Semantic Image Segmentation via Fully Convolutional Networks

Matthew Falkenhainer

Introduction

Image segmentation has a wide variety of uses ranging from removing backgrounds to increase classification accuracy to simply wanting to copy and object onto another image. Many algorithms exist that use user generated input to help guide the segmentation process. While these perform well, it is beneficial to create a segmentation with as little user interaction as possible.

My goal with this project is to create an 8 stride fully convolutional network in Caffe based on the one described in the paper Fully Convolutional Networks for Semantic Segmentation[1]. Once a model is trained on this network, the model, given only an image, should be able to successfully segment out any sneakers contained in the image.

I begin first by training a 32 stride, 16 stride and 8 stride network on a dataset of images with white backgrounds, each network using the previous network's weights to speed up the training process. Later, I train an 8 stride network on a dataset of images containing a wide variety of backgrounds.

Data

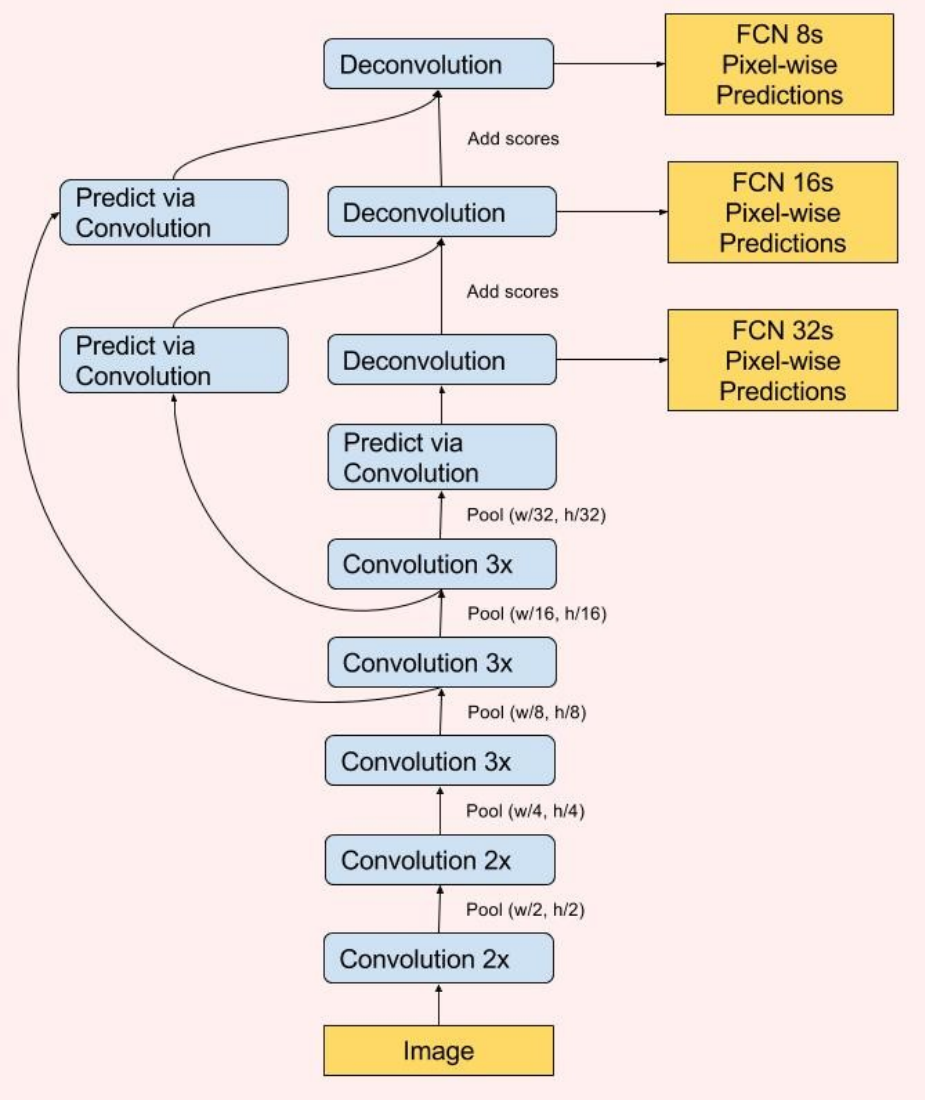
The dataset used to train the networks contains 4,500 sneakers from Amazon on white backgrounds, each with a size of 224x224. By setting the white backgrounds to a value of 0 and anything not within some small range of white to 1, we create a binary mask the network can be trained on.

During the training on background images, these masks were used to paste only the shoes at random scales and positions on random images from the VOC Pascal 2012[2] dataset containing many different scenes. This method of pasting shoes onto images removes the need manually drawing a segmentation line around a shoe in thousands of images. The downside of this method is it can create a bias against shoes in positions that do not normally appear in e-commerce images.

Network

The 32 stride network used is based off the VGG-16[3] network used to classify entire images, replacing the fully connected layers with convolutional layers, followed by a deconvolution layer to map the classification to the same number of pixels in the input images. Because the 32 stride's classification layer has been pooled down to a size of 7x7, the final output segmentation is severely limited to a bilinear interpretation of this 7x7 mask.

The 16 stride network uses the 32 stride network as a base, then classifies the 5th pooling layer and combines this with the lowest level class predictions, allowing it to classify more accurately at a finer pixel scale. The 8 stride network uses the same idea on the 16 stride network, classifying the 4th pooling layer and combining this with the last prediction layer of the 16 stride network.



Detailed description of the networks used.

Training

In order to speed up efficiency of the training process, the networks were trained in steps, beginning with the 32 stride network. After each network was trained, the layer weights were used to initialize the next networks layers. If we were to train the 8 stride network from scratch, it would take significantly longer to train the convolution layers in the first half of the network.

The 32s network was trained with 50,000 iterations using a fixed learning rate of 1e-8 decreasing down to 1e-10 after the first 20,000 iterations, a momentum of 0.99, and a batch size of 15. Both the 16s and 8s networks were trained using 20,000 iterations using a 1e-10 learning rate and 0.99 momentum. These networks were all trained using the sneakers on white backgrounds.

Initially for the images containing background information, I attempted to train the networks from scratch, beginning with the 32 stride network. However, I discovered that using the weights from the 8 stride network trained on the white backgrounds allowed the network to quickly filter out the background after only 10,000 iterations with a learning rate of 1e-10. I found this very interesting because it shows that, even without any other information being shown to the network, it was still able to learn what a sneaker looks like. This leads me to believe that better segmentation on backgrounds could be gained by training on the white backgrounds more, which I did not have enough time to do. Lastly, I fine tuned the network with another 20,000 iterations at a learning rate of 1e-10.

Results

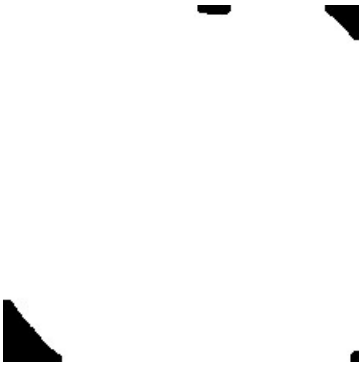
Input

FCN 32s

FCN 16s

FCN 8s

Input



No Background Network

Background Net

Input

Output

Input

Output

Conclusion

Using Fully Convolution Networks, I was able to segment out the background of an image, leaving a significant part of the object of interest, in this case sneakers. Due to the time restriction, and each 10,000 iterations taking several hours, I was not able to get a perfect segmentation. But, with more training I am certain this method would be able to return a very accurate segmentation prediction.

Because this method involves no user input, it is favorable for many applications that aim to provide an quick and easy experience. In the future, finer segmentation might be found by using a wider variety of layer combinations, as well as training the background network on real images with sneakers in them rather than just pasting a shoe on top of a background.

References

[1] T. Darrel, J. Long, E. Shelhamer. Fully Convolutional Networks for Semantic Segmentation. UC Berkeley, 2014.

[2] M. Everingham, L. Van~Gool, C. K. I. Williams, J. Winn, A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.htm

[3] K. Simonyan, A. Zisserman. Very Deep Convolutional Networks For Large-Scale Image Recognition. Visual Geometry Group, Department of Engineering Science, University of Oxford, 2014.