Semantic Segmentation with Histological Image Data: Cancer Cell vs. Stroma

Adam Abdulhamid
Stanford University
450 Serra Mall, Stanford, CA 94305
adama94@cs.stanford.edu

Abstract

With the introduction of end-to-end trainable neural models, several tasks across the field of computer vision have seen enormous success, including image classification, semantic segmentation, and many more. This paper explores the application of convolutional neural networks to the task of semantic segmentation on histological images from cancer patients, obtained from Stanford Medical School.

1. Introduction

Semantic segmentation has become an important task in computer vision over the past several years. With the introduction of AlexNet [1], and since then, many deeper network architectures like VGG [2] and ResNet [3], image classification has achieved accuracies on par, if not better than, human performance. Naturally, the next step was an end-to-end trainable convolutional neural network for semantic segmentation, which was first proposed by Jonathan Long and Evan Shelhamer at UC Berkeley [4].

This paper aims to apply the work done in the field of semantic segmentation to a dataset consisting of histological images of breast cancer patients in the Stanford Medical School.

2. Motivation

Before discussing the model and performance itself, it is useful to motivate finding a solution to the task at hand. The task at a high level is to segment images into cancerous sections and not cancerous sections. Given a robust classifier designed to perform this task, there are several useful applications of this classifier. One good example would be to try and categorize how different types of cancers behave. Given an automated way to go from the histological image to the labeled image, performing a large scale study on the behaviors and evolution of the cancer itself becomes much more feasible. Another more tangible example would be to use this labeled output from our classifier as an input to another system, the goal of which is to predict life expectancy and/or best treatments for an individual patient. Personalized medicine aims to provide patient specific treatment, and a robust and accurate classifier for a task like this will be very useful.

3. Related Work

The primary paper in the field of semantic segmentation that used end-to-end convolutional neural networks was that titled "Fully Convolutional Networks for Semantic Segmentation", written by Jonathan Long et. al [4]. In the paper they proposed a network architecture that is trained pixels to pixels, directly for semantic segmentation.

They adapted and tuned several modern deep networks, such as AlexNet [1], VGG [2], and GoogLeNet [5], to the specific task of image segmentation instead of image classification. With this, they achieved state of the art performance on a few datasets used to test image segmentation, such as PASCAL VOC [6], and NYUDv2. Also discussed was the relative efficiency with which inference can be completed. Inference requires just one forward pass through the convolutional network, which now contains no fully connected layers at the end. This provides quick inference, which is quite useful for real world tasks that need to be performed in near real time. Note that the advances in this paper rely not only on the success of previous networks such as AlexNet [1] and VGG [2], but also on the recent successes of transfer learning, and because of this the ability to fine tune models that have already been trained successfully.

4. Data

4.1. Dataset

As briefly mentioned above, the dataset contains histological images from real tumors. The tumors were extracted and imaged in the Stanford Medical School. The labels were hand generated by the same group in the Stanford Medical School as well. Overall, the dataset consists of 158 image/label pairs. The labels themselves are segmented into three categories: cancer, stroma, or background. The images and labels themselves are 1128x720 pixel images. This somewhat alleviates the issue of having a very small amount of data, because these images are roughly 10x larger than a 256x256 image which you will find in many other compute vision tasks. Here is an example image/label pair from the validation dataset.

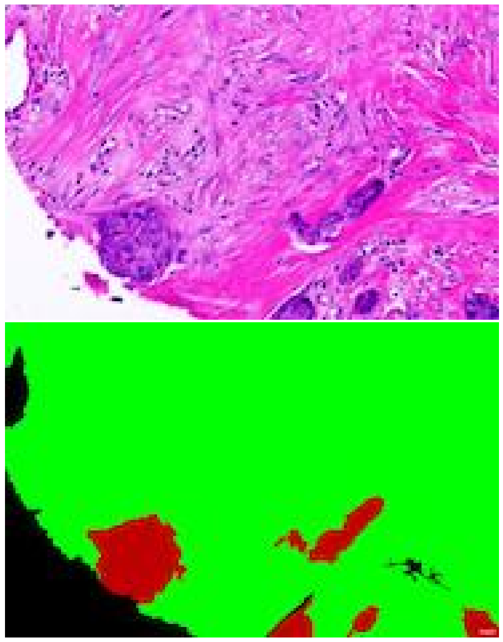
Figure 1: Example validation image and label number 148.

We can see here that our labels consist of only green, red, and black pixels, corresponding to stroma, cancer, and background respectively.

4.2. Data Augmentation

Vision tasks such as this usually require large labeled training sets, which as mentioned above, wasn't available for this specific task. The dataset was quite small with 158 image/label pairs. To try to alleviate this issue, a few data augmentation techniques were applied to give more training data. The one that was most promising and ended up being used was mirroring the data. To augment the data set, each training image and label was mirrored around both the x and y axis. This provided three times as much training data as we originally had, while still providing novel information because the kernels move left to right and top to bottom.

Other data augmentation techniques that were promising but would have required more time to explore fully are adding Gaussian noise to each image while leaving the labels unchanged, and tiling the images. Gaussian noise has the attempt to make the classifiers more robust, because small noise should not change the output from the classifier. Tiling the images allows the network to operate on smaller, more local inputs, and can aid in computational efficiency as well. Again, these other techniques were not explored in full but are likely to have positive effects on the efficacy of the final classifier.

4.3. Evaluation

The chosen evaluation metric will be some loss measured between the labeled images, and the models predictions. Two different types of evaluations were explored in this paper. The first was both L1 and L2 norms of the difference in output image from the model and corresponding label. Note different combinations of L1 and L2 norm were tried, and discussed later down in the results section, but here is the general formulation of loss using L1 and L2 norm:

$$L = \frac{1}{N}\sum_{i=1}^{N}(\lambda_1 L1_i + \lambda_2 L2_i)$$

Where $L1_i$ and $L2_i$ are:

$$L1_i = \|y_i - \hat{y}_i\|_1$$
$$L2_i = \|y_i - \hat{y}_i\|_2$$

Where y_i is the true label, and $\hat{y}_i$ is the predicted image. We take a combination of the $L1$ and $L2$ norm between the target and predicted image, and then average across all images to get a single scalar loss value.

The other evaluation metric used was softmax cross entropy loss. Our output image has three channels, one corresponding to each of the three classes: cancer, stroma, or background. If we frame the task as a classification at each pixel, it makes sense to take the softmax cross entropy loss between every pixels probability distribution, and the ground truth distribution from the corresponding labeled image. These pixel-wise losses are then averaged to give a scalar loss per image, and these again are averaged to get a scalar loss for the entire train, validation, or test set. The full form of the softmax loss for this task looks as follows:

$$L = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{P}\sum_{j=1}^{P}\sum_{k=1}^{3} p_{jk}\log(\hat{p}_{jk})$$

Where N is the total number of examples in the data set we are looking at, and P is the total number of pixels in one of the images. Here, p_j represents the true probability distribution over the 3 classes for the j-th pixel, and $\hat{p}_j$ represents the predicted distribution for the same pixel j. This formulation simplifies slightly because out target distributions are all one hot. It can be rewritten as follows:

$$L = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{P}\sum_{j=1}^{P}\log(\hat{p}_{jk})$$

Here, k represents the index of the true label for pixel j. Overall, these two different evaluation methods were used, and results are discussed below.

5. Approach

5.1. Overview

The network used in this paper is built on two main ideas. The first is transfer learning. Several layers from VGG16 [2] were used as the building blocks for the rest of the network. The second idea is more specific to semantic segmentation, which is the transpose convolutional layer. This layer is used to upsample the spatial dimensions of the image in a learnable way, as opposed to upsampling via other methods like pooling. With these two ideas put together, all the experimented models look similar. The images are first passed through a number of the VGG16 layers, where the spatial resolution shrinks as the volumes get further into the network. Then, these volumes are passed through a series of transpose convolutional layers and convolutional layers to upsample the spatial resolution back to the same size as the initial image, and to provide the model with the expressivity required to perform well on a task like semantic segmentation. A few different model architectures from various experiments are discussed below. Note, ReLu nonlinearities were used after each convolutional and transpose convolutional layer, but are omitted in the figures for brevity.

5.2. Architecture

Figure 2 in the appendix is a figure of the VGG16 architecture for reference. The purple block at the bottom is what was used as the transfer learning component to most of the models I experimented with. This means that the images were fed through VGG16 until right before the second pooling layer, and then extracted to build on top of. The future diagrams will contain the same purple block for clarity.

5.2.1 Experiment 1

The first experiment was run with the model architecture in Figure 3 in the appendix. The red layers are, as introduced previously, the transpose convolutional layers. In the case of experiment one, because the VGG layers only pooled once, the inputs to the transpose convolutional layer are exactly have the spatial dimensions of the original image, or 564x360. These are upsampled with 128 filters to get back a volume of dimensions 1128x720x128 which is then passed through several more convolutional layers until we get an output of size 1128x720x3 from which we take our loss directly. Large kernel sizes (11x11 and 9x9) were used in the first experiment to try to increase the receptive field. This was done because it is often useful to have a slightly larger perspective about the cells the kernel is passing over and what large groups within the image they belong to.

5.2.2 Experiment 2

The architecture for experiment two can be found in Figure 4 in the appendix. Experiment two looks quite similar to experiment one, but with kernel sizes slightly different. This was done in attempt to have a smoother transition from large to small kernel sizes while still maintaining the relatively large receptive field.

5.2.3 Experiment 3

The architecture for experiment three can be found in Figure 5 in the appendix. Experiment three modified the existing architecture in a few ways. One was the decrease in kernel size, so that all filters are 3x3. It was shown in the context of ResNet [3] that stacking smaller 3x3 filters can have the same effective receptive field as one larger 7x7 filter for example. The other change is the deeper channel depth. This was done to give the model a bit more expressive power to capture some of the more intricate features of the task space.

5.2.4 Final Architecture

Finally, the model architecture in Figure 6 was arrived at with a few more modifications. The deeper channel depth and smaller filter sizes have been retained, but there are two noteworthy changes. First, the VGG layers extracted now include one more pooling layer and two more convolutional layers. As a result of this, the volume coming from the VGG layers now has spatial dimensions one fourth of the original image size, or 282x180. In order to end up with images of the same spatial resolution as the inputs, two transpose convolutional layers must be used. Each transpose convolutional layer upsamples by a factor of two, so we will recover the dimensions needed for the loss metrics. These are the main changes that were used in the final model architecture.

6. Experiments & Results

6.1. Quantitative Analysis

As mentioned earlier, several experiments were performed with the different model architectures shown above. Table 1 presents a comparison of the different models performance using the softmax cross entropy loss function descried above. Note that the loss values presented here are loss values for the entire validation set. These experiments were run with a small hyperparameter search for best results, and were trained for roughly 5-10 epochs, or until no improvements were seen.

Experiment	Final Loss Value
One	0.849
Two	0.774
Three	0.748
Final experiment	0.697

Table 1. Comparison of loss values on validation set.

Dataset	Final Loss Value
Training	0.563
Validation	0.697
Test	0.685

Table 2. Comparison of loss values on validation set.

On the final model, the following hyperparameters were used: learning rate of 0.0001, batch size of 4, learning decay rate of 0.96. The Adam optimizer was used for optimization as well. Note that dropout along with standard L2 regularization were both implemented, but neither proved very useful. Again this is likely due to the small dataset, so any penalty on the model's expressivity ended up hurting performance all around. A scenario with little to no regularization on a small dataset is ripe for overfitting, but the model architecture itself is not incredibly complex, so the resulting gap between training error and validation/test error is acceptable. Table 2 presents the final loss values across the three datasets.

Looking at the final loss values, we see there is a gap between training and validation/test, but it is not too large. In addition, the validation loss and test loss are quite close, which is promising and means the model is likely to generalize to unseen examples well.

6.2. Qualitative Analysis

In addition to looking at the quantitative results of the model in terms of loss values, it can also be useful to qualitatively analyze how the model performs and hypothesize why it does well in some cases and not so well in others. To do so, we can look at a few examples of training example, training label, predicted label triplets and analyze where the predicted label differs. Below are two examples of images from the validation set.

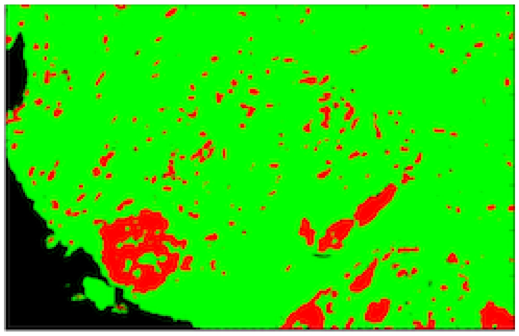

Figure 7: Example predicted label for image number 148.

This first example corresponds to the validation images and labels above in Figure 1. Referring back to the original image and label above, we can see that this produces quite reasonable results, with a few caveats. Overall, it seems to capture the main areas that truly contain the cancer. By visual inspection, we can notice that this corresponds to the darker purple spots in the original training image. The human visual system can quickly identify the pattern of darker, denser spots as likely to contain cancer, and it seems the convolutional neural network has done the same here. The difference is that this image seems to be overall much noisier. Looking back to the original image, it seems like the network classifies many of the individual cell nuclei as cancerous, likely because they are also generally darker and denser looking than the surrounding tissue. Here is a second example of validation image and true label.

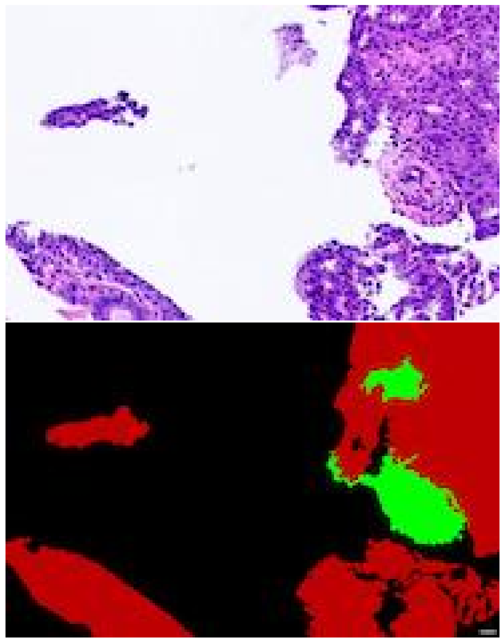

Figure 8: Example validation image and label number 154.

Looking at these two, it is quite difficult to visually separate what apparently are the true cancerous regions from the rest. The next figure contains the predicted model output.

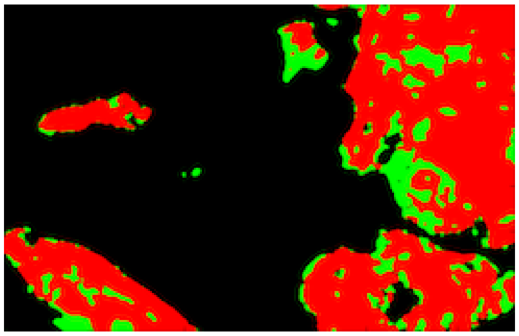

Figure 9: Example predicted label for image number 154.

We can see here many of the same qualities as we found in the previous example. It generally identifies the broad regions that are cancerous quite accurately, but the model seems to have much more noise than the labels themselves. Small groups, perhaps even individual pixels, seem to be misclassified in otherwise continuous large blocks.

Perhaps the noise comes from the fact that we have a relatively small dataset, and we are likely to end up overfitting to very small intricacies like we are seeing here. Even with regularization techniques, it is difficult to prevent overfitting on such a small dataset.

It is also interesting to note some of the deficiencies or shortcomings of the true labels themselves. In both examples here there are areas that are clearly part of the tissue that have been classified in the true label as background. Likely this is due to the fact that the process of hand generating these labels is expensive and tedious, and it is likely human oversight to classify these as background. Take for example the small green area in the center of Figure 9. Looking at the corresponding image in Figure 8 from the dataset, we see there is a small purple area there, yet again it is classified as background in the true label, also found in Figure 8.

Overall, it seem the model identifies quite well general cancerous areas, but fails to have the very fine precision and smoothness that the true labels do. This observation prompts the idea of adding another loss term to help promote smoothness. Even a simple loss function that has a small penalty for neighboring pixels differing, summed across all pixels, would likely help with the issue here of our predicted images not looking smooth. This idea, which shares many similarities to conditional random fields, was proposed in the context of semantic segmentation by Chen et. al [11] in 2014. This has an intuitive biological explanation as well. Cancer cells are likely to originate in one area and grow outward, not spawn up randomly and individually in many places. This is why we end up with one large tumor as opposed to small groups of cancer cells scattered across a large distance.

7. Conclusion

This paper aimed to build a semantic segmentation network for a given histological image dataset taken from Stanford Medical School. Overall, given the limitations in dataset size and time constraints, the results are promising and likely there could be room for much improvement. It is likely that the biggest limiting factor is dataset size. Therefore, given more time, this would be the area of focus. As discussed earlier, different Gaussian noise techniques or tiling techniques could prove very beneficial. It is also worth looking into augmenting the data with images from entirely different datasets as well. The Cancer Genome Atlas has quite a lot of similar histological images, but unfortunately no labels. It would be worth contacting the group who organized this dataset to see if it would be possible to obtain more high quality data.

Another next step would be to take this to a pathologist with domain knowledge to analyze more qualitatively where the model performs well and where it does not. This might be helpful in determining the shortcomings of the model, and could provide useful insights for designing new, potentially better model architectures.

8. References

[1] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. ImageNet Classification with Deep Convolution Neural Networks. *Proceedings of the 2012 Conference on Neural Information Processing Systems (NIPS)*, 2012.

[2] Karen Simonyan, Andrew Zisserman. Very Deep onvolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556, 2014.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385, 2015.

[4] Jonathan Long, Evan Shelhamer, Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. *CoRR*, abs/1411.4038, 2015.

[5] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. Going Deeper with Convolutions. *CoRR*, abs/1409.4842, 2014.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results.

[7] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor Segmentation and Support Inference from rgbd Images. In *European Conference on Computer Vision (ECCV)*, 2012.

[8] Jifeng Dai, Kariming He, Jian Sun. Convolutional Feature Masking for Joint Object and Stuff Segmentation. *CoRR*, abs/1412.1283, 2015.

[9] B. Hariharan, P. Arbelez, R. Girshick, and J. Malik. Simultaneous Detection and Segmentation. In *European Conference on Computer Vision (ECCV)*, 2014

[10] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, Yichen Wei. Fully Convolutional Instance-aware Semantic Segmentation. *CoRR*, abs/1611.07709, 2017.

[11] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, Alan L. Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. *CoRR*, abs/1412.7062, 2014.

[12] Carreira, J., Caseiro, R., Batista, J., and Sminchisescu, C. Semantic Segmentation with Second-order Pooling. In *European Conference on Computer Vision (ECCV)*, 2012

[13] Carreira, J. and Sminchisescu, C. Cpmc: Automatic Object Segmentation Using Constrained Parametric Mincuts. *PAMI*, 2012.

[14] Cogswell, M., Lin, X., Purushwalkam, S., and Batra, D. Combining the Best of Graphical Models and Convnets for Semantic Segmentation. *CoRR*, abs/1412.4313, 2014.

[15] Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *CVPR*, 2014.

9. Appendix

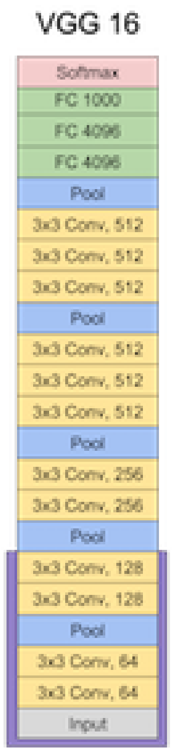

Figure 2: VGG16 Architecture

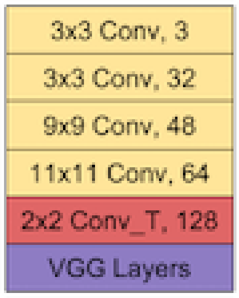

Figure 3: Experiment one network architecture

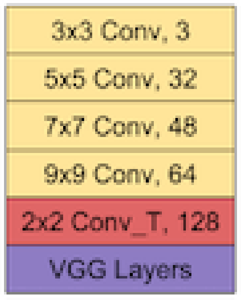

Figure 4: Experiment two network architecture

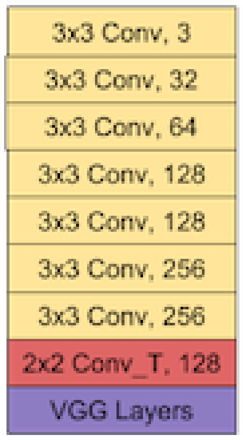

Figure 5: Experiment three network architecture

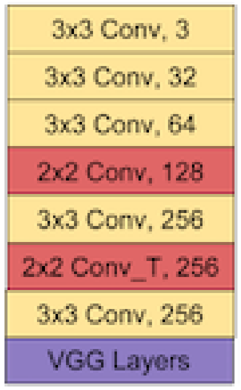

Figure 6: Final experiment network architecture