Tutorial: Using Transformers to Segment Glioblastoma Multiforme (GBM)

Keith Bova

Department of Electrical and Computer Engineering
The University of New Mexico

UNM ECE 533
May 18, 2025

Outline

Outline

Introduction

Motivation

Digital image processing is a powerful field with interesting applications across many disciplines.

Why do we care?

Fewer than 5% of people diagnosed with GBM survive more than five years [TJ17].

What is the goal?

The goal is to recreate the results from the UNETR++ paper and learn about how loss functions and custom datasets were used with 3D U-Net to accurately segment GBM.

Introduction: Literature Review

Image segmentation is necessary

Binary classifiers and object detection is good, but segmentation is better for GBM detection [Ba24].

There are ethical concerns

Bonada et al. discuss how the integrity of a dataset and the ability to make unbiased inferences is important. [Ba24]

The properties of the tumor are important

Porz et al. discuss how classifiers should prioritize characteristics of the afflicted tissue (necrotic, quiescent, etc.) [Por+14]

Supervised learning is the most ideal

Computers can segment GBM using supervised and unsupervised learning, with the former outperforming the latter in terms of accuracy [JA+15].

Outline

Metrics for Image Segmentation

Hausdorff Distance

The longest distance that a person can be forced to traverse between two finite sets, whereby they must travel from one set into another [HKR93]. The *Hausdorff Distance* draws the boundaries between cancerous and noncancerous cells in GBM segmentation.

Dice

Defines the ratio of the set of pixels in a prediction to the set of pixels in a ground truth. A perfect dice is 1.0.

Intersection Over Union (IoU)

IoU is similar to dice, but is calculated by dividing the area of overlap by the area of the union. Similar to dice, a perfect IoU is 1.0.

Existing Methods

3D U-Net [Wol+20]

A convolutional neural network (CNN) designed to perform volumetric segmentation on 3D data.

UNETR++ [Sha+24]

A transformer based model that implements 3D U-Net that should provide higher accuracy than a convolutional neural network.

UNETR++ using SWIN Transformers: link here

One of the most accurate modern techniques for GBM volumetric segmentation.

What They Missed

Simplicity

The README of UNETR++ describes a build process that is not up to date. Also, the repo requires re-training to generate missing files. Overall, it is difficult to build from source. The evaluation script in UNETR++ returned an error for both the CPU and GPU memory space.

Accuracy

The UNETR++ source code says it is using un-labled data for validation. This might be corrected elsewhere; however, I think this is a bug. Best accuracy was 0.79600 with $dice = \begin{bmatrix} 0.839 & 0.665 & 0.859 \end{bmatrix}$ at epoch 958.

Visualization

The UNETR++ repository doesn't directly describe in-depth a way to visualize the results (a github issue describes 3DSlicer).

Outline

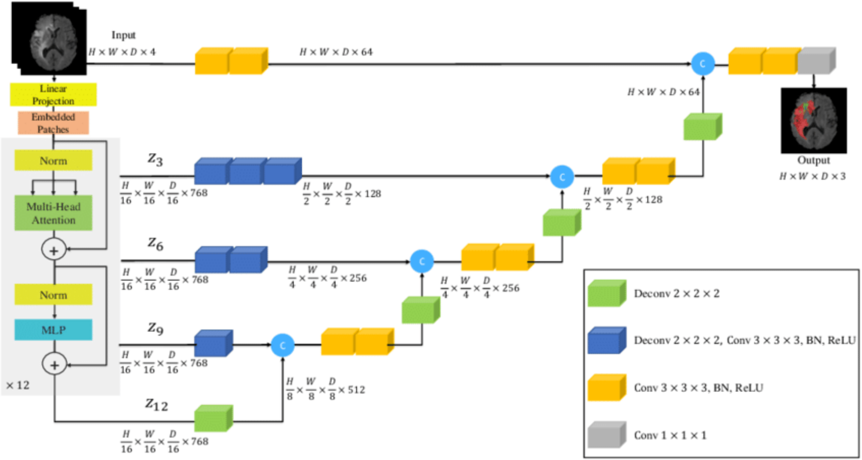

Figure 3.1: System diagram for the *UNETR++*, as defined by [Sha+24]

Custom Datasets

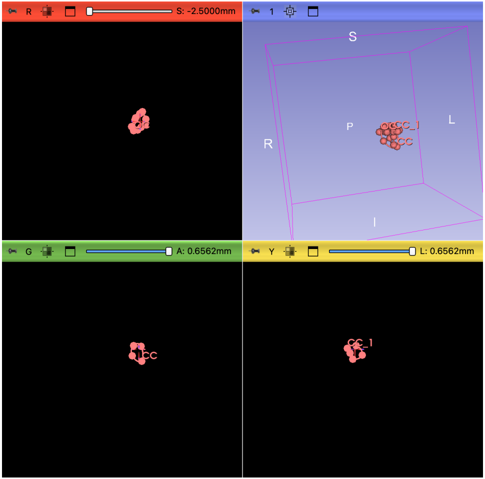

Figure 3.2: Example of how the 3DSlicer program can be used to generate a volume in three dimensions for data labeling with UNETR++.

Loss and Performance

Node	Accelerator	Using CUDA	Learning Rate	Training Loss	Validation Loss	Theoretical Run Time
AMD Opteron 4284	GTX 1060	No	0.009991	-0.4399	-0.4194	243.62 days
Apple Mac M3 Pro	Internal	N/a	0.009991	-0.438	-0.4477	65.63 days
i9 9900k	P100 16GB	No	0.009991	-0.4399	-0.4338	85.38 days
i9 9900k	P100 16GB	Yes	0.009991	-0.4372	-0.4703	3.58 days
Xeon Platinum 8528y	H200 141GB	Yes	0.009991	-0.4427	-0.4319	1.85 days
Xeon Platinum 8470	H100 80GB	Yes	0.009991	-0.444	-0.4418	2.26 days
Xeon Silver 4314	A100 80GB	Yes	0.009991	-0.4465	-0.4223	4.94 days
Xeon kvm	A100 40GB	Yes	0.009991	-0.4462	-0.4042	3.36 days
Xeon Gold 6130	V100 32GB	Yes	0.009991	-0.4415	-0.4183	2.66 days

Table 3.1: Loss and performance at EPOCH 0 for each tested node. Actual training took 3.46 days ($24x$ increase using the CUDA compiler optimization!) on a system that I built. When using the CUDA compiler optimization, the memory footprint was roughly consistent at about $\approx 10gb$ in each GPU. The number of CPU cores and CPU memory varied when compiled for the CPU memory space.

Loss and Performance

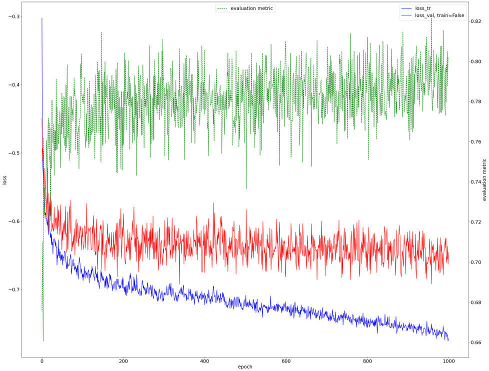

Figure 3.3: Training progress. Red is validation loss. Blue is training loss. Green is the evaluation metric.

Outline

Side by Side Comparison: Training Dataset

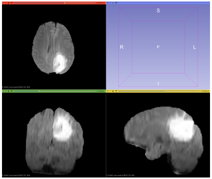

Figure 4.1: Task003_tumor-imagesTr-BRATS_397_0000

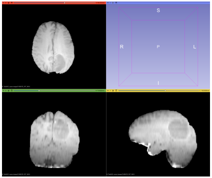

Figure 4.2: Task003_tumor-imagesTr-BRATS_397_0001

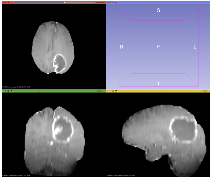

Figure 4.3: Task003_tumor-imagesTr-BRATS_397_0002

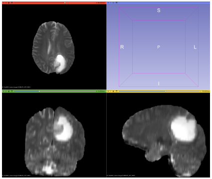

Figure 4.4: Task003_tumor-imagesTr-BRATS_397_0003

Side by Side Comparison: Results

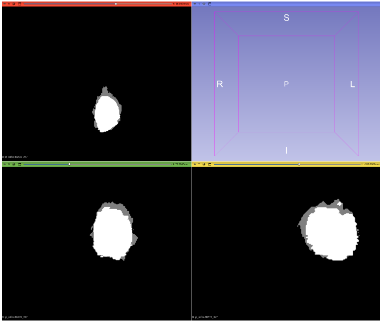

Figure 4.5: ground truth: gt_niftis-BRATS_397

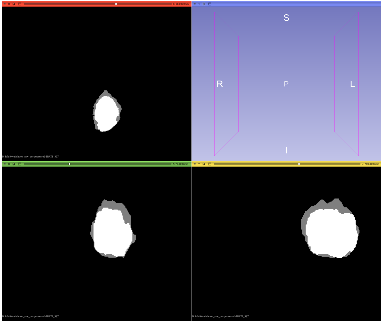

Figure 4.6:
fold-0-validation_raw_postprocessed-BRATS_397

Outline

Discussion

Results

GBM segmentation is a very difficult task and necessitates a high degree of precision. The GPU memory space is desirable for digital image processing because of their wide vector registers and large memory bandwidth.

Other Methods

Unetr++ is a good code with a high degree of precision. It is substantially better than 2D techniques, but lags behind SWIN Transformers.

Loss Functions

Unetr++ has four primary loss functions that include cross-entropy, deep-supervision, dice, and TopK.

Outline

Conclusion

Remarks

Volumetric GBM segmentation is possible with the use of transformers. Using modern GPU technology, results can be obtained in a reasonable time.

Accomplishments

Removed CUDA and Anaconda dependency. Substantially streamlined the install process. Calculated estimated runtime for an assortment of nodes. Trained network and obtained results in qualitative agreement with the paper.

Future Work

Recreating the results from this paper about SWIN Transformers.