Segmentation of Breast Cancer Tumors using Deep Learning

jason0, ematsu

December 9, 2019

1 Introduction

Breast cancer is one of the leading causes of death for women in the United States. Many researchers are working on ways to find a cure for this terrible disease, and there are are currently several treatments that exist for it. These treatments target certain hormone receptors in the breast cells or the HER2 protein, all of which can stop the spread of malignant breast cancer cells. However, if the cancer does not have hormone receptors and it also doesn't respond to the HER2 treatments, then it's considered a 'triple negative' cancer, meaning no known treatments will work on it. These are the types of tumors that we will be segmenting in this project, and it's extremely important that we're able to help in the treatment of this disease.

Furthermore, we're choosing to use deep learning to segment the tumors in order to help the medical researchers automatically segment images of tumors, rather than forcing them to manually segment tumors in medical imagery. Manual segmentation can be a very tedious and long task, and we want these medical researchers to be able to devote as much of their time as possible to solving the main problem at hand regarding finding a cure, rather than doing a repetitive task that's already been solved.

2 Dataset

Our dataset consists of 222 patient MRIs, each with pathology-proven breast cancers (solitary or multiple). Each MRI has a single line annotation, marking the extent of the tumor across the largest diameter. This dataset was provided to us by our faculty sponsor at the Stanford School of Medicine. It was chosen for several factors. For one, it's a unique dataset consisting of triple negative breast cancer tumors that hasn't been automatically segmented before, which means we would be solving a novel problem in the space. For reason two, this dataset had the most fully annotated ground truth tumors available compared to other datasets that the faculty sponsor was showing us. This allowed us to work quickly by determining the loss of our training algorithms on real data.

The data is provided to us in the format of Nifti images, which are 3D images commonly used in the medical field, and they are from a 3D scan of a patient through an MRI. In order to make the problem easier, and also to get more training data, we decided to turn each of these 3D images (of size 512 x 512 x **depth**), into **depth** images of size 512 x 512. This means we dealt with 2D images in our project, which we sliced from the original dataset in our code provided above. Each image also is grayscale, so we do not have 3 separate channels for R, G, and B that other image segmentation algorithms often have to deal with.

There were several modifications to the dataset that we made in order to help our training and/or evaluation. First, since our training was running very slowly, we made a separate dataset by downsampling the existing 512 by 512 images to 256 by 256 images. The idea here was by decreasing the image size, we could still retain most of the information regarding tumor location, but decrease the training time by a factor of four.

Additionally, we normalized our datasets by subtracting the mean from each datapoint and dividing by the standard deviation. We wanted to standardize the data to account for a large range of values in the original images. Feature normalization can also help decrease the training speed by making convergence faster. Since we struggled with slow training speed due to the constraints of the Nero system, we thought this would be able to help increase our iteration speed as well.

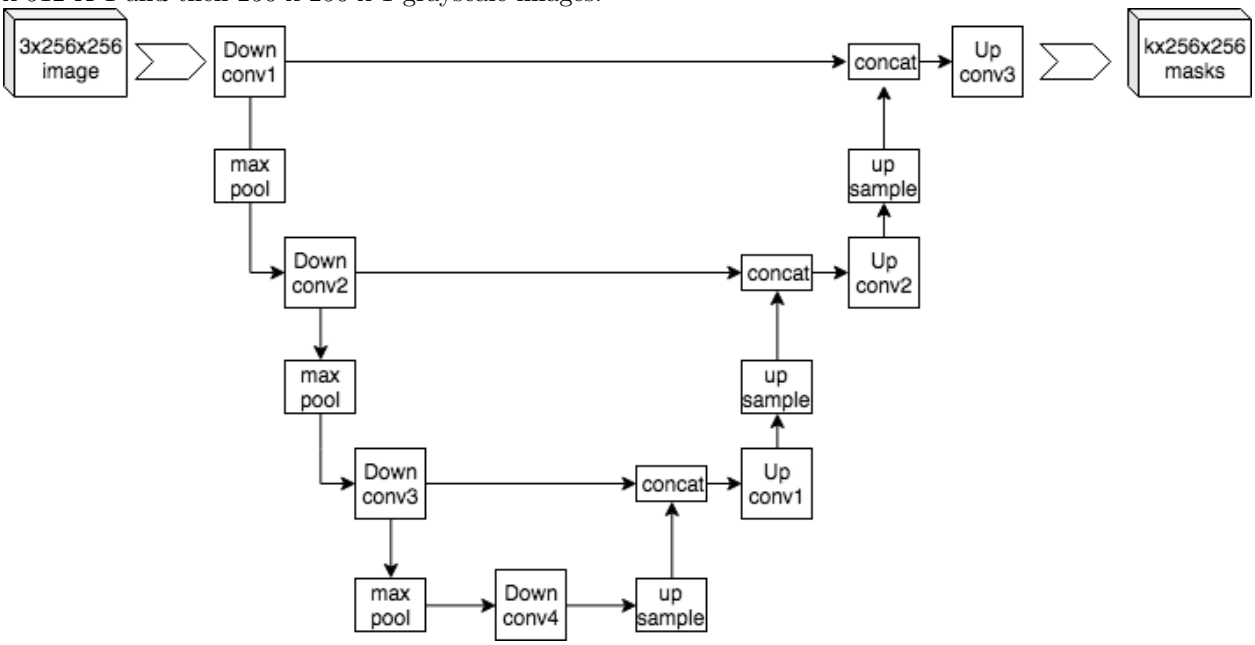
Finally, the dataset was a very unbalanced one, with not many images actually containing pixels of tumors. This is partly due to our decision to slice the 3D images into 2D images, as this lead to many images not having a tumor within them. Furthermore, even without taking this into effect, the tumors themselves are an extremely small part of each image, so there are many more negative pixels than positive pixels. This means that in order to alleviate some of this class imbalance, we created additional training sets that ignored all images with no tumor in them. This aimed to help the classifier avoid predicting only negative results, as well as again decreasing the training time by decreasing the training set size. We are able to do this because the problem we're tasked with guarantees that there will be a tumor inside each image we wish to segment.

We split our data into training and validation sets with no formal test set. We did this to ensure that we used as much of our labeled data for training purposes instead of keeping a formal separate test set. We also used splitting functions that shuffled the examples so as to avoid too many similar images being grouped together while training (since we sliced into our 3D volumes). The train/validation split was 80/20.

3 Approach

We decided to use a U-Net architecture for segmentation (references 3, 9). U-Net is a convolutional neural network aimed at segmenting biomedical imagery. It's a fully convolutional network, inspired by the fully convolutional network architecture proposed by Long and Shelhamer. The idea for a fully convolutional network is to replace the final layers of a convolutional neural network, normally consisting of several fully connected layers, with more convolutional layers. It uses these final convolutional layers to upsample the convolution into a pixel map of the final output, which should correspond to which pixels belong to which class.

The benefit of using U-Net in particular is that it is able to work with fewer training images. This is critical, as one of the pitfalls of our dataset is that it is relatively small, and only a portion of them are properly annotated with a ground truth label. U-Net achieves this, and differentiates itself from the original proposed Fully Convolutional Network architecture, by using far more channels during its upsampling process as it creates the final output image. This allows it to get a higher resolution output image with less context in the input data, and also leads to a U-shaped architecture, hence its name "U-Net". After we have the desired amount of pixel masks, we do a pixel-wise sigmoid over each mask to determine the final desired pixel class to output. Below is the implementation of the U-Net architecture, which we have adapted for 512 x 512 X 1 and then 256 x 256 x 1 grayscale images:



We trained the U-net model using the Adam optimization algorithm with a learning rate of 1e-4 for 5 epochs. We trained the model using three different loss functions. The first loss we used was a cross-entropy

loss function that weighs pixels near borders of the segmented tumor more than pixels elsewhere. This causes the algorithm to focus more heavily on getting more precise segmentations, and also specifically helps us in the biomedical field by allowing smaller cells to be properly segmented. In an attempt to get better metrics on our training data, we tried training on two other loss functions that are popularly used in image segmentation. The first was dice coefficient loss, which is defined by 2 times the intersection of the grouth truths and our predictions, divided by the sum of the grouth truths plus the sum of our predictions. We also used Jaccard distance as a loss function, which is defined by the union of the ground truths and our predictions minus the intersection, all divided by the union. These two loss functions were chosen because they perform well when dealing with unbalanced classes in image segmentation.

We also experimented with changing the dropout on the initial model to decrease since our model was performing poorly on the training set. This yielded no substantial difference in the performance of the model. We also tried experimenting with the learning rate. Reducing the learning rate did seem to improve model performance.

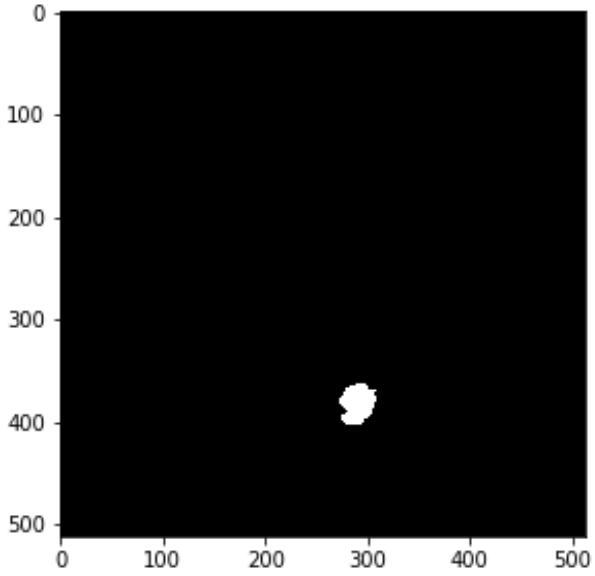
We also adopted a model that recently won the BRATS prize in 2018 for brain tumor segmentation. It was introduced by Andriy Myronenko at Nvidia. However, due to the limited constraints of our environment, we were not able to train our model. Due to HIPAA compliance and the confidential nature of our dataset, we had to train on Stanford Medicine's computing platform, Nero. The training had high memory requirements and our job requests on this platform were not able to be filled when requesting so much memory. When run with a lower amount of memory, our scripts would run into out of memory errors. This model allowed for multiple segmentation annotations as labels, but since we only had one type of annotation, we concatenated the data to be compatible with the model. We also cropped the MRI volumes into (256, 256, 16) volumes to be compatible with the network architecture.

4 Results

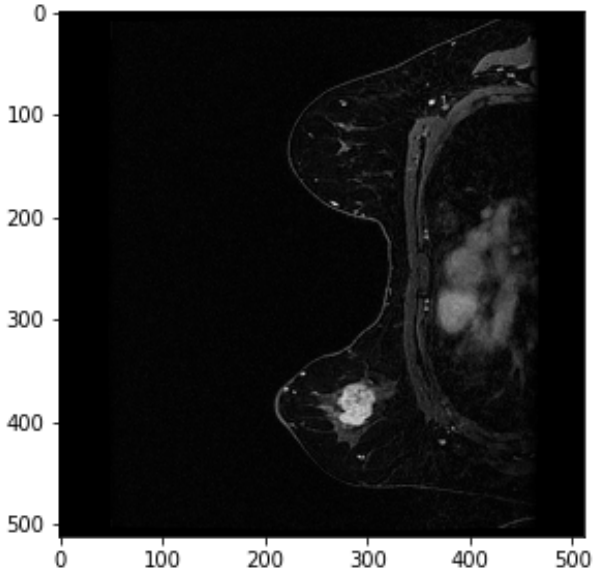
Our models had limited success in segmenting tumors in the images. Several achieved very high accuracy at the cost of recall and precision. Given the large class imbalance, it makes sense that our models were able to achieve high accuracy by predicting non-tumor for most pixels. Looking at the actual predictions, it is clear though that the models are learning something. In particular, it seems the models are able to recognize the lighter parts of the image as potential tumor pixels. Indeed, in the example included in this report, many of the tumor pixels are predicted as tumor. Varying the threshold also allows more or fewer pixels to be labeled as tumor pixels. Thus, we can increase our recall slightly, but at the expense of precision.

In order to combat the problem of predicting all negative results at the expense of recall, we decided to create another dataset of just images that contained tumors, which we call our "Balanced" datasets. This ensured that every image we trained on would have at least some portion of a tumor within it. Unfortunately, after running our training algorithms overnight on these balanced datasets, we found that the algorithm essentially flipped sides, and started predicting all positive results. This meant that our accuracy became extremely low, but our recall became near 1.

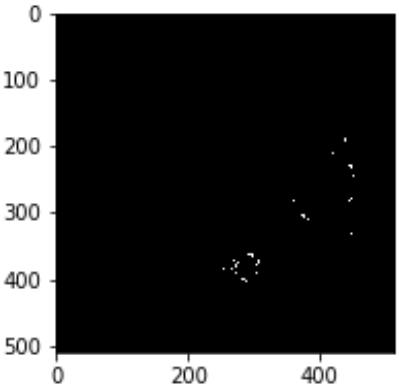
Ground Truth



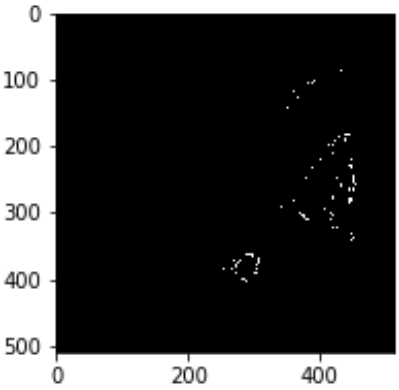
Test Image



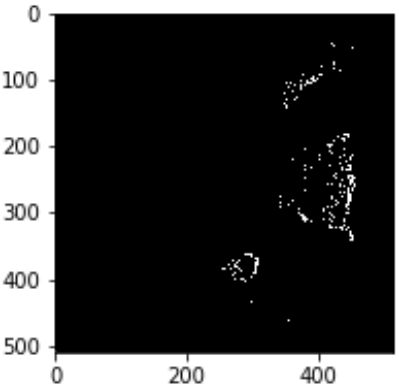
High Threshold



Medium Threshold



Low Threshold



Below is a table of our training results on both full and balanced datasets with each of our three loss functions. We also included a row with data for one of our runs for full cross entropy that had a lowered learning rate. It actually looked the most promising out of everything that we tried, but due to time constraints we could only make it to epoch 4/5, so we are reporting results from that epoch. Note that the cross entropy full model only has accuracy because we only compiled that model with the accuracy metric at that time.

	Accuracy	Precision	Recall	F1
Cross Entropy Full	0.999	-	-	-
Dice Full	0.975	0.000	0.000	0.000
Jaccard Full	0.975	0.000	0.000	0.000
Cross Entropy Balanced	0.001	0.137	0.994	0.237
Dice Balanced	0.004	0.142	1.000	0.243
Jaccard Balanced	0.001	0.138	1.000	0.240
Cross Entropy Full Lower Learning Rate	0.741	0.074	0.678	0.125

Below is a table of our validation results on both full and balanced datasets with three loss functions.

	Accuracy	Precision	Recall	F1
Cross Entropy Full	0.997	0.005	0.078	0.009
Dice Full	0.976	0.000	0.000	0.000
Jaccard Full	0.976	0.000	0.000	0.000
Cross Entropy Balanced	0.18	0.12	0.99	0.21
Dice Balanced	0.43	0.13	1.000	0.22
Jaccard Balanced	0.18	0.13	1.000	0.22

5 Conclusion

In conclusion, the segmentation of breast cancer tumors proved to be a difficult problem to tackle. A large part of this difficulty stems from the inherent class imbalance within the training data. Our initial efforts showed high accuracy but low recall, and our efforts to fix these issues created results that returned high recall but low accuracy, since it was over-predicting pixels as positive. Additionally, we see that there is high potential in decreasing the learning rate of the model. This suggests that our original learning rate caused the model to oscillate around or diverge from the optimal answer.

6 Future Work

Due to the constraints of our limited compute resources on the shared Nero platform, there were many experiments we would have liked to run but did not have time or were not allocated the resources to do so.

We started experimenting with trying to balance the dataset, which led to an over-correction in many cases (predicting all positive pixels for tumor). Some future experiments that might prove useful might be to try to balance the dataset, but in a more measured way with a more negative examples than our balanced datasets but with fewer negative examples than the full datasets. In addition, different normalization techniques on the data.

We also began to tune hyperparameters such as decreasing dropout or the learning rate. While decreasing the dropout did not seem to have a large effect, decreasing the learning rate showed promising results. More experimentation with hyperparameters might prove useful.

We also would like to try out other models. The model from Myronenko from Nvidia seems very promising and we have the data and model ready, but did not have enough time to get the memory resources needed to train it successfully. As mentioned in our "Approach" section, we ran into out of memory issues when training and were not allocated the resources to increase memory. Other models that seem interesting are Mask RCNN or Fully Convolutional Networks. The idea behind this paper is that it simply takes existing convolutional network architectures, such as AlexNet, VGG, or GoogLeNet, takes off their final fully connected layers, and replaces them with upsampling convolutional layers.

7 Contributions

Eric worked on preprocessing the data and normalizing it as well as training and evaluating models and viewing results. Jason worked on experimenting with balancing the datasets and different loss functions and training and evaluating models.

Thank you to Dr. Haruka Itakura in the Stanford School of Medicine for providing us with the dataset and project direction.

8 Code

Our code can be found on github at this link: `https://github.com/jason2249/cs230`.

9 References

1. https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf
2. https://www.breastcancer.org/symptoms/diagnosis/trip_neg
3. https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/
4. https://arxiv.org/pdf/1810.11654.pdf
5. https://arxiv.org/abs/1703.06870
6. https://arxiv.org/pdf/1606.04797.pdf
7. https://github.com/tensorflow/models/tree/master/research/deeplab
8. https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Dai_BoxSup_Exploiting_Bounding_ICCV_2015_paper.pdf
9. https://arxiv.org/abs/1505.04597