Medical Image Computing CAP5516 Assignment#02

Ashmal Vayani, UCF ID: 5669011

Deep Learning-based Brain Tumor Segmentation Using MRI

Training Code: The training and implementation code can be accessed **here**.

1 Task Visualization:

For the input task, I run four samples on ITKSoftware

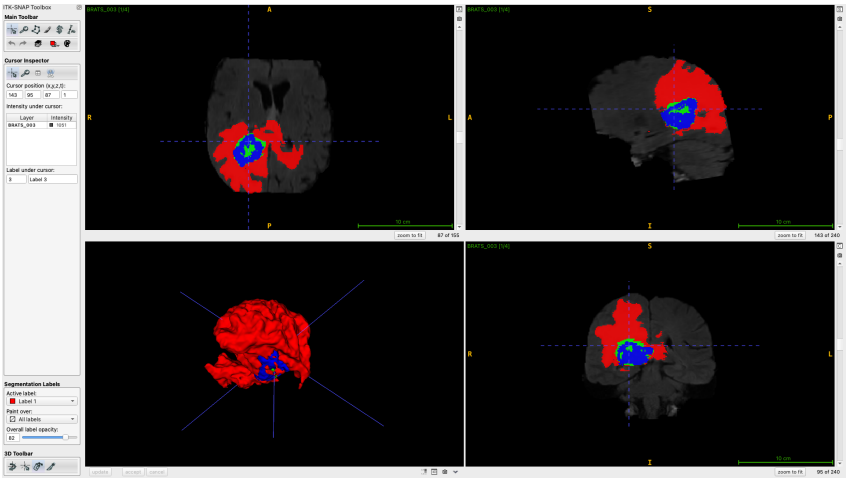

Figure 1: Visualization of input image and the segmentation mask.

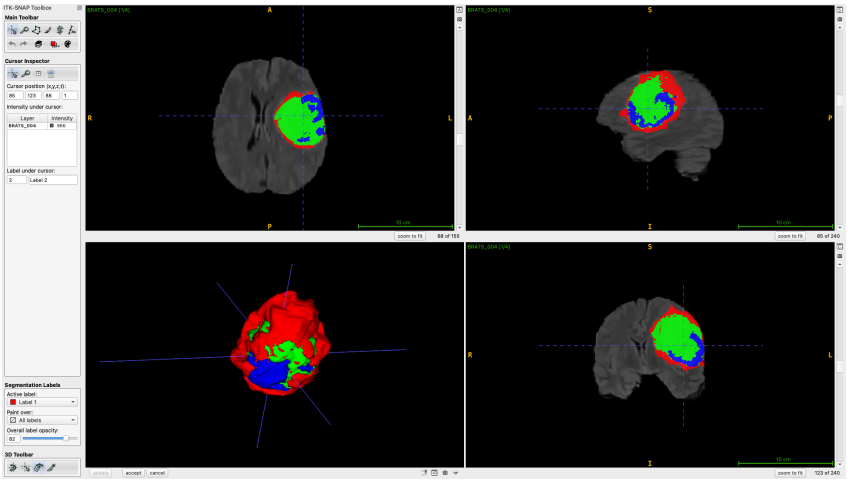

Figure 2: Visualization of input image and the segmentation mask.

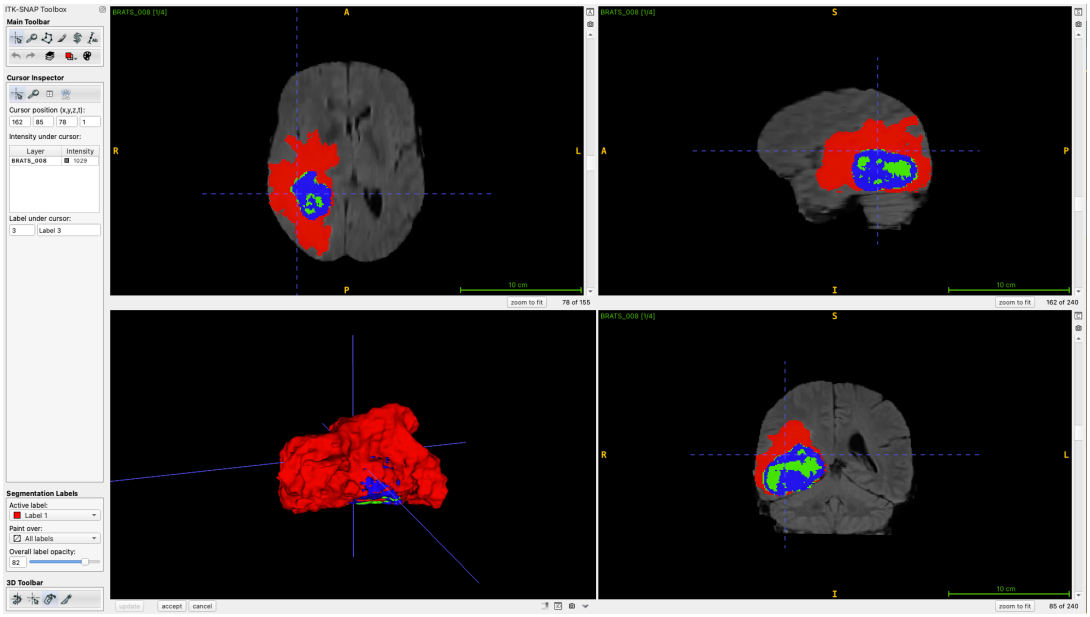

Figure 3: Visualization of input image and the segmentation mask.

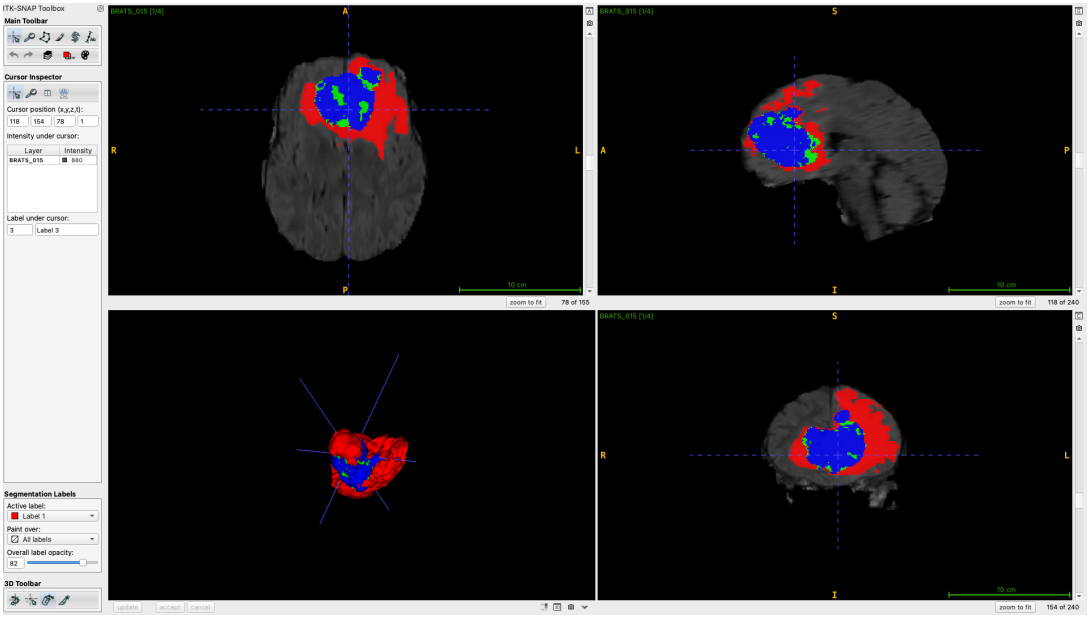

Figure 4: Visualization of input image and the segmentation mask.

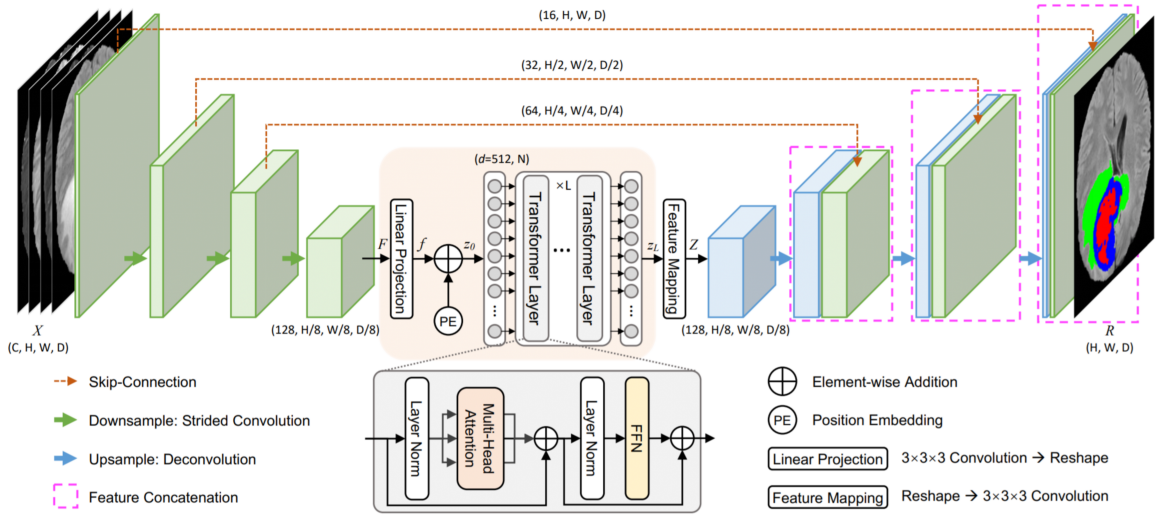

Figure 5: The figure presents the main architecture figure of the TransBTS framework.

2 Implementation Details:

In this assignment, I referred from the codebase of TransBTS repository and made adjustments as the code wasn't directly compatible with our sue case. It seems like the variant they were using was different than what we used to report our results. Next, we describe the detailed implementation approach that was used.

2.1 Intermediate Sequential Module

The **IntermediateSequential** module extends `nn.Sequential` to track intermediate activations during forward propagation. This allows debugging, feature visualization, and auxiliary outputs. Unlike standard sequential models that return only the final output, this module stores and provides feature maps from different network stages, improving interpretability and control over intermediate representations.

2.2 Positional Encoding

Since Transformers lack inherent spatial understanding like CNNs, **positional encoding** provides relative spatial context. The model employs *fixed sinusoidal positional encodings*, precomputed based on the position index and added to input embeddings. These encodings help retain spatial relationships, which is crucial for segmentation tasks.

2.3 Transformer Encoder

The model's core is a **multi-head self-attention Transformer encoder** that captures global dependencies across input features. Unlike CNNs with local receptive fields, this encoder enables long-range interactions. Each encoder layer consists of:

- **Multi-head self-attention**: Computes attention scores across feature locations, allowing focus on relevant regions.
- **Feed-forward network**: Applies non-linear transformations for enhanced feature representation.
- **Layer normalization and residual connections**: Ensures stable training and smooth gradient flow.

2.4 Query Token Mechanism

The model utilizes **query-based representation**, where learnable query tokens interact with image features via self-attention layers. These tokens condense feature representations, later used for predictions.

The number of query tokens is a hyperparameter determining the granularity of feature extraction.

2.5 Segmentation Head

Final segmentation predictions are generated using a **decoder head**, which refines Transformer encoder outputs and upscales them to match the original resolution. This module includes:

- **Deconvolution layers** for upsampling feature maps.
- **Linear projections** mapping features to class probabilities.
- **Softmax activation** to generate final segmentation masks.

2.6 Loss Function and Training Strategy

The model is trained using a combination of:

- **Cross-entropy loss** for multi-class segmentation.
- **Dice loss** to handle class imbalance and improve boundary precision.
- **Adam optimizer with learning rate scheduling** to optimize convergence.

Training involves **data augmentation techniques** such as flipping, rotation, and color jittering to improve generalization performance.

3 Dataset Pre-processing:

3.1 5-fold Evaluation:

To ensure robust model evaluation, we implement a 5-fold cross-validation strategy to split our brain tumor data set.

- **Dataset Splitting**: The data set is divided into five subsets (folds), ensuring diverse training and testing samples.
- **Shuffle for Randomness**: Files are shuffled before splitting to prevent bias.
- **Training & Testing**: Each fold uses 4 subsets for training and 1 for testing, ensuring that all samples contribute to model evaluation.
- **Balanced Evaluation**: Reduce variance and provide a more reliable assessment of model performance.
- **Reproducibility**: A fixed random seed ensures consistent and repeatable splits across different runs.

3.2 Dataset Preprocessing:

Since the initial files are not in the required format directly, we have to first convert them into the *pickle* files to be able to use them in training.

- The script processes MRI brain tumor images for deep learning tasks.
- It loads a dataset JSON file that contains paths to training images and labels.
- Full file paths for images and labels are extracted and structured properly.
- MRI images are loaded and converted into a standardized format.
- The images undergo z-score normalization, adjusting intensity values while preserving background regions.
- Normalized images and labels are saved as pickle files for efficient loading.
- The processed files are organized for training and evaluation in medical image segmentation.

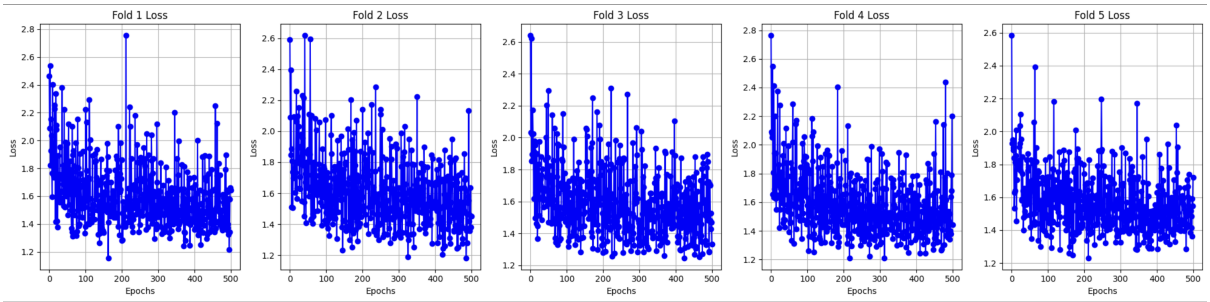

Figure 6: The above graphs shows the training loss for 500 epochs on each fold.

4 Training

The training is done on the below parameter setting:

Category	Argument	Description
Basic Information	`--user`	Name of the user performing the experiment
	`--experiment`	Experiment name
	`--date`	Date of the experiment
	`--description`	Description of the experiment
DataSet Information	`--root`	Path to the root directory of the dataset
	`--train_dir`	Directory containing the training images
	`--train_file`	File containing the list of training data
	`--dataset`	Name of the dataset
Model & Input Configuration	`--input_C`	Number of input channels (e.g., for images)
	`--input_H`	Height of the input image
	`--input_W`	Width of the input image
	`--input_D`	Depth of the input image
	`--num_class`	Number of output classes (e.g., tumor segmentation classes)
Training Hyperparameters	`--batch_size`	Number of samples per batch
	`--num_workers`	Number of workers for data loading
	`--lr`	Learning rate for optimization
	`--weight_decay`	Weight decay for regularization
	`--amsgrad`	Whether to use AMSGrad optimizer variant
	`--criterion`	Loss function used for training
	`--seed`	Random seed for reproducibility
GPU & Resource Configuration	`--gpu`	GPU device to use (comma-separated for multiple GPUs)
Training Time Control	`--start_epoch`	Epoch to start training from (for resuming)
	`--end_epoch`	Epoch to end training at
Model Checkpoints & Output	`--resume`	Path to resume a checkpoint from (optional)
	`--load`	Whether to load from checkpoint
Cross-Validation	`--k_fold`	Fold number for k-fold cross-validation

5 Results

For our results, we present our values on two different metrics:

Dice Scores: The Dice The Dice Similarity Coefficient (DSC) measures the overlap between predicted and ground truth tumor segmentations. It is computed separately for Enhancing Tumor (ET), Tumor

Core (TC), and Whole Tumor (WT) by comparing binary masks for each class. A higher Dice score indicates better segmentation accuracy.

Hausdroff Score: The 95th percentile Hausdorff Distance (HD95) evaluates boundary mismatch between predicted and ground truth segmentations. It focuses on the 95th percentile of distances, reducing outlier influence. Using extracted surface voxels, HD95 is computed for ET, TC, and WT, providing insights into boundary accuracy.

Next, we present the results for each fold in the below table.

Fold	HD (ET)	HD (TC)	HD (WT)	Dice (ET)	Dice (TC)	Dice (WT)
fold_1	5.56	5.98	6.43	0.7673	0.8125	0.8749
fold_2	6.22	5.20	6.21	0.7485	0.8246	0.8986
fold_3	8.46	4.92	5.44	0.7207	0.8270	0.9064
fold_4	7.34	5.09	5.50	0.7895	0.8208	0.8856
fold_5	7.12	5.19	6.30	0.7704	0.8188	0.8870
Average	6.94	5.48	5.97	0.7593	0.8207	0.8905

Table 2: Hausdorff Distance (HD) and Dice Score for different folds. HD measures the maximum surface distance between predicted and ground truth segmentations, while the Dice Score quantifies overlap between them. ET = Enhancing Tumor, TC = Tumor Core, WT = Whole Tumor.

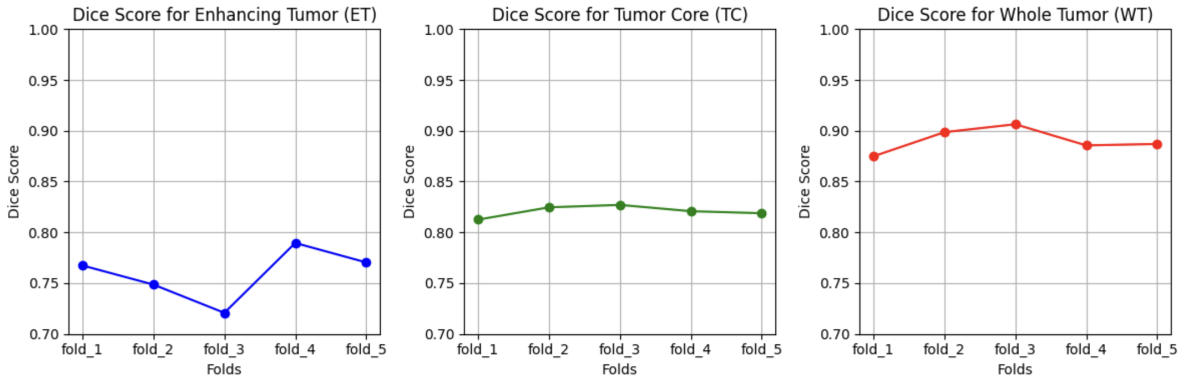

Figure 7: This row presents the Dice scores for three tumor regions: Enhancing Tumor (ET), Tumor Core (TC), and Whole Tumor (WT) across five different folds. The Dice score represents the overlap between the predicted and ground truth segmentation, with higher values indicating better performance.

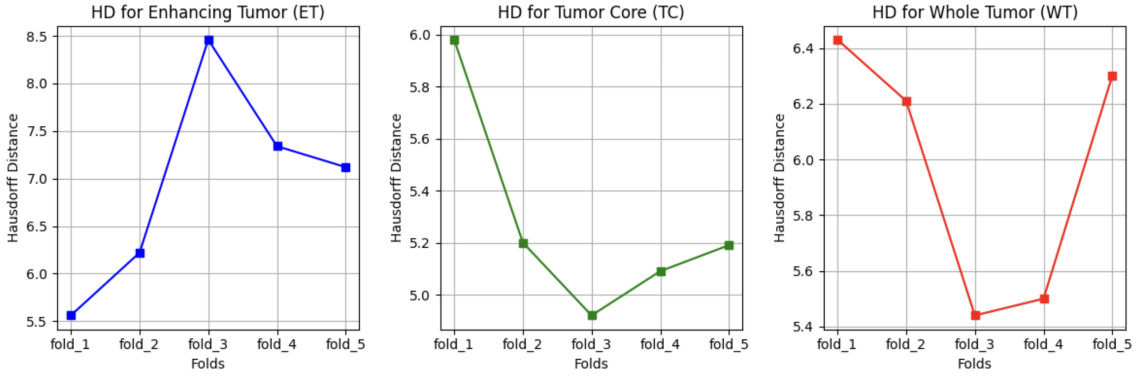

Figure 8: This row shows the Hausdorff Distance (HD) for Enhancing Tumor (ET), Tumor Core (TC), and Whole Tumor (WT) across five folds. The HD measures the worst-case boundary mismatch between predictions and ground truth, with lower values indicating better accuracy in capturing tumor boundaries.

6 Results Visualization

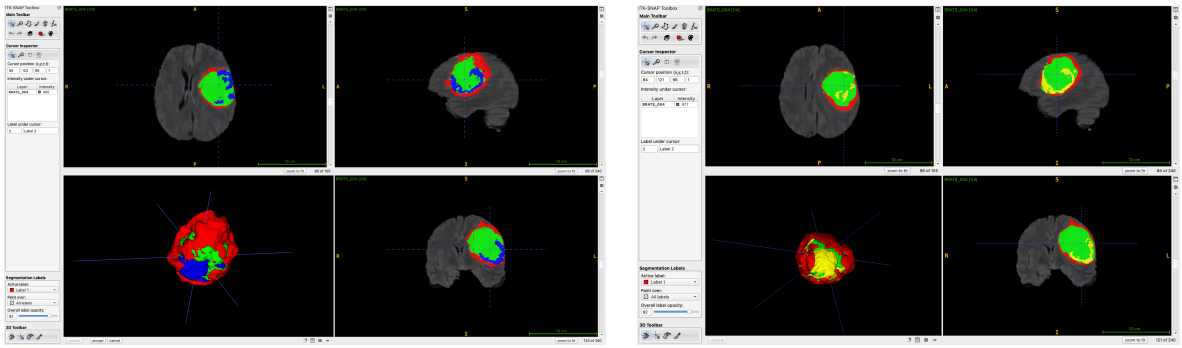

Figure 9: Left: Ground Truth with given image and segmentation mask. Right: Predicted segmentation values for 5th fold.

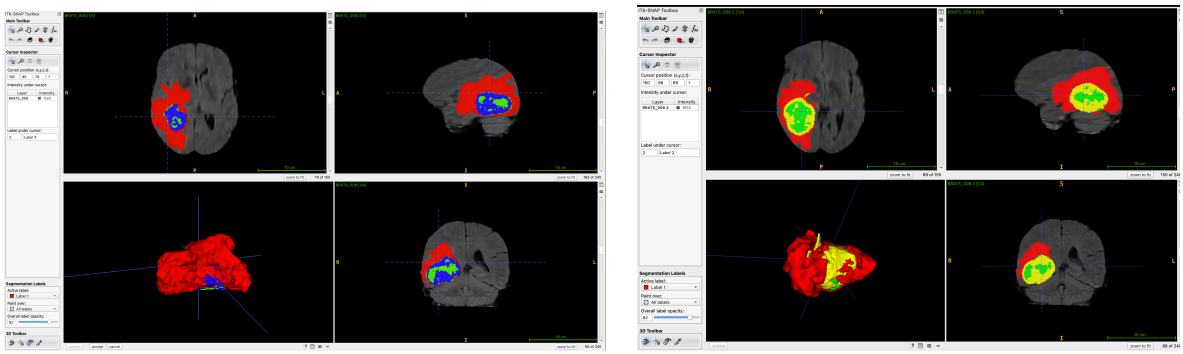

Figure 10: Left: Ground Truth with given image and segmentation mask. Right: Predicted segmentation values for 5th fold.

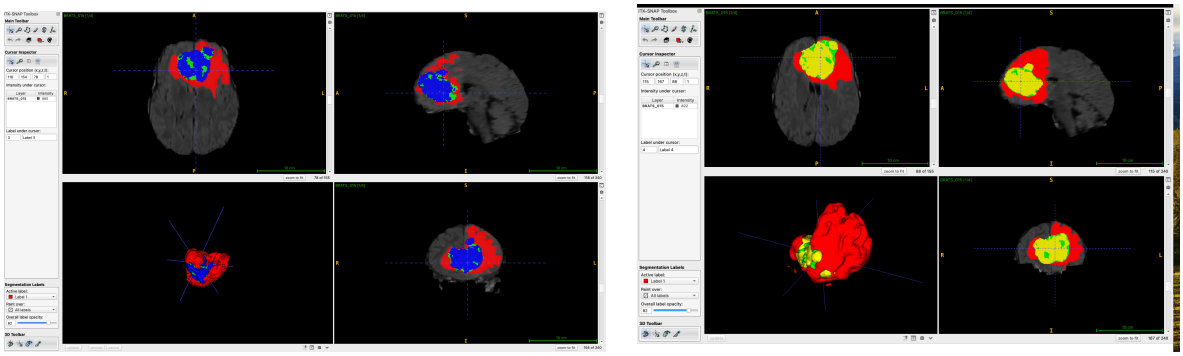

Figure 11: Left: Ground Truth with given image and segmentation mask. Right: Predicted segmentation values for 5th fold.