CMPE-685 COMPUTER VISION PROJECT REPORT: SEMANTIC SEGMENTATION

Humza Syed & Yuri Yakovlev
Department of Computer Engineering
Rochester Institute of Technology
Rochester, NY 14623
hxs7174@rit.edu

April 29, 2019

1 Introduction

Semantic segmentation is a method in which object classes are segmented precisely in an image to gain an understanding of what is within an image. In contrast to to other computer vision techniques, such as image classification or creating bounding boxes around detected objects, semantic segmentation allows for pixel-wise precision. This can be used for applications such as medical diagnosis, autonomous driving, and robotic vision. For this project, the main goal consists of detecting labeled object classes in an image and creating a pixel-wise segmentation that separates the labeled object classes from the background and from each other. Two objects of the same class may not be distinctly separated if they are close together as they would be in instance segmentation, but the classes themselves should be clearly separated. Other goals consist of improving understanding of this area and achieving an intersection over union, a metric discussed later for determining the accuracy of a semantic segmentation result, of over 0.25 on each dataset for the models.

2 Related Work

Previous works have shown that advances in image classification and object detection have grown in recent years. Architectures such as ResNet [1] and R-CNN [2], have allowed for implementations of convolutional neural networks (CNNs) that achieve state-of-the-art accuracy in both of these tasks.

A highly efficient version of R-CNN, in terms of execution time, is Faster R-CNN. A diagram of Faster R-CNN is shown in Figure 1. The purpose of R-CNN is to have a CNN detect objects with a bounding box indicating where that image was detected. In the Faster R-CNN architecture an image is first run through a CNN to create feature maps. Then, a Region Proposal Network (RPN) uses the feature maps to propose regions of interest. This network needs to be trained such that it can propose regions where there is a chance that an object exists. Without a neural network doing this, region proposal must either be arbitrary or done through a separate region proposal algorithm. Once some proposed regions are found, the regions are analyzed to try to find an object. The region in which the object was found becomes the bounding box indicating the position of the object.

Faster R-CNN was successful in detecting objects and their positions roughly via a bounding box. However, this project explores the usage of semantic segmentation as it's a more precise method of detecting object classes down to the pixel level. Semantic segmentation has been performed using CNNs that contain an encoding-decoding structure. The encoder portion of the network is a conventional CNN, in which an input image is downsampled across layers to gain various features, while the decoder portion consists of deconvolutional layers to upsample an image relative to the features extracted. This methodology was initially performed in Fully Convolutional Networks (FCNs) [4], in which the output of specific pooling layers in the network were upsampled to create pixel-wise segments of the input image. This allowed for the network to learn pixel-wise accuracy along with achieving a high intersection over union compared to conventional computer vision techniques. Improving upon FCNs, multiple networks have been developed including SegNet [5], U-Net [6], PSPNet [7], etc. With each network, the overall performance of semantic segmentation tasks has improved over time.

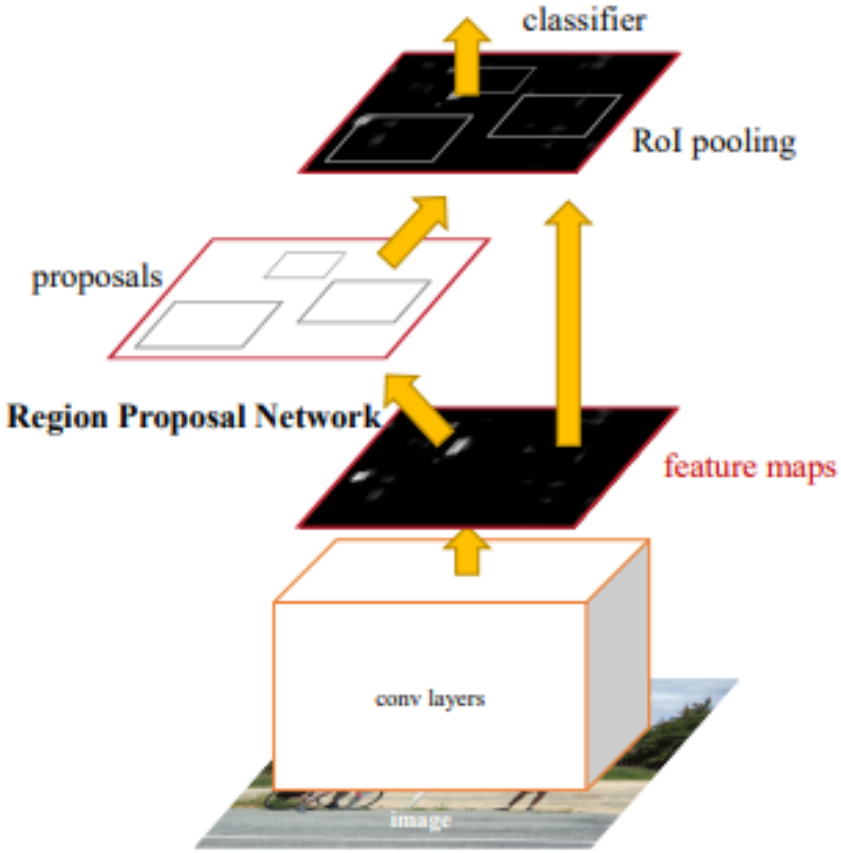


Figure 1: Faster R-CNN diagram [3]

Normally a CNN requires that an image input is set to a specific size to ensure that the classification layer at the end of the architecture consists of a vector containing just the class scores. However, this leads to a bottleneck if you have images of arbitrary size as downsizing these images to a specific size or cropping them leads to loss in features. This is true especially in the case of semantic segmentation as classes can be lost during cropping or resizing. In [4] the authors introduce FCNs that enable an end-to-end system that performs semantic segmentation on arbitrary sized images. Typically a CNN consists of convolutional layers as well as fully connected layers. Additional pooling and normalization layers may be added to the architecture to enable for spatial reduction in feature map sizes and to transform outputs of each layer to a normalized range. However, in these networks, the fully connected layers can be represented as convolutional layers as illustrated in Figure 2. For example, in this figure, a feature map output of $5 \times 5 \times 256$ from a convolutional layer can be fed to a fully connected layer of 4096 neurons to receive an output size of 1×4096. Alternatively, the feature map output can be convolved with 4096 filters of kernel size $5 \times 5 \times 256$ to receive an output feature map of size $1 \times 1 \times 4096$. This was the intuition behind creating a fully convolutional network as arbitrary sized images could then be sent through the network to receive output feature maps that could be $x \times x \times y$ where x would be the feature map width and height and y would be equivalent to the original number of fully connected neurons. A softmax would then occur on this output to receive the classification as well as pixel localization of the objects in the image. Additional details on the FCN are discussed in Section 3.

3 Methods

Two network architectures were used for this project: FCN [4] and SegNet [5]. FCN was one of the first networks used in semantic segmentation that yielded solid results. An illustration of the FCN network can be seen in Figure 3. The architecture consists of an initial CNN that has its fully connected layers changed to convolutional layers. In this work the convolutional layer structure of the VGG16 network is utilized. Deconvolution layers are then used to upscale and create the segmentation from various portions of the network. More specifically, skipped connections are

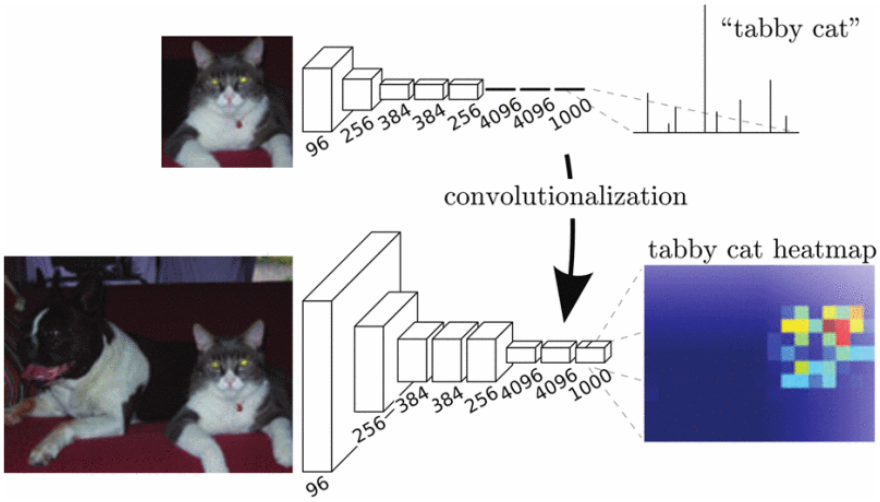


Figure 2: Example of how fully connected layers can be represented as convolutional layers for pixel extraction of an image [4].

employed from the output of pooling layers and upscaled to create multiple segmentation masks. The FCN32 network uses only the final output layer convolution and upscales this, while FCN16 takes an intermediate pooling layer as well as the final output layer convolution, upscales them, and then concatenates their feature maps. Lastly, FCN8 makes use of two intermediate pooling layer outputs as well as the final output layer convolution, upscales all layers, and then concatenates them. In this work, skipped connections from each pooling layer is upscaled and concatenated as an output model result of FCN.

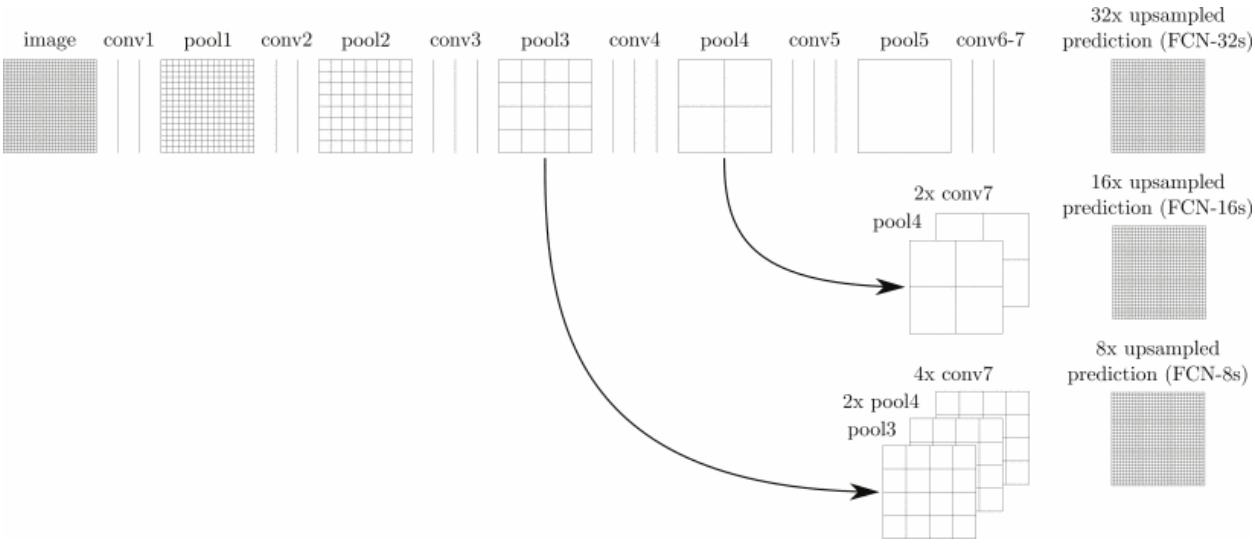


Figure 3: Illustration of the FCN network and its skipped connections [4].

SegNet uses an architecture that consists of an encoder half and a decoder half. The two halves are mirrors of each other. Figure 4 shows an overview of the SegNet architecture. The encoder half, like in FCN, is based on VGG16. The difference between the networks is in how they perform upscaling as highlighted in Figure 5. Rather than using deconvolution, SegNet saves max-pooling indices during the downscaling of the network and uses these indices for upsampling. This in turn leads to sparse feature maps from each of the individual pooling layers of the encoder. The decoder then performs convolution and batch normalization on to create a feature map in the decoder pass. The feature map generated by a single decoder layer (upsampling + convolution + batch normalization), is passed to the next decoder layer to then combine with its pooling layer input. After all the decoding operations are done, a softmax is taken to produce the final segmentation output.

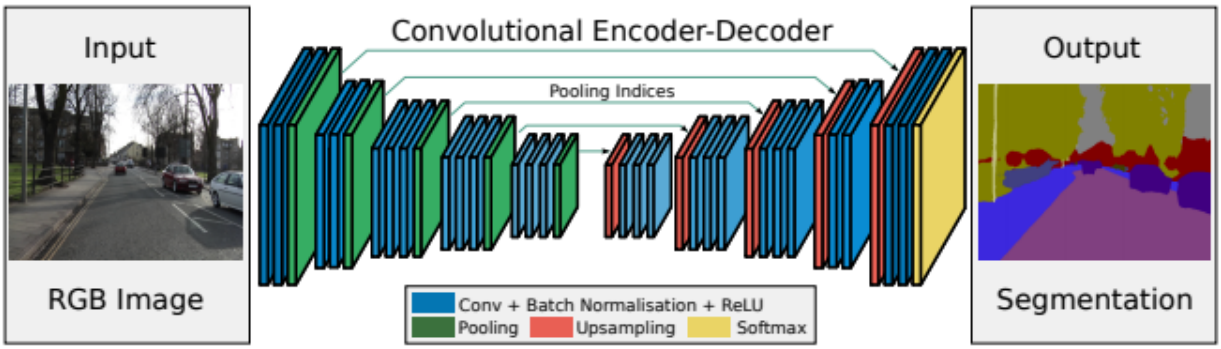


Figure 4: Formulation of the SegNet architecture's end-to-end system [5].

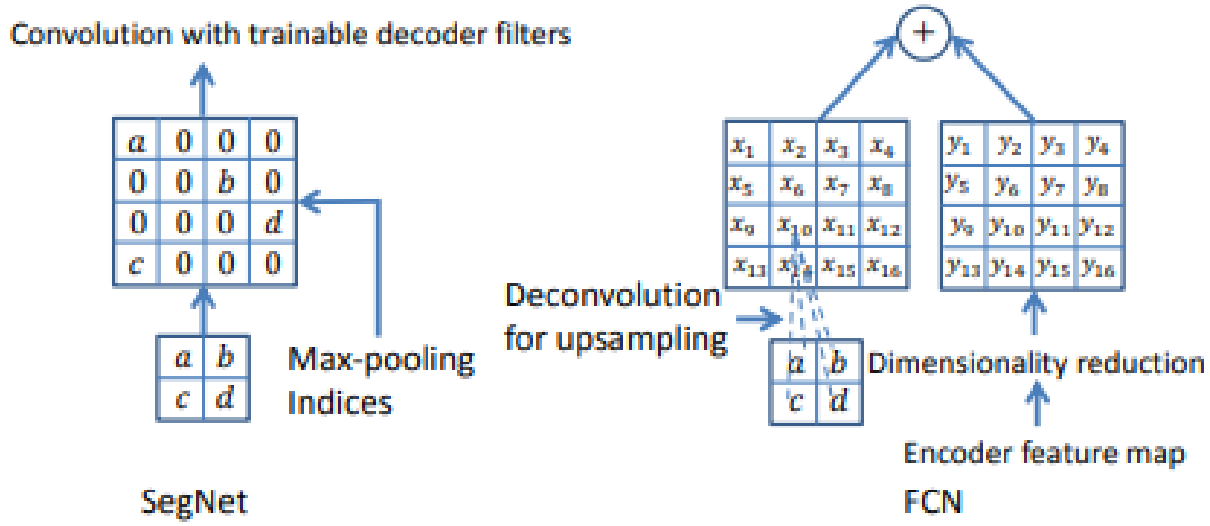


Figure 5: Differences between upscaling methodologies used in SegNet and FCN [5].

4 Experiments

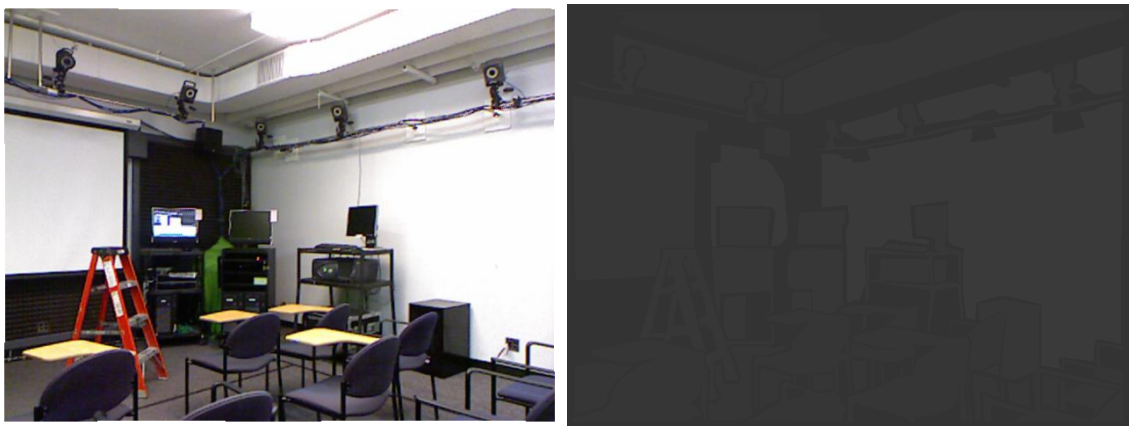
The PyTorch framework [8] was utilized to for all experiments to create the architectures and to train and test them. PyCharm was used for development, and GitLab was used for version control. Training and testing of the networks was done using two datasets: Cityscapes[9, 10] and NYU Depth Dataset V2 (NYUv2) [11].

The Cityscapes dataset consists of 5,000 finely annotated images of street scenes and 30 classes, including road, sidewalk, person, etc. The images in the dataset were all captured on a dash cam of a car driving through some cities, so the images are all of the same size. A coarsely labeled Cityscapes dataset also exists, but only the finely labeled dataset was used for the experiments in this project. Of the 5,000 images, 2,975 images are used for training, 500 images are used for validation, and lastly 1,525 images are used for testing. Additionally, of the 30 classes, 11 are set to void, such as bridge, tunnel, parking, rail track, etc. These classes aren't as relevant as person, road, sidewalk, etc. and therefore are commonly left out to leave 19 classes in total. The original image sizes are of width 1024 and height of 2048. These were downscaled to a width of 256 and a height of 512 before sending them through the network to speed up training time and reduce GPU memory usage. Figure 6 shows an example of a segmentation label image from Cityscapes.

The NYUv2 dataset consists of 1,449 RGB with 14 classes. All of the images were taken on the NYU campus, and the default split is 795 images for training 654 images for testing. This is quite close to a 50/50 split, so it may not actually be ideal for training. Additionally, the dataset isn't nearly as large as Cityscapes, so it's likely that the results would not be as accurate with the same training parameters. All images are 640x480 and were not resized during training. An

Figure 6: Cityscapes segmentation example [9, 10].

example of an image and its segmentation label for this dataset is shown in Figure 7. The contrast for the label image was increased to make the segmentation more visible as the actual image is much darker.



(a) NYUv2 image

(b) NYUv2 segmentation

Figure 7: NYUv2 segmentation example [11]

For both datasets training was conducted using stochastic gradient descent with a batch size of 1, a learning rate of 0.001, a momentum of 0.9, and a weight decay of 0.0016 are used. Cross entropy 2D loss was used as the loss function to tune the weights of the networks. Both architectures are trained for 100 epochs and have their output weights saved to be tested on.

To evaluate the accuracy of the segmentation results, a number of metrics were employed. The first metric was pixel accuracy, which reports the percentage of pixels that were classified correctly in an image across all classes. The second metric was the mean pixel accuracy, which takes the mean accuracy across all classes. The calculations for these two metrics can be seen in Equations 1, 2, and 3. In these equations n_{ii} is the number of pixels of class i predicted by class i, t_i is the total number of pixels of class i, and lastly n_{cl} is the number of classes.

$$t_i = \sum_j n_{ij} \tag{1}$$

$$pixel\ accuracy = \sum_i n_{ii} / \sum_i t_i \tag{2}$$

$$mean\ pixel\ accuracy = (1/n_{cl}) \sum_i n_{ii}/t_i \tag{3}$$

The third metric was the mean intersection over union (IoU) across all classes. Figure 8 shows an illustration of how IoU is computed. This metric takes the known annotations and the results of the segmentation and computes both the intersections and the unions of the classes. The area of intersection is divided by the area of the union to produce the metric. A result that lines up perfectly would get an IoU of 1.

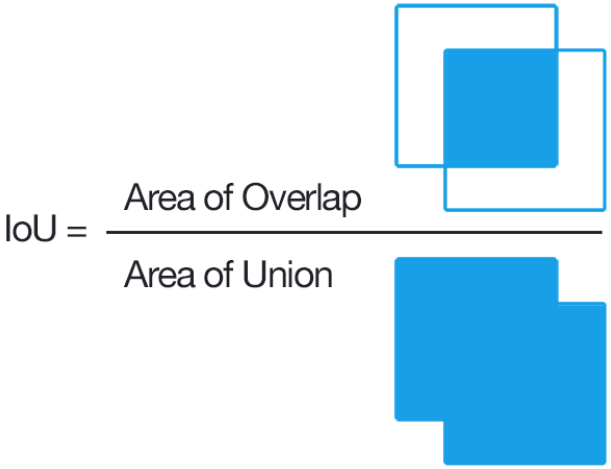


Figure 8: Visualization of IoU Computation [12].

With these datasets and architectures, the anticipated result was an IoU of around 0.25 or greater after training for 100 epochs. Training on Cityscapes was run on an Nvida GeForce RTX 2080 for both the SegNet and FCN architectures since it is the much larger dataset. The NYUv2 dataset training was run on an Nvidia GeForce GTX 970.

5 Results

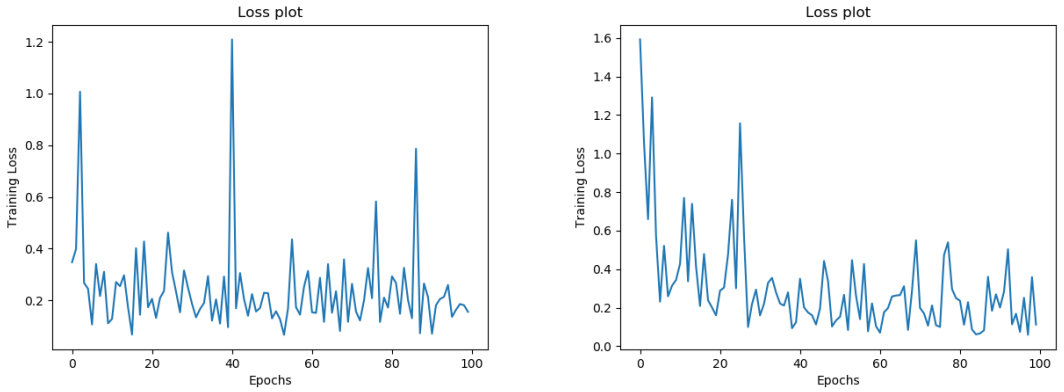
The results from running both SegNet and FCN on both datasets can be seen in Table 1. The results shown are the results that produced the highest IoU throughout training, rather than the last epoch. From the results it can be seen that FCN outperforms SegNet on both the Cityscapes and NYUDv2 datasets. This is most likely due to how the FCN model has each of its pooling layers upsampled and concatenated. This leads to a more precise mask to obtain a greater IoU. SegNet on the other hand only upsamples from 1 output all the way to the original image. FCN also makes use of deconvolution operations to obtain more features in comparison to SegNet's use of max-pooling indices to obtain sparse feature maps. However, overall SegNet's performance was not substantially lower as it still was able to obtain an IoU close to FCN. Additionally in both cases the pixel accuracies and mean pixel accuracies were fairly close. The goal of an IoU of over 0.25 was achieved using both models on the Cityscapes dataset but this wasn't the case for the NYUDv2 dataset. In this case SegNet reported an IoU of 0.2281, while FCN reported 0.3618. This is most likely due to the small number of training samples in the dataset in comparison to Cityscapes. Additionally, a greater IoU may have been obtained using a reduced learning rate and additional epochs.

To further assess the training of each network, the loss for both datasets was recorded. Plots of the loss over time for Cityscapes can be seen in Figure 9. It can be seen here that the loss for Cityscapes initially dips low but then continues to oscillate by a fair amount over time. High increases in loss can be seen for both models. This means that a learning rate scheduler or reduced learning rate with additional epochs of training could have led to a trained model that performed well. Similarly to the Cityscapes loss plots, the loss plots for the NYUDv2 dataset can be seen in Figure 10. Behavior similar to Cityscapes loss can be seen as there is an initial decrease in loss then oscillation occurs. Once again

Table 1: Results using each model on each dataset.

Dataset	Model	Pixel Accuracy (%)	Mean Pixel Accuracy (%)	IoU (%)
Cityscapes	FCN	91.58	56.92	49.68
Cityscapes	SegNet	90.16	48.59	40.68
NYUDv2	FCN	59.77	48.64	36.18
NYUDv2	SegNet	50.15	33.17	22.81

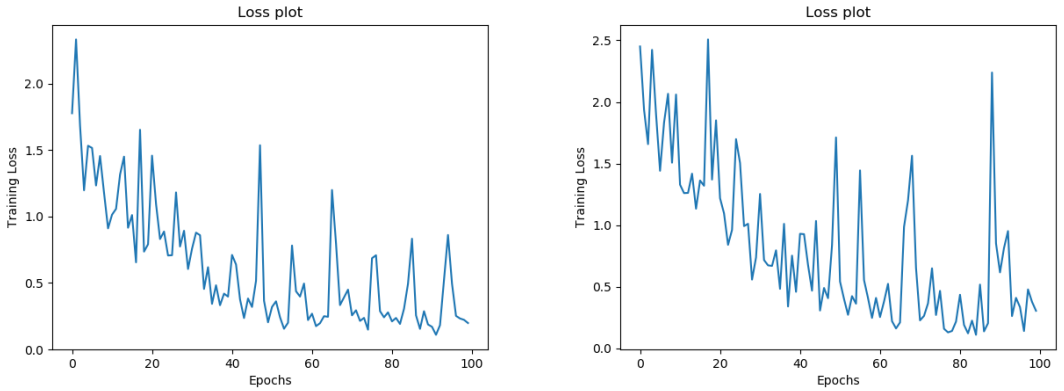
it is reiterated that a reduced learning rate or learning rate scheduler would have enabled for more effective training of the models.



(a) Plot of FCN loss on the Cityscapes dataset.

(b) Plot of SegNet loss on the Cityscapes dataset.

Figure 9: Plot of loss for both models on the Cityscapes dataset.



(a) Plot of FCN loss on the NYUv2 dataset.

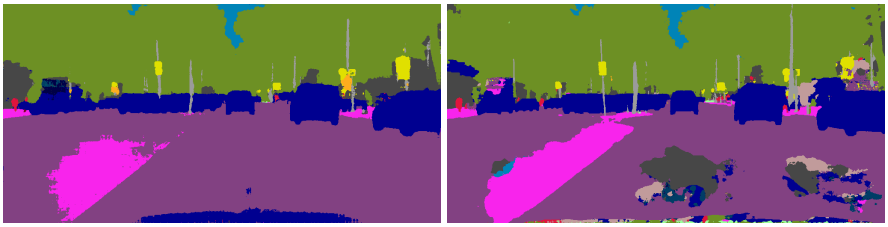
(b) Plot of SegNet loss on the NYUv2 dataset.

Figure 10: Plot of loss for both models on the NYUv2 dataset.

To finalize the results semantic segmented images are shown using example images from both datasets. A test image from the Cityscapes dataset can be seen in Figure 11. The segmentations of this image using both models can be seen in Figure 12. It can be seen that the cars in the image were sucessfully segmented in both cases, where the FCN performs a better job by also segmenting the hood of the car that is being driven in. SegNet seems to be having problems with segmenting part of the road while FCN showed a much smoother segmentation. Similarly, a test image from the NYUv2 dataset can be seen in Figure 13. The segmentations of this image can be seen in Figure 14. These segmentations show a vastly different output from the Cityscapes segmentations. The objects in the image are slightly segmented but, as

noted before, a small IoU was obtained for this dataset. The segmentation further highlights this as few of the classes were entirely segmented out.

Figure 11: Example test image from the Cityscapes dataset [9, 10].



(a) Semantic segmentation using the FCN model on the Cityscapes image.

(b) Semantic segmentation using the SegNet model on the Cityscapes image.

Figure 12: Semantic segmentation using both models on a Cityscapes image.

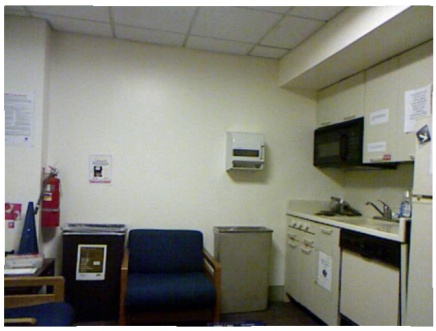
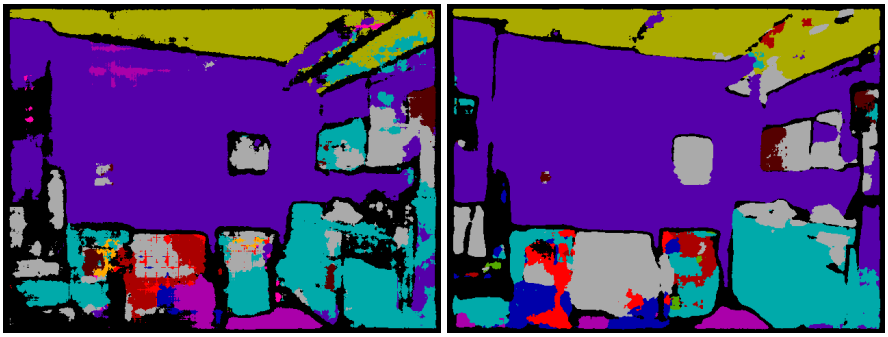


Figure 13: Example test image from the NYUv2 dataset [11].

6 Discussion

The Cityscapes dataset yielded high IoUs for both the FCN and SegNet architectures. After training for 100 epochs, the best IoU produced by the FCN architecture was 49.68%. SegNet had a lower, but still reasonable IoU of 40.68%. The dataset had preferable splits for training, validation and testing, as well as many images overall in comparison to the NYUv2 dataset. This in turn led to a higher IoU for both models over NYUv2. With more training time and a reduction in learning rate, both models would show more impressive results. However, with the given learning rate and number of training epochs, a fairly high IoU was still obtained.

For the NYUv2 dataset, the FCN architecture met the desired goal with an IoU of 36.17%. SegNet, however, did not perform as well and the IoU was only 22.81%. Results for both architectures definitely can be improved. The loss started oscillating rather early in training. It's likely that, with better scheduling of a decaying learning rate, the networks would learn more gradually toward the end and produce better results. Additionally, the train/test split was likely suboptimal. As mentioned before, the train/test split for NYUv2 is close to 50/50. Moving some of the images from the test test to the training set could yield much better results. Perhaps a split close to 90/10 or 80/20 would be best. Of course, more data would also be very beneficial as the dataset only included 1,449 images, so it's no surprise that the overall performance after 100 epochs of training was not as good as it was with the Cityscapes dataset.



(a) Semantic segmentation using the FCN model on the NYUv2 image.

(b) Semantic segmentation using the SegNet model on the NYUv2 image.

Figure 14: Semantic segmentation using both models on a NYUv2 image.

Across both datasets, the FCN model performed much better in comparison to SegNet. SegNet performs a different upscaling procedure compared to the FCN's deconvolution layers. It's likely that SegNet requires drastically different hyperparameters compared to FCN for proper training. The main limitation of these experiments was the few number of training epochs as well as a fairly high learning rate. In future work, additional experimentation to fine tune the models hyperparameters can potentially lead to more finely trained models with increased performance.

7 Teamwork

For this project portions were allocated to each team member to work on. The work split was roughly 60 to 40, for Humza Syed and Yuri Yakovlev respectively, and can be seen in Table 2.

Table 2: Teamwork during project completion.

Humza Syed	Yuri Yakovlev
Created models	Loaded in data
Created training and test	Created metrics to use
Created logging information and stored models	Refactored code
Cityscapes dataset runs	NYUv2 dataset runs

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[4] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, June 2015.

[5] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[7] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. *CoRR*, abs/1612.01105, 2016.

[8] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[9] 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.

[10] https://www.cityscapes-dataset.com/.

[11] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[12] https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.