Brain Tumor Segmentation Project Report

Problem Statement

Can a machine learning model detect brain tumors in MR images of human brains with accuracy over 95% in order to automate the segmentation of brain tumors?

Introduction

A brain tumor is a cluster of mutated cells that can be identified in patients through Magnetic Resonance Imaging (MRI), a non-invasive technique used to capture images of slices of the brain. Brain tumors can be seen on an MRI as an abnormal mass that disrupts the structure of healthy brain tissue.

Previously, brain tumors in MR images often needed to be manually sectioned off, or segmented, from healthy brain tissue by expert radiologists in order to measure the tumor, make a diagnosis, and inform treatment; however, manual segmentation of brain tumors is an onerous task. Manual segmentation requires an expert to go through every slice of an MRI individually and delineate the tumor, which can be extremely time consuming, and produces a suboptimal jagged segmentation (Prastawa et al., 2003). In addition, manual segmentation is often subjective among different radiologists and is prone to human error. Manual segmentation of brain tumors uses up precious hospital resources, but with recent advancements in machine learning techniques, tumor segmentation can be done automatically in order to improve this process.

Some approaches of automated tumor segmentation include unsupervised machine learning techniques that rely on image intensities such as thresholding and edge detection, which work well when boundaries are well defined. However, the current state of the art for image segmentation are convolutional neural networks (CNNs) and various versions of CNNs such as fully convolutional networks (FCNs) that do not have any dense layers (Seo et al., 2020). Below, I will outline my approach with a classic U-Net architecture, an FCN built specifically for biomedical image segmentation, developed by Ronneberger (2015).

Data

The data used for this project came from a subset of The Cancer Genome Atlas Low Grade Glioma (TCGA-LGG) dataset (Pedano et al., 2016) and can be downloaded from kaggle. The dataset contains 3,929 Images from 65 patients with low grade gliomas, a type of slow-growing brain tumor. Each image is a 256x256 axial cross section from an MRI and has a corresponding segmentation mask which marks all tumor pixels as 255 and all non-tumor pixels, i.e. healthy brain tissue and background, as 0. These masks were validated by a board-certified radiologist from Duke University.

Exploratory Data Analysis

After downloading the data, I created a dataframe of file paths for the mri scan and the corresponding segmentation mask and visualized some samples to verify that the masks matched the images. Figure 1 displays some samples of the mri, the mask, and an overlay of the mri and the mask.

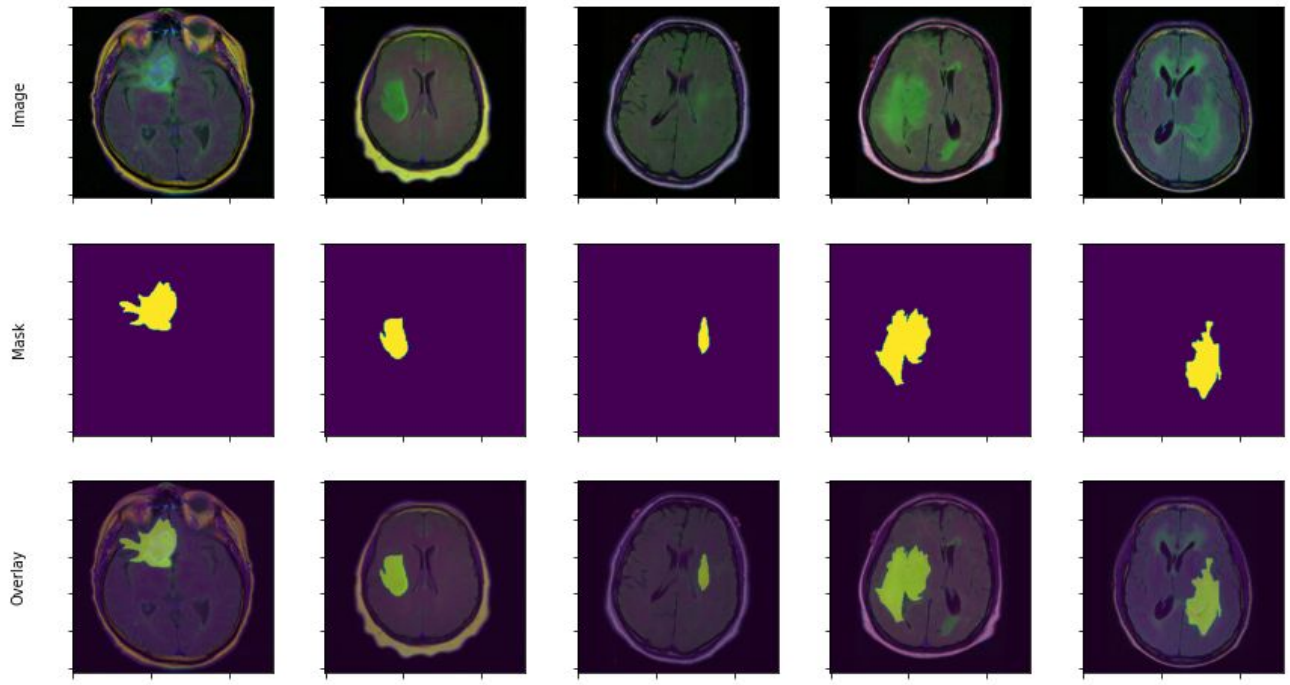

Figure 1: sample of MRI (top), mask (middle), and an overlay of the MRI and mask (bottom)

In addition, I calculated the number of images without tumors and found that there were 2,556 images without a tumor present, compared to 1,373 images with a tumor present. I also looped through every pixel and calculated the percentage of tumor pixels out of all pixels. Only 1.03% of all pixels were tumor pixels, meaning that there was a major class imbalance to be considered in the model.

Preprocessing

To prepare the data for the model, I performed some transformations on the images and masks together. With the consideration of the large class imbalance, I decided to drop all 2,556 images that did not have a tumor present as a way of undersampling the majority class of non tumor pixels. This increased the percentage of tumor pixels to 2.9% while decreasing the size of the dataset to a total of 1,373 image and mask pairs. In addition, I created a function that crops

the image around the tumor, with a random amount of padding on each side of the tumor. The purpose of this function was to further increase the ratio of tumor pixels to non tumor pixels, while not sacrificing too much of the surrounding information necessary for the model to learn properly. With the transformation, the percentage of tumors increased to 6%. After cropping the images, I resized all images back to 256x256 and scaled all pixels by 255 so that the pixel data was normalized from 0 to 1. A sample of the transformations can be found below in figure 4. With the randomized padding, some images are unchanged from the original (image 1) and some images are slightly shifted and/or zoomed in (images 2-5).

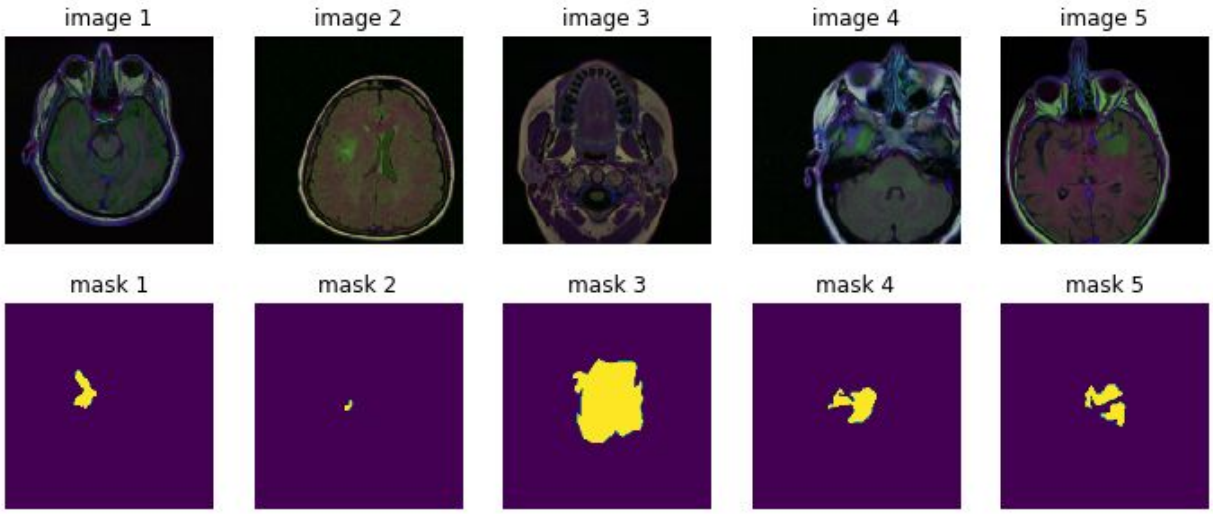

Figure 2: A sample of mri images and masks after transformation

Modeling

U-Net

The architecture I decided to use for my neural network was the U-Net which is an FCN that performs well on biomedical image segmentation, shown below in figure 3. This network involves an encoder path that downsamples the images to 16x16 pixels and provides context information with 1024 feature maps through several convolution and pooling layers. I also added batch normalization after the convolution layers in order to combat overfitting on the small training set. Then the images are decoded through deconvolution layers and upsampling the images back to their original size to provide spatial information. Then there is a final convolution layer that outputs a binary classification for every pixel. One major component of the U-Net is that the upsampled layers from the decoder path are concatenated with the corresponding convolution layers from the encoder path through skip connections so that context information is not lost. In total, the U-Net has 22,384,577 parameters to train.

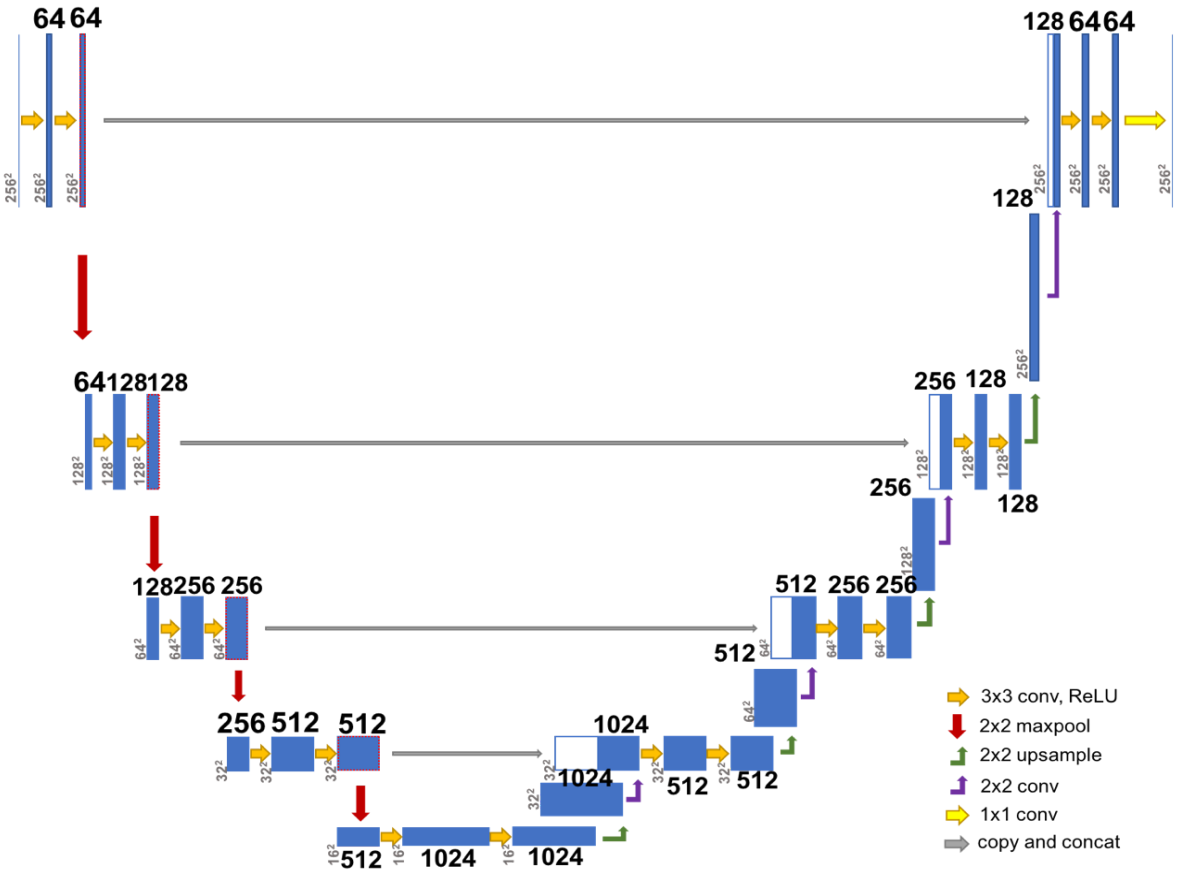

Figure 3: U-Net model architecture (Padhy, 2017)

Loss Function

For the metrics and loss function, I used a custom dice coefficient function. The dice coefficient ranges from 0 to 1 and is a better representative of model performance over binary accuracy because it acts the same as the F1 score for binary data (see formula below), meaning that the dice coefficient handles class imbalance better than binary accuracy. A smoothing parameter was added to the numerator and denominator to ensure that the function was differentiable. For the loss function, I simply subtracted the dice coefficient from 1.

$$Dice = \frac{2 \times intersection}{union} = \frac{2TP}{2TP + FP + FN} = F1$$

Figure 4: formula for the dice coefficient

Hyperparameter Tuning

In order to tune the model, I ran many experiments to test out a number of different hyperparameters with keras. I tested various optimizers and found that the Adam optimizer works best with a learning rate of .001 and learning rate decay. The model also worked best with a mini batch size of 32 images per batch. For the convolutions, I used a ReLU activation function, a kernel initializer that draws weights randomly from a normal distribution centered around 0, and added padding so that the output shape matches the input shape. I also tried additional regularization such as kernel regularization and dropout but found that the model works best without the additional regularization. I let the model run for 100 epochs and added model checkpoints so that I could save and load the best model.

Results

The best model had a dice score of .68 and a binary accuracy of .984 on unseen data. In figure x, you can see the train and validation scores for every epoch. Although the train score keeps improving, the score for the validation set caps at approximately .70 after 100 epochs.

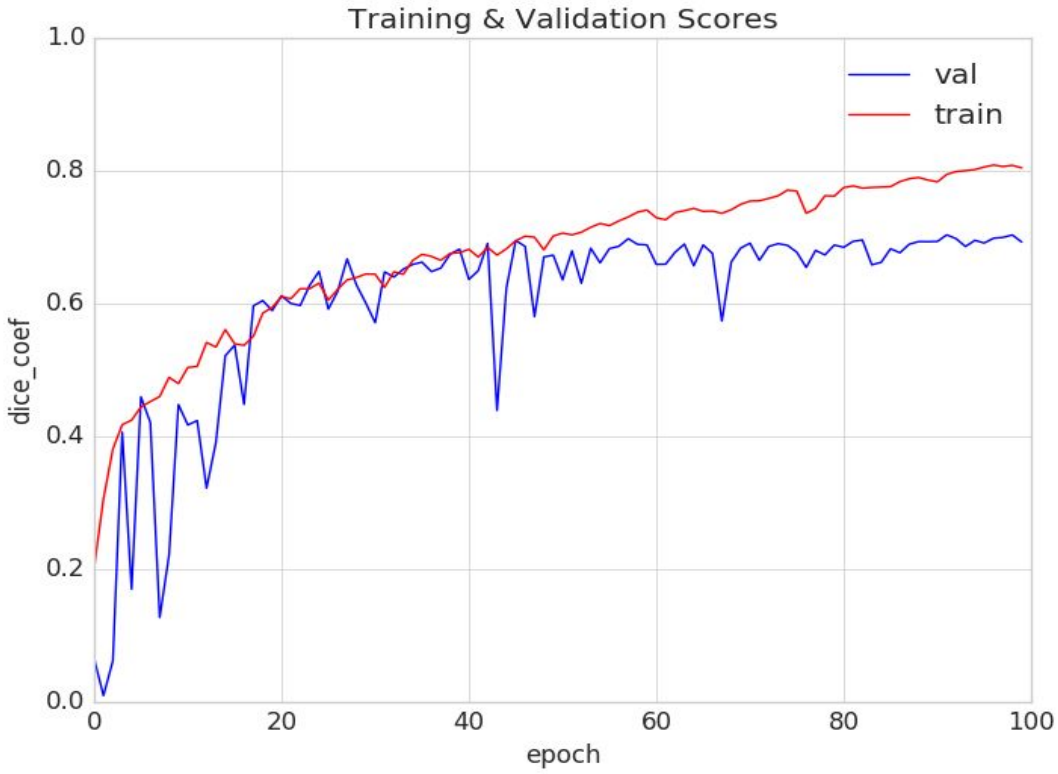

Figure 5: A graph of the training and validation dice coefficients for each epoch

Figure 6 shows a few sample predictions of the test set. The model does well with simple tumor shapes, but lacks the complexity needed to capture oddly shaped tumors.

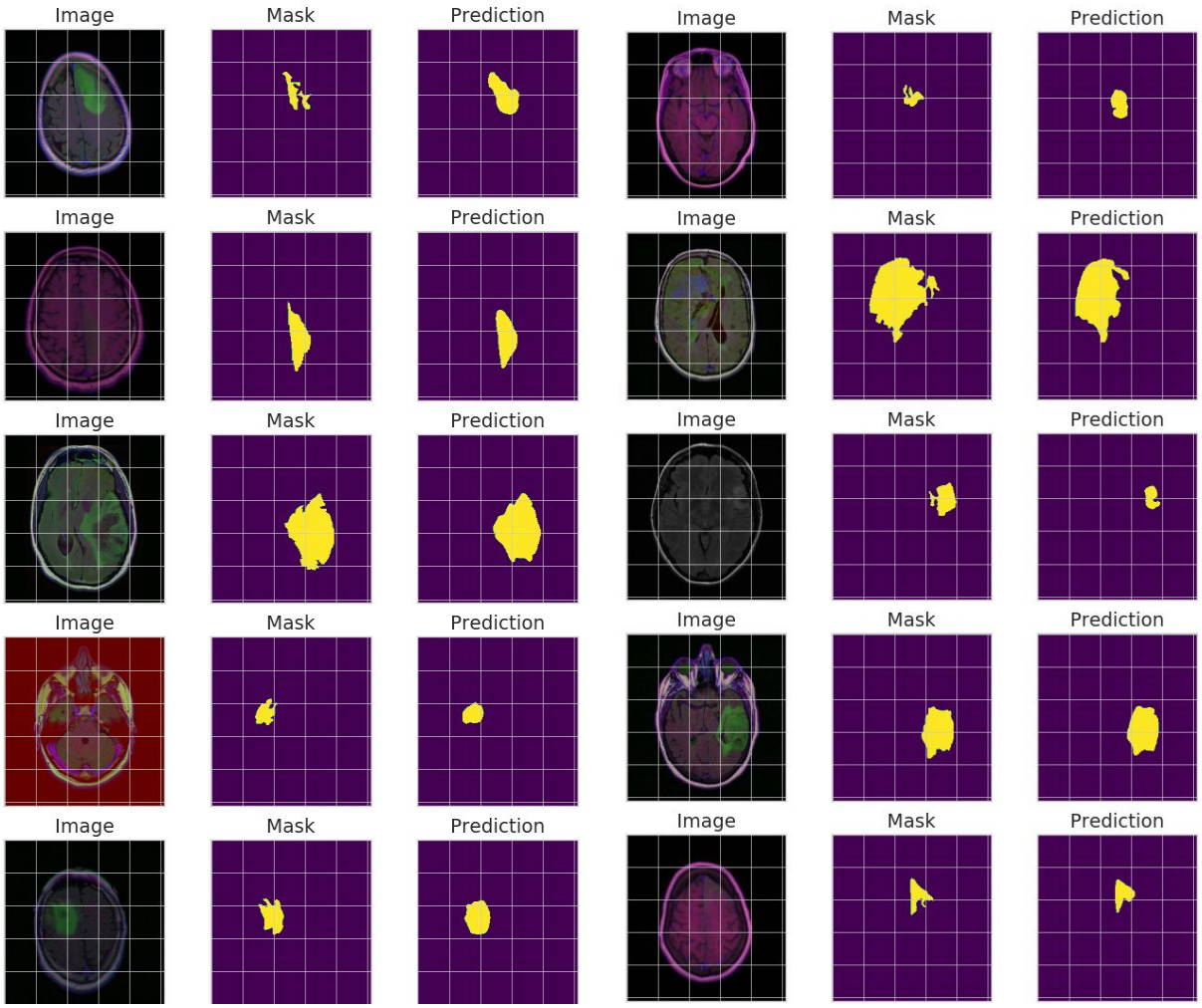

Figure 6: Sample predictions

Conclusion

With the small amount of data available and the size of the network, the model performed considerably well. While the current model may not be accurate enough for fully automated brain tumor segmentation, it would be very useful for semi-automated segmentation in order to reduce the burden on radiologists. Moving forward, fully automated segmentation may be achieved with more data which would greatly improve model performance. In addition, while the classic U-Net performs well, other variations of the U-Net may perform better for this specific task. Additionally, the task could be further specified to categorize the different components of the tumor such as edema (accumulation of fluid), necrosis (accumulation of dead cells), and enhancing/non-enhancing tumor. The use of machine learning and deep learning has proved useful to the field of medicine in many areas, including MRI brain tumor segmentation.

Citations

Padhy, S. (2017). *Unet Zoo*. Awesome Open Source. https://awesomeopensource.com/project/shreyaspadhy/UNet-Zoo

Pedano, N., Flanders, A. E., Scarpace, L., Mikkelsen, T., Eschbacher, J. M., Hermes, B., … Ostrom, Q. (2016). *Radiology Data from The Cancer Genome Atlas Low Grade Glioma [TCGA-LGG] collection*. The Cancer Imaging Archive. https://wiki.cancerimagingarchive.net/display/Public/TCGA-LGG

Prastawa, M., Bullitt, E., Moon, N., Van Leemput, K., & Gerig, G. (2003). Automatic Brain Tumor Segmentation by Subject Specific Modification of Atlas Priors. *Academic Radiology*, *10*(12), 1341–1348. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2430604/

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *ArXiv:1505.04597 [Cs]*. http://arxiv.org/abs/1505.04597

Seo, H., Khuzani, M. B., Vasudevan, V., Huang, C., Ren, H., Xiao, R., Jia, X., & Xing, L. (2020). Machine learning techniques for biomedical image segmentation: An overview of technical aspects and introduction to state-of-art applications. *Medical Physics*, *47*(5), e148–e167. https://doi.org/10.1002/mp.13649