Capstone Project:Deep Learning Analysis of Brain Pathologies

Julie D. Blum

February 18, 2019

1 Definition

1.1 Project Overview

The purpose of this project is to use deep learning techniques to determine whether brain pathologies are present in patient magnetic resonance imaging (MRI) images and to determine which parts of the brain are affected by disease by localizing the abnormalities, i.e. segmentation of the images. Specifically, MRI images from patients with brain tumors affecting the glial cells of the brain which can be classified as high-grade (HGG) with a poor prognosis or low-grade (LGG) with a better prognosis will be considered. Additionally, patient MRI's with brain lesions caused by ischemic stroke will be analyzed.

Brain tumors and strokes are terribly destructive to the lives of many people. Most people know at least one person, usually older, who has been affected by stroke and although brain tumors are not as ubiquitous, there are frequently people in the news such as John McCain with this devastating disease. The manual labor of analyzing MRI images to determine appropriate treatment regimens for these diseases is difficult, time-consuming, and dependent on the ability and subjective interpretations of radiologists. An accurate automated procedure would be beneficial for early detection and treatment of these diseases. Image segmentation removes brain structures from the image not associated with the the disease, thus improving the reliability of the diagnosis and avoiding unnecessary damage to healthy brain regions.[1]

1.2 Problem:Segmentation of Brain MRIs

In this project a deep learning network will be trained on 3-dimensional brain MRI images of patients with brain pathologies, and the network will output a brain image for each patient labeled by zeroes and ones. Optimizing the match between the output of these training images and MRI images of the same patients that have been manually labeled with zeroes and ones according to the absence or presence of pathologies (segmented) by expert radiologists will be the training goal. The ability of the trained network to segment new patient MRI's will be determined by test and validation runs. Various metrics described below will quantify the agreement between the output of the network and the manually segmented images.

1.3 Evaluation Metrics

The most important "metric" used in segmentation problems to quantify the agreement of two images is the Sørenson-Dice coefficient (DSC)[2]. The formula for the DSC compares an output image from a neural net and a ground truth image labeled by ones and zeroes. The overlap between ones on the ground truth and ones on the output correspond to true positives (TP), the overlap between zeroes on the ground truth and ones on the output correspond to false positives (FP), while the overlap between ones on the ground truth and zeroes on the output corresponds to false negatives (FN) and the overlap between zeroes on the output and zeroes on the ground truth correspond to true negatives (TN). Thus, the number of ones on the ground truth is $TP+FN$ and on the output is $TP+FP$. The formula measures the sum of overlapping, correct ones on both images divided by the total number of ones on both images:

$$DSC = \frac{2 * TP}{2 * TP + FP + FN}$$

.

Slightly modifing DSC allows $1 - DSC$ to be used as a differentiable loss function for the deep network. Ground truth pixels are denoted by g_j and output pixels by o_j and ϵ is a small positive number.

$$DSC = \frac{2 * \sum_j g_j * o_j}{\sum (g_j^2 + o_j^2) + \epsilon}$$

Since the distance function, 1-DSC, does not satisfy the triangle inequality, DSC is not actually a metric. Other measures including accuracy, sensitivity, and specificity can be determined from comparing the output image and ground truth.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$sensitivity = \frac{TP}{TP + FN}$$

$$specificity = \frac{TN}{TN + FP}$$

Sensitivity and specificity are sensitive to the sizes of images with a larger penalty for incorrect labels in smaller images[3]. Accuracy needs to be complemented with other measures because true negatives swamp true positives in the segmentation results.

Note that the DSC avoids the problem of class imbalance in MRI's due to the far greater number of background voxels (zeroes) than ones by not including true negatives in its definition. Because DSC is not a distance metric, the penalty for non-overlapping regions of the output does not increase with their distance from the overlap region. Incorrect labels receive a uniform penalty. The coregistration of images from each patient is important here to avoid misaligned segmentations.

The Hausdorff distance metric based on the surface distance between the segmentation result and the ground truth will be considered. This metric is sensitive to outliers and avoids the problems Dice has with a small image dimension. The stroke MRI images in this project have small sized third dimensions.

Note that outliers cannot be ignored in the medical context. The Hausdorff metric yields infinity if the segmentation is empty (all zeroes) so Dice is a better measure for this case and the empty segmentation must be removed for averaging Hausdorff. These aspects of DSC and other measures are discussed in [3, 4].

2 Analysis

2.1 Data Exploration

The BrATS (Multimodal Brain Tumor Segmentation Challenge) competition [5] has been a yearly contest to benchmark algorithms that segment MRI images with brain tumors since 2012 while the ISLES (ISCHEMIC STROKE LESION SEGMENTATION) challenge [6, 7] has attracted contestants since 2015. For brain tumor segmentation both unsupervised and supervised learning methods have been tried. The data for the analysis comes from the medical image segmentation challenges, BRATS2015 and ISLES2017, consisting in the BRATS case of four types of three-dimensional MRI images; T1-weighted, T2-weighted (including Fluid-Attenuated Inversion Recovery, i.e., FLAIR), and gadolinium enhanced T1-weighted imaging sequences(T1c). For each anonymous patient there is a manually segmented image in which labels necrosis (1), edema (2), non-enhancing (3), and enhancing tumor (4) designate the corresponding parts of the tumor. A brief explanation of some of the medical aspects of these labels can be found for the tumor labels [8] and for an explanation of the types of MRI's used.[9] The training set provided by BRATS2015 consists of 220 HGG patients and 54 LGG patients. The image preprocessing consisted of coregistration with the T1c MRI which had the finest spatial resolution and resampling and interpolation to $1 \times 1 \times 1$ mm^3 so that the image size was $240 \times 240 \times 155$.[1]

The ISLES2017 training set consists of 43 anonymous patients with seven types of MRI's not including the ground truth manual segmentation. The MRI's of each patient are coregistered but there is no further preprocessing. The MRI types include diffusion maps (DWI and ADC) as well as perfusion maps (CBV, CBF, MTT, TTP, and TMAX) For an explanation of these MRI's see [10].

For my project the training set provided by these challenges is further processed and split into training, validation, and test sets for the deep learning networks. The stroke MRI's require further preprocessing (resampling) so that voxel sizes of all images are identical.

2.2 Exploratory Visualization

2.2.1 Brain Tumors

Here are slices of the different types of MRI's for patients with high grade and low grade gliomas. Note that there are different colors representing the four aspects of the tumors in the ground mri.

Figure 1: HGG-flair

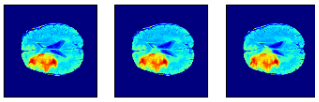


Figure 2: HGG-T1

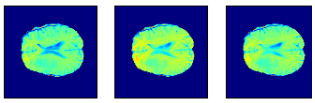


Figure 3: HGG-T1c

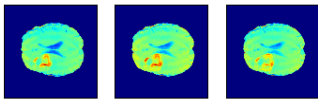


Figure 4: HGG-T2

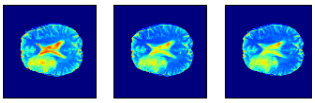


Figure 5: HGG-ground

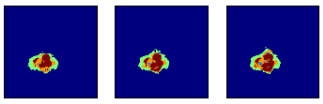


Figure 6: LGG-flair

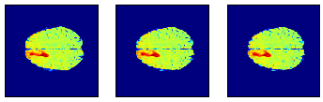


Figure 7: LGG-T1

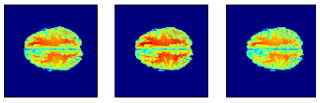


Figure 8: LGG-T1c

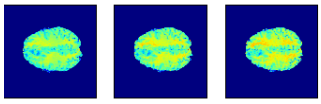


Figure 9: LGG-T2

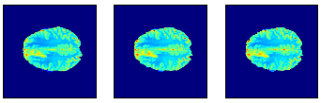
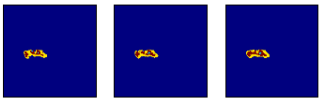


Figure 10: LGG-ground



2.2.2 Strokes

Here are some two-dimensional slices of the various types of MRI's showing the brain of a patient suffering a stroke. The ground MRI, the result of segmentation by an expert radiologist, has a tiny red blob at the stroke location. I have normalized these MRI's to mean zero and standard deviation one.

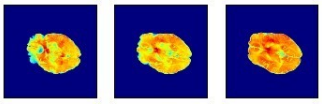


Figure 11: DWI

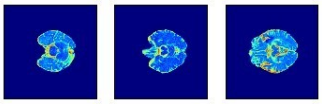


Figure 12: ADC

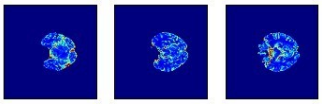


Figure 13: CBF

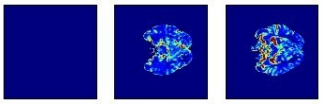


Figure 14: CBV

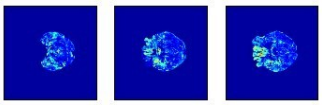


Figure 15: MTT

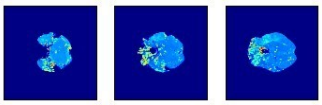


Figure 16: TTP

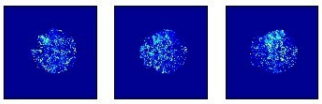


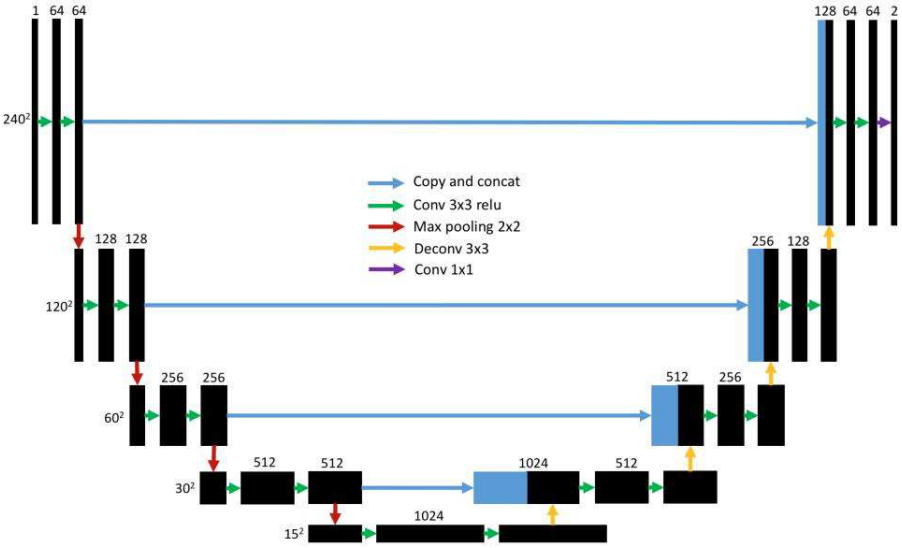
Figure 17: Tmax

Figure 18: ground

2.3 Algorithms and Techniques

In this project the BRATS2015 and ISLES2017 datasets will be trained and tested on a U-net and 3D CNN using publicly available software. [11, 12] To set up a gpu to take advantage of these algorithms, it is necessary to install a large amount of software including TensorFlow from source [13], TensorLayer [14], and deepmedic which used Theano but now uses Tensorflow [12].

Figure 19: U-net [1]



The figure above schematically describes the U-net. An image of size 240^2 pixels enters the network on the top left side, and is processed through a standard convolution network on the left side with the third dimension (channels) increasing from 1 to 64 after the first convolution. The output of blocks of three convolutions is concatenated to the output of a deconvolution network on the right side of the diagram. At the top right, the output of the first convolution block combines with that of the last deconvolution block. A sigmoid function converts the output pixels to zeroes and ones.

The MRI images need to be converted to numpy arrays. For each type of MRI, excluding the ground truth, the array of patients is normalized to zero mean and standard deviation one. The resulting arrays are stacked together. The data is then split into training, validation, and test sets. The arrays are shuffled and various random deformations are applied to each set of five patient MRI types (including ground truth) before sending consecutive batches through the network. The segmented output images are compared to the ground truth by computing the dice and other metrics.

The three dimensional convolution neural network (3D CNN) can be very inefficient and costly to implement because of the extra dimensional parameters held in memory. The DeepMedic set-up [15] shown below increases the efficiency in several ways. It uses a dense-inference technique that inputs image segments larger than the small patches of previous 3D CNN's. This results in feature maps of the classification layer that are image segments larger than one voxel.

Figure 20: DeepMedic 3D CNN

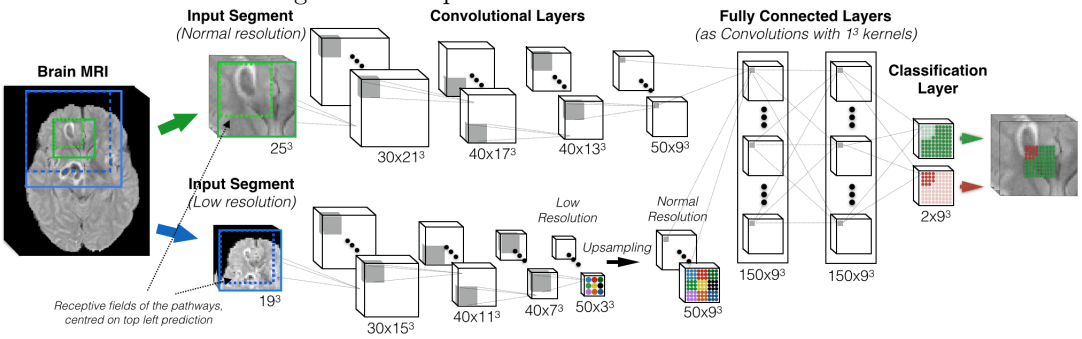


Figure 21: Multi-scale 3D CNN with two convolutional pathways. The kernels of the two pathways are here of size 5^3 (for illustration only to reduce the number of layers in the figure). The neurons of the last layers of the two pathways thus have receptive fields of size 17^3 voxels. The inputs of the two pathways are centered at the same image location, but the second segment is extracted from a down-sampled version of the image by a factor of 3. The second pathway processes context in an actual area of size 51^3 voxels. The DeepMedic 11-layers architecture results from replacing each layer of the depicted pathways with two that use 3^3 kernels. Number of FMs and their size depicted as $(Number \times Size)$.[15]

Besides obviating the need for redundant computations on the same voxels in overlapping patches, the batch size is effectively multiplied by the segment size in the classification layer without increasing the memory requirements. The larger training image segments can be chosen to balance the classes (ones and zeroes here). Deepmedic uses smaller kernel sizes (3^3 instead of 5^3) to conserve computer resources. The third dimension also can result in noise amplification. DeepMedic decreases this undesired effect by using a Batch Normalization technique.[16] Similarly to the U-net approach, DeepMedic adds a second convolution pathway where the input images are downsized to provide contextual (large-scale) information while the first pathway provides the local (small-scale) information. The output of the second pathway is upsized and concatenated with that of the first pathway. There are two additional hidden layers before the classification layer. Noise from the input images and deep learning minimization of loss (due to incorrect classification) that results in local rather than global minima in the output can negatively impact the predictive power of neural networks. DeepMedic adds a three-dimensional Conditional Random Field (CRF) after the 3D CNN that is fully connected (there is a potential term between every pair of voxels) based on the two-dimensional CRF[17] to smooth anomalous "holes" in the segmentation. Recently, the CRF has been formulated as a recurrent neural network (RNN) that can be coupled to a CNN[18] which may be possible to include in my set-up.

For DeepMedic all MRI images need to be resampled to identical voxel sizes while the means and standard deviations of patients for each MRI type (excluding the ground truth) needs to be normalized to zero and one respectively. DeepMedic saves a lot of computer memory by not requiring that Nifti images

be converted to Numpy arrays of identical sizes before running the 3D CNN. However, the preprocessing steps require back and forth conversions between Numpy arrays and Nifti images which are conveniently handled with Nilearn and NiBabel.

2.4 Benchmark

The results for the BRATS2015 challenge can be found at [19]. The top Dice score for the complete segmentation which treats all the four labels the same is 0.87, 0.76 for the core (labels 1, 3, and 4), and 0.66 for the enhance (label 4). The testing leaderboard for ISLES2017 are found at [20]. The present top Dice score there is 0.36.

3 Methodology

3.1 Data Preprocessing

3.1.1 Brain Tumors

As mentioned above the BRATS2015 MRI's were previously preprocessed to uniform image size and uniform voxel size, and the MRI's for each patient have been coregistered or aligned. I implemented the following steps to prepare the images for the U-net. The .mha formatted images were converted to numpy arrays with the help of medpy (The inverse transformation utilized simpleITK). I needed to obtain the means and standard deviations of patient arrays for the four types (not including ground truth) of MRI and with the arrays including both categories of tumors (HGG and LGG). Thus, there were a total of four means to calculate. The corresponding patient arrays were normalized to mean zero and standard deviation one. The normalized arrays and the ground truth array were split into training (132 HGG and 32 LGG), validation (44 HGG and 11 LGG), and test (44 HGG and 11 LGG) arrays. Then the different types of MRI arrays were stacked together. In the stacking step, the (240, 240, 155) shape of the four types of MRI images for each patient becomes (155, 240, 240, 4) and the patients are added to a numpy array of shape (155* number of patients, 240, 240, 4) in preparation for sending through the U-net. Memory limitations forced me to split the the HGG training stacks into two sets. The exact details of the data processing are given in the included code file "U_net_get_data.html".

3.1.2 Strokes

The ISLES2017 data preprocessing was more complicated. The first task was to create range of interest masks for the brain MRI's. The code for this can be found in the "make_niimg_masks" function of the included file "make_brain_mask.html". I needed to treat the DWI MRI separately from the others because it had some extra dimensions including a time series. I needed to squeeze the size one dimensions and take the mean of the time dimension. For the nilearn code to work, one also needs to smooth the image before calculating a brain mask. The brain masks from the other types of MRI's and the DWI were then intersected to produce a mask for each patient. By excluding the region outside the range of interest from subsequent calculations, one can improve the

dice scores by several hundredths. The bulk of the data processing is in the "U_net_get_data(stroke).html" file. For the deepmedic net to work properly, one needs to resample all images in the database to the same voxel size, and in my case that was $1 \times 1 \times 1$ mm^3. Note that all of the images of each patient are the same size, but the sizes differ among patients. After resampling I calculated the mean and standard deviation of the patients for each MRI type by restricting these calculations to the non-empty mask regions. The same normalization as in the brain tumor case was used. The configuration files and data files for running the deepmedic algorithm were created in the included code file "create_3d_data_from_patientIms.html" Here, I split the set into a training, testing, and validation set. Because the number of patients was quite small, I rotated the training and validation into a five-fold cross-validation.

3.2 Implementation

3.2.1 Brain Tumors

The implementation of the U-net for BRATS2015 is presented in the included code file "U_net_train_test.html" which is mostly taken from [11]. I introduced an interactive feature in the two included data retrieval python files, "get_train_valid_test.py" and "get_mixed_datasets.py", to determine which part of the data to import into the U-net to not overburden the computer memory capacity. The "get_mixed_datasets.py" mixed the HGG and LGG patients while the "get_train_valid_test.py" did not. The heart of the U-net is the function "run_u_net". This function will output a segmented image for each patient to be compared with the ground truth images for the corresponding patients. If the default variable "all" is chosen, then all four labels, necrosis (1), edema (2), non-enhancing (3), and enhancing tumor (4), are kept in the ground truth image as ones while unlabeled parts of this image are zeroes. One can pick out necrosis, for example, by putting all labels except ones as zeroes by choosing the "necrosis" variable for "run_u_net". Images are randomly distorted before entering the u-net through the function "distort_imgs". As discussed in [21], these deformations increase the amount of data leading to better results. The elastic deformation is useful in changing the shape of tumors which are amorphous blobs.[1]. I did not experiment too much with the U-net which had already been successfully implemented by [11] but had as always to do some initial debugging. The batch size could not be increased very much without crashing my computer. I did try the "u_net_bn" which adds a batch normalization layer and found in a cursory implementation that it did not work as well as the plain U-net for brain tumors. However, in my initial work with stroke MRI's adding the batch normalization to the U-net definitely improved the results. After finding that the good results of the U-net did not generalize to the test set of BRATS2015, I reimplemented the U-net with batch normalization.

3.2.2 Strokes

The deepmedic algorithm is fairly straightforward to use. The training and validation configuration file contains a large number of parameters and optimization techniques that can be varied. The model configuration file sets the batch sizes and contains parameters associated with the architecture of the neural net. I experimented with just about all of these as I slowly improved my results. I might

not have done so much experimentation had I looked earlier at the ISLES2017 benchmark since my results were not as bad as I thought in comparison to the benchmark. I made some early mistakes such as not noticing that the ground truth was being inputted instead of the mask image for the range of interest restriction which gave me better results than what was warranted.

I include here my best choices of model and training parameters in the "modelConfig.cfg" and "trainConfig.cfg" files. The main limitation on achieving better results, given the best choice of parameters, was computer memory. I found that using the whole image sampling with the largest possible dimensions gave the best results. I increased the size of image sampling up to the point that the program collapsed due to insufficient memory. I also needed to restart my computer and sometimes rest the computer after memory intensive tasks to avoid an unnecessary collapse. The resulting image sampling size in the two large dimensions was significantly less than the full image. However, the smaller sampling size included a good portion of the mask image and probably most of the ground image. In any case using an incomplete whole image improved on the deepmedic default sampling. In the subsampling pathway I had a larger downsizing factor for the two larger dimensions and a smaller one for the third dimension of the MRI images. For this stroke project the Adam optimizer with "selu" activation improved on the default RMSprop optimizer with "prelu" activation.

I added a post-processing CRF to the deepmedic algorithm that used the public code of [17]. The included code files are "postDeepmedicCRF.html", "experimentCRFparams.html", and "MakeCRFims.html" There are twelve parameters in the CRF smoothing potential, and I set these randomly in 100 iterations to determine the best improvement on the average dice score in the "postDeepmedicCRF.html" code. I then made small perturbations on the best solution in "experimentCRFparams.html" to obtain a small further improvement. Additionally, I used [22] to calculate the Hausdorff distance while omitting empty segmentations when calculating the Hausdorff average to avoid an infinite result. I took the test sets from the five-fold cross-validation and ran them through the CRF with the best choice of parameters. The results can be seen under the fifth code block of the "MakeCRFims.html". I followed all of the above steps with some minor changes in submitting training and test results to the ISLES2017 contest.

3.3 Refinement

3.3.1 Brain Tumors

The U-net configuration did not have a lot of free parameters. I did a small amount of experimentation with the learning rate and batch size, but the default parameters gave the best result. I did decrease the number of epochs in the default. I found that starting the "all" training by not mixing the HGG and LGG MRI's using the code in "get_train_valid_test.py" worked best. After training on all five sets of data, I used the "get_mixed_datasets.py" to further train on five sets mixing the HGG and LGG. Note that because of the random data augmentation deformations the training set passing through the U-net was not repeated. I used the pretrained U-net to then train on an HGG-LGG mix for the "core" training where the core of the tumor includes the labels 1, 3,

and 4. From the "all" training I could deduce the 0 labels, and then could infer the 2 labels from the "core" training. The "core" training used the pretrained "all" network. In order to deduce label 4, the enhancing tumor, I picked out labels 4 using the pretrained "core" network. Note that I had to stop the enhance training after one epoch to avoid reaching an extremum of the loss function with 100% loss. The included code files "U-net_train_test_all", "U-net_train_test_core", and "U-net_train_test_enhance" contain the code as well as results. I added one further machine learning step to try to distinguish between the low-grade and high-grade gliomas. Using ideas in [23] and an adaptation of scikit learn face recognition code [24] with a support vector machine, I achieved 75% accuracy on a test set that was 10% of the training set while the training accuracy was 99%. I used this algorithm to label the unknown test set with no way of ascertaining the accuracy. The code for this is included in the "classify_hgg_lgg-BRATS2015_test.html" file. I had tried to look at the distribution of the four tumor types among the LGG and HGG cases but found no pattern to distinguish the grade. I had to do some further work to obtain segmentation images for the BRATS2015 contest in the .mha format.

3.3.2 Strokes

I started my stroke project with the default deepmedic configuration parameters without using range of interest masks or an additional CRF step. I did all sorts of non-systematic variations on these parameters, and my results remained poor. Adding the masks improved my results dramatically as mentioned above until I realized I was using the ground truth images rather than the masks. Then I went through the model and training configuration files systematically varying almost all of the possible parameters, one at a time. For discrete parameters like the type of sampling or optimization, I kept whatever choice gave the best average dice score. If a continuous parameter change improved the dice, I would tune the parameter by moving in the direction of improvement until I obtained a decrease in the dice and keep the best result. I found in the model configuration that the default size of the convolution net and kernel sizes worked best. I changed the downsizing factor in the subsampling pathway to obtain a better result. I also needed to decrease the training batch size as I increased the sampling size to not crash my computer. In the training configuration, increasing the foreground sampling fraction over the background for the foreground/background sampling where the foreground is the ground truth image initially improved the result, but the whole image sampling with the largest size possible gave the best result. Optimizing the number of epochs and subepochs required a decrease in the default training time. I had a much smaller learning rate with exponential decay starting after the first training epoch. My momentum was slightly lower and my regulation "L1" and "L2" were orders of magnitude higher. Note that the original deepmedic had been optimized for ISLES2015 where MRI sizes were uniform and the third dimension was comparable to the others.

4 Results

4.1 Model Evaluation and Validation

4.1.1 Brain Tumors

Here are some sample plots of the dice cost, hard dice, and intersection over union (iou) measures for 26 epochs of training and validation of the first fifth of the data. The results level off with little improvement training and validating on the rest of the data.

Figure 22: All: Dice Cost, Hard Dice, IOU

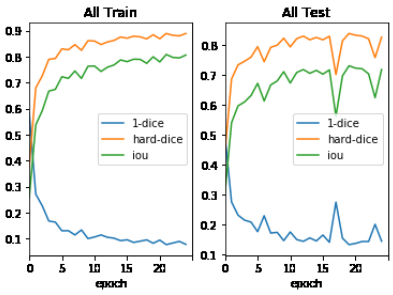


Figure 23: Core: Dice Cost, Hard Dice, IOU

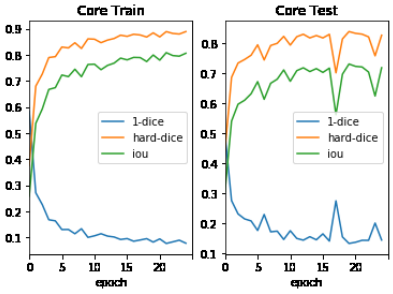
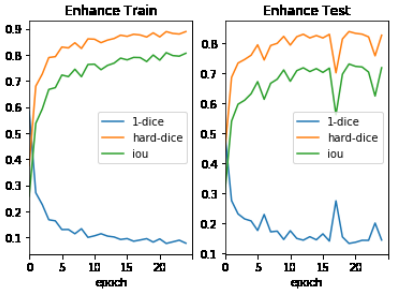
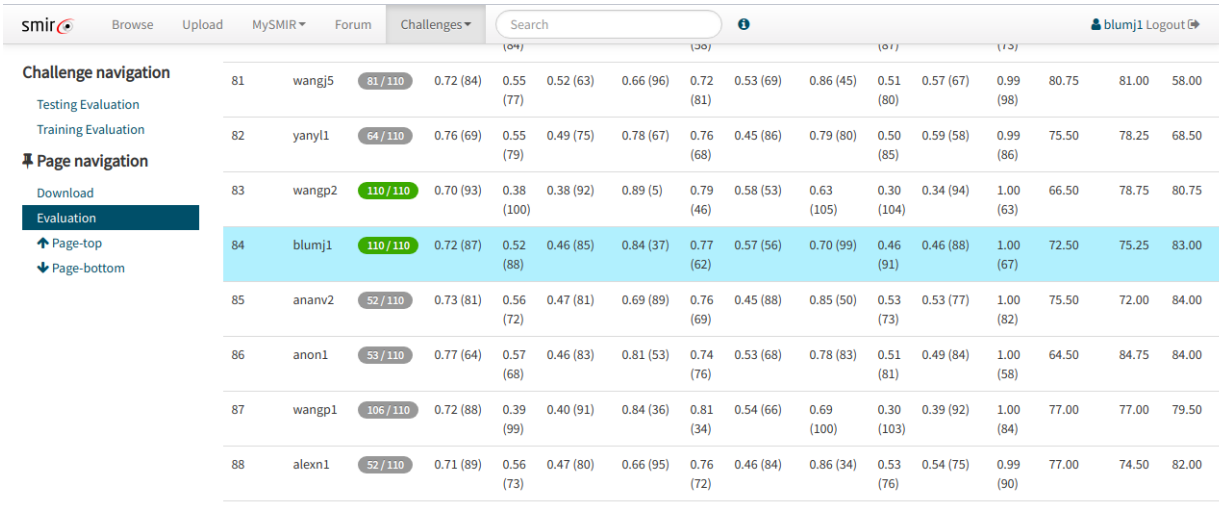


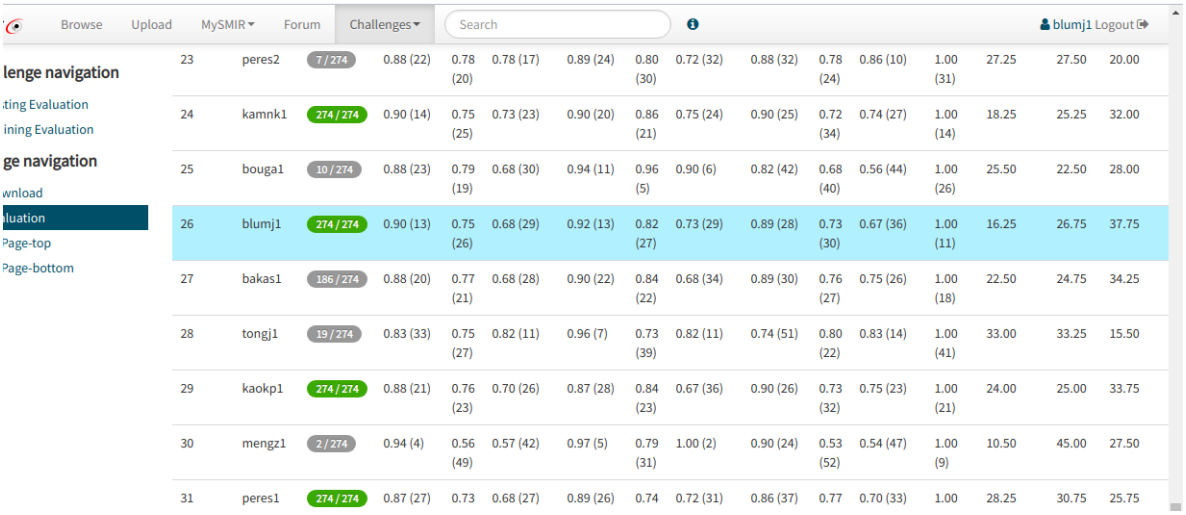
Figure 24: Enhance: Dice Cost, Hard Dice, IOU



The validation results closely track the training results and seem to indicate that the U-net successfully segments brain tumor images. Unfortunately, the results for the BRATS2015 test set submission to the contest were disappointing.



There was no systematic problem because the training set submission to the contest achieved results similar to what I attained at home.



I am currently working on possible improvements.

4.1.2 Strokes

Here are some sample plots using all of the data except seven patients reserved for testing.

Figure 28: Training without Validation Measures

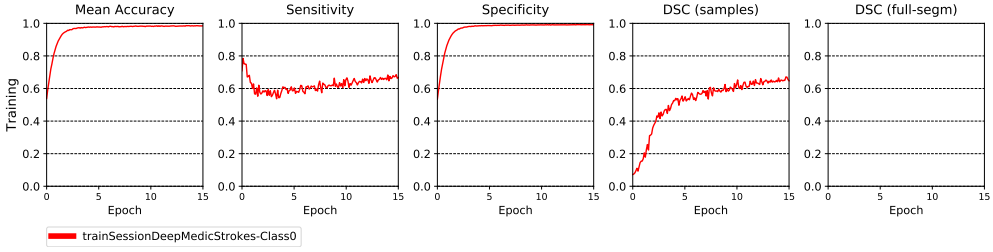


Figure 25: Training Cost and Validation Accuracy

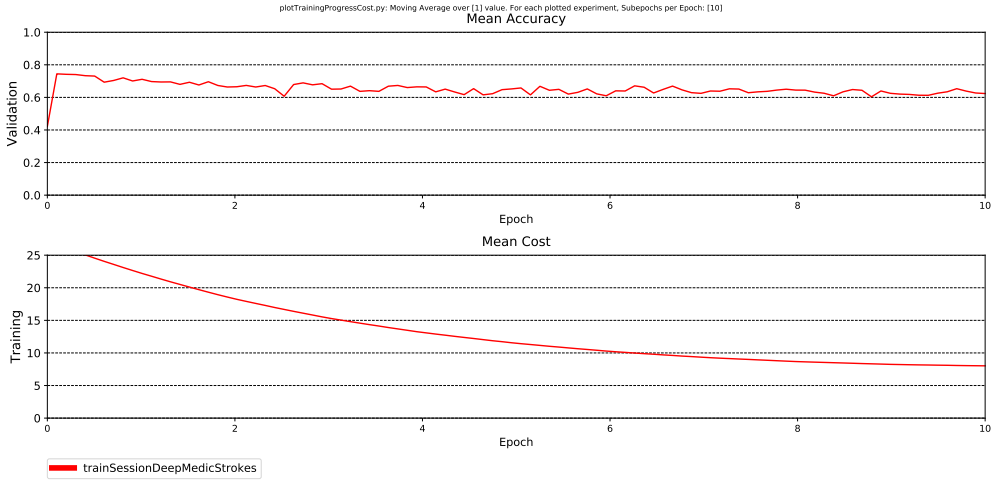


Figure 26: Training and Validation Measures

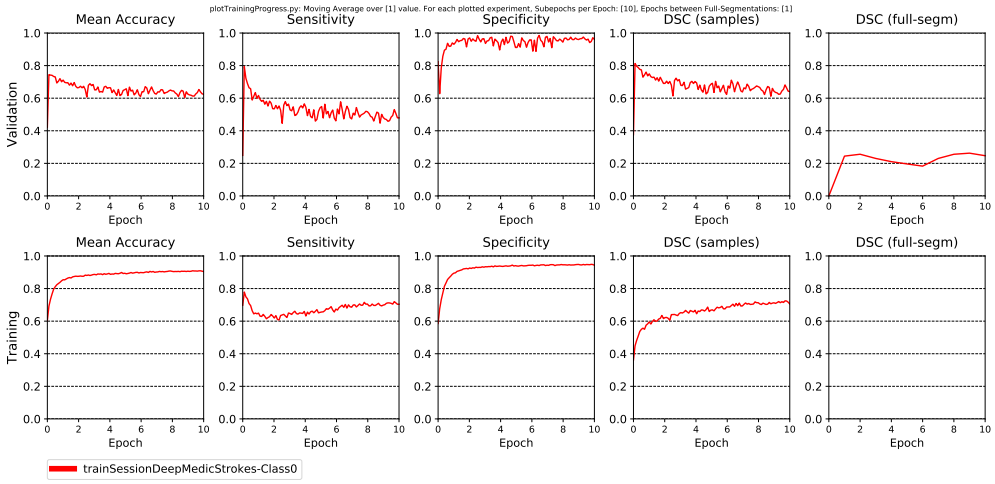
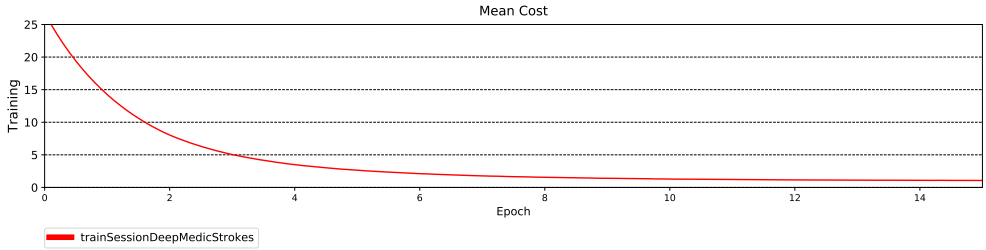
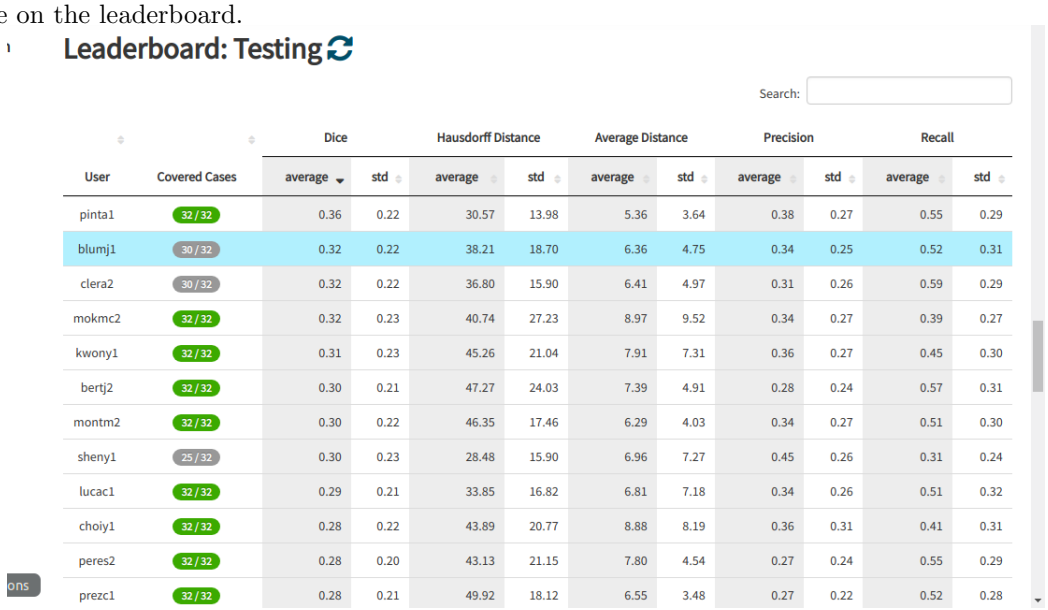


Figure 27: Training Cost without Validation



Note that I obtained better results by increasing the number of epochs from ten to fifteen. The best validation that the final model results are robust is my submission to the ongoing ISLES2017 contest. I submitted both the test results where no ground truth was provided and the training results. Note that the test set is different from what I used to train, validate, and test the deepmedic network. My test dice score tied the top result from the original contest and is

currently in the top five on the leaderboard. My test Hausdorff distance score is better than the top result of the original contest and is currently in the top five on the leaderboard.

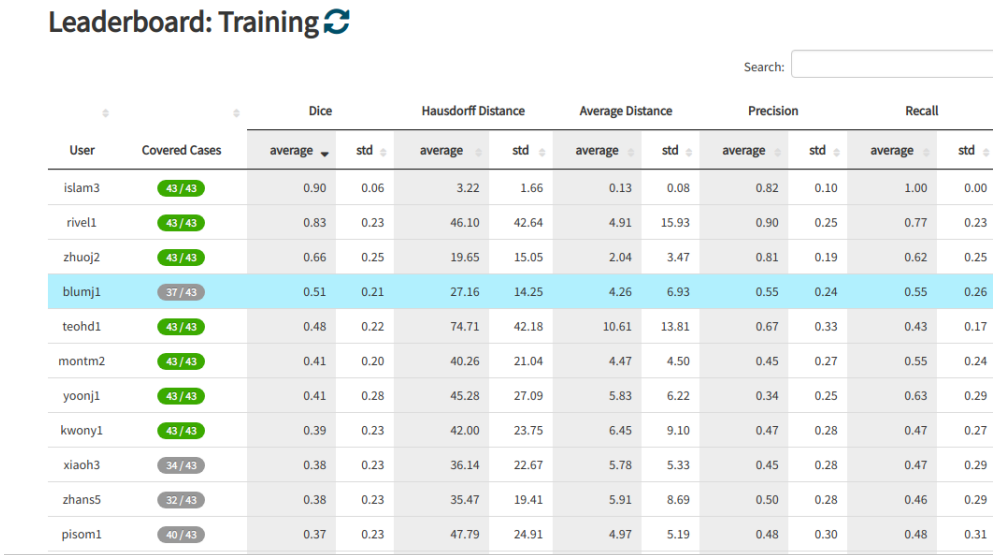


Leaderboard: Testing

Search:

		Dice		Hausdorff Distance		Average Distance		Precision		Recall	
User	Covered Cases	average	std	average	std	average	std	average	std	average	std
pinta1	32 / 32	0.36	0.22	30.57	13.98	5.36	3.64	0.38	0.27	0.55	0.29
blumj1	30 / 32	0.32	0.22	38.21	18.70	6.36	4.75	0.34	0.25	0.52	0.31
clera2	30 / 32	0.32	0.22	36.80	15.90	6.41	4.97	0.31	0.26	0.59	0.29
mokmc2	32 / 32	0.32	0.23	40.74	27.23	8.97	9.52	0.34	0.27	0.39	0.27
kwony1	32 / 32	0.31	0.23	45.26	21.04	7.91	7.31	0.36	0.27	0.45	0.30
bertj2	32 / 32	0.30	0.21	47.27	24.03	7.39	4.91	0.28	0.24	0.57	0.31
montm2	32 / 32	0.30	0.22	46.35	17.46	6.29	4.03	0.34	0.27	0.51	0.30
sheny1	25 / 32	0.30	0.23	28.48	15.90	6.96	7.27	0.45	0.26	0.31	0.24
lucac1	32 / 32	0.29	0.21	33.85	16.82	6.81	7.18	0.34	0.26	0.51	0.32
choiy1	32 / 32	0.28	0.22	43.89	20.77	8.88	8.19	0.36	0.31	0.41	0.31
peres2	32 / 32	0.28	0.20	43.13	21.15	7.80	4.54	0.27	0.24	0.55	0.29
prezc1	32 / 32	0.28	0.21	49.92	18.12	6.55	3.48	0.27	0.22	0.52	0.28

My training results are also in the top five on the leaderboard. The one problem with my solution is that a small number (2/32 for the test set) and (6/43 for the training set) yield empty segmentations for small but nonempty ground truth images. By adding a further preprocessing step, I could possibly alleviate this issue.



Leaderboard: Training

Search:

		Dice		Hausdorff Distance		Average Distance		Precision		Recall	
User	Covered Cases	average	std	average	std	average	std	average	std	average	std
islam3	43 / 43	0.90	0.06	3.22	1.66	0.13	0.08	0.82	0.10	1.00	0.00
rivel1	43 / 43	0.83	0.23	46.10	42.64	4.91	15.93	0.90	0.25	0.77	0.23
zhuoj2	43 / 43	0.66	0.25	19.65	15.05	2.04	3.47	0.81	0.19	0.62	0.25
blumj1	37 / 43	0.51	0.21	27.16	14.25	4.26	6.93	0.55	0.24	0.55	0.26
teohd1	43 / 43	0.48	0.22	74.71	42.18	10.61	13.81	0.67	0.33	0.43	0.17
montm2	43 / 43	0.41	0.20	40.26	21.04	4.47	4.50	0.45	0.27	0.55	0.24
yoonj1	43 / 43	0.41	0.28	45.28	27.09	5.83	6.22	0.34	0.25	0.63	0.29
kwony1	43 / 43	0.39	0.23	42.00	23.75	6.45	9.10	0.47	0.28	0.47	0.27
xiaoh3	34 / 43	0.38	0.23	36.14	22.67	5.78	5.33	0.45	0.28	0.47	0.29
zhans5	32 / 43	0.38	0.23	35.47	19.41	5.91	8.69	0.50	0.28	0.46	0.29
pisom1	40 / 43	0.37	0.23	47.79	24.91	4.97	5.19	0.48	0.30	0.48	0.31

5 Conclusion

5.1 Free-Form Visualization

5.1.1 Brain Tumors

The included file "visualize_tumor_seg.html" which shows a 2-dimensional slice of the third dimension of the segmented image next to the corresponding slice of the ground image for each patient in the training set indicates that the U-net has successfully segmented these images. The average dice scores for the all, core, and enhance segmentations are 90%, 75%, and 70% respectively which is the same as the result for the BRATS2015 contest except the enhance dice score was 68%.

5.1.2 Strokes

I enclose the file "visualize_stroke_seg-Copy1.html" which shows the slice of the third dimension with the maximum number of ones (evidence of stroke) from the segmented image alongside the corresponding ground image slice for each patient in the training set. A cursory glance at this file reveals that the stroke segmentation is considerably less successful than the brain tumor one, and the average dice score is 44%. The result for the ISLES2017 contest was 51%.

5.2 Reflection

In this project a two-dimensional U-net was used to segment three-dimensional MRI images of patients with brain tumors while a three-dimensional convolution neural network was used to segment MRI images of patients with strokes. The four types of tumor MRI images needed to be separately normalized to average zero and standard deviation one before implementing the U-net. The U-net is essentially a 2-d convolution neural network attached to a symmetric deconvolution net which enables the capture of local and large-scale features at the same time. The neural net needs to accurately label individual voxels but also learn large scale patterns in the areas of the brain affected by the tumor. Images were deformed before entering the U-net to increase the amount of data and take account of the irregular contours of tumors. A final machine learning step was added to distinguish between patients with high-grade and low-grade tumors. Brain masks were used in this step. Brain masks did not improve results in the U-net, and in some cases made them worse.

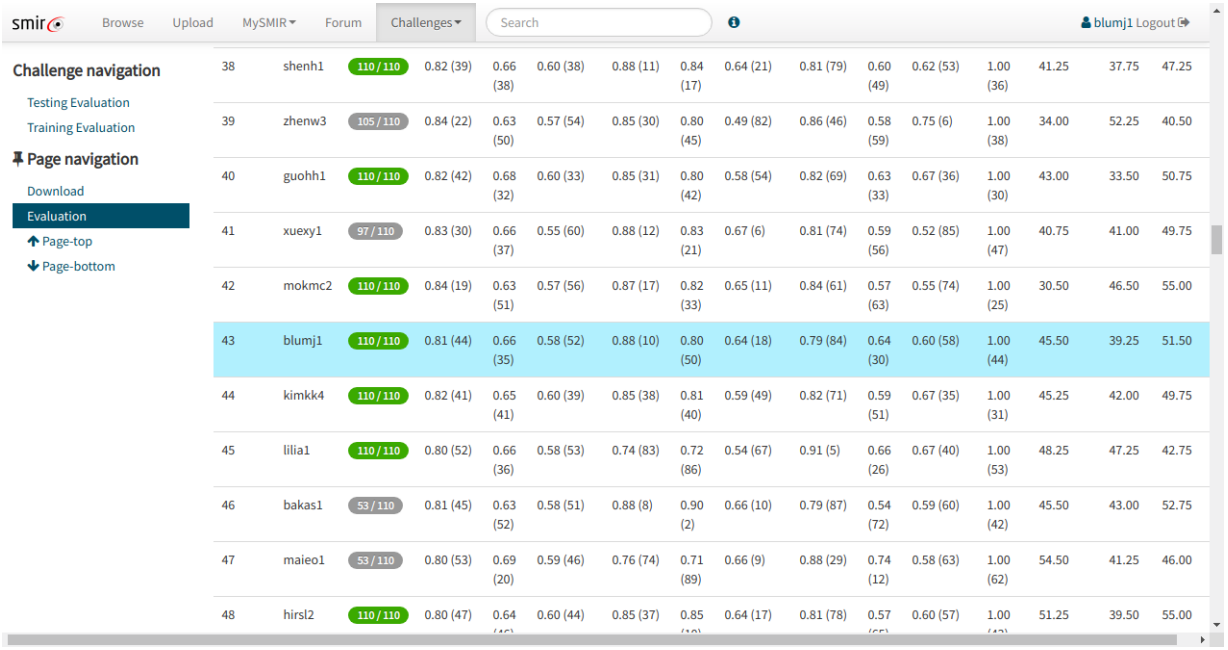
The stroke MRI's required further data preprocessing so that all images had the same voxel size. Computing brain masks improved the 3d CNN results. Again the seven types of MRI's needed to be separately normalized to norm zero and standard deviation one. A lot of experimentation was required to parameterize and improve the results of the 3d CNN. The 3d CNN contained two pathways (more were possible), one without rescaling of images, and another that shrunk the images to capture both the local and large-scale information. I added a final conditional random field to reduce noise leading to incorrect classification. A slower learning rate, stronger regularization, an asymmetric shrinking of images, and sampling a larger portion of the image for training improved the results as compared to the default parameters of deepmedic.

The U-net produced good results without a lot of experimentation but did not appear to be robust as the results for a different dataset, the BRATS2015 test set, were significantly worse. I am currently trying to ameliorate the U-net. The results using the 3d CNN on the brain tumor data were less successful than those of the U-net. On the other hand, while the 3d CNN produced better results for the stroke data than the U-net, the stroke segmentation is nowhere near ready for prime time. However, the 3d CNN seems robust since the results for the ISLES2017 test set were strong and similar to what I attained at home. Some of the images with very small evidence of stroke produced empty segmentations so a better method of preparing the data is needed. More importantly, a larger dataset would significantly improve the results. The size of the brain tumor dataset used for training was more than six times that of the stroke dataset.

5.3 Improvement

5.3.1 Brain Tumors

The disappointing results of the BRATS2015 test set have led me to consider some possible improvements to the U-net. As a first try I have replaced the plain U-net with one that includes batch normalization, increased the batch size from 10 to 15 which is the maximum possible without crashing my computer, and have foregone the last training on the portion of the training set that was used for validation. Although my all dice score decreased from .72 to .62, my core dice score increased from .52 to .65 and my enhance dice score increased from .46 to .53. This indicated that there was a problem with the edema label 2 in this improvement. I am currently getting some promising results with a second improvement that changes the loss function of the first improvement to a 50-50 split of dice loss and cross entropy loss. Since the cross entropy emphasizes accuracy, I thought that this might ameliorate the mislabeling of edema. The results for training are currently several points higher than my previous best results. After further training on the entire training set, my test set also improved in the contest with my all dice score .81, my core dice score .66, and my enhance dice score .58.



Ten more epochs of training increased the all dice to .82 but decreased the core dice to .65 and the enhance dice to .56 on the test set. Further improvement may be possible.

5.3.2 Strokes

To improve the stroke analysis significantly would necessitate training on a much larger patient dataset. However, I obtained good improvement by using the largest possible size for the whole image sampling. The limitation was my computer memory, and I would expect an even better result on a computer with enough memory to accomodate the whole image. The other problem concerned patients with little evidence of stroke with the resulting segmentations empty. Perhaps, a preprocessing step that enhances the contrast between low and high intensity parts of the image would help.

References

[1] Hao Dong, Guang Yang, Fangde Liu, Yuanhan Mo, and Yike Guo. *Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks*. 2017. URL: `https://arxiv.org/abs/1705.03820`.

[2] URL: `https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient`.

[3] Abdel Aziz Taha and Allan Hanbury. *Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool*. 2015. URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4533825/pdf/12880_2015_Article_68.pdf`.

[4] Abdel Aziz Taha and Allan Hanbury. *An Efficient Algorithm for Calculating the Exact Hausdorff Distance*. 2015. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7053955.

[5] Menze et al. *The Multimodal Brain TumorImage Segmentation Benchmark (BRATS)*. 2015. URL: https://hal.inria.fr/hal-00935640.

[6] Oskar Maier et al. *ISLES 2015 - A public evaluation benchmark for ischemic stroke lesion segmentation from multispectral MRI Medical Image Analysis*. 2016. URL: https://www.ncbi.nlm.nih.gov/pubmed/27475911.

[7] Kistler et al. *The virtual skeleton database: an open access repository for biomedical research and collaboration*. 2013. URL: http://doi.org//10.2196/jmir.2930.

[8] N Upadhyay and A D Waldman. *Conventional MRI evaluation of gliomas*. 2011. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3473894/.

[9] David C Preston MD. *Magnetic Resonance Imaging (MRI) of the Brain and Spine: Basics*. 2006. URL: http://casemed.case.edu/clerkships/neurology/Web%20Neurorad/MRI%20Basics.htm.

[10] URL: http://www.isles-challenge.org/.

[11] URL: https://github.com/zsdonghao/u-net-brain-tumor.

[12] URL: https://biomedia.doc.ic.ac.uk/software/deepmedic/.

[13] URL: https://www.tensorflow.org/install/install_sources.

[14] URL: http://tensorlayer.readthedocs.io/en/latest/user/installation.html.

[15] Konstantinos Kamnitsas, Christian Ledig, Virginia F.J. Newcombe, Joanna P. Simpson, Andrew D. Kane, David K. Menon, Daniel Rueckert, and Ben Glocker. *Efficient Multi-Scale 3D CNN with fully connected CRF for Accurate Brain Lesion Segmentation*. 2017. URL: https:arxiv.org/pdf/1603.05959.pdf.

[16] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. URL: https:arxiv.org/pdf/arXiv:1502.03167.pdf.

[17] Philipp Krahenbühl and Vladlen Koltun. *Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials*. 2012. URL: https://arxiv.org/pdf/1210.5644.pdf.

[18] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. *Conditional Random Fields as Recurrent Neural Networks*. 2016. URL: https://arxiv.org/pdf/1502.03240.pdf.

[19] URL: https://www.smir.ch/BRATS/Start2015.

[20] URL: https://www.smir.ch/ISLES/Start2017.

[21] Patrice Y. Simard, Dave Steinkraus, and John C. Platt. *Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis*. 2003. URL: http://cognitivemedium.com/assets/rmnist/Simard.pdf.

[22] Harry Ashkam. *Surface Distance Based Metrics*. 2018. URL: https://github.com/deepmind/surface-distance.

[23] Evangelia I. Zacharaki. *Classification of brain tumor type and grade using MRI texture and shape in a machine learning scheme*. 2009. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2863141/.

[24] URL: https://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html.