Brain Tumor Segmentation Using CNN

Miguel Aguilar
Department of Electrical Engineering
Stanford University
Stanford, CA
miguel6@stanford.edu

Ahmed Ahmed
Department of Computer Science
Stanford University
Stanford, CA
ahmedah@stanford.edu

Stephen Lopez
Department of Electrical Engineering
Stanford University
Stanford, CA
slopez27@stanford.edu

Abstract

The segmentation of magnetic resonance imaging (MRI) scans is an important tool used for examination, screening, diagnosis, and management of many life-threatening diseases. Automated MRI segmentation for the use of localizing tumors would allow for an increase in work throughput and potentially increase the chance at correct diagnoses and decrease false detection. Brain MRI scans are 3-dimensional data that are interpreted in 3 different coordinate planes, axial, coronal, and sagittal, each of which are interpreted differently by a physician. For the sake of time and computing power available to us, we focused on automating the segmentation of brain tumors from the axial view of the MRI scans. In this paper, we are attempting to automate the segmentation of brain tumors from an axial point of through the use of a U-Net Convolutional Neural Net (CNN) architecture in a similar manner as the original U-Net paper.

1 Problem statement

Segmentation and the subsequent quantitative assessment of lesions in medical images provide valuable information for the analysis of neuropathologies and are important for the planning of treatment strategies, monitoring of disease progression and prediction of patient outcome. For this paper, we will be focusing on automating the segmentation and the quantitative analysis of brain tumors in MRI scans provided in the BraTS dataset [8] for both high-grade gliomas and low-grade gliomas. In particular, we will be assessing the localization of brain tumors from the axial point-of-view of a generic brain tumor scan, similar to Figure 1.

2 Related Works

To get us started on this project, we took a look at a number of existing semantic segmentation deep learning networks. The first network we came across was the Fully Convolutional Network [11] that introduced the concept of using a convolutional neural network that consists of no fully-connected layers, but instead the fully-connected layers are replaced by convolutional layers. However, after further reading, we noticed that the U-Net architecture [7] might be more robust due to its use of skip connections in addition to the FCN so that the later layers of the model can retain the features that were extracted in the earlier layers of the model. The U-Net architecture was also first applied to

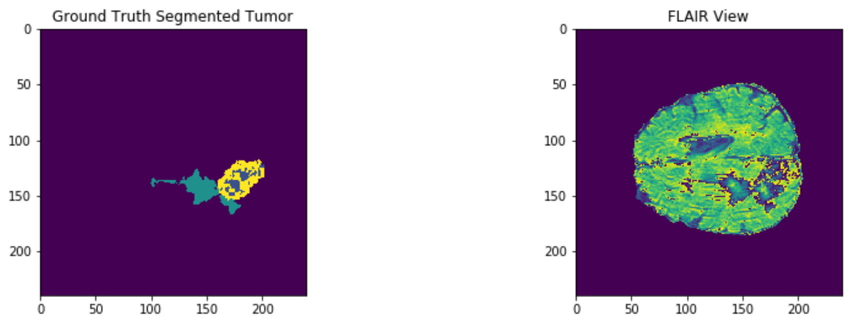


Figure 1: Example of a tumor segmented

other biomedical imaging semantic segmentation and as a result of this additional feature retainability, we chose to proceed with the U-Net architecture.

3 Data

The dataset that we obtained was the BraTS dataset provided by the Center for Biomedical Image Computing and Analytics (CBICA) in collaboration with the University of Pennsylvanica Pellman School of Medicine [8]. The BraTS data set contains 280 subjects who had a brain MRI scan, and the data set comes with 4 types of the neuroimage all as a separate image: Flair, T1, T1ce, and T2. Additionally, each subject comes with a ground truth segmented image, where the neuroimage is segmented with one of 5 colors indicating either lack of tumor, necrotic core, edema, non-enhancing core, and enhancing core. For the purpose of this project, we did not utilize the information of the classes of tumors in any way other than treating them as different classes. The MRI scans came in a neuroimaging format, .nii.gz, that needed to be extracted utilizing NiBabel tools. A large amount of work was solely focused on figuring ways to copy our data into usable formats without causing memory errors. Each image was in the shape of 240 x 240 x 155 pixels in size. Each different index represented a different viewing plane. Holding the first 2 dimensions constant yielded the axial view of the scan, holding the middle 2 dimensions constant yielded the coronal view of the scan, and holding the last 2 dimensions constant yielded the sagittal view of the scan. For this project, we decided to focus on segmentation of the axial point of view of the scan using all four types of the MRIs instead of analyzing the 3-dimensional scan as a whole. As a result, this allowed the input to our CNN to be 240x240x4 pixels in size. One thing we noticed in our data, however, was the imbalance in the classification of pixels in the scans. Roughly speaking, *only* 5-10% of our inputs would be input scans would be classified as one of the four classes of tumors. As a result, 90-95% of the data that our model would learn on would classify as not having a tumor, a clear imbalance in the classification of the pixels. We attempted to resolve this issues by randomly sampling 30x30 patches of our axial images in hopes of getting a more even balance of classes in addition to using class weights when training our model.

4 Baseline and Oracle

For our project's baseline, we thought about what would be the most intuitive way to segment the images of MRI scans we received by inspecting the data. We noticed by inspecting examples that tumors are often where there is a starkly darker color, which in our case means a much lower value than the surrounding pixels. We then used the OpenCV library within python to help us segment the FLAIR images such that we would only be left the area of a tumor, as it is an easy to use tool equipped with plenty of useful functions for the task of image segmentation [1]. First, we used a smoothing image filter on the FLAIR images we had, in order to reduce the noise that is often present in MRI scans. We choose OpenCV's Bilateral smoothing filter, because it allowed us to smooth over the few darker pixels that were sprinkled throughout areas without a tumor which were just noise, but unlike a Gaussian filter [2]:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

which takes the neighbors around the pixel and finds the weighted average and does not take into account important features for our scenario such as whether pixels might have a similar intensity, which would give us a better image to segment in that it would have more clearly defined edges. The Bilateral filter [3]:

$$I^{filtered}(x) = \frac{1}{W} \sum_{\text{all other pixels } y} I(y) G_i(||I(y) - I(x)||) G_s(||y - x||)$$

takes into account another component which is a function of the different pixel intensity differences, leading to a more useful smoothed image for our analysis. Here, $I^{filtered}$ returns the value of a pixel at our filtered image, I returns the original value of the pixel, G_i is a Gaussian function that represents the kernel for smoothing differences in intensities, G_s is a Gaussian function that represents the spatial kernel for smoothing differences in locations. The mean variance values we decided on were achieved through experimenting with which ones gave us a blur that balanced clearing out noise while keeping from removing too much detail.

As for the actual task of segmentation, the easiest and best performing method seemed to be a threshold segmentation method. using OpenCV we segmented the image such that the output was the residual image after we removed all pixels that had an intensity p higher than a threshold $thres$ by setting their values to zero. This allowed us to retain the lower intensity pixels that most likely belong to a tumor. After this, we simply marked whichever pixels that had a non-zero value as tumors, to be compared with the true-label. We found the best threshold by using a method within the OpenCV library that utilizes Otsu's method [9]:

$$\sigma_w^2 = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

Which minimizes variance between classes of a photo, where Weights ω_0 and ω_1 are the probabilities of the two classes separated by a threshold t ,and σ_1 and σ_0 are variances of these two classes. This is great for our baseline because we only have two classes, namely tumor (1) and not-tumor (0), and this will allow us to get a more realistic prediction than a global threshold. We want to note that we tried several other thresholds, that did not yield adequate segmentation. We also tried running an edge detection algorithm to potentially find a region of interest for a tumor, but OpenCV's tools would often highlight multiple areas within the MRI scan, severely reducing accuracy. Despite all this effort, when using the Intersection over Union (or Jaccard index) evaluation metric [4][5]:

$$\text{IoU(A, B)} = \frac{|\text{A} \cap \text{B}|}{|\text{A}| + |\text{B}| - |\text{A} \cap \text{B}|}$$

for which $0 \leq$ Iou(A, B) ≤ 1, our baseline only scored 0.005324 rounded to the fourth significant digit. Finally, we should mention that our oracle in this case is our dataset, as we have the true labels of all the tumors created by doctors from our scans, and there is no other viable, outside oracle.

5 Model and Results

For the architecture of our model, we utilized the U-Net architecture, shown in Figure 2, that has been utilized for medical imaging semantic segmentation [7]. Keep in mind that the image in Figure 2 does not the correct dimensions of the model we used in training. In deciding on the inputs and outputs of our model, however, we took a look at a number of approaches. The first approach we thought of was to take in the entire 3-dimensional brain scan and create a 3D CNN that would output a 3D segmented scan, limiting us to 280 total data points. Ultimately, we concluded that this approach would require a larger dataset to prevent us from overfitting as well as a lot of computational capacity. The second approach that we took into consideration and decided on was to take only the axial view of the images and to utilize all four types of the MRI scans. With this approach, we implemented two models: one model that attempted to segment all 5 classes of tumor and another model that attempted to simply distinguish between tumor and no tumor. This allowed us to have a total of 25650 data points that we split the data points into a $80/10/10$ training/val/test split. The inputs of our model, x will be of

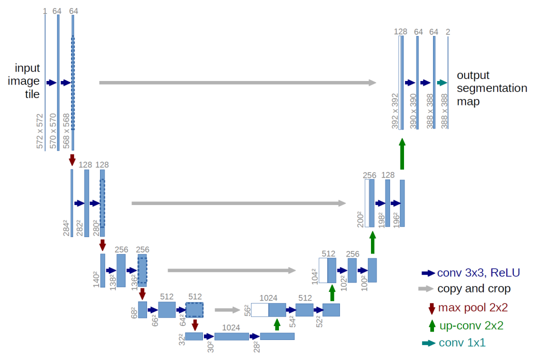


Figure 2: U-Net Architecture

dimensions 240x240x4, or $x \in \mathbb{R}^{240\text{x}240\text{x}4}$, where the first two dimensions indicate the dimensions of the axial image, and the last dimension indicates the type of the MRI scan. The output of our models, $\hat{y}$, will be of dimension 240x240x5 and 240x240x2, or $\hat{y} \in \mathbb{R}^{240\text{x}240\text{x}5}$ or $\hat{y} \in \mathbb{R}^{240\text{x}240\text{x}2}$, where the first two dimensions indicate the dimensions of the axial image, and the last dimension indicates the total number of classes that our model will be segmenting. The loss function that we used for training was the Cross Entropy Loss function, defined as such:

$$CE = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}),$$

where $M = 2 \text{or} M = 5$, y is the binary indicator (0 or 1) if the class label c predicted for the observation is correct for pixel o, and p is the predicted probability that the observation is of the labeled class. For the optimizer, we experimented with two different algorithms, the first of which was stochastic gradient descent (SGD) with Nesterov momentum:

$$v_{t+1} = \mu v_t - \eta \nabla L(\theta + \mu v_t)$$
$$\theta_{t+1} = \theta_t + v_{t+1},$$

where μ is the momentum parameter. The second optimizer that we experimented with was the Adam optimizer:

$$w^{(t+1)} = w^{(t)} - \eta \frac{\hat{m_w}}{\sqrt{\hat{v_w}} + \epsilon},$$
$$m_w^{(t+1)} \leftarrow \beta_1 m_w^{(t)} + (1 - \beta_1) \nabla_w L^{(t)},$$
$$v_w^{(t+1)} \leftarrow \beta_2 v_w^{(t)} + (1 + \beta_2)(\nabla_w L^{(t)})^2,$$
$$\hat{m}_w = \frac{m_w^{(t+1)}}{1 - \beta_1^{(t+1)}},$$
$$\hat{v}_w = \frac{v_w^{(t+1)}}{1 - \beta_2^{(t+1)}},$$

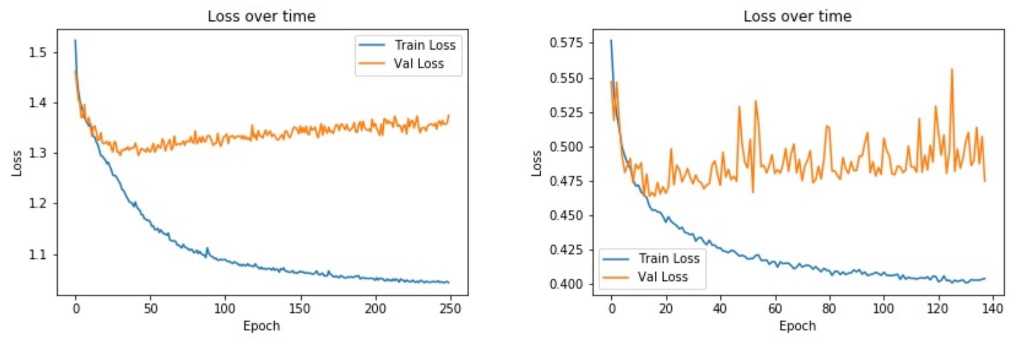
where β_1 and β_2 are the forgetting factors for gradients and second moments of gradients, respectively. During training, we were having more success with the SGD with Nesterov momentum optimizer, so we stuck with that for the rest of the project.

5.1 Results

In Figure 3 are the final results of the two models' loss plotted over iteration. Additionally, you can see our mean Intersection Over Union graphs as training went on in Figures 4 and 5 for both models. Recall that Intersection over Union is defined as:

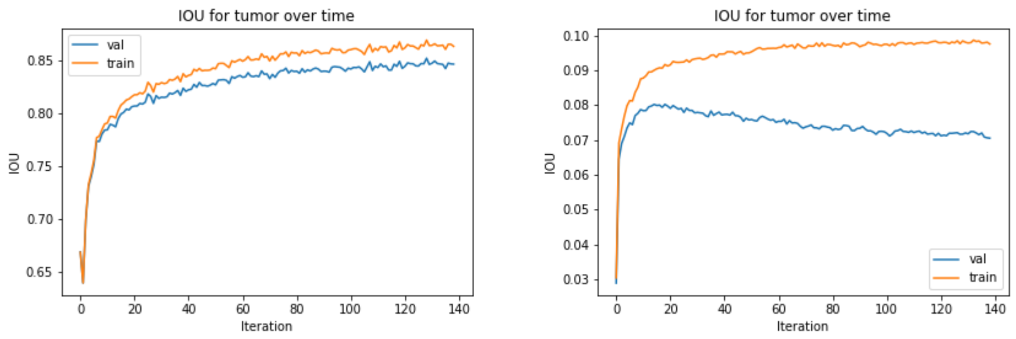
$$\text{IoU(A, B)} = \frac{|\text{A} \cap \text{B}|}{|\text{A}| + |\text{B}| - |\text{A} \cap \text{B}|}.$$

The source code for our project can be seen at: `https://github.com/aguilarmg/cs221-final-project`.



(a) Averaged Loss Curves

Figure 3: Loss Over Time



(a) IoU Curves for Binary Segmentation Problem

Figure 4: IoU Curves for Binary Tumor Segmentation

6 Error Analysis

6.1 Overfitting

As you can see from our IoU curves in Figure 4, our model began to overfit when predicting whether pixels would contain a tumor or not. We added more regularization, but due to the lack of time, we needed to regularize our loss function further to account for this overfitting. Another option we can look into in future works is early-stopping, where we would stop the model from training as soon as we see it begin to overfit so that it does not overfit any further.

6.2 Class Imbalance

One major issue with our dataset was the severe imbalance in classes: $< 7\%$ of our pixels mapped to tumors. As a result of this, the models that we were training were learning a lot about how to classify a pixel as a blank pixel that did not contain a tumor, but it didn't have enough exposure to be able to classify a pixel as a tumor pixel consistently. We tried to mitigate against this issue by random sampling our dataset into 30x30 patches to attempt to get a more even distribution of pixels across the classes. This ultimately did not solve our issue, so we then attempted to utilize class weights. This helped a bit as we were beginning to achieve approximately IoU values of 0.10 when detecting for a tumor in the binary tumor segmentation problem, but it still was not enough. We believe that another approach to attempting to mitigate against this issue would be through the use of data augmentation to generate more "tumor pixels" that are not identical to what we have right now.

Acknowledgements

We'd like to acknowledge our Mentor TA Shervine Amidi, for his mentorship in finding a feasible problem, and how to better improve our IOU results, reduce noise and which metrics and techniques we should have considered for our problem formulation, loss function, and network architecture.

References

[1] Image Thresholding. (2017, December 22). Retrieved from https://docs.opencv.org/3.4.0/d7/d4d/tutorial_py_thresholding.html

[2] Gaussian blur. (2018, November 20). Retrieved from https://en.wikipedia.org/wiki/Gaussian_blur

[3] Bilateral filter. (2019, June 09). Retrieved from https://en.wikipedia.org/wiki/Bilateral_filter

[4] Intersection over Union (IoU) for object detection. (2018, June 24). Retrieved from https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detect

[5] Jaccard index. (2019, April 01). Retrieved from https://en.wikipedia.org/wiki/Jaccard_index

[6] Sørensen–Dice coefficient. (2019, April 09). Retrieved from https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient

[7] Ronneberger, O., Fischer, P., & Brox, T. (2015, May 18). U-Net: Convolutional Networks for Biomedical Image Segmentation [Scholarly project]. Retrieved from https://arxiv.org/abs/1505.04597

[8] Section for Biomedical Image Analysis (SBIA). (n.d.). Retrieved from https://www.med.upenn.edu/sbia/brats2018/data.html

[9] Otsu's method. (2019, May 17). Retrieved from https://en.wikipedia.org/wiki/Otsu's_method

[10] Hui, J., & Hui, J. (2018, March 07). MAP (mean Average Precision) for Object Detection. Retrieved from https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

[11] Shelhamer, E., Long, J., & Darrell, T. (2016, May 20). Fully Convolutional Networks for Semantic Segmentation [Scholarly project]. Retrieved from https://arxiv.org/abs/1605.06211