Brain Tumor Segmentation with Deep Neural Networks

Joshua Crowther
Georgia Institute of Technology
jcrowther9@gatech.edu

Lukas Dauterman
Georgia Institute of Technology
ldauterman3@gatech.edu

Barry Cheng
Georgia Institute of Technology
barrycheng@gatech.edu

Julio Moscon
Georgia Institute of Technology
jmoscon3@gatech.edu

Abstract

Identifying brain tumors in MRIs is a critical and time-consuming task for radiologists, taking days of a radiologist's time to identify possible cancers. In order to reduce the time it takes to address these issues, we are training deep neural networks on BRaTs MRIs to segment tumors within the volumes. In this study, we apply Unet, DeepLabV3+, and our own custom network on 3D image segmentation tasks. Upon thorough experimentation, we found that our custom network performed the best on all tracked metrics, Unet had the lowest false positive rate, and DeepLabV3+ had resolution issues. Code for this study can be obtained at https://github.com/jshcrowthe/tumor-segmentation.

1. Introduction

Radiologists are a finite and expensive resource for identifying brain tumors in MRIs [9]. Can we train a deep neural network to predict tumors in MRIs faster and more accurately than a radiologist? The time saved by using a model could mean the difference between the patient's life or death.

To attempt this, we trained Unet, DeepLabV3+, and a custom model to identify gliomas and related tissue disorders in MRIs.

1.1. MRI Background

MRIs are used to get detailed 3D image of patients brains. Radio waves stimulate radio emissions of water molecules in the brain in a strong magnetic field to generate MRIs. The pulse frequency of the radio wave results in image types base on tissue characteristics. High frequency pulses are T1 images, medium frequency are T2 images, and low frequency are FLAIR images [12].

Radiologists choose the MRI type depending on the cancer and use the volumes to examine slices and identify tumors [9].

1.2. Data

We used the BRaTs 2021 dataset [2], created by The Radiolocial Society of North America to drive research on glioma segmentation, to train our models. The MRIs were labeled by one of 4 radiologists and use the outputs of previous best models. These MRIs were aligned and reshaped to match dimensions. The data set contains 1251 subject MRIs with T1, T2, FLAIR, and segmentation images saved as $.nii.gz$ images, one per image type per subject, with labels 0 for no tumor, 1 for necrotic core, 2 for invaded tissue, 3 for whole tumor (omitted), and 4 for GD-enhancing tumor. These MRIs have had their skulls striped, anonymizing the patients.

We used the IXI dataset [8] as a test set to measure the false positive rate and calibrate model sensitivity. This data is made available by the Imperial College of London and contains 600 T1 and T2 MRIs from healthy individuals. These MRIs have not been skull-stripped and can potentially be used to reconstruct the patient's identities.

The common pre-processing steps for the MRIs are to first convert to the common format, down sampling the image, reshape to (48, 64, 48), then normalizing it by the histogram of values. The training data is augmented by applying random motions, bias field, noise, flips, Affine, and Elastic Transformations.

2. Approach

2.1. Frameworks

We utilized the TorchIO [11] package for data-loading. TorchIO contains a collection of useful functions for reading and augmenting MRIs for different tasks. Because these MRIs were provided as a Kaggle dataset without a default data loader, we extended the TorchIO package and created methods for reading and preprocessing the BRaTs dataset

and for training, validating, and testing the trained DNNs. We also utilized TorchIO's built in method for downloading and loading the IXI dataset.

For defining the data-loaders and the training methods we used PyTorch Lightning [4], with PyTorch [10] as a general framework for defining flexible training loops. PyTorch Lightning requires that we implement our own data-loaders using their method hooks, and that we define how training is performed on each iteration. It removes the need for us to manage where the data is processed and loaded on device, allowing us to focus on experiments. It also handled data logging and hyperparameter storage of each model run. We used PyTorch to implement the models, loss functions, and metrics.

2.2. Models

The DeepLabV3+ model is an updated version of DeepLabV3 [7]. The architecture was improved by the authors of the original model by adding an encoder-decoder structure. The encoder in DeepLabV3+ is similar to DeepLabV3 but with the addition of the Separable Atrous Convolution. The Decoder consists of up-sampling operations to upscale the output of the encoder. In our DeepLabV3+ implementation, we replaced all 2D operations with 3D equivalent operations and implemented a 3D version of ResNet-101 as the network backbone.

The Unet model [3] was first introduced for 2D biomedical image segmentation tasks as an improvement upon the sliding window approach to localize and label regions around each pixel. Unet utilizes a contracting path to increase output resolution and an expanding path to combine results from the contracting path to localize high resolution features. 2D Unet has provided good performance on biomedical segmentation tasks when combined with data augmentation. In this study, we implemented 3D Unet to experiment whether the strong 2D segmentation performance also applies to image volumes.

The custom model was based off of an existing 3D Unet implementation (see [18]). Given the Unet base, many of the characteristics of the Unet model are present in the custom model as well. Unet is designed with max pooling layers in between stacks of convolutional layers. The custom model explores the effect of different aggregation and generalization techniques. This model uses a combination of max pooling, average pooling, as well as 3D dropout layers to attempt to improve gradient flow and better propagate feature representation throughout the network.

2.2.1 Custom Model

Both DeepLabV3+ and Unet have well documented architectures that can be explored in [1] and [14] respectively. The custom model used for these experiments, while taking queues from both models, is distinct from both.

In convolutional neural networks, pooling is used as a down-sampling methodology that helps with overfitting while also improving computational speeds by reducing the dimensionality of the inputs. There are a number of different pooling strategies that approach the task from mathematically unique perspectives. For this effort several methods explored in [5] were considered. However, to simplify the optimization space, due to time restrictions, only max pooling and average pooling were used.

As with Unet, the custom model uses stacks of convolutional layers as encoding layers each separated by pooling layers, each encoding stack increasing the channel depth of the previous stack. The first two encoding layers use max pooling, while the second two use average pooling. Dropout layers are used at several transitory parts of the network to facilitate feature generalization. The architecture can be seen in Figure 1.

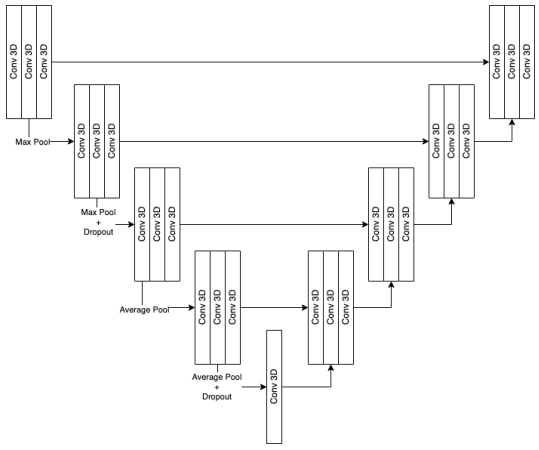

Figure 1: Custom network model architecture

2.3. Methodology

2.3.1 Hyperparameter Selection

We used Dice Loss, Log Cosh Dice Loss, and Focal Loss to optimize the models. Dice Loss [6] is defined as

$$L_{DC}(y,p) = \frac{TP(y,p) + \epsilon}{TP(y,p) + .5 * FP(v) + .5 * FN(y,p) + \epsilon} \quad (1)$$

where TP, FP, FN are approximations of the true positive and false positive and negative rates, and ϵ is a smoothing factor. These are calculated by taking the sum of the one hot encoding of the segment mask and multiplying by the predictions such as $TP = \text{sum}(p * y)$, $FP = \text{sum}(p * (1 - y))$, and $FN = \text{sum}((1 - p) * y)$. This function is not directly convex and may have issues during training.

To address this it has been proposed that the Log Cosh [6] of the function can be used to smooth out the non-convex nature of Dice Loss.

$$L_{LC}(y, p) = \log(\cosh(L_{DC}(y, p))) \tag{2}$$

Another potential issue is overfitting to the classes that have better representation in the data. For cross-entropy, this can be done by adjusting the equation to only learn well on examples where is is less confident, hence Focal Loss [6]:

$$L_{FL}(V) = \frac{1}{|V|} \sum_{v \in V} \sum_{j=0}^{|C|-1} y(C_j|v)(1 - p(C_j|v))^{\gamma} \log(p(C_j|v)) \tag{3}$$

where C is the set of class, V is the set of pixel elements in a volume, v is a pixel in V, y is the label if it is that class or not, and C_j is a specific class. This has the advantage of being a convex loss function but does not directly optimize for matching the segmentation.

For the optimizers we chose Adam, Adagrad and RMSprop. Adagrad decays learning rate based on the update history, RMSprop normalizes the update step by the size of the gradient, and Adam applies a moving average to both what Adagrad and RMSprop do.

For the learning schedules, Exponential decay is frequently used for segmentation. Cosine annealing is a frequently used for hard to train networks that might not be convex optimization spaces. The Step learning rate schedule is simple and serves as a baseline to measure the performance of the other learning rate schedules.

For the data types, we did not consider linear combinations due to the fact that T1,T2, and FLAIR focus on different tissue response times, and averaging them would cause us to lose important contrastive details.

2.4. Success Metrics

Pixel accuracy [15] can be used to measure the accuracy of the model to predict the classes for each pixel in a volume. This can be done by taking the $argmax$ of the prediction at each pixel, checking if matches ground truth, then taking the average across the volume.

$$acc = \frac{1}{N|V|} \sum_{i=0}^{N-1} \sum_{v \in V} y(v|C_i) == P(v|C_I) \tag{4}$$

Mean intersection over union (mIoU) [15] can be used to tell how well the predicted segmentation for each class intersects with the ground truth. This is done as follows:

$$mIoU = \frac{1}{N} \sum_{i=0}^{N-1} \frac{|C_i \cap P_i|}{|C_i \cup P_i|} \tag{5}$$

Where C_i is the set of pixel with labeled ground truth for class i, and P_i is the model predictions for that class. Another measure of the class intersection is the Dice coefficient [15] :

$$dice\_coef = \frac{1}{N} \sum_{i=0}^{N-1} \frac{2|C_i \cap P_i|}{|C_i| + |P_i|} \tag{6}$$

Dice coefficient is correlated with intersection of union, but it is another metric for measuring performance and can be another way to distinguish if one model is performing better than another.

2.4.1 Hyperparameter Tuning

There are a variety of different hyperparameters that we could optimize in this scenario. By limiting the search space to only the hyperparameters mentioned above, we were able to perform a comprehensive search of the desired parameter space. To do this we implemented grid search over the optimizers, learning schedule, data types, and loss function spaces.

This resulted in 81 different hyperparameter permutations. Given the variance in results, the best model was decided by how well it performed across all tracked metrics. We were fortunate that this resulted in a single set of parameters that outperformed multiple metrics. Considerations were not made for the case of what constituted the "best" model if no single set of parameters dominated the metrics.

2.4.2 Issues Encountered

We did not end up using a Pixel Shuffle [16] or Depth to Space [17] model. These methods did not generalize well to 3D data and the models always had the metrics drop to zero, meaning that backpropagation was not working well for the task.

During early training attempts, the model outputs did not predict anything besides the negative class despite having low loss values and high metric values. To fix this issue, the loss functions were modified such that each class has a loss value separately calculated then has a weighted average calculated from them. This allowed us to down weight the negative class to .01, while the other classes were set to 1. We selected this value because the tumor makes up about 1-2 % of the volume such that the negative class was weighed about equally to the tumor in the loss functions,

making it so the gradient updates are more evenly weights. We dropped the negative class from the final metrics calculation because it did not provide an indication of how well the model performed on actual tumor segmentation.

Another issue was the amount of time it would take to run a large parameter search due to the slowness of model training. To address this, we made it so that the MRIs would be preprocessed once and saved prior to training, resulting in sizeable time savings. We also faced issues running patches versus entire volumes as input - patches broke for Unet and the custom model while DeepLabV3+ had no issue due to the model architecture. We chose to omit patches in the interest of time.

For the data, the small volume size is potentially holding back model performance due to low feature fidelity, but the number of data points per volume go as n^3 which would drastically increase train sizes by scaling them up. We ended up not using the Episurg dataset [13] as the MRIs were not aligned for the same subject, and the labels were inconsistent shapes. Preprocessing this data would require further research into different software tools.

We also encountered unique difficulties during individual model implementation. For Unet, we faced with the complexity of including encoding and decoding layers of variable depths and formulating a 3D cropping method to utilize residual connections. Unet with default parameters trained extremely slowly due to the large number of channels, and this setting actually hurt overall performance. Decreasing the number of channels per depth level to adapt to the small image volume size significantly improved Unet training speed and overall model performance.

In implementing the custom model we explored the use of PyTorch's experimental lazy layers. While the output shape was correct the model would fail with vague errors in our training harness. Reproducing the same model using the non-experimental layers proved to be successful and was the approach we ultimately utilized.

2.4.3 Novel Additions

In this task we implemented 2 existing models as well as a completely custom model from scratch. The custom model (described in detail above) was an attempt to create a model tailored to the characteristics of the task and dataset. Over the course of the experiments, this unique model allowed for the exploration of the comparative effects of network architectures on image segmentation.

We also proposed in our experiments the use of a 3D version of DeepLabV3+ and used it for the MRI segmentation task. Originally, this model was designed for 2D images, but due to the nature of our dataset, we changed all 2D operations to 3D operations to accept MRI samples.

3. Experiments and Results

3.1. Relative Performance

All 3 models were able to successfully segment tumor regions in MRI images. The custom model was best suited to the task, outperforming the other two models across all metrics by a consistent margin. DeepLabV3+ and Unet were comparable across all metrics with DeepLabV3+ outperforming by a slight margin across all metrics (see Table 1). The relative performance of the best performing hyperparamaters for each model can be seen in Figures 6, 3, and 4.

Metric	Unet	DeepLabV3+	Custom
Loss	0.0059	0.0055	0.4155
Pixel Accuracy	0.5454	0.5806	0.6681
Dice	0.3826	0.3872	0.3889
mIoU	0.2489	0.2482	0.2546

Table 1: Metrics for best Unet, DeepLabV3+, and custom model on test set for each metric.

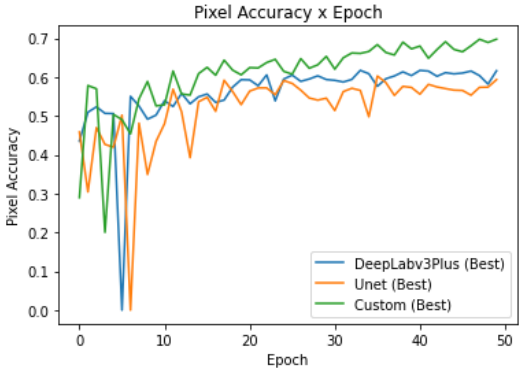

Figure 2: Relative validation pixel accuracy of each model

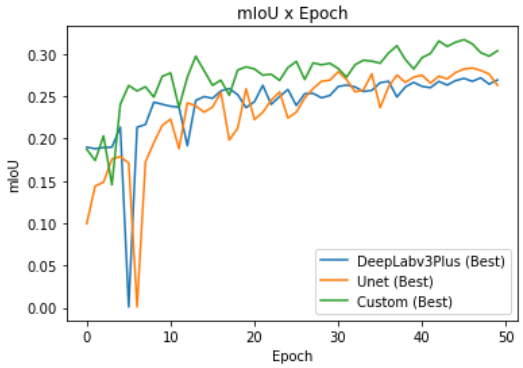

Figure 3: Relative validation mIoU of each model

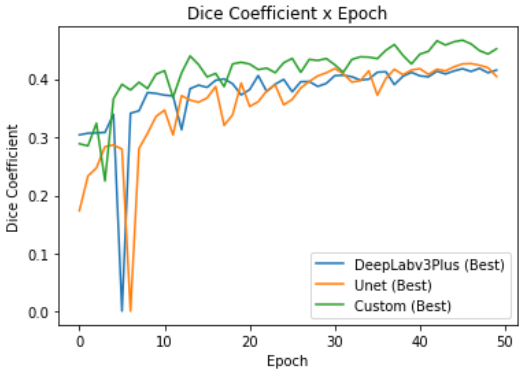

Figure 4: Relative validation dice coefficient of each model

3.2. Unet

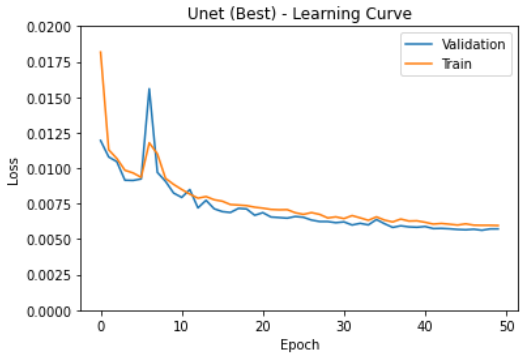

Figure 5: Unet Model: Loss over Epoch

Our 5-layer Unet model was implemented using 3x3 same-padded 3D convolution blocks with encoder and decoder paths expanding and contracting respectively from the initial 16 channels. Performing grid search on the hyperparameter space revealed that Focal Loss, Adam optimizer, and an Exponentially decaying learning rate schedule allowed Unet to perform well on T2 images. Unet performs very similarly to DeepLabV3+ at a glance, resulting in very similar loss, pixel accuracy, Dice Coefficient, and mIoU values. Unet was still able to train efficiently despite starting with a small loss value, meaning that these 3D segmentation networks are already quite effective without much parameter adjustment. Unet benefits from a smaller set of trainable parameters (10M compared to 40M in that of DeepLabV3+ and the Custom model), resulting in faster training and testing times. We observe the largest difference in performance between the models when we analyze its susceptibility to false positives. We hypothesize that the low false positive rate is linked to Unet's ability to efficiently retain tumor-related feature information throughout the encoding and decoding layers avoiding obfuscation with the no-cancer class. Unet's resilience against false positives makes it a prime candidate for helping radiologists quickly sift through benign MRI images.

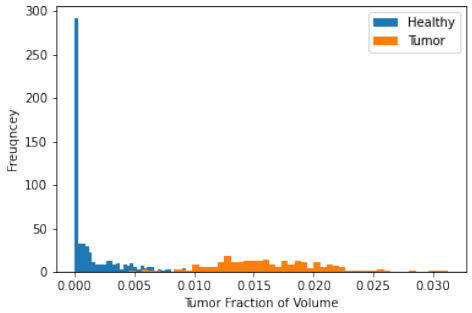

Figure 6: Unet false positive rate

Although other models had lower performance on false positive rates, our loss functions ensured all models leaned away from generating false negatives, a dire diagnosis in the medical industry.

3.3. DeepLabV3+

The best DeepLabV3+ configuration found by our grid search experiments uses Focal Loss, RMSProp optimizer, Exponential learning rate and FLAIR segmentation. Figure 7 shows the loss curve for this configuration. This plot indicates that the training and validation loss curves decreases to a point of stability converging to similar values. The small gap between the two curves reveals that this model generalizes well on new data. One particular interesting feature in this plot is the big jump around epoch 6 in the validation curve. A possible explanation for this spike is that, in the beginning, the model was over-fitting and could not generalize well for the mini-batch images received in this epoch resulting in a very large loss. After the spike, the curve rapidly stabilizes. Interestingly, we see this behavior across all model loss curves and is possibly linked to random seeding for training reproducibility.

3.4. Custom Model

The best performing custom model used Focal Loss, Adam optimizer, Exponential learning rate and FLAIR segmentation. The custom model outperformed the other two models across all of the initially defined metrics. The combined usage of max and average pooling in conjunction with the addition of intermediate dropout layers seemed to be well suited this task. Additionally, as can be seen in the loss graph (i.e. Figure 8), the model loss had not yet stabilized suggesting that continued training would quickly yield even better performance (Unet and DeepLabV3+ saw loss curves that had started to plateau by 50 epochs, see Figures 5 and 7 respectively).

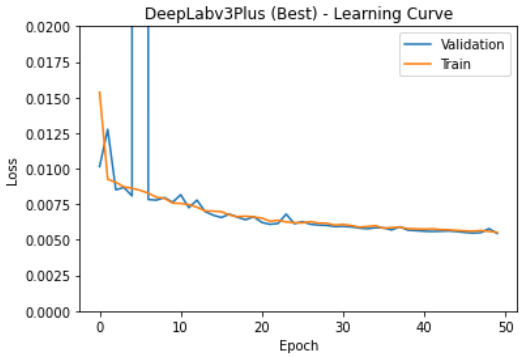

Figure 7: DeepLabV3+ Model: Loss over Epoch

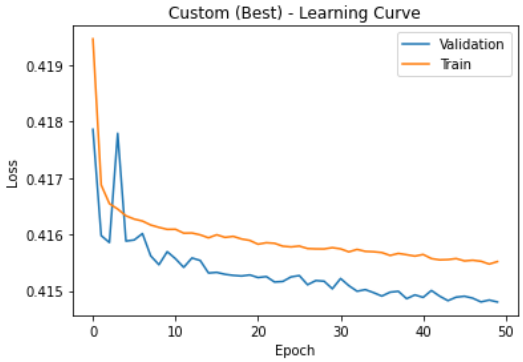

Figure 8: Custom Model: Loss over Epoch

3.5. Interesting Findings

The models all exhibited varying levels of performance across the classes present in this dataset. However, one of the things we observed was that the same network architectures, when trained on binary classification rather than multi-class segmentation, exhibited useful results across all models.

In Figure 9 the comparative predictions of binary segmentation vs. the multi-class segmentation can be seen. Binary segmentation is, by comparison, a simpler task to that of multi-class segmentation, and as such our experiments showed slightly better performance of a binary segmentation model vs. a multi-class segmentation model. However, the results from both models are remarkably accurate and demonstrate the utility of this technique in MRI analysis (see Figure 9 for an example of a binary and multi class prediction side-by-side).

4. Future Research

Brain tumor identification is critical work for radiologists, and further research into improving the segmentation quality and accuracy will be paramount to the success of neural network assisted cancer identification. Potential

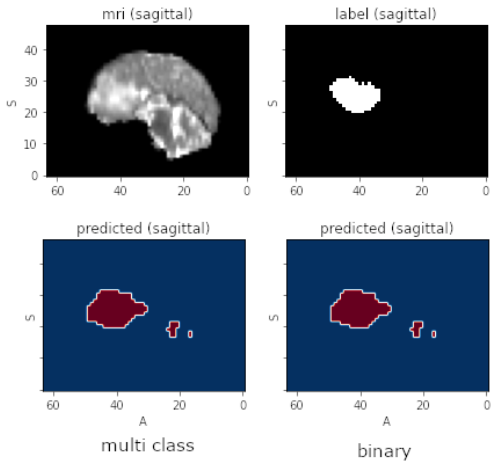

Figure 9: Multiclass segmentation vs. Binary segmentation

routes for further research include investigating model performance with more epochs, further searching the hyperparameter space to find optimal settings for 3D segmentation, and applying the networks to generate high fidelity image segmentation on variable image volumes. Additionally, there is no single model that performed better on all MRI data types, and it may be worthwhile to investigate and dedicate specific models for segmenting T1, T2, and FLAIR images.

In future work, we would also like to investigate different DeepLabV3+ model configurations. In our experiments, we used a rather deep ResNet-101 as the backbone for the DeepLabV3+ model. Riding off the custom model's success with fewer layers, it may be fruitful to experiment with shallower backbone networks such as ResNet-18 and ResNet-34; perhaps yielding similar results with less training time.

There are also opportunities to adapt these brain tumor segmentation techniques beyond 4 classes, and even to general tumor types around the body. We hope our work can be a primer for effective, efficient, and accurate tumor segmentation that can provide doctors the information needed for diagnosis and reduce the waiting time for all patients.

To reduce the impact of class imbalance it would be interesting to look at creating a two network system, one to predict a bounding box around the tumor, and another network that focuses and doing the prediction within the bounding box. This way the model prediction could be less down-sampled and more targeted to a specific region of the brain, prevents outliers from being predicted.

References

[1] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with

atrous separable convolution for semantic image segmentation, 2018. 2

[2] Baid et al. The rsna-asnr-miccai brats 2021 benchmark on brain tumor segmentation and radiogenomic classification, 2021. https://arxiv.org/pdf/2107.02314.pdf. 1

[3] Ronneberger et al. U-net: Convolutional networks for biomedical image segmentation, 2015. https://arxiv.org/pdf/1505.04597.pdf. 2

[4] William Falcon et al. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3, 2019. 2

[5] Hossein Gholamalinezhad and Hossein Khosravi. Pooling methods in deep neural networks, a review, 2020. 2

[6] Shruti Jadon. A survey of loss functions for semantic segmentation, 2020. https://arxiv.org/pdf/2006.14822.pdf. 2, 3

[7] George Papandreou Florian Schroff Hartwig Adam Liang-Chieh Chen, Yukun Zhu. Encoder-decoder with atrous separable convolution for semantic image segmentation, 2016. https://arxiv.org/abs/1802.02611v3. 2

[8] Imperial College London. Ixi dataset, 2021. https://www.nitrc.org/projects/ixi_dataset. 1

[9] Stanford Medicine. Faqs about radiologists, 2021. https://med.stanford.edu/neuroimaging/patients.html. 1

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 2

[11] Fernando Pérez-García, Rachel Sparks, and Sébastien Ourselin. Torchio: a python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning. *Computer Methods and Programs in Biomedicine*, page 106236, 2021. 1

[12] David C Preston. Magnetic resonance imaging (mri) of the brain and spine: Basics, 2021. https://case.edu/med/neurology/NR/MRI%20Basics.htm. 1

[13] Roman; Alim-Marvasti Ali; Sparks Rachel; Duncan John; Ourselin Sebastien Pérez-García, Fernando; Rodionov. Episurg: a dataset of postoperative magnetic resonance images (mri) for quantitative analysis of resection neurosurgery for refractory epilepsy, 2020. University College London. Dataset. https://doi.org/10.5522/04/9996158.v1. 4

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 2

[15] Ekin Tiu. Metrics to evaluate your semantic segmentation model, 2019. https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2. 3

[16] Ferenc Huszar-Johannes Totz Andrew P. Aitken Rob Bishop Daniel Rueckert Zehan Wang Wenzhe Shi, Jose Caballero. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, 2016. https://arxiv.org/pdf/1609.05158.pdf. 3

[17] Ferenc Huszar-Johannes Totz Andrew P. Aitken Rob Bishop Daniel Rueckert Zehan Wang Wenzhe Shi, Jose Caballero. Width, depth and space, 2016. https://arxiv.org/pdf/1607.00945.pdf. 3

[18] Adrian Wolny, Lorenzo Cerrone, Athul Vijayan, Rachele Tofanelli, Amaya Vilches Barro, Marion Louveaux, Christian Wenzl, Sören Strauss, David Wilson-Sánchez, Rena Lymbouridou, Susanne S Steigleder, Constantin Pape, Alberto Bailoni, Salva Duran-Nebreda, George W Bassel, Jan U Lohmann, Miltos Tsiantis, Fred A Hamprecht, Kay Schneitz, Alexis Maizel, and Anna Kreshuk. Accurate and versatile 3d segmentation of plant tissues at cellular resolution. *eLife*, 9:e57613, jul 2020. 2

Student Name	Contributed Aspects	Details
Lukas Dauterman	Identified Datasets	Did the work to identify and Collect datasets appropriate for use in the project
Lukas Dauterman	Data Loading	Implemented the method for loading using TorchIO and reprocessing the BRaTs Dataset
Lukas Dauterman	Training Methods	Created Framework for running experiments in PyTorch lighting, fully integrating the data-loaders, models, and metrics and prepossessing pipeline
Lukas Dauterman	Losses and Metrics	Wrote from scratch the loss functions and metrics, and adjusted them once the class imbalanced issue was identified.
Lukas Dauterman	Implemented Pixel Shuffle and Model	Wrote a 3D Implementation of Pixel shuffle due to PyTorch not supporting 3D, and wrote model using Pixel Shuffle based on Unet
Lukas Dauterman	Model output Data Visualization	Wrote the methods for visualizing the outputs of the model to compare to ground truth
Lukas Dauterman	Model training Logs parsing	Wrote method for parsing tensorboard logs to enable data visualization
Josh Crowther	Collaboration Setup	Integrated workflow tooling to facilitate collaboration between members of the team
Josh Crowther	Platform standardization	Consolidated the various training implementations that were being used for reproducible training and validation
Josh Crowther	Experiment Scripting	Gathered requirements on hyperparameter tuning, authored, and validated scripts used for running all model experiments
Josh Crowther	Custom Model Design/Implementation	Researched the strategy for diverse aggregation methods and implemented the custom model based on that research
Julio Moscon	DeepLabV3+ implementation	Researched DeepLabV3+ paper and implemented this model using PyTorch. Adapted the model to 3D inputs replacing all 2D operations to 3D to support our input dataset.
Julio Moscon	ResNet-101	Researched and implemented a 3D version of the ResNet-101 network to be used as backbone for the DeepLabV3+ model.
Julio Moscon	Data Visualization	Wrote PyTorch script to collect data from the training logs and to plot the graphs used in the paper.
Julio Moscon	Data Extraction	Wrote PyTorch script to extract data to be used in the paper analysis and tables.
Barry Cheng	Unet Model Design and Implementation	Researched the Unet architecture and adapted it to implement a 3D version of the model that can intake variable channel depths
Barry Cheng	Repository Organization	Organized and refactored code in the group repository to align with best practices and ease of use. Researched and provided feedback for a general set of hyperparameters to perform grid search on.
Barry Cheng	Group Project Logistics	Scheduled weekly sync-ups with group members to align on project scope, requirements, and potential paths for exploration
Barry Cheng	Paper Writing and Proofreading	Proofread paper to keep information consistent and fluid. Bookkeeping to ensure paper fit within the project guidelines and scope.

Table 2: Contributions of team members.