SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation

Goutham Mittadhoddi GM7PS@VIRGINIA.EDU
Ishan Mathur ANM3HH@VIRGINIA.EDU
Shubhankar Poundrik UFH6FT@VIRGINIA.EDU
Vishakha Gohel UBK8EY@VIRGINIA.EDU

Abstract

This paper presents a deep-learning approach for semantic image segmentation. SegNet is a deep learning architecture for image segmentation developed by researchers at the University of Cambridge. It is a convolutional encoder-decoder network that uses a series of convolutional and max pooling layers in the encoder part of the network to extract image features and transposed convolutional layers in the decoder part of the network to upsample the feature maps and generate the segmented output. SegNet is known for its efficiency and ability to learn rich feature representations from images, which makes it well-suited for applications such as object recognition and scene understanding. The authors evaluate the performance of SegNet on several benchmark datasets, showing that it achieves superior results compared to previous state-of-the-art methods while using fewer resources.

Keywords: Segnet, Convolutional Neural Network, Encoder-Decoder, Image Segmentation.

1. Introduction (Shubhankar P, Goutham M, Ishan M, Vishakha G)

The authors address the problem of semantic image segmentation. This is a challenging task in computer vision, where the goal is to assign a class label to each pixel in an image, enabling the system to identify and segment objects and regions in the scene. The problem is difficult because the segmentation must be performed at a high resolution, and the labels are often complex and overlapping. Traditional approaches to image segmentation rely on hand-crafted features and rules, which are difficult to design and often lack generalization. The authors propose a deep learning approach, using the SegNet architecture, to overcome these limitations and improve the performance of image segmentation.

The authors also mention the difficulty of training deep learning models on large-scale datasets, due to the computational cost and the need for efficient optimization algorithms. These challenges are addressed in the paper by proposing the SegNet architecture, which is designed to handle high-resolution images and complex labels and can be trained efficiently on large datasets.

The best methods for categorizing individual pixels before the invention of deep neural networks tend to rely heavily on human-crafted traits. A patch is commonly run through a classifier, such as Random Forest or Boosting, in order to estimate the class probabilities of the center pixel. The NYU dataset made indoor RGBD pixel-by-pixel semantic segmentation more widely used by illustrating the importance of the depth channel for improving segmentation. Their technique used pixel location, RGB-SIFT, depth-SIFT, and other information as input to a neural network classifier in order to anticipate pixel unaries.

The authors drew inspiration from the unsupervised feature learning architecture put out by Ranzato. The primary learning module from the architecture is a network of encoders and decoders. To create the feature maps, an encoder uses convolution with a filter bank, element-wise tanh non-linearity, max-pooling, and sub-sampling. Using the pooled indices that have been saved, the decoder upsamples the feature maps. Image super-resolution and depth map prediction from a single image are two further applications where pixel-by-pixel predictions are performed using deep networks.

2. Background (Shubhankar P, Goutham M, Ishan M, Vishakha G)

There were many previous attempts at developing algorithms for semantic image segmentation. Some of the most notable include the Fully Convolutional Network (FCN) developed by researchers at UC Berkeley, the Deconvolution Network (DeconvNet) developed by researchers at the University of Toronto, and the Multi-Scale Context Aggregation by Dilated Convolutions (DeepLabv2) developed by researchers at Google.

These algorithms, like SegNet, use convolutional neural networks (CNNs) and encoder-decoder architectures to perform semantic segmentation. However, each one has its own unique approach and set of innovations that differentiate it from the others. SegNet, for example, uses a novel approach to handling large images and retaining spatial information, which sets it apart from other algorithms.

Since its release, SegNet has been widely adopted in the research community and has been used as the basis for many other semantic segmentation algorithms. For example, the DeepLab and PSPNet algorithms, which were introduced in 2016 and 2017 respectively, are based on SegNet and significantly improve upon its performance. Additionally, many other papers have been published that describe applications of SegNet in various domains, such as medical imaging, remote sensing, and robotics.

Overall, SegNet has been a significant contribution to the field of deep learning and has been widely adopted in many different applications. Its success has led to the development of many other algorithms that build upon its architecture and improve upon its performance.

3. Methodology (Shubhankar P, Goutham M, Ishan M, Vishakha G)

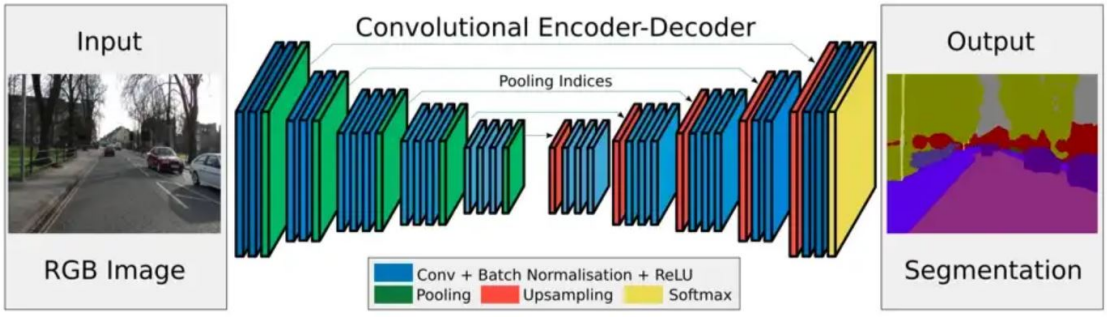

Figure 1: Encoder-Decoder Architechture

SegNet is a deep learning network that uses a convolutional encoder-decoder architecture to perform semantic image segmentation. The network consists of two main parts: an encoder network and a decoder network.

3.1. Encoder

The encoder network is responsible for extracting features from the input image. This is done by applying a series of convolutional filters to the image, followed by a nonlinear activation function such as ReLU. The output of each convolutional layer is then downsampled using a max-pooling operation, which reduces the spatial resolution of the feature maps. This process continues until the encoder network has extracted the most relevant features from the input image.

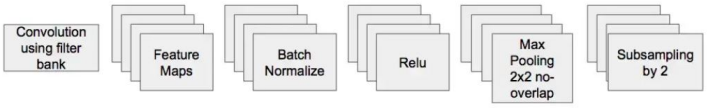

Figure 2: Encoder Layers

3.2. Decoder

The decoder network takes the output of the encoder network and uses it to generate a segmentation map for the input image. This is done by applying a series of transposed convolutional layers to the encoder's output, followed by a nonlinear activation function. The output of the decoder network is a segmentation map with the same spatial resolution as the input image, where each pixel has been assigned a label indicating the class or category to which it belongs.

One key innovation in SegNet is the use of a trainable upsampling layer in the decoder network. This layer uses learned weights to upsample the feature maps from the encoder network, rather than using a fixed interpolation technique. This allows the decoder network to better preserve spatial information and improve the accuracy of the segmentation.

Another innovative feature of SegNet is the use of a max-pooling layer with unpooling in the decoder network. This allows the network to "remember" the locations of the most salient features in the encoder's output, and use them to better align the segmentation map with the input image. This helps to improve the overall accuracy and performance of the network.

4. Experiment (Shubhankar P, Goutham M, Ishan M, Vishakha G)

4.1. Data Preprocessing

The cardiac image dataset we were provided consisted of 3 sets of images. The first was 100 training images with their corresponding correct segmentation mask. The second was 10 testing images with their corresponding correct segmentation mask. The final set was 16 testing images without correct outputs to check against. The images were not of uniform dimensions. We scaled every image to a size of 256x256.

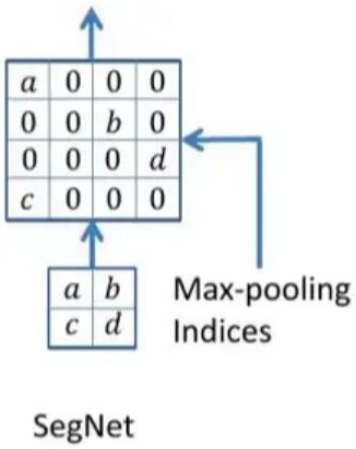

Figure 3: Remembering Indices

The variety among these images and the scarcity of training images pushed us to perform data augmentation to generalize our model. The fact that there were only 100 training images severely limited the range of outputs our model could produce. To mitigate this, we performed rotation and translation operations on the images and their masks to augment the dataset to a total of 5000 training samples (4900 artificial samples). The unlabeled testing images had a different distribution from the training images. We varied the contrast in some of the augmented images to account for this variability and make our model generalize beyond looking at just the brightness and focus more on shapes. Finally, we applied histogram matching to all the datasets.

4.2. Loss selection

We also had to be creative about how we punished the model for an incorrect output. The outputs of the model are mostly black with a single white ring at the segment of the image we are interested in. Due to this, even if the model returns a completely black square, it would still get the majority of the pixels correct.

To mitigate this issue, we use a Weighted Cross Entropy Loss with a higher loss for mistaking a white pixel as black than mistaking a black pixel as white. After running experiments on the model, we determined that the best value for this weight was 25.

4.3. Training

We used the Rivana servers provided by the University with 4 Core vGPUs and 128GB RAM. We trained the model on the augmented 5000 sample dataset for a total of 100 epochs using a batch size of 16. Training the model took less than 1.5 hours. We used an SGD optimizer with a learning rate of 0.01 and momentum of 0.8.

4.4. Testing

We tested the model we had trained on the 10 labeled test images and the 16 unlabeled test images. For the labeled test images we calculated the dice score and for the unlabeled test images, we inspected the results visually.

5. Results (Shubhankar P, Goutham M, Ishan M, Vishakha G)

We achieved a dice score of 0.75 on the labeled test samples. The dice score before training (for random outputs) was about 0.03.

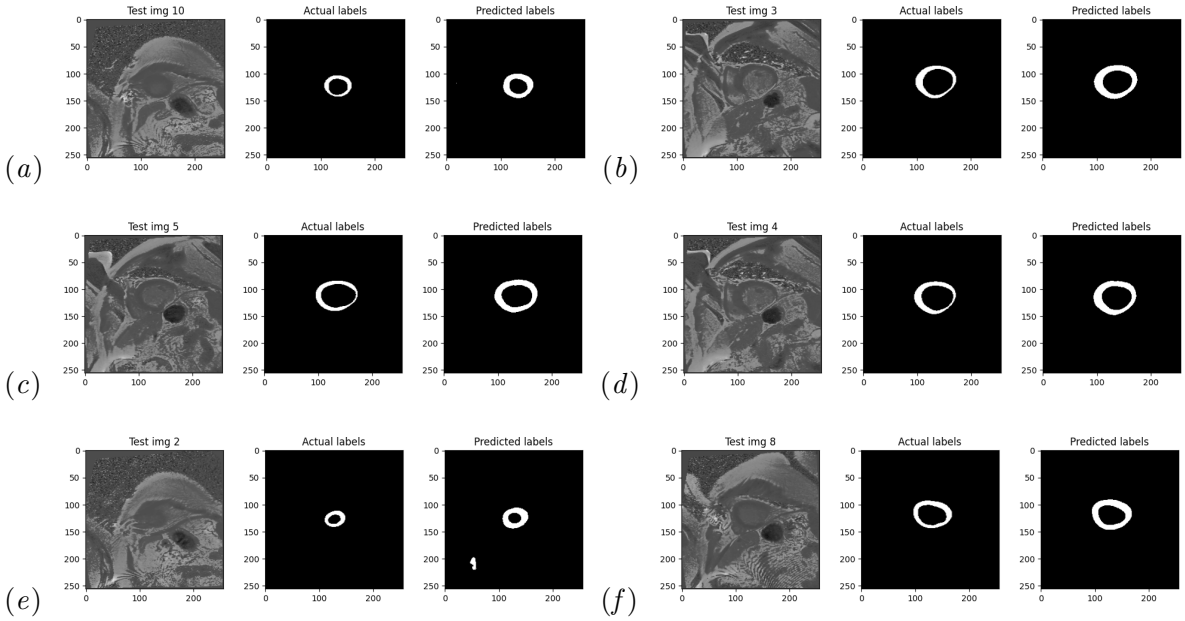

Figure 4: Test images, Actual segmentation, and Predicted segmentation for 6 labeled test images

Test sample	Dice Score
1	0.7659
2	0.5121
3	0.7681
4	0.8191
5	0.7518
6	0.7884
7	0.7767
8	0.8060
9	0.8166
10	0.6899
Mean for all test images	0.7495

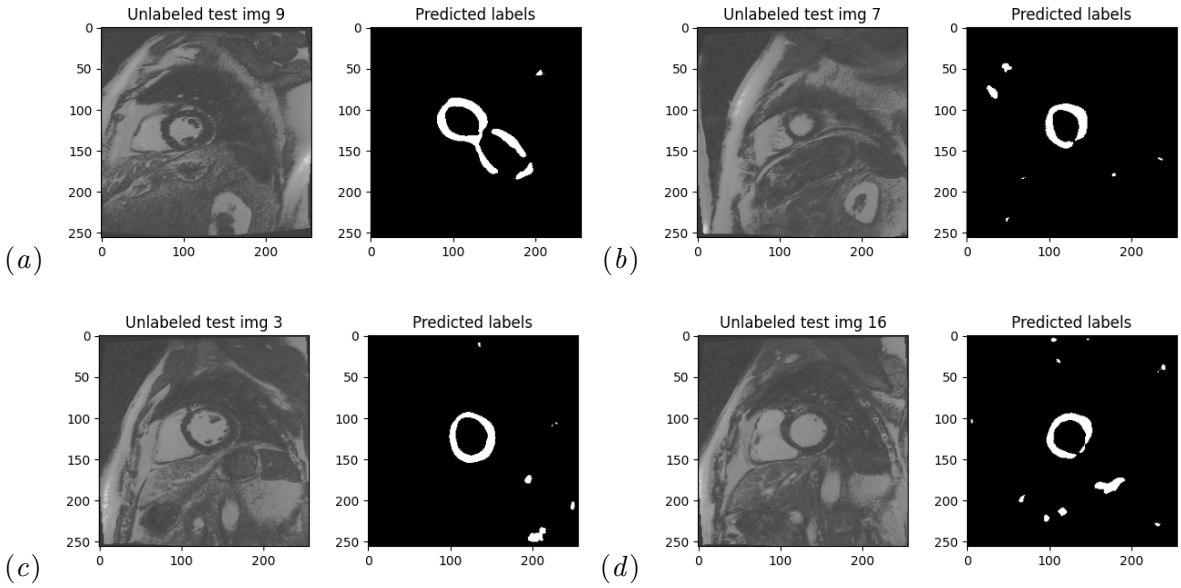

Figure 5: Test images and Predicted segmentation for 4 unlabeled test images

6. Discussion (Shubhankar P, Goutham M, Ishan M, Vishakha G)

We learnt a lot from implementing this project. The concept was simple and we used few resource available online to complete the implementation. We faced some challenges in the process. For example, in order to extend this exercise it would benefit us to have a larger and more varied training dataset to build a more generalized model. Another thing we noticed was that the training and test with label images were more similar when compared to the test images without label, which resulted in less than expected accuracy. Our results would also benefit from more hyper-parameter tuning, which was difficult in this case due to the computational limitations and the time needed for each training cycle. Apart from this, it would be interesting to see the real-life application of this project and research in different areas of security and autonomous driving.

7. Conclusion (Shubhankar P, Goutham M, Ishan M, Vishakha G)

Overall, our model produces mostly accurate results. The goal of the exercise was to create a proof of concept for the application of an encoder-decoder network in image segmentation using the Segnet model. Since we achieved results with a dice score of around 75%, the goal can be considered reached. We were able to achieve the goal and it opened a lot of different opportunities for us like one in the paper of building this pipeline to work on a video. We can further extend this work to work on live time series data of autonomous vehicle imagery. The possibilities are endless and we are going to keep on making the progress on this.

References

Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla. 2016. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation.

SegNet-A-Deep-Convolutional-Encoder-Decoder-Architecture-for-Image-Segmentation