Enhancing Semantic Segmentation with Depth Information: A Comparison of Network Architectures on the NYUv2 Dataset

Jacek Dlugosz
University of Michigan
dlugoszj@umich.edu

Franklin Volcic
University of Michigan
fvolcic@umich.edu

Broderick Riopelle
University of Michigan
broderio@umich.edu

Dhillon Patel
University of Michigan
dhipatel@umich.edu

1. Introduction

Image segmentation is the process by which the pixels of an image are partitioned into multiple segments based on shared characteristics that can include color, texture, and intensity. The goal of semantic image segmentation is to predict a class for every pixel in an image. Semantic image segmentation has important applications in fields such as medical imaging [13] and autonomous vehicles [4].

In this paper, we investigate the use of depth information in semantic image segmentation. We evaluate the performance of two network architectures, Efficient Network (ENet) and UNet, on the task of semantic image segmentation using both RGB and RGB-D images. Our experiments were conducted using the NYUv2 dataset. A combination of Dice Loss and Cross Entropy Loss were used to stabilize the losses. Our results demonstrate the benefits of incorporating depth information in improving the accuracy of semantic image segmentation.

2. Related Work

There are various approaches to semantic segmentation, including FCN [12], UNet [3] [10], ResNet [6], and SegNet [2]. Convolutional neural networks (CNNs) can struggle with loss of spatial information during downsampling, but techniques such as concatenating feature maps and implementing max pooling layers can help preserve this information [**?**]. FCN, UNet, and SegNet all utilize these techniques. Another common issue with CNNs is vanishing gradients, which can prevent parameters from being learned. ResNet addresses this issue by including skip connections, allowing the gradient to propagate through deep layers without loss of accuracy [6] [**?**].

Like the networks mentioned prior, Efficient Net (ENet) is a type of CNN that is designed to be both accurate and efficient [9]. It achieves this by using a combination of specific bottleneck building blocks and scaling up the network in a way that balances accuracy and efficiency. Some of the advantages of ENet include its ability to achieve high accuracy while using fewer parameters and requiring less computation compared to other networks [9]. This makes it a good choice for deployment in resource-constrained environments, such as on mobile devices or in edge computing applications.

The accuracy of the segmentation can be improved by supplementing the 2D RGB channels of an image with information about its 3D environment. This information can provide context about the relative distances and positions of objects in the scene, which can be useful for distinguishing between objects. Often, this information is represented as an additional depth channel (RGB-D), point clouds, projected images, voxels, or mesh [7].

3. Methods

We implemented a majority of the code for the ENet, but adapted some pieces written by Yassine Ouali (yassouali on GitHub) [14]. The main idea behind ResNet that made it popular was the introduction of skip connections to each residual block or bottleneck [6]. If a certain bottleneck decreased the network's performance, the skip connection could be used to ensure that the performance was not affected.

An example of a ResNet bottleneck can be seen in Figure 1. The ResNet bottleneck architecture begins with a 1x1 convolution called a projection, followed by a 3x3 convolution and another 1x1 convolution called an expansion. In the projection, the number of channels is decreased by an arbitrary factor, and then in the expansion, the number of channels is increased back to the original amount. This increases the depth of the network while limiting the number of parameters it has. An ENet bottleneck is similar to a

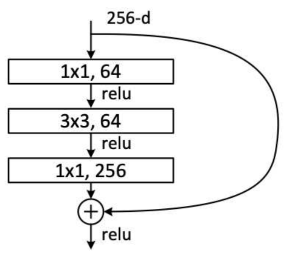

Figure 1. Example of ResNet Bottleneck [6]

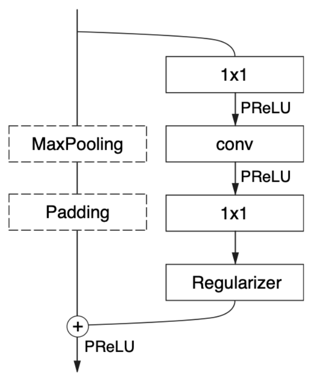

Figure 2. Example of ENet Bottleneck [9]

ResNet bottleneck, but with some key differences, as shown in Figure 2.

The ENet architecture consists of seven stages, as shown in Figure 3. There are several types of bottlenecks, which can be divided into two groups: those that involve downsampling and upsampling, and those that involve dilated and asymmetric bottlenecks. For downsampling bottlenecks, max pooling and padding are added to the skip connection, and the projection convolution is changed to 2x2. For upsampling bottlenecks, a spatial convolution and a max unpooling are added to the skip connection. For dilated bottlenecks, the dilated and padding parameters in the 3x3 convolution are set to the respective values. Finally, for asymmetric bottlenecks, the 3x3 convolution is replaced with two convolutions: one 1x5 and one 5x1. This simulates a 5x5 convolution, but with fewer parameters and increased computational speed. It's also worth noting that after each convolution, there is a batch normalization and an activation function (PReLU in this case) [9].

To compare the results of our ENet implementation, we decided to implement a simple UNet as a baseline. The UNet we used was based on the architecture shown in Figure 4, with some modifications from what we did in class for PS6. The main difference between the UNet we used and the one from PS6 is that we replaced the batch normalization after each layer with max pooling, as shown in the architecture in Figure 4 [10].

We trained our models using a combination of Dice loss and cross entropy loss. Dice loss is a loss function specifically designed for semantic segmentation, as it helps compare the output mask with the ground truth mask. As seen in Equation 1, the numerator of the dice coefficient is 2 times the intersection between the outputted mask and the ground truth mask, and the denominator is the union of the two. To evaluate the accuracy of our models, we used mean intersection over union (IoU) and mean pixel accuracy. Mean IoU is calculated by dividing the intersection of the model

Name	Type	Output size
initial		$16 \times 256 \times 256$
bottleneck1.0	downsampling	$64 \times 128 \times 128$
4× 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		$C \times 512 \times 512$

Figure 3. ENet Architecture [1]

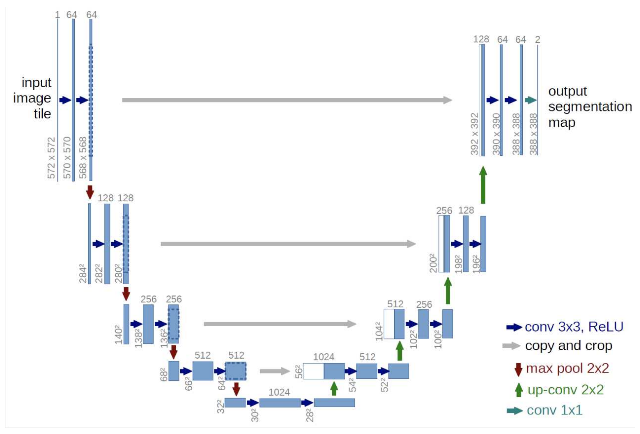

Figure 4. UNet Architecture [10]

$$\text{Dice Loss} = 1 - \frac{2 * |A \cap B|}{|A| + |B|} \tag{1}$$

1. Dice Loss using Dice Coefficient between sets A and B [1]

output and ground truth by the union of the two. On the other hand, mean pixel accuracy calculates the number of pixels predicted correctly compared to the total number of pixels in the image.

4. Experiments

The goal of this project was to compare the impact of depth data on semantic segmentation accuracy. Specifically, we wanted to determine whether adding depth data to RGB images would improve the mean IoU and mean pixel accuracy of the networks when compared to networks trained without depth information. Our hypothesis was that the inclusion of depth data would lead to significant improvements. To test this hypothesis, we conducted experiments using the NYUv2 dataset, which consists of RGB images with optional depth data and segmented versions of the images. An example of the NYUv2 dataset is shown in Figure 5. To use the NYUv2 dataset in our experiments, we adapted a dataloader written by Mihai Suteu (xapharius on GitHub) [11] to load the images into memory.

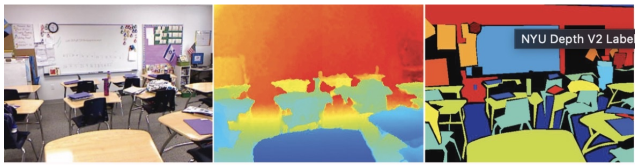

Figure 5. Example of a NYU data set image with RGB (left), Depth (Center), and Labeled (Right) [8]

Each network was trained twice: once using only RGB images and once using RGB and depth data. To increase the amount of available training data, we applied random flipping as a data augmentation during training. For the loss function, we used a combination of cross-entropy loss and dice loss, denoted as Loss $=$ CE Loss $+$ Dice Loss. We found that this combination resulted in the fastest and most accurate training of the networks. As the optimizer, we chose stochastic gradient descent with momentum, setting the initial learning rate to 0.1 and the momentum to 0.9. We also employed an automatic learning rate decay function provided by PyTorch, namely `lr_scheduler.ReduceLROnPlateau`, to decay the learning rate once a learning plateau was reached. In total, each network was trained for 400 epochs on the NYUv2 training set. Figure 7 shows the output of each model when given the RGB image in Figure 6 as input.

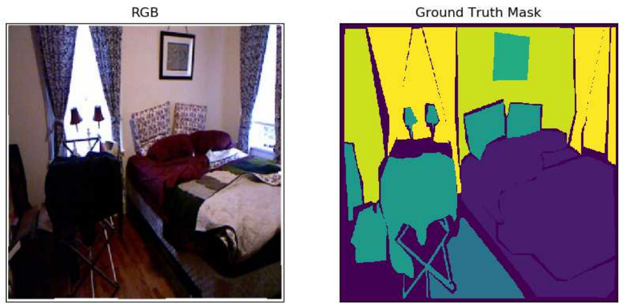

Figure 6. Input image and the ground truth

After our models were trained and produced the output shown in Figure 7, we ran some statistical analyses to assess their accuracy. Mean intersection over union (mIoU) and mean pixel accuracy (MPA) are common metrics for evaluating the performance of a semantic segmentation model.

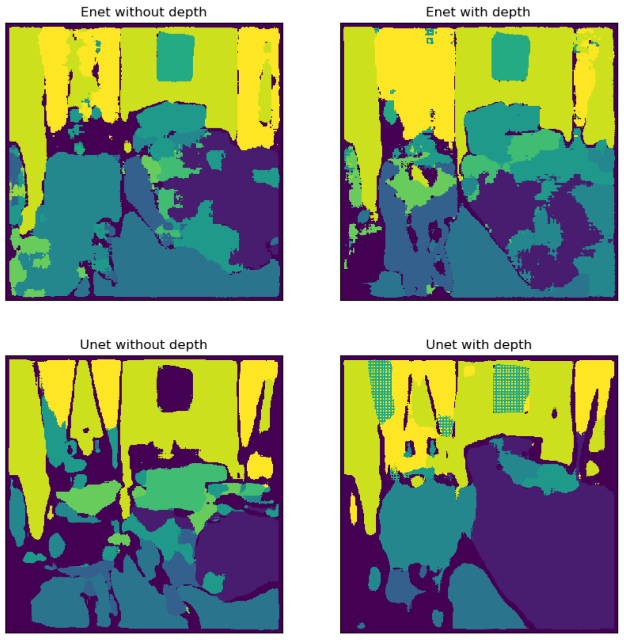

Figure 7. Output segmentation mask from the input shown in Figure 6 by model

Model	**mIoU**	**mpa**
ENet	30.57%	53.86%
Depth ENet	32.38%	58.33%
UNUNet	31.69%	50.64%
Depth UNet	48.80%	57.57%

Table 1. mean IoU and mean pixel accuracy (MPA)

We calculated the mIoU and MPA for each example in the test set and present the results in Table 1.

Both ENet and UNet demonstrated improvements in semantic segmentation accuracy. ENet showed a smaller improvement of approximately 2% in mIoU and 5% in MPA, while UNet saw a more substantial improvement of 17% in mIoU and 7% in MPA. Interestingly, our simple UNet architecture paired with depth data achieved an mIoU of 48.80%, which is comparable to many leading models trained and tested on the NYUv2 dataset [5]. Overall, our results indicate that depth information is a crucial factor in semantic segmentation and has the potential to significantly improve segmentation outcomes.

5. Conclusion

Our research has demonstrated that the inclusion of a depth channel in RGB images improves the performance of semantic segmentation models. We evaluated four different architectures, ENet, Depth ENET, UNet, and Depth UNet, on the NYUv2 dataset and found that the models with a depth channel outperformed their counterparts without a depth channel in terms of both mean intersection-over-union (mIoU) and mean pixel accuracy (MPA). Our findings highlight the importance of considering depth information in visual representations for semantic segmentation tasks.

References

[1] Amine. The difference between dice and dice loss, Sep 2021. 2

[2] 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. 1

[3] Bhakti Baheti, Shubham Innani, Suhas Gajre, and Sanjay Talbar. Eff-unet: A novel architecture for semantic segmentation in unstructured environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 358–359, 2020. 1

[4] Guo Cheng and Jiang Yu Zheng. Semantic segmentation for pedestrian detection from motion in temporal domain. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 6897–6903, 2021. 1

[5] Papers With Code. Semantic segmentation on nyu depth v2. https://paperswithcode.com/sota/semantic-segmentation-on-nyu-depth-v2, 2022. Accessed: 2022-12-10. 3

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 1, 2

[7] Yong He, Hongshan Yu, Xiaoyan Liu, Zhengeng Yang, Wei Sun, Yaonan Wang, Qiang Fu, Yanmei Zou, and Ajmal Mian. Deep learning based 3d segmentation: A survey. *arXiv preprint arXiv:2103.05423*, 2021. 1

[8] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 3

[9] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation, 2016. 1, 2

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 1, 2

[11] Mihai Suteu. Pytorch nyuv2 dataset class. https://github.com/xapharius/pytorch-nyuv2, 2022. Accessed: 2022-12-10. 3

[12] Yongfeng Xing, Luo Zhong, and Xian Zhong. An encoder-decoder network based fcn architecture for semantic segmentation. *Wireless Communications and Mobile Computing*, 2020, 2020. 1

[13] Ruixin Yang and Yingyan Yu. Artificial convolutional neural network in object detection and semantic segmentation for medical imaging analysis. *Frontiers in oncology*, 11:638182, 2021. 1

[14] yassouali. Pytorch segmentation models. https://github.com/yassouali/pytorch-segmentation/blob/master/models/enet.py, 2022. Accessed: 2022-12-10. 1