As is the standard practice in state-of-the-art semantic segmentation works like Chen et al. [2017], we use the "poly" learning rate scheduler where the learning rate decreases from the base learning rate to zero during the course of training according to the formula $lr_{iter+1} = lr_{iter} \left(1 - \frac{iter}{max_iter}\right)^{power}$. A base learning rate of 0.01 and $power$ of 1 has been used for our training. We use the Cross Entropy loss function and Stochastic Gradient Descent (SGD) optimizer with weight decay of 0.0001 and momentum of 0.9.

We have used a batch size of 8 for the training as well as validation steps. Due to limitations of memory on GPUs, we resize the input images to a `base size` of 400. Following this the we perform random rescaling between 0.5 and 2 and cropping to a `crop size` of 380. This is followed by data augmentation steps of random rotation between -10 and 10 degrees, random horizontal flip, and random Gaussian blurring. Evaluation metrics and losses for the training and validation epochs have been plotted in Fig. 4, and the final results have been reported in Table 2. Visualizations of segmentation maps obtained can be found in Fig. 6

### 3.4 Semantic Segmentation using R2UNet and PSPNet

We outline the experimental details for Task II (R2UNet) and Task III (PSPNet) in this section. The hyperparameters used for experimentation are same for both tasks and differ solely in the model architecture. PSPNet is trained with ResNet-50 backbone. Both models are trained on the Cityscapes dataset without any image resizing (`base size` 1024) for 80 epochs. As explained in the previous section, we use "ploy" learning rate scheduler with base learning rate of 0.01 and $power$ of 1. The Cross Entropy loss function and Stochastic Gradient Descent (SGD) optimizer with weight decay of 0.0001 and momentum of 0.9 has been used. Due to limitations of GPU memory we have used a batch size of 3 for training as well as validation steps. We perform random rescaling of input image between 0.5 and 2 followed by cropping to a `crop size` of 512. This is followed by data augmentation steps of random rotation between -10 and 10 degrees, random horizontal flip, and random Gaussian blurring. Evaluation metrics and losses for the training and validation epochs have been plotted in Fig. 5 in the form of a comparison between ENet and PSPNet, and the final results have been reported in Table 2. Visualizations of segmentation maps for R2Unet and PSPNet can be found in Fig. 7 and Fig. 8, respectively.

### 3.5 Comparative Analysis

A comparison of complexities of the three semantic semantic segmentation models has been depicted in Table 1 and the performance measures for the models is shown in Table 2 and Fig. 5. It is easy to observe that while PSPNet gives superior performance in terms of evaluation metrics and output segmentation maps compared to R2UNet, the model complexity is much higher. ENet is clearly the most efficient in terms of model complexity, while giving satisfactory segmentation performance.

Table 2: Training and Validation Metrics

| | Training Metrics | | | | | | Validation Metrics | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Architecture** | **Dice** | **F-One** | **Mean IoU** | **Pixel Accuracy** | **Sensitivity** | **Loss** | **Dice** | **F-One** | **Mean-IoU** | **Pixel Accuracy** | **Sensitivity** | **Loss** |
| **ENet** | 0.3098 | 0.3098 | 0.2236 | 0.8131 | 0.2935 | 0.5091 | 0.3686 | 0.3686 | 0.2721 | 0.8041 | 0.3828 | 0.6908 |
| **R2Unet** | 0.4395 | 0.4395 | 0.3591 | 0.9014 | 0.4181 | 0.3918 | 0.4056 | 0.4056 | 0.3312 | 0.9016 | 0.3889 | 0.3378 |
| **PSPNet** | 0.8493 | 0.8493 | 0.7515 | 0.9574 | 0.8305 | 0.1690 | 0.7806 | 0.7806 | 0.6608 | 0.9505 | 0.7598 | 0.1466 |

# 4 Conclusion and Future Work

In this project, we implemented the task of Semantic Segmentation on three Neural Networks based architectures using two datasets. For Task I, we implemented ENet on Pascal VOC dataset and observed better results with dataset augmentation with very few computational parameters. For Task II and Task III, we used the architecture of R2UNet and PSPNet respectively on Cityscapes dataset and compared their evaluation metrics, to find that that PSPNet performed better than R2UNet in terms of evaluation metrics, but requires greater model complexity. As our possible future scope of this project, we can aim to incorporate an attention mechanism which can compute feature importance at various conditions and produce a more accurate global prior for the semantic segmentation.

# References


Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

Nils Plath, Marc Toussaint, and Shinichi Nakajima. Multi-class image segmentation using conditional random fields and global classification. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 817–824, 2009.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

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. *arXiv preprint arXiv:1802.06955*, 2018.

Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338, 2010.

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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *2011 International Conference on Computer Vision*, pages 991–998. IEEE, 2011.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

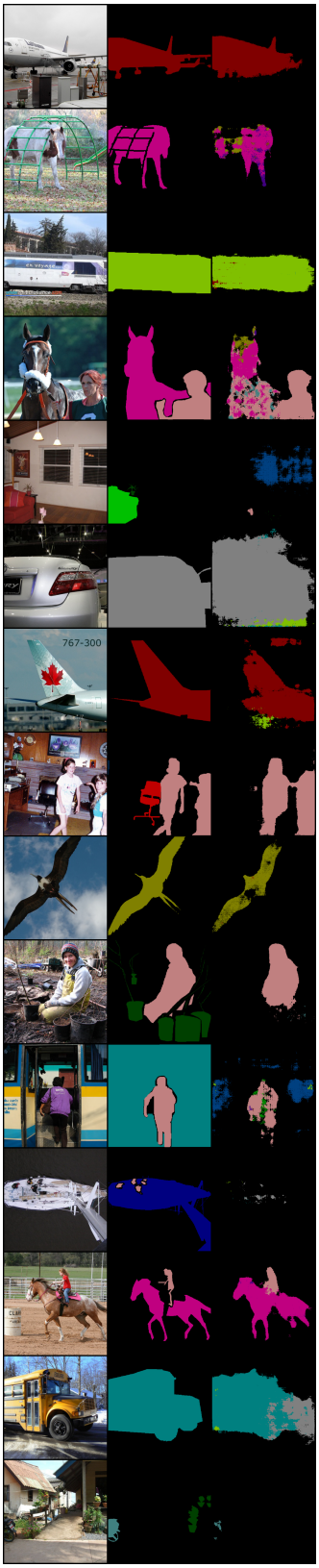


Figure 6: Segmentation maps obtained using ENet. (a) Original RGB Image, (b) Ground Truth, (c) Segmentation Output

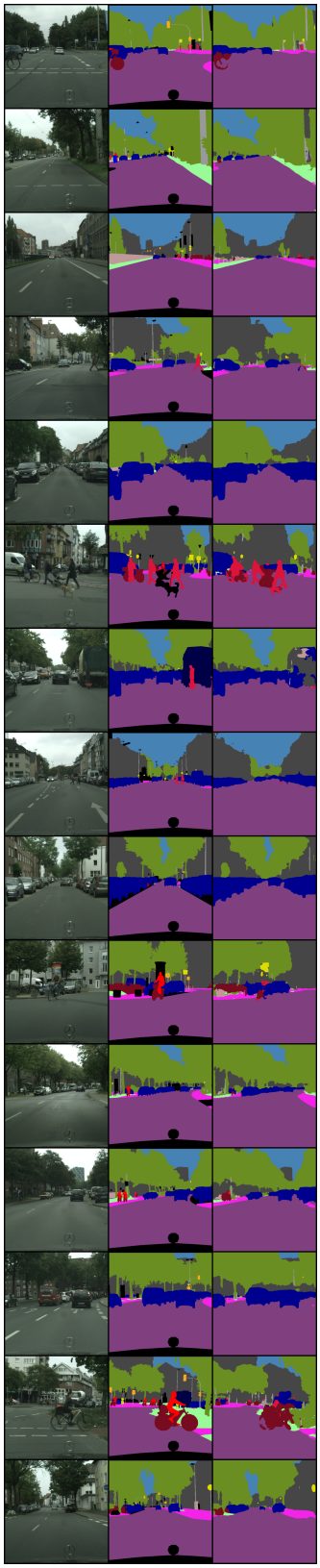


Figure 7: Segmentation maps obtained using R2Unet. (a) Original RGB Image, (b) Ground Truth, (c) Segmentation Output

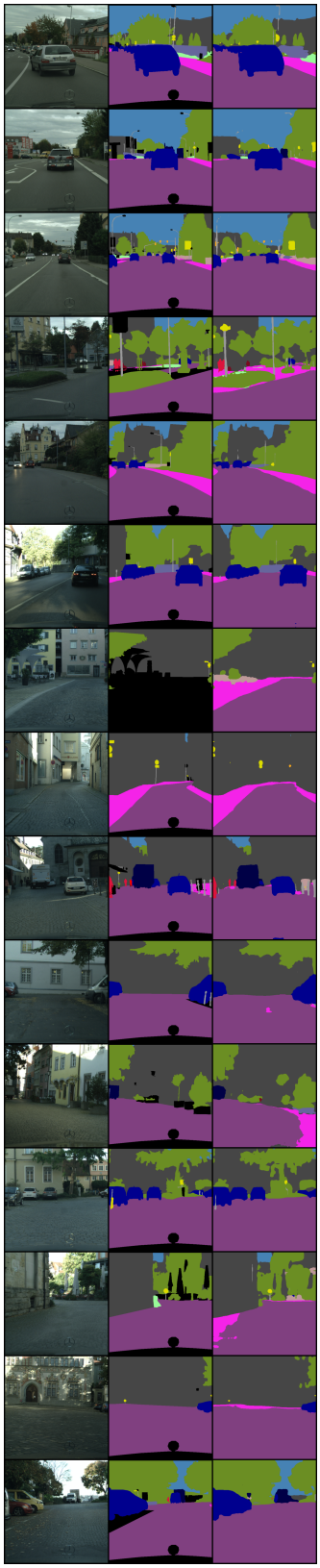


Figure 8: Segmentation maps obtained using PSPNet. (a) Original RGB Image, (b) Ground Truth, (c) Segmentation Output