3D Brain Tumor Segmentation via Multi-Channel 2D Slice Encoding of Volume

Anonymous CVPR submission

Paper ID *****

Abstract

Modern deep learning has led to an explosion in our ability to automate numerous tasks, including brain tumor segmentation. However, there is an unfortunately small number of training examples for this task, as it requires expensive MRI imaging, as well as expert-level knowledge for labelling inputs. This has led some models to leverage the power of 2D image segmentation models, as they have orders of magnitude more training data. One example is SAM3D [3], which leverages the image encoder from Meta's Segment Anything Model (SAM) [5] to encode each 2D slice of a 3D volume, then stacks the slice encodings back into a volume to feed to a 3D decoder for voxel segmentation.

Limited by computational resources, we choose to build upon SAM3D by implementing a lightweight slice encoder, as opposed to the vision transformer used by SAM, as well as considering each slice of the MRI volume as a four channel image, whereas SAM3D worked with grey scale (single channel) images. We implement this model within the MONAI framework for loading, manipulating, and training networks on medical imaging. Our early results show promise and indicate this model architecture is worth expanding in size and training for longer.

1. Introduction

Brain tumor segmentation technologies are used by doctors to diagnose patients with brain tumors. This involves first taking an MRI image of the patient's brain, which results in a 3D volume, then feeding this 3D volume to some model that makes voxel-level predictions if a part of the brain is a tumor or not.

Figure 1 shows a 2D slice of our input and output volumes. The input consists of four channels corresponding to the four MRI modalities used, and the output consists of three channels corresponding to the three types of tumor tissue we want prediction masks for. MONAI uses this multi-channel approach to multi-class segmentation, as opposed to outputting a single non-binary channel. This is useful, as it boils down the problem to transforming a four channel volume to a three channel volume of the same spatial dimension. It also is possible, and common, for a part of the brain to fall into multiple categories, so it makes sense to predict each output category separately in a binary fashion, instead of a predicting a single label per-voxel.

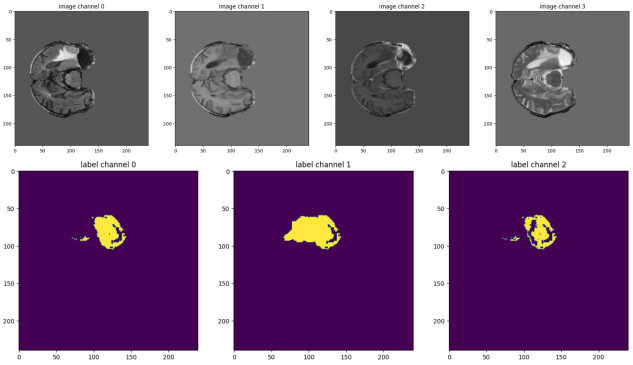

Figure 1. A single slice of the volumetric input and output. Each image corresponds to a separate channel of the same slice. Top row shows the four channels of an MRI image: T1-weighted MRI (T1), T2-weighted MRI (T2), T1-weighted MRI with gadolinium contrast enhancement (T1-Gd) and Fluid Attenuated Inversion Recovery (FLAIR). Bottom row shows the three channels of tumor prediction: peritumoral edema, GD-enhancing tumor, and non-enhancing tumor core.

2. Related Work

Bui, et al. [3] did similar work on SAM3D related to brain tumor segmentation, but they were more generally focused on segmentation of any medical imaging (Figure 2). They also included experiments for organ segmentation, lung tumor segmentation, and automatic cardiac diagnosis, and these datasets all feature single channel input volumes.

This is the major difference in their approach from ours. Since they had more single channel input volumes to consider, they used grey scale image slices as inputs to their image decoder, while we use all four channels of the brain MRI. Since their image decoder expected RGB images with three channels, the authors tiled the 1xHxW grey scale in-

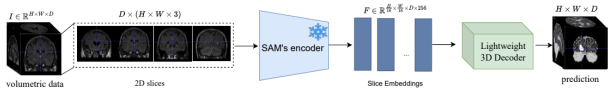

Figure 2. Original SAM3D network proposed by Bui, et al. [3]. The major difference between this network and our network is the handling of multi-channel inputs. We keep the multiple channels together, while they process each channel individually as tiled greyscale (1xHxW → 3xHxW).

puts to be 3xHxW, where each channel is the same. We flagged this as a flaw to fix, as the spatial connections between channels are pointless if each channel is the same.

They did not specify how they handled multi-channel inputs, although they do mention that their model for brain tumor segmentation featured 4x the number of parameters as the other models, so it's possible they process each channel separately through its own network and perform some kind of pooling. It's better to make predictions by considering connections across channels, rather than doing a final pooling at the end, so we process our slices as four channel images.

3. Method

Our overall model architecture follows a typical U-Net structure (Figure 3). We use a slice encoder to decrease spatial dimension while increasing feature dimension, then use a volume decoder to increase spatial dimension back to original while decreasing feature dimension. We have skip connections connecting volumes of the same spatial dimension from the encoder to decoder, and a final convolution layer produces three channels for our three predicted masks.

We start by splitting our input volume into slices, then passing each to a 2D encoder before stacking the encoding into a volume again and feeding that to a 3D decoder.

3.1. Slice Encoder

Our slice encoder is based upon the PyTorch implementation [2] of DenseNet [4], which features an initial Conv2d-BatchNorm2d-ReLU-MaxPool2d block before the dense convolution layers. We repeat this initial block structure three times, reducing the image spatial dimension by eight. To store skip connections, each intermediate group of slices is grouped into a volume and forwarded to the decoder as a volume of decoded slices.

We initially planned on using a vision transformer for encoding, but even the smallest model didn't fit in our RAM with a batch size of one.

3.2. Volume Decoder

Our volume decoder is based upon the volume decoder of SAM3D [3], which uses four Conv3d-InstanceNorm3d-

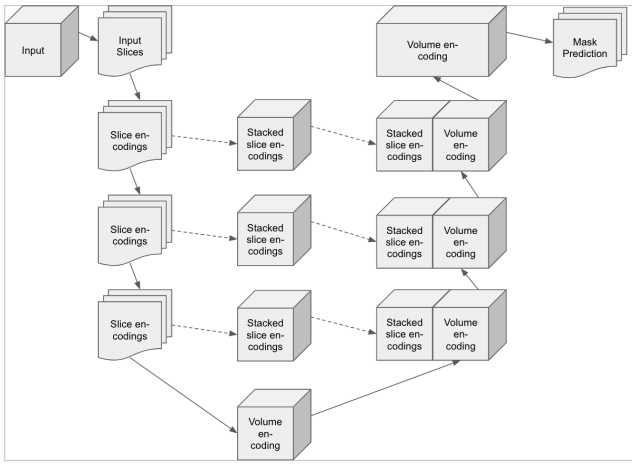

Figure 3. Our Custom SAM3D architecture.

LeakyReLU-Upsample blocks before a final prediction block. We simply use three of these blocks so our upsampling matches our downsampling. Since our downsampling takes place in the 2D context, our 3D upsampling only takes place along the HxW dimensions (depth dimension is untouched).

We finish by passing our final upsampled features to another Conv3d layer for predicting our three output masks.

3.3. Error Metric

We use the standard DICE error metric of two volumes, which measures

$$\frac{2 \times \text{intersection}}{\text{union}}$$

and ranges from zero (no overlap) to one (perfect overlap). We compute the DICE metric for each output channel, then average these results for our average DICE metric, which we use to compare two models overall.

4. Experiments

4.1. MONAI

Much of our training pipeline was borrowed from a Google Colab tutorial from MONAI [1]. MONAI, or Medical Open Network for AI, is an open source framework dedicated to loading medical datasets, designing models, and running training and inference pipelines (Figure 4). The MONAI framework is built entirely on top of PyTorch, so it's relatively straight-forward to incorporate a custom PyTorch model into the workflow.

4.2. Dataset

We start by loading the BraTS dataset from Medical Decathlon, a common source for loading medical imaging data, which is easily loaded by MONAI. The BraTS dataset

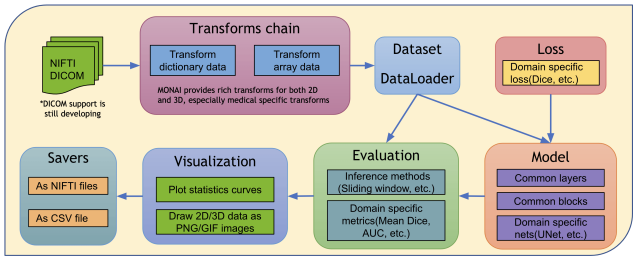

Figure 4. Typical MONAI training pipeline. Our focus was on changing the model.

is used each year for the annual BraTS Tumor Segmentation challenge, and Medical Decathlon gives us easy access to the 2016 and 2017 BraTS datasets. This includes 750 four channel 3D volumes (484 training, 266 testing) of brain MRI images. Each volume has been pre-processed to remove the skull, so we're left with simply the brain.

The MONAI framework already loads the datasets for us, and uses different "transforms" which are supposed to enable dynamic data augmentation by randomly adjusting intensities, though this is something we could not confirm, lacking extensive knowledge of MONAI.

We used a batch size of one, as some of our original experiments didn't support larger batch sizes. Exploring adaptive batch sizing may change training or inference behavior. We keep the batch size of one, as our training data is so small, so we want to maximally learn from each training example.

4.3. Training

First, we ran the original MONAI tutorial, with the given network they defined. The tutorial defined a SegResNet based upon [6], which uses residual connections and a variational autoencoder to make tumor mask predictions. The original SegResNet model has great final results, achieving over a 0.9 average DICE metric, and the MONAI implementation achieves about 0.8 average DICE metric after training, so this model is a good baseline to compare our model.

We ran the MONAI model for ten epochs and achieved an average validation DICE metric of 0.675.

Next, we ran our custom network for ten epochs, and achieved an average DICE metric of 0.7286.

4.4. Evaluation

Below shows the best metrics achieved by each model in ten epochs, as well as mask predictions made by our Custom SAM3D on a given input.

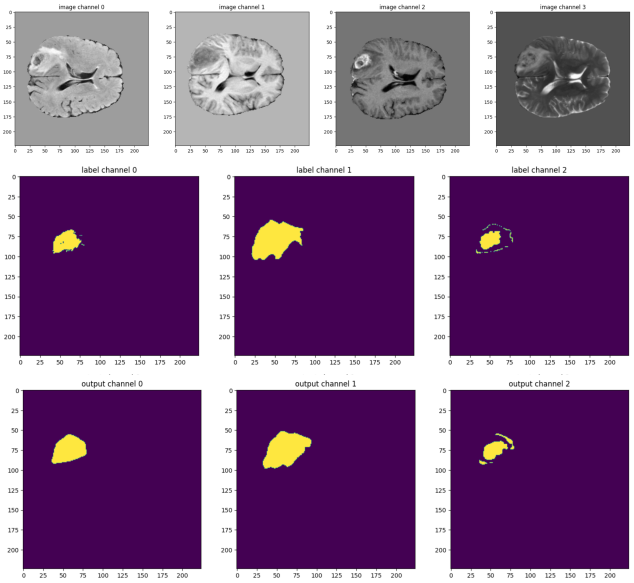

Figure 5. Single slice prediction of a 3D volume. Top row shows an input slice from an input volume (four channels), bottom row shows output predictions for that slice (three channels), and middle row shows ground-truth masks.

	SegResNet	Our Custom SAM3D
Epochs	10	10
Mean DICE	0.6752	**0.7286**
TC DICE	0.7338	**0.7794**
WT DICE	0.8719	**0.8811**
ET DICE	0.4197	**0.5253**

5. Conclusion and Further Work

This paper lays out a great blueprint for future experiments to build upon. Even though we're using a relatively shallow custom model, with a few easily-identifiable improvements needed, we still perform better than the prebuilt MONAI SegResNet model for the first ten epochs. The SegResNet model is based on a paper that achieves over a 0.9 DICE metric, which is considered nearly state-of-the-art, so the hope is our model architecture would show similar results if expanded and trained longer.

Expanding the network would mean simply making it deeper. Adding more convolution layers, or adding more channels to layer, would make the model deeper and potentially improve results. Specifically, it should help to add more layers in the decoder when processing a new skip connection. This is typically done in a few layers per skip connection, while we only use one layer to reduce model size. We should also add the original input volume as a skip connection for our final prediction layer, as we originally overlooked this skip connection.

The task of 3D volume segmentation is computationally expensive, so we cannot speak conclusively on this model until hundreds of epochs have been run across many many hours, as is typically done for this task. But we can say that we show strong promise, as our early performance closely matches that of a nearly state-of-the-art model. More exploration is needed to determine the full capabilities (and limitations) of 3D segmentation via 2D slice segmentation, however, we've shown that we should expect good results if we continue to pursue this method, and potentially even a new state-of-the-art.

References

[1] Brain tumor 3d segmentation with monai. https://github.com/Project-MONAI/tutorials/blob/main/3d_segmentation/brats_segmentation_3d.ipynb. Accessed: 2024-05-17. 2

[2] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, Apr. 2024. 2

[3] Nhat-Tan Bui, Dinh-Hieu Hoang, Minh-Triet Tran, Gianfranco Doretto, Donald Adjeroh, Brijesh Patel, Arabinda Choudhary, and Ngan Le. Sam3d: Segment anything model in volumetric medical images, 2024. 1, 2

[4] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018. 2

[5] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023. 1

[6] Andriy Myronenko. 3d mri brain tumor segmentation using autoencoder regularization, 2018. 3