50.039 Deep Learning Project Report
Brain Tumor Segmentation in 3D Pre-Operative MRI Scans

Bryan Tan (Chen Zhengyu) | 1004318 (Running of experiments and report writing)
Christy Lau Jin Yun | 1005330 (Debugging of serious issues and architecture research)
Lee Pei Xuan | 1005513 (Data visualisation code and presentation materials)

Computer Science and Design (CSD)
Singapore University of Technology and Design

April 17, 2023

Abstract

There is a need for automated approaches to brain-tumour segmentation as it can help reduce hospital workloads and save lives. Existing models that have proven to be suitable for the problem of tumour segmentation include 3D-UNet and Swin UNETR. Using the BraTS2020 dataset, we test several approaches for brain tumour segmentation such as developing novel models we call 3D-ONet and 3D-SphereNet, our own variant of 3D-UNet with more than one encoder-decoder paths. We aim to maximise performance (defined as high dice and jaccard scores) while minimising parameter count. We trained each model from scratch for 50 epochs using BCE-Dice Loss as the objective and evaluated them by their dice and jaccard score. We found that the performance of our proposed 3D-ONet exceeds that of NVIDIA's Swin UNETR (a model considered state-of-the-art) in data and resource-constrained environments while being much more parameter efficient. However, more work is needed to validate its performance, especially under data and resource-abundant conditions. The code and instructions for running them can be found here.

1 Introduction

Image segmentation is a fundamental task in the field of computer vision and image processing, serving as the foundation for numerous applications across diverse domains such as autonomous vehicles, robotics, remote sensing, and medical imaging. The primary objective of image segmentation is to partition an image into distinct regions, each corresponding to a specific object or area of interest. This process enables the extraction of meaningful information from images, facilitating subsequent tasks such as object recognition, tracking, and scene understanding.

One such task is 3D Brain Tumour Segmentation; an important challenge in biomedical image processing. Though tumour segmentation is traditionally done manually by doctors worldwide, it can be done very quickly with the help of 3-dimensional image processing and computer vision. Such methods can facilitate the early detection of brain tumours which is vital for the successful treatment of patients. One of the most prevalent types of primary brain tumours is Giloma, with a survival rate of just over 6.8% and which comprises about 30% of brain tumours and 80% of all malignant brain tumours. Automated brain-tumour segmentation approaches can hence help reduce hospital workloads and save lives.

2 Dataset

We utilise the Medical Image Computing and Computer Assisted Interventions (MICCAI) Brain Tumor Segmentation (BraTS 2020) [1] dataset which consists of 369 labelled training samples and 125 unlabelled validation samples of preoperative MRI Brain scans from 19 different institutions. Each sample comprises an image with 240x240x155 voxels saved in the Neuroimaging Informatics Technology Initiative (NIfTI) file format with the file extension ".nii.gz". For the purpose of training and evaluation, we discard the 125 unlabelled validation samples and split the remaining 369 labelled training samples into train-val-test splits of 263-53-53 (a ratio of approximately 5:1:1).

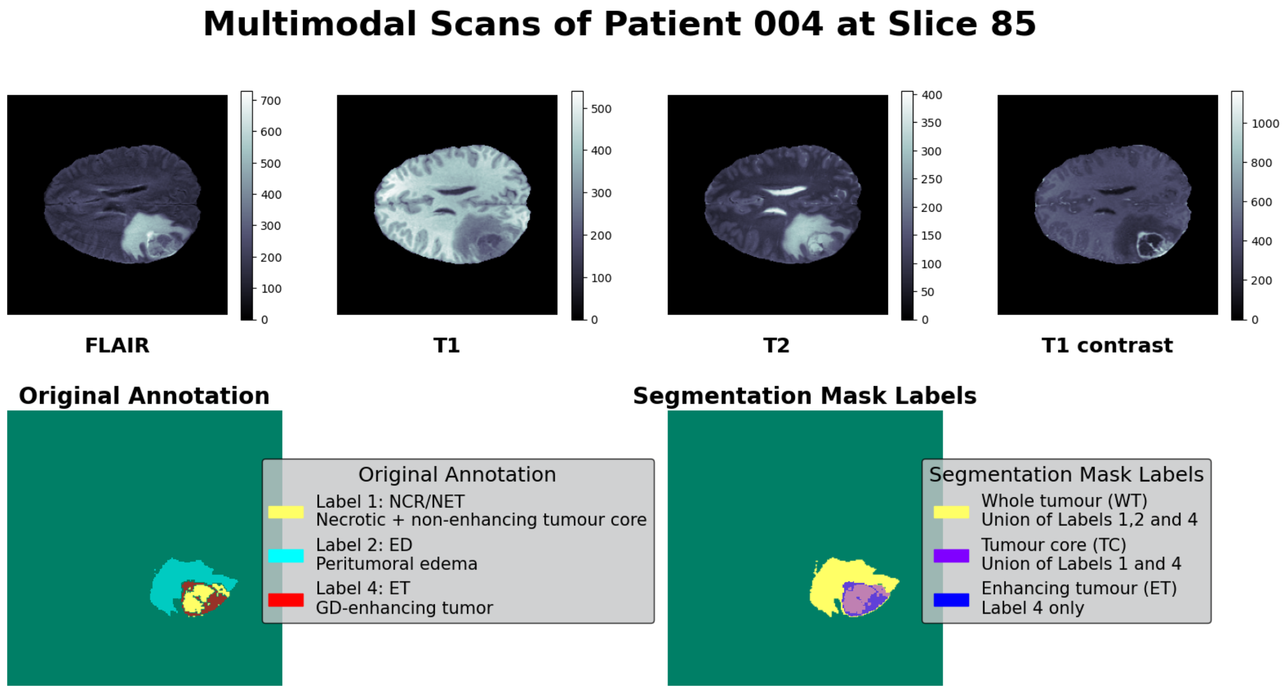


Figure 1: Multi-modal Brain Tumour Scans, original annotations and processed segmentation labels

2.1 Channels

Each sample is comprised of 4 channels/modalities: T1-Weighted (T1), Post-contrast T1-Weighted (T1c), T2-Weighted (T2) and T2 Fluid Attenuated Inversion Recovery (T2-FLAIR). By combining different types of images, doctors gain a more complete picture of a patient's condition and make more informed decisions about treatment.

- **T1-weighted** images are created using specific settings during an MRI scan that highlight differences in the relaxation times of tissues in the body. In medical imaging, T1-weighted images are particularly useful for imaging the brain and other soft tissues.
- **T1c images** are "contrast-enhanced", meaning that a contrast agent is used during the scan to make pertinent structures in the brain more visible.
- **T2-weighted** images are characterized by their ability to show the differences in the relaxation times of tissues in the body. T1-weighted images are good at showing details of anatomical structures, while T2-weighted images are useful in detecting abnormalities in fluid-filled spaces and soft tissues.
- **T2-Flair** is similar to T2-weighted imaging in that it uses a magnetic field to highlight differences in tissue relaxation times in the body. However, it also incorporates a technique called fluid-attenuated inversion recovery, which suppresses the signal from cerebrospinal fluid (CSF) in the brain, making it easier to identify abnormalities in adjacent tissues.

2.2 Original Annotations

The original annotations in the dataset comprise the following:

- **Label 1**: Necrotic and non-enhancing tumour core (NCR/NET)
- **Label 2**: Peritumoral edema (ED)
- **Label 4**: GD-enhancing tumor (ET)

2.3 Segmentation Classes

The above regions make up the following segmentation classes considered for evaluation in the BraTS 2020 challenge:

- **Whole tumour (WT)**: Union of NCR/NET, ED and ET (Labels 1,2 and 4).
- **Tumor core (TC)**: Union of NCR/NET and ET (Labels 1 and 4)
- **Enhancing tumour (ET)**: Label 4 only.

2.4 Pre-processing

We perform the following pre-processing steps within the BraTS Dataset class:

2.4.1 Min-Max Normalization

Image normalization is performed on input images to assist with convergence during training. This is carried out according to the equation below:

$$Normalized\ value = \frac{value - value_{min}}{value_{max} - value_{min}}$$

2.4.2 Cropping

The image is cropped from size $(240, 240, 155)$ to size $(224, 224, 128)$. This reduces the image volume and speeds up training without major performance losses due to the margins of empty spaces in the original images. This also allows for the evaluation of MONAI's implementation of Swin UNETR, which only accepts 3D inputs with dimensions that are multiples of 32.

2.4.3 Masking

Segmentation performance in the BraTS challenge is evaluated on three partially overlapping sub-regions of tumours: whole tumour (WT), tumour core (TC), and enhancing tumour (ET). To adhere to this, 3 sets of one-hot segmentation masks are created and stacked from unions of the original annotations.

3 Existing Works

U-Net [2] is a popular convolutional neural network for biomedical image segmentation tasks consisting of a roughly symmetric pair of contracting (encoder) and expansive (decoder) paths which yields a u-shaped architecture. While both paths feature repeated applications of convolutional layers followed by the ReLU operation, the contracting path features downsampling and max-pooling operations while the expansive path features upsampling and batch-norm operations. Features from both the previous layer and the corresponding convolution layer in the contractive path are concatenated during upsampling in the form of multiple skip connections. This allows the model to retain information from previous layers and extract features from various resolutions. A 3D version of the U-Net model exists in the form of 3D U-Net [3], a variant [4] of which we utilise as a strong baseline to modify for use in BraTS2020. As a benchmark for ideal performance, we investigate Swin UNETR [5], a model by NVIDIA considered to be among the top-performing approaches for biomedical image segmentation tasks.

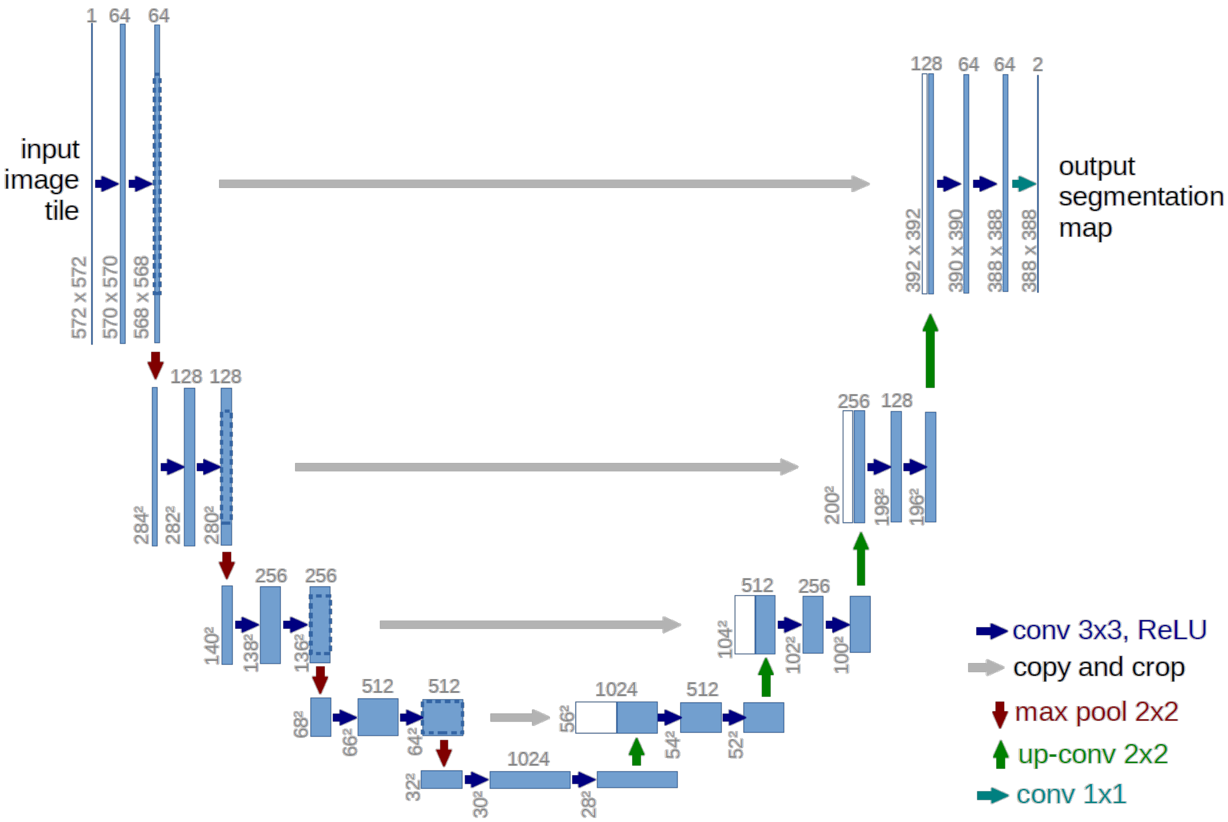


Figure 2: U-Net as implemented in "U-Net: Convolutional Networks for Biomedical Image Segmentation" by Ronneberger et al.

3.1 3D U-Net

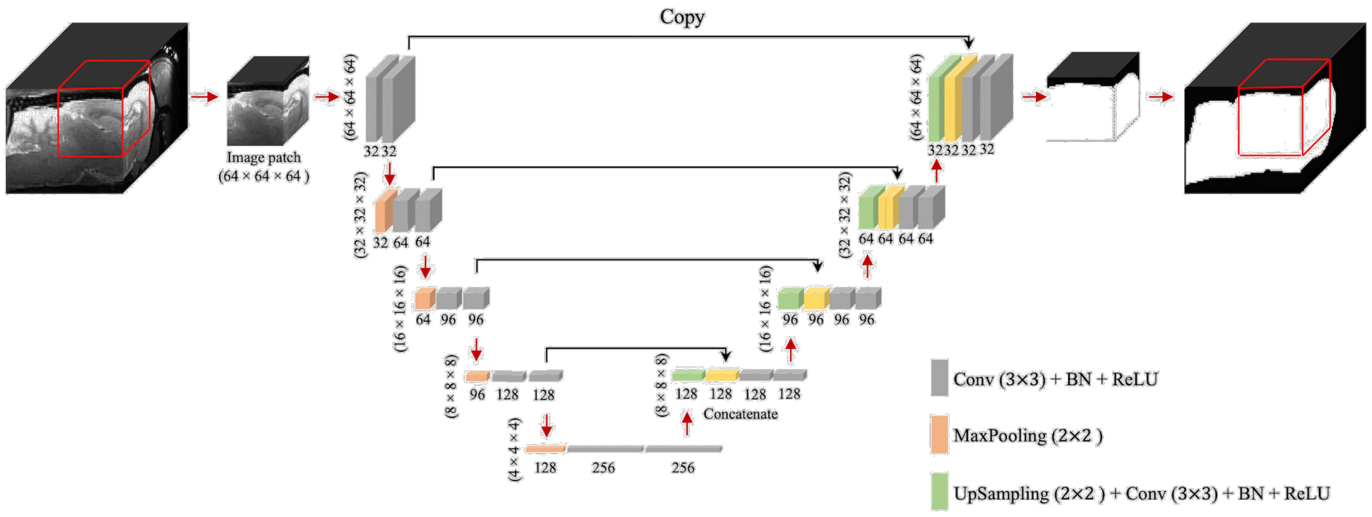


Figure 3: 3D U-Net as implemented in "3D U-Net Improves Automatic Brain Extraction for Isotropic Rat Brain Magnetic Resonance Imaging Data" by LM Hsu et al.

3D U-Net features a similar architecture as that of the classic U-Net, with the core difference being that it uses 3D convolutions instead of 2D convolutions. Other changes include the addition of a batch normalization layer before each ReLU operation. We consider this model to be a strong baseline for our task.

3.2 Swin UNETR

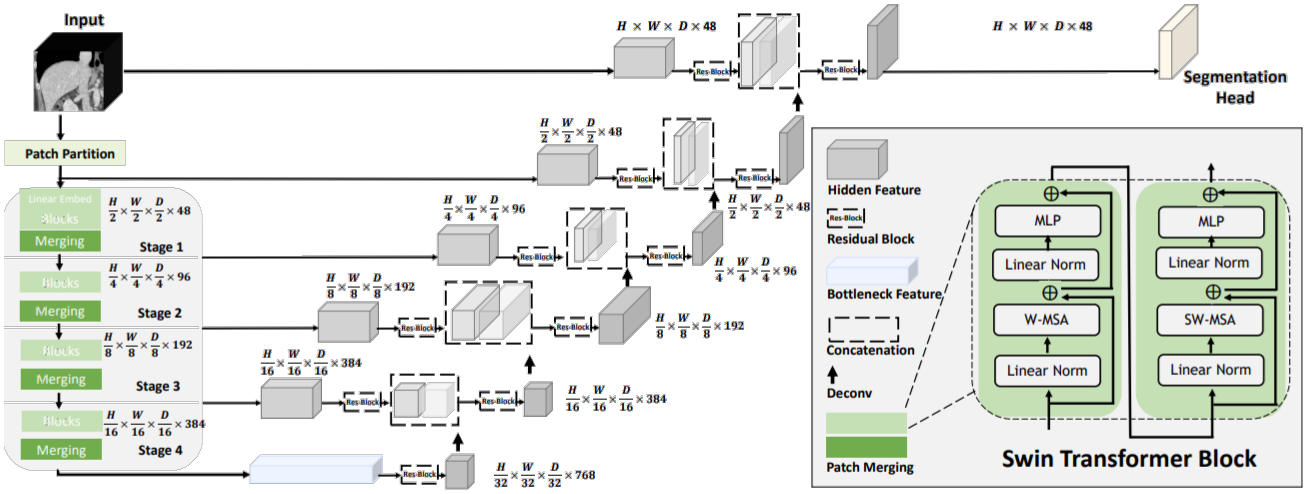

Input
Patch Partition
Linear Embed
Blocks
Merging
Stage 1
Stage 2
Stage 3
Stage 4
$H \times W \times D \times 48$
$\frac{H}{2} \times \frac{W}{2} \times \frac{D}{2} \times 48$
$\frac{H}{4} \times \frac{W}{4} \times \frac{D}{4} \times 96$
$\frac{H}{8} \times \frac{W}{8} \times \frac{D}{8} \times 192$
$\frac{H}{16} \times \frac{W}{16} \times \frac{D}{16} \times 384$
$\frac{H}{32} \times \frac{W}{32} \times \frac{D}{32} \times 768$
Res-Block
Segmentation Head
Hidden Feature
Residual Block
Bottleneck Feature
Concatenation
Deconv
Patch Merging
MLP
Linear Norm
W-MSA
SW-MSA
Swin Transformer Block

Figure 4: Swin UNETR Architecture from "Swin UNETR: Swin Transformers for Semantic Segmentation of Brain Tumors in MRI Images" by A Hatamizadeh et al.

Swin UNETR is a model developed by NVIDIA which is considered state-of-the-Art for 3D medical segmentation tasks [6]. It utilises Swin Transformers [7] [8], a hierarchical transformer specialised for computer vision tasks, a guide to which can be found in this medium blog post [7].

4 Methods

We use 3D U-Net as a strong baseline and aim to improve this to match or exceed the performance of Swin UNETR while maintaining a low parameter count. We do this by first making minor changes such as adding dropout layers, changing the optimisers, replacing the activation functions and changing the depth and size of channels. We then create novel architectures by making significant modifications to the structure of the baseline.

4.1 Models evaluated

We evaluated a total of 18 different models each falling under one of the four model families below.

4.1.1 3D U-Net

Our implementation of 3D U-Net replaces batch normalization with group normalization by Wu et al. [9] and features a variable hidden layer channel size as an argument. This determines the output size of the first convolution operation which doubles with each successive encoder layer. All convolution layer uses a kernel size of 3 with same-padding. The 7 models falling under this family are initialised as follows:

1. **3DUnet**: `UNet3d(in_channels=4, n_classes=3, n_channels=24)`. The default 3DUnet model as defined above.

2. **3DUnet_32_Channels**: `UNet3d(in_channels=4, n_classes=3, n_channels=32)`. The default 3DUnet with n_channels=32 instead of 24.

3. **3DUnet_GELU**: `UNet3d_GELU(in_channels=4, n_classes=3, n_channels=24)`. 3DUnet with all ReLU replaced by GELU.

4. **3DUnet_SELU**: `UNet3d_SELU(in_channels=4, n_classes=3, n_channels=24)`. 3DUnet with all ReLU replaced by SELU.

5. **3DUnet_Dropout**: `UNet3dDropout(in_channels=4, n_classes=3, n_channels=24)`. 3DUnet with a dropout layer of p = 0.2 before each ReLU.

6. **3DUnet_SingleConv**: `UNet3dSingleConv(in_channels=4, n_classes=3, n_channels=24)`. 3DUnet with single convolution instead of double convolution per upscale/downscale.

7. **3DUnet_Atten**: `UNet3d_atten(in_channels=4, n_classes=3, n_channels=24)`. 3DUnet with attention gates as implemented in a repository based on "Attention U-Net: Learning Where to Look for the Pancreas" by Ozan et al [10].

4.1.2 Swin UNETR

We utilise Project MONAI's implementation of Swin UNETR. The 3 models falling under this family are initialised as follows:

1. **SwinUNETR**: `SwinUNETR(in_channels=4, out_channels=3, img_size=(128, 224, 224), depths=(1, 1, 1, 1), num_heads=(2, 4, 8, 16))`. The default Swin UNETR model.

2. **SwinUNETR_AdamW**: `SwinUNETR(in_channels=4, out_channels=3, img_size=(128, 224, 224), depths=(1, 1, 1, 1), num_heads=(2, 4, 8, 16))`. Swin UNETR model trained with an AdamW optimiser instead of Adam.

3. **SwinUNETR_DoubleLayerDepth**: `SwinUNETR(in_channels=4, out_channels=3, img_size=(128, 224, 224), depths=(2, 2, 2, 2), num_heads=(2, 4, 8, 16))`. Swin UNETR model with depths=(2, 2, 2, 2) instead of depths=(1, 1, 1, 1)

4.1.3 3D O-Net (Ours)

We experiment with the notion of having two sets of encoder-decoder sections which are concatenated before the output convolution layer. This forms what can be visualised as 2 parallel sets of U-Net in approximately an "O" shape. To maintain a similar parameter count as that of U-Net, we mainly experiment with single (rather than double) convolutions for each upscale and downscale step. Additionally, while the kernel size of convolution layers at the bottom encoder-decoder half is fixed at 3, we vary and experiment with different kernel sizes at the top encoder-decoder half. This is motivated by the intuition that the two halves ought to learn feature representations at different scales from each other. The 7 models falling under this family are initialised as follows:

1. **3DOnet_SingleConv_Kernel5**: `ONet3d(in_channels=4, n_classes=3, n_channels=24)`. 3D-ONet with convolutions of kernel size 5 for the upper encoder-decoder section.

2. **3DOnet_SingleConv_Kernel3**: `ONet3d_v2(in_channels=4, n_classes=3, n_channels=24)`. 3D-ONet with convolutions of kernel size 3 for the upper encoder-decoder section.

3. **3DOnet_SingleConv_Kernel1**: `ONet3d_v3(in_channels=4, n_classes=3, n_channels=24)`. 3D-ONet with convolutions of kernel size 1 for the upper encoder-decoder section.

4. **3DOnet_SingleConv_Kernel1_32_Channels**: `ONet3d_v3(in_channels=4, n_classes=3, n_channels=32)`. 3D-ONet with convolutions of kernel size 1 for the upper encoder-decoder section and n_channels=32 instead of 24.

5. **3DOnet_SingleConv_Kernel1_GELU**: `ONet3d_v3_GELU(in_channels=4, n_classes=3, n_channels=24)`. 3D-ONet with convolutions of kernel size 1 for the upper encoder-decoder section and all ReLU replaced with GELU.

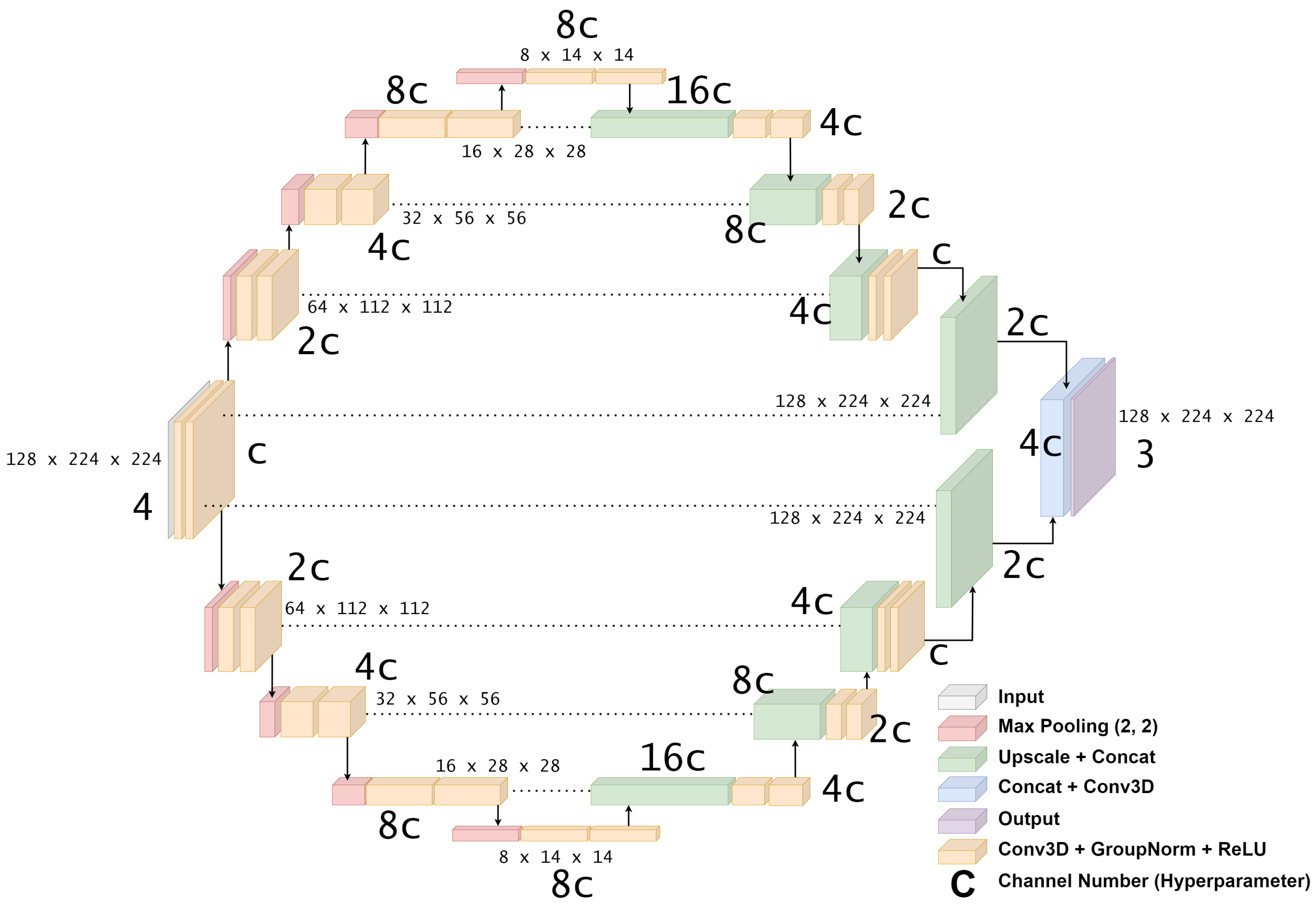


Figure 5: Double-convolution variant of our 3D O-Net architecture, consisting of 2 sets of Encoder-Decoder pairs. To ensure the parameter count is roughly the same as the original 3D U-Net, we mainly experiment with the single-convolution variant.

6. **3DOnet_SingleConv_Kernel1_GELU_AdamW**: `ONet3d_v3_GELU(in_channels=4, n_classes=3, n_channels=24)`. 3D-ONet with convolutions of kernel size 1 for the upper encoder-decoder section, all ReLU replaced with GELU and trained with AdamW instead of Adam.

7. **3DOnet_DoubleConv_Kernel1**: `ONet3d_v3_DoubleConv(in_channels=4, n_classes=3, n_channels=24)`. 3D-ONet with convolutions of kernel size 1 for the upper encoder-decoder section and with double convolutions (as per the original U-Net) rather than single convolutions.

4.1.4 3D SphereNet (Ours)

On top of adding an extra set of encoder-decoder pair, we experiment with having a total of 4 sets of encoder-decoder pairs. These 4 sets of encoders can be visualised as being at the bottom, top, left and right in a 3D diagram, forming a sphere of sorts. Convolution layers in each of these encoder-decoder sections have different kernel sizes of 1, 3, 5 and 7 respectively (which results in this model having the largest parameter count despite having the number of n_channels reduced from 24 to 16). The one model falling under this family is initialised as follows: `SphereNet3d(in_channels=4, n_classes=3, n_channels=16)`.

4.2 Performance Metrics and Loss

We evaluate the performance of our models based on the Dice and Jaccard score, with Dice Score being the main evaluation metric. We use BCE-Dice Loss as the objective function, which is a simple average of Dice Loss and Binary Cross Entropy Loss.

4.2.1 Dice Score

The Dice Score is a measure of the overlap between the predicted segmentation mask and the ground truth mask and is defined as twice the intersection of the two masks divided by the sum of the pixels in both masks. The score ranges between 0 and 1, with higher values indicating a better overlap between the predicted and ground truth masks. This will be the main metric for evaluation.

$$Dice\ Score = \frac{2 * |X \cap Y|}{|X| + |Y|}$$

4.2.2 Jaccard Score

The Jaccard Score, also known as the Intersection over Union (IoU) Loss, is another measure of the overlap between the predicted and ground truth segmentation masks. It is defined as the intersection of the two masks divided by their union. The score also ranges between 0 and 1, with higher values indicating better segmentation accuracy.

$$Jaccard\ Score = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|}$$

4.2.3 BCE-Dice Loss

Though our objective is to maximise Dice Score, Dice Loss only accounts for the final predictions. In order to penalize both confident misclassifications and unsure correct predictions, we use a simple average of Dice Loss and Binary Cross Entropy (BCE) Loss as the objective function. BCE is a commonly used loss function in binary classification tasks, whereby the output of a model is a probability between 0 and 1 representing the likelihood that a given input belongs to a particular class.

$$Dice\ Loss = 1 - Dice\ Score$$

$$BCE\ Loss = -Y \log(\sigma(Y')) - (1 - Y) \log(1 - \sigma(Y'))$$

$$BCEDice\ Loss = 0.5 * Dice\ Loss + 0.5 * BCE\ Loss$$

🟦	3DUnet_SingleConv
🟦	3DUnet_Dropout
🟦	3DUnet_SELU
🟦	3DUnet
🟦	3DUnet_GELU
🟦	3DUnet_Atten
🟦	3DUnet_32_Channels
🟪	3DOnet_SingleConv_Kernel1_GELU_AdamW
🟪	3DOnet_SingleConv_Kernel1
🟪	3DOnet_SingleConv_Kernel1_GELU
🟪	3DOnet_SingleConv_Kernel1_32_Channels
🟪	3DOnet_DoubleConv_Kernel1
🟪	3DOnet_SingleConv_Kernel3
🟪	3DOnet_SingleConv_Kernel5
🟩	SwinUNETR_AdamW
🟩	SwinUNETR
🟩	SwinUNETR_DoubleLayerDepth
🟧	SphereNet3D

Table 1: **Model Legend**. Darker shades corresponds generally to a greater parameter count. Blue corresponds 3D-UNet variants; Purple corresponds to 3D-ONet variant; Green corresponds to Swin UNETR variants; Orange corresponds to 3D-SphereNet.

4.3 Training

All models are trained for 50 epochs using the Adam optimiser with default parameters unless otherwise stated. All computations are performed on the Ubuntu-22.04 OS via WSL, with an Intel i5-13600k, 32GB RAM and an RTX 3090 with 24GB VRAM. All models are evaluated on the same 53 test samples by loading the best checkpoint (epoch in which the model obtained the lowest validation loss). For reproducibility during training, we set `seed=42` wherever possible. The code for training is built upon this Kaggle notebook [11]. Larger versions of most figures can be found in the Appendix.

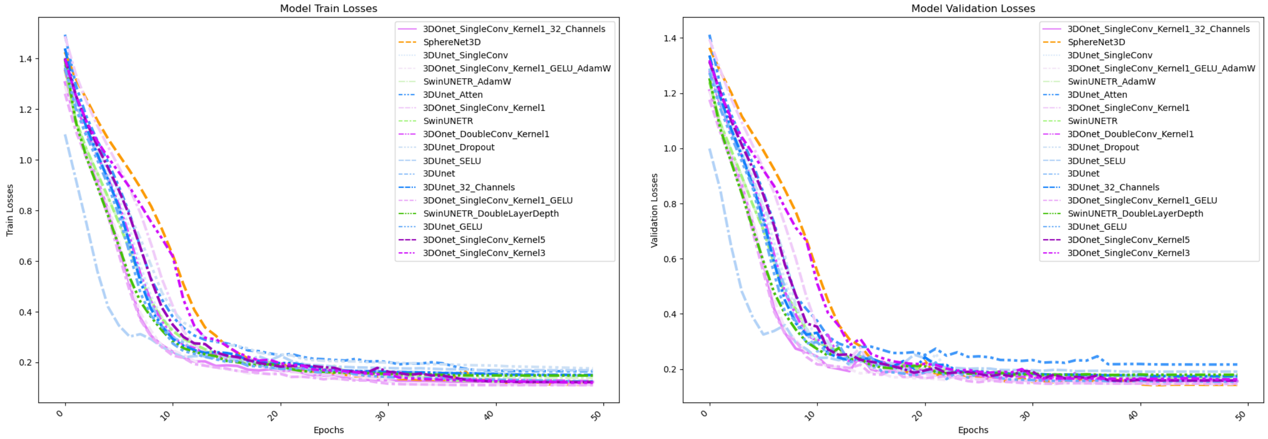


Figure 6: Train and validation losses respectively, over 50 epochs

5 Results

Bold represents best result within model family; **underline** represents best result across all models.

Models	WT dice	WT jaccard	TC dice	TC jaccard	ET dice	ET jaccard	Infer time (s)	Train time (h)	Params (1e6)
3DOnet_DoubleConv_Kernel1	0.8801	0.8022	0.7058	0.6017	0.6600	0.5678	27.41	**5.850**	5.90
3DOnet_SingleConv_Kernel1	0.8753	0.7956	0.7282	0.6363	0.7093	0.6172	22.71	10.533	**3.15**
3DOnet_SingleConv_Kernel1_32_Channels	0.8860	0.8085	0.7365	0.6300	0.6706	0.5761	29.72	12.167	5.60
3DOnet_SingleConv_Kernel1_GELU	0.8817	0.8050	0.7449	0.6435	<u>**0.7218**</u>	<u>**0.6266**</u>	23.18	6.650	**3.15**
3DOnet_SingleConv_Kernel1_GELU_AdamW	<u>**0.8879**</u>	<u>**0.8097**</u>	<u>**0.7572**</u>	**0.6565**	0.6948	0.6006	**22.53**	7.750	**3.15**
3DOnet_SingleConv_Kernel3	0.8593	0.7797	0.7217	0.6264	0.6597	0.5667	24.98	14.167	6.03
3DOnet_SingleConv_Kernel5	0.8705	0.7897	0.7064	0.5962	0.6877	0.5905	30.87	10.017	16.86
3DUnet	**0.8750**	0.7900	0.6736	0.5661	0.6414	0.5425	18.06	4.633	5.65
3DUnet_32_Channels	0.8651	0.7810	0.6849	0.5722	0.6508	0.5498	22.69	5.317	10.05
3DUnet_Atten	0.8290	0.7334	0.6216	0.5163	0.6267	0.5231	19.04	3.617	6.09
3DUnet_Dropout	0.8527	0.7719	0.6775	0.5690	0.6427	0.5393	17.58	3.883	5.65
3DUnet_GELU	0.8730	**0.7944**	**0.7262**	**0.6323**	**0.6849**	**0.5888**	18.50	5.833	5.65
3DUnet_SELU	0.8554	0.7690	0.6390	0.5303	0.6191	0.5144	17.88	<u>**3.567**</u>	5.65
3DUnet_SingleConv	0.8731	0.7901	0.6820	0.5785	0.6246	0.5210	<u>**15.20**</u>	5.567	<u>**3.01**</u>
SphereNet3D	**0.8667**	**0.7929**	**0.7526**	<u>**0.6582**</u>	**0.6860**	**0.5930**	**36.35**	**10.450**	**6.13**
SwinUNETR	0.8666	0.7902	**0.6888**	**0.5758**	0.6640	0.5749	20.39	**4.100**	**14.98**
SwinUNETR_AdamW	0.8674	0.7861	0.6786	0.5657	0.6611	0.5722	**20.24**	6.800	**14.98**
SwinUNETR_DoubleLayerDepth	**0.8733**	**0.7964**	0.6643	0.5569	**0.6706**	**0.5759**	24.15	15.233	15.64

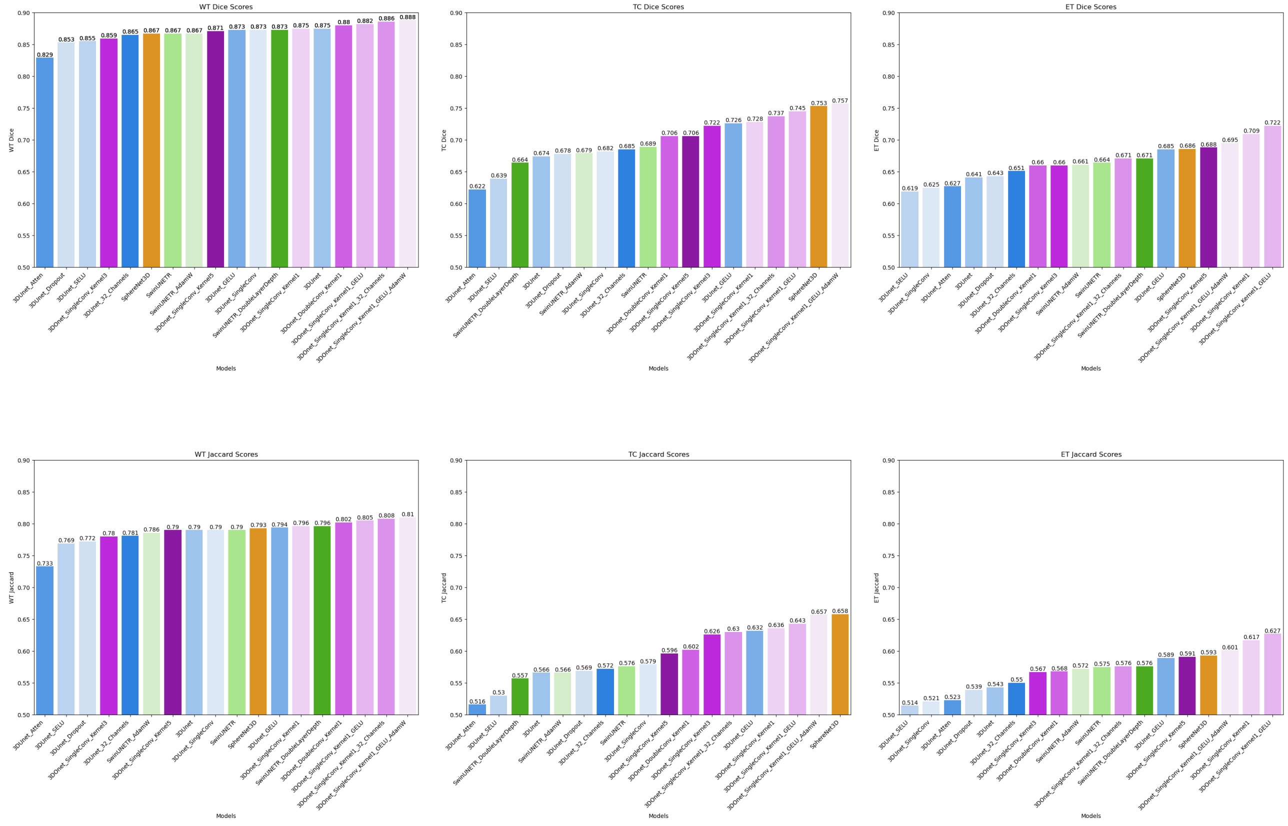


Figure 7: Dice (top) and Jaccard (bottom) score for WT (left), TC (middle) and ET (right) classes respectively, ranging from 0.5-0.9

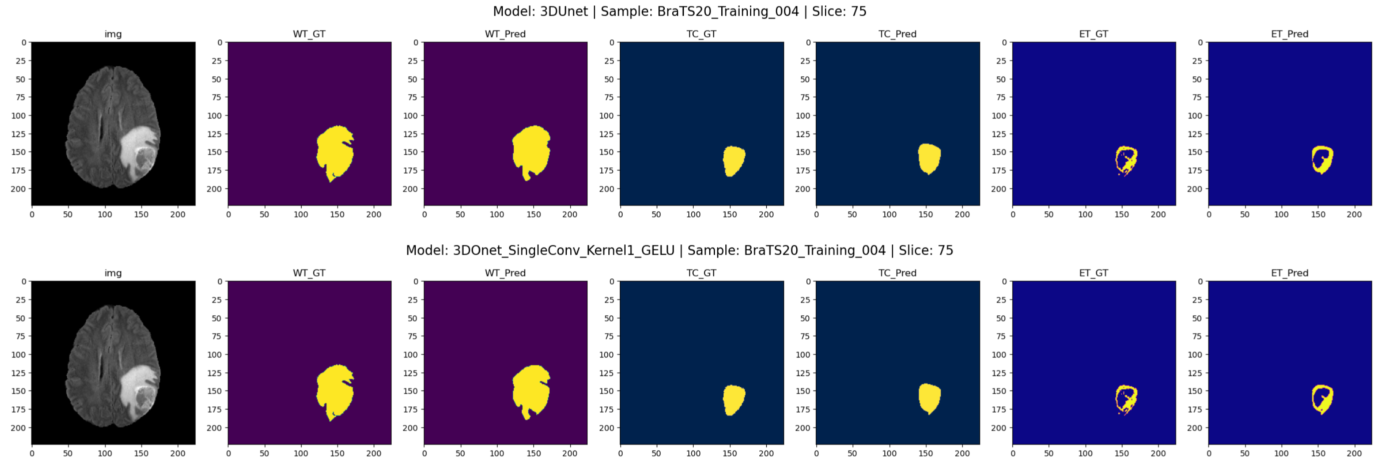

Model: 3DUnet | Sample: BraTS20_Training_004 | Slice: 75
img
WT_GT
WT_Pred
TC_GT
TC_Pred
ET_GT
ET_Pred
Model: 3DOnet_SingleConv_Kernel1_GELU | Sample: BraTS20_Training_004 | Slice: 75
0
25
50
75
100
125
150
175
200
0
50
100
150
200

Figure 8: Sample model prediction against ground truth by `3DUnet` (top) and `3DOnet_SingleConv_Kernel1_GELU_AdamW` (bottom)

6 Discussion

6.1 Observations

6.1.1 The smallest 3D O-NET models (ours) outperform other models

Variants of 3D O-Net models with kernel size = 1 for the upper encoder-decoder section tend to outperform other models; These variants achieved the highest score for 5 out of 6 of the tracked metrics while having among the lowest parameter count. The performance difference is especially pronounced for the TC and ET classes which cover a much smaller area in segmentation masks and hence require greater precision. The greater performance of small 3D O-Net models (kernel size = 1 for the upper encoder-decoder section) could be attributed to the fact that the upper encoder-decoder section is learning a more precise feature representation with its small kernel size.

These results are superior even to that of NVIDIA's Swin UNETR which can be considered state-of-the-art. One possible reason is the differences in the training environment and the fact that Vision Transformer might need more data to unleash its predictive potential; NVIDIA performed much more robust data augmentation/preprocessing steps, used a much larger dataset, and trained their model on a DGX-1 cluster with 8 NVIDIA V100 GPUs for 800 epochs instead of 50. Such steps may be required to utilise Swin UNETR to its fullest potential, and loading a Swin UNETR checkpoint trained by NVIDIA will likely result in much greater accuracy than using our own checkpoint. Nonetheless, our model 3D O-Net models show promising results, especially when compared to other models trained from scratch in data and resource-constrained environments.

6.1.2 Longer inference time is required for 3D O-Net and 3D Sphere-Net

3D O-Net and 3D Sphere-Net models have greater inference time compare to both 3D U-Net and Swin UNETR. This is despite the fact that many 3D O-Net models have a much lower parameter count compared to 3D U-Net and Swin UNETR. One explanation is that backpropagation has to be performed across more layers due to its sequential nature, despite the presence of seemingly parallel paths. This also shows that a lower parameter count does not always corresponds to lower training/inference time.

6.1.3 Replacing ReLU with GELU improves performance

3D U-Net and 3D O-Net show noticeable improvements in performance when the ReLU activation function in their default variants is replaced by GELU. There is a significant increase in performance for 4 out of the 6 metrics between 3DUnet and its GELU variant 3DUnet_GELU. Additionally, 3DUnet_GELU outperformed Swin UNETR and its variants for nearly every performance metric. This may indicate that the GELU activation function is much more suitable for this particular segmentation task.

6.1.4 Simpler models tend to outperform more complex ones

Increasing or decreasing the number of convolution layers per upscale/downscale steps appears to (un-intuitively) have the opposite effect on performance. For example, one might expect that the performance of a model increases when double convolution is used over a single convolution. Yet, the performance for `3DOnet_DoubleConv_Kernel1` actually decreased for 4 out of the 6 metrics compared to its simpler single convolution counterpart, while performance for `3DUnet_SingleConv` increased for 3 out of the 6 metrics compared to its more complex double convolution counterpart. This may be a manifestation of overfitting due to small dataset size or a result of other complex interplay between model architecture and hyperparameters.

6.2 Challenges faced

6.2.1 Incompatible input sizes

One of the initial challenges faced was the fact that the default input size of `(240, 240, 155)` was incompatible with one of the evaluated models (specifically MONAI's implementation of Swin UNETR). We hence implement the function `crop_3d_array` to crop the input to size `(224, 224, 128)` with dimensions that are multiples of 32.

6.2.2 Long training time

Due to the complexity of the task and model, training is prone to interruptions as it can take up to several hours to complete 50 epochs. To address this, the trainer always saves the model checkpoints `your_best_model_{YYYYMMDD_HHmmss}_{epoch}.pth` in the model's `./Logs` folder for the epoch with the best validation accuracy in the format. This allows the checkpoint to be loaded in the future using the Trainer's `load_pretrain_model(path)` method.

6.3 Limitations

There are several limitations faced by our group during the training and evaluation of our models.

6.3.1 Limited compute power

The size and complexity of the dataset used and the models evaluated prevent us from performing extensive hyperparameter tuning and exploring further techniques such as increasing the number and sizes of convolution layers. This also prevents us from replicating the conditions used to train NVIDIA's state-of-the-art approach (Swin UNETR). The task complexity also restricted the number of train, validation and test samples we can use, meaning that the trained models are potentially overfitted to the 53 test samples.

6.3.2 Unusually poor results for certain samples

Certain samples show unusually poor results for classes TC and ET, with dice/jaccard scores of less than 0.5 and 0.001 for TC and ET. One explanation is that these samples are anomalously labelled, or have input images that are misrepresented by their labels. Another explanation is that the area covered by the ground truth TC and ET masks is unusually small for these samples, hence requiring precision that exceeds the model's ability.

6.3.3 Lack of Data Augmentation

Due to the greater compute power required and the danger of generating anatomically incorrect examples, we did not perform any data augmentation. The impact of incorporating anatomically unrealistic examples into training sets still remains an open issue [12].

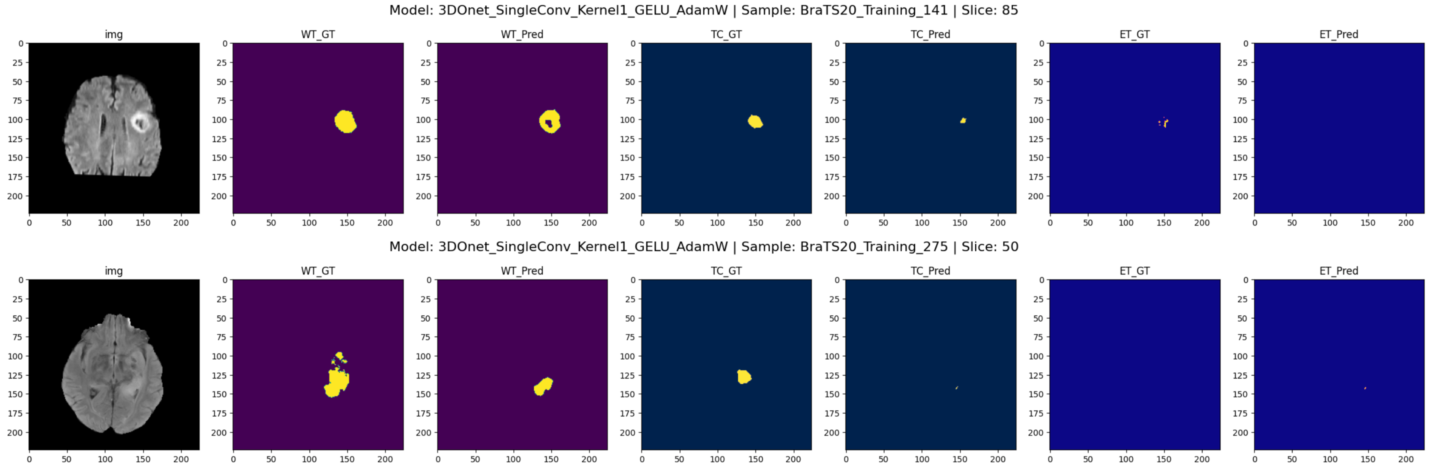

Model: 3DOnet_SingleConv_Kernel1_GELU_AdamW | Sample: BraTS20_Training_141 | Slice: 85
img
WT_GT
WT_Pred
TC_GT
TC_Pred
ET_GT
ET_Pred
Model: 3DOnet_SingleConv_Kernel1_GELU_AdamW | Sample: BraTS20_Training_275 | Slice: 50
img
WT_GT
WT_Pred
TC_GT
TC_Pred
ET_GT
ET_Pred
0
25
50
75
100
125
150
175
200
0
50
100
150
200

Figure 9: Examples of challenging samples

6.4 Future improvements

There are a variety of areas one could explore to improve performance. For example, some of the best-performing models could have been combined into an ensemble model. Other possible methods include increasing the size of the dataset and artificially generating additional training samples via robust data augmentation methods that can generate anatomically correct examples.

7 Conclusion

In this paper, we introduced 3D-ONet which is a novel, performant and parameter-efficient architecture for 3D brain tumour segmentation using multi-modal MRI images. We demonstrated the potential of increasing the number of encoder-decoder paths and validated the effectiveness of this approach through test results which showed that 3D-ONet is competitive with state-of-the-art approaches when trained from scratch under data-scarce resource-constrained environments. Increasing the number of encoder-decoder paths in UNet-style architectures is hence a promising approach that ought to be explored. However, more work is required to further verify the performance of the model compared to state-of-the-art approaches, especially under data-abundant conditions.

8 Acknowledgments

We thank Dr. Matthieu De Mari for his guidance and support.

References

[1] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, L. Lanczi, E. Gerstner, M.-A. Weber, T. Arbel, B. B. Avants, N. Ayache, P. Buendia, D. L. Collins, N. Cordier, J. J. Corso, A. Criminisi, T. Das, H. Delingette, C. Demiralp, C. R. Durst, M. Dojat, S. Doyle, J. Festa, F. Forbes, E. Geremia, B. Glocker, P. Golland, X. Guo, A. Hamamci, K. M. Iftekharuddin, R. Jena, N. M. John, E. Konukoglu, D. Lashkari, J. A. Mariz, R. Meier, S. Pereira, D. Precup, S. J. Price, T. R. Raviv, S. M. S. Reza, M. Ryan, D. Sarikaya, L. Schwartz, H.-C. Shin, J. Shotton, C. A. Silva, N. Sousa, N. K. Subbanna, G. Szekely, T. J. Taylor, O. M. Thomas, N. J. Tustison, G. Unal, F. Vasseur, M. Wintermark, D. H. Ye, L. Zhao, B. Zhao, D. Zikic, M. Prastawa, M. Reyes, and K. Van Leemput, "The multimodal brain tumor image segmentation benchmark(Brats)," *IEEE Transactions on Medical Imaging*, vol. 34, no. 10, pp. 1993–2024, Oct. 2015. [Online]. Available: http://ieeexplore.ieee.org/document/6975210/

[2] O. Ronneberger, P. Fischer, and T. Brox, “U-net: convolutional networks for biomedical image segmentation,” May 2015, arXiv:1505.04597 [cs]. [Online]. Available: http://arxiv.org/abs/1505.04597

[3] Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, “3d u-net: learning dense volumetric segmentation from sparse annotation,” Jun. 2016, arXiv:1606.06650 [cs]. [Online]. Available: http://arxiv.org/abs/1606.06650

[4] L.-M. Hsu, S. Wang, L. Walton, T.-W. W. Wang, S.-H. Lee, and Y.-Y. I. Shih, “3d u-net improves automatic brain extraction for isotropic rat brain magnetic resonance imaging data,” *Frontiers in Neuroscience*, vol. 15, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2021.801008

[5] A. Hatamizadeh, V. Nath, Y. Tang, D. Yang, H. Roth, and D. Xu, “Swin unetr: swin transformers for semantic segmentation of brain tumors in mri images,” Jan. 2022, arXiv:2201.01266 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2201.01266

[6] “Novel transformer model achieves state-of-the-art benchmarks in 3d medical image analysis,” Jun. 2022. [Online]. Available: https://developer.nvidia.com/blog/novel-transformer-model-achieves-state-of-the-art-benchmarks-in-3d-medical-image-analysis/

[7] S.-H. Tsang, “Review: swin transformer,” Aug. 2022. [Online]. Available: https://sh-tsang.medium.com/review-swin-transformer-3438ea335585

[8] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: hierarchical vision transformer using shifted windows,” Aug. 2021, arXiv:2103.14030 [cs]. [Online]. Available: http://arxiv.org/abs/2103.14030

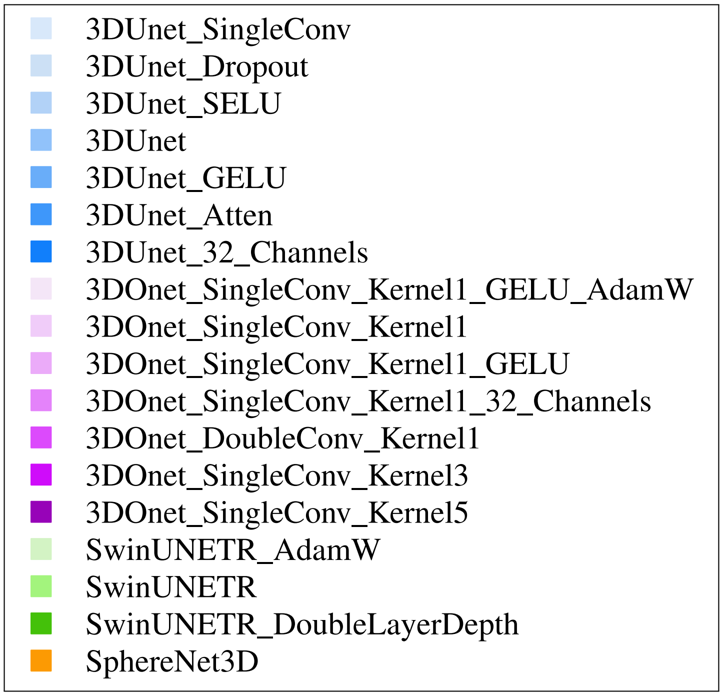
[9] Y. Wu and K. He, “Group normalization,” Jun. 2018, arXiv:1803.08494 [cs]. [Online]. Available: http://arxiv.org/abs/1803.08494

[10] O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz, B. Glocker, and D. Rueckert, “Attention u-net: learning where to look for the pancreas,” May 2018, arXiv:1804.03999 [cs]. [Online]. Available: http://arxiv.org/abs/1804.03999

[11] “Brats20_3dunet_3dautoencoder.” [Online]. Available: https://kaggle.com/code/polomarco/brats20-3dunet-3dautoencoder

[12] J. Nalepa, M. Marcinkiewicz, and M. Kawulok, “Data augmentation for brain-tumor segmentation: a review,” *Frontiers in Computational Neuroscience*, vol. 13, 2019. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fncom.2019.00083

Appendix



🟦	3DUnet_SingleConv
🟦	3DUnet_Dropout
🟦	3DUnet_SELU
🟦	3DUnet
🟦	3DUnet_GELU
🟦	3DUnet_Atten
🟦	3DUnet_32_Channels
🟪	3DOnet_SingleConv_Kernel1_GELU_AdamW
🟪	3DOnet_SingleConv_Kernel1
🟪	3DOnet_SingleConv_Kernel1_GELU
🟪	3DOnet_SingleConv_Kernel1_32_Channels
🟪	3DOnet_DoubleConv_Kernel1
🟪	3DOnet_SingleConv_Kernel3
🟪	3DOnet_SingleConv_Kernel5
🟩	SwinUNETR_AdamW
🟩	SwinUNETR
🟩	SwinUNETR_DoubleLayerDepth
🟧	SphereNet3D

Table 2: **Model Legend**. Darker shades corresponds generally to a greater parameter count. Blue corresponds 3D-UNet variants; Purple corresponds to 3D-ONet variant; Green corresponds to Swin UNETR variants; Orange corresponds to 3D-SphereNet.

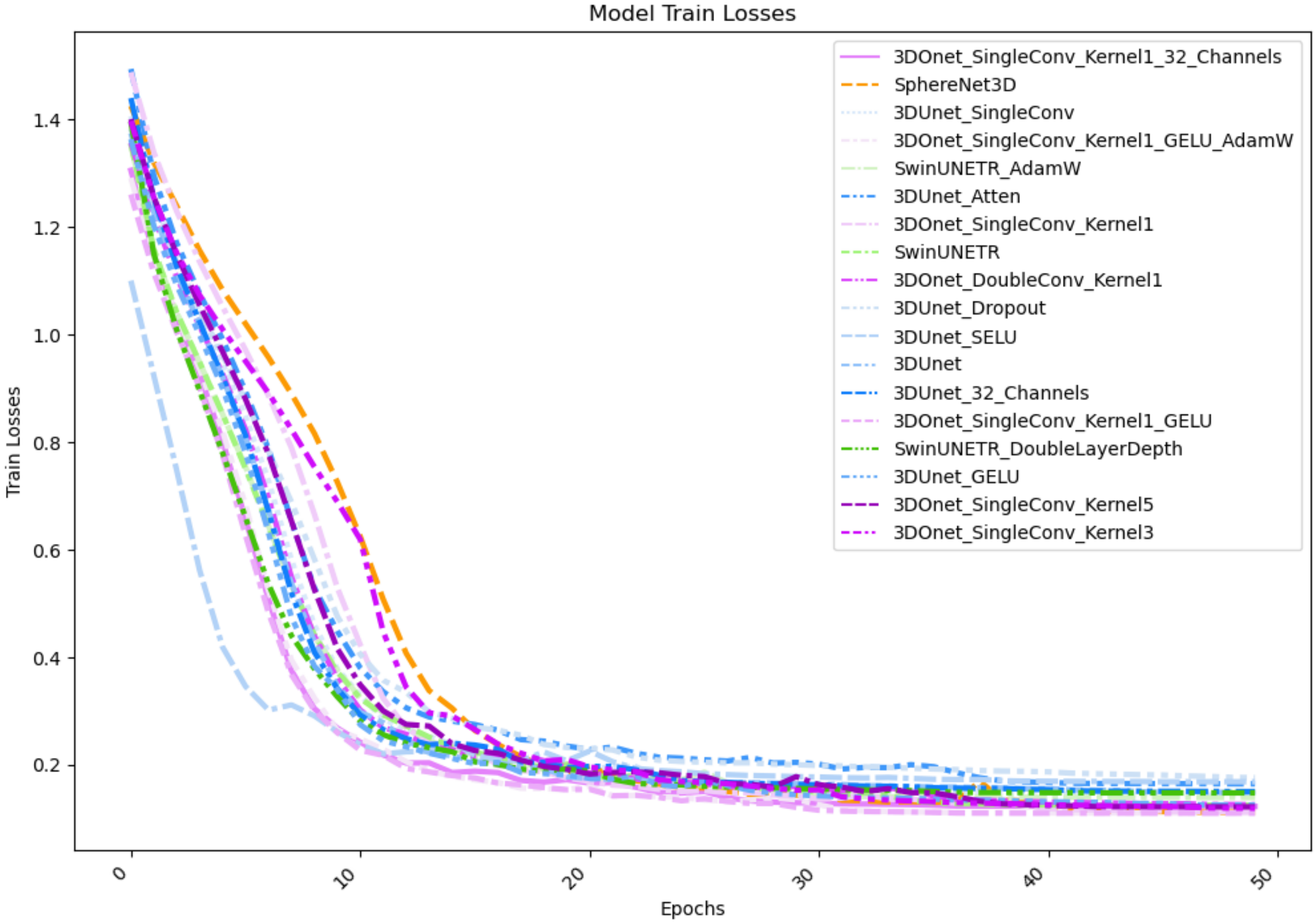


Figure 10: Train losses for all models over 50 epochs

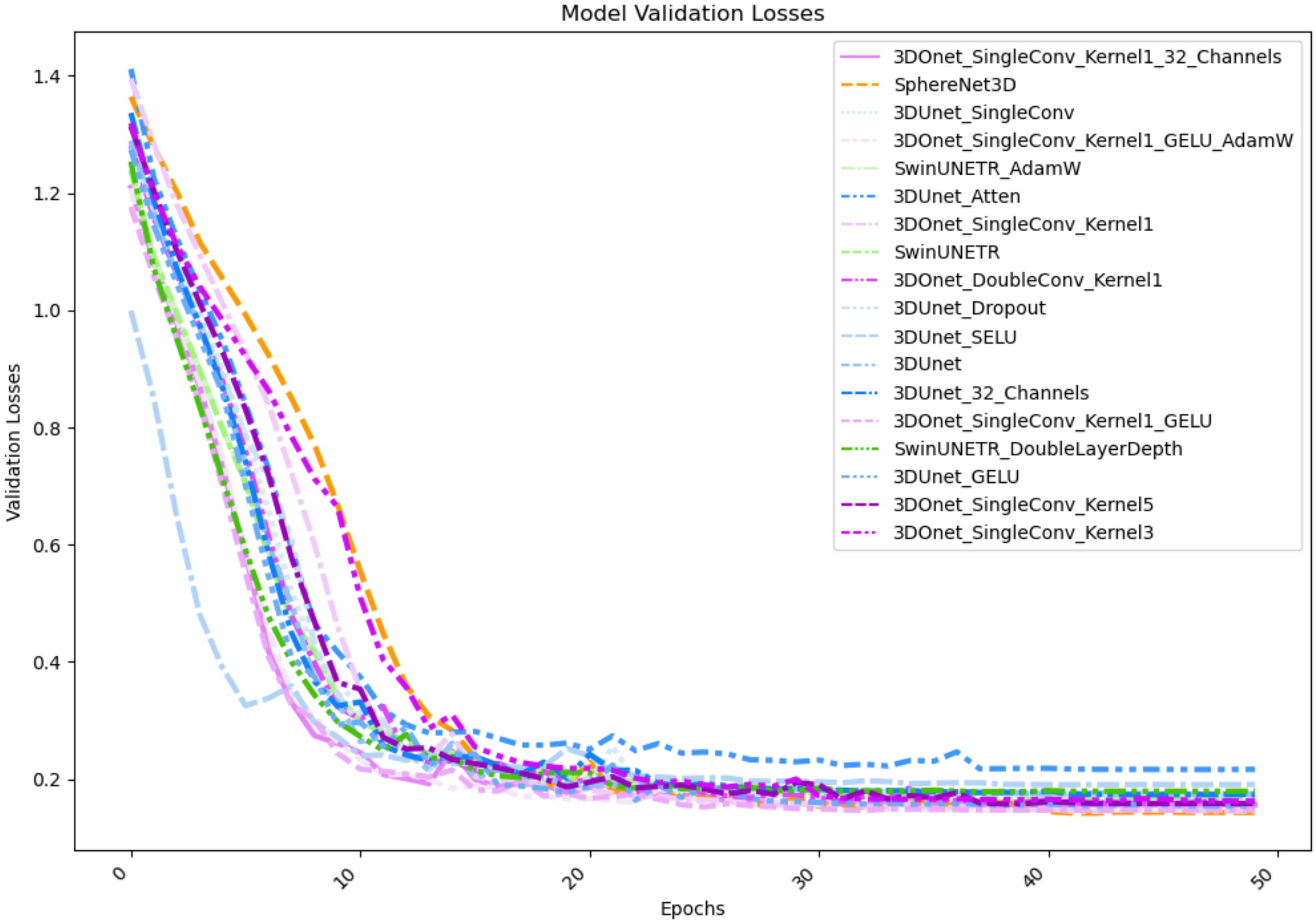


Figure 11: Validation losses for all models over 50 epochs

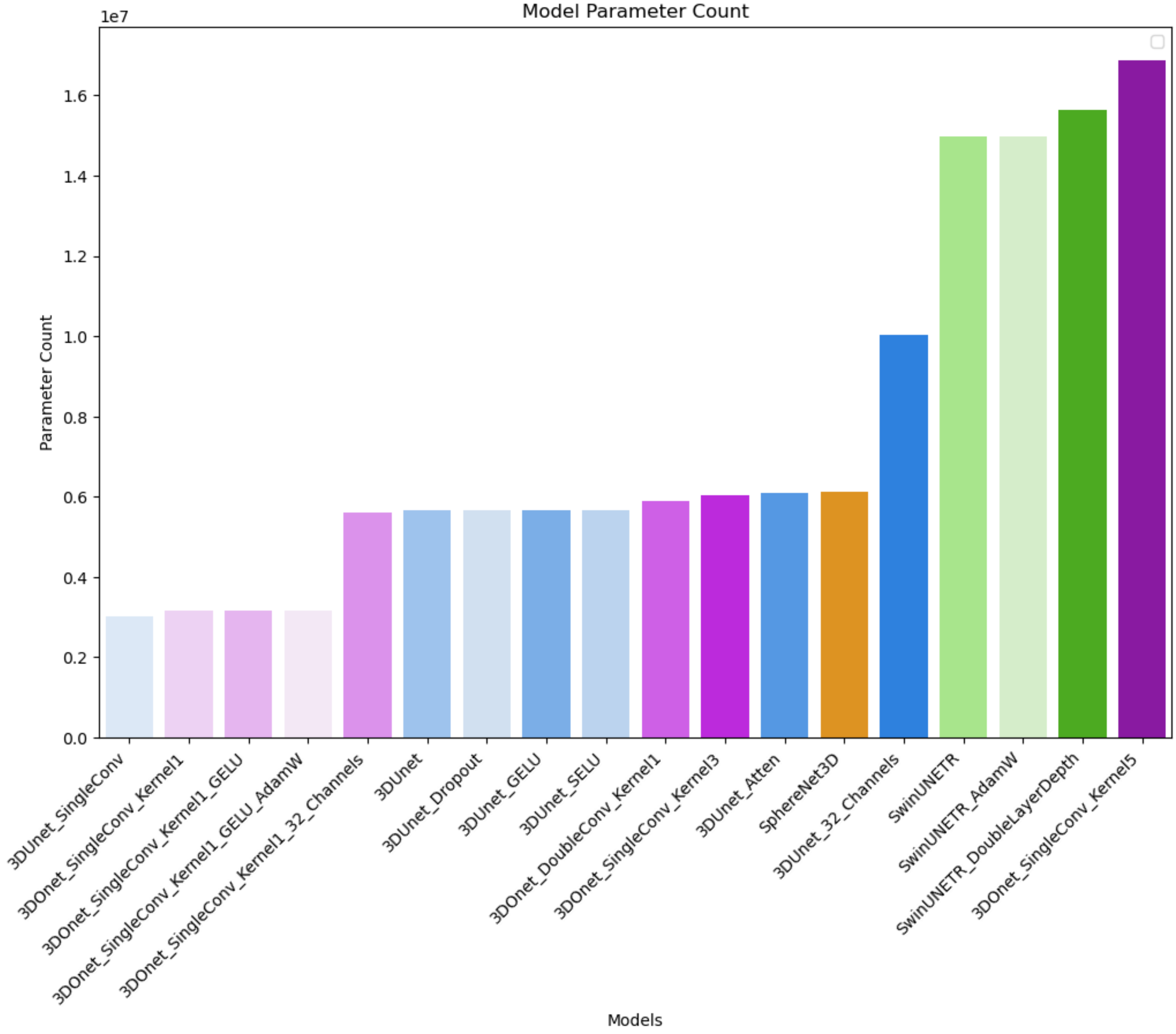


Figure 12: Parameter count for all models in millions

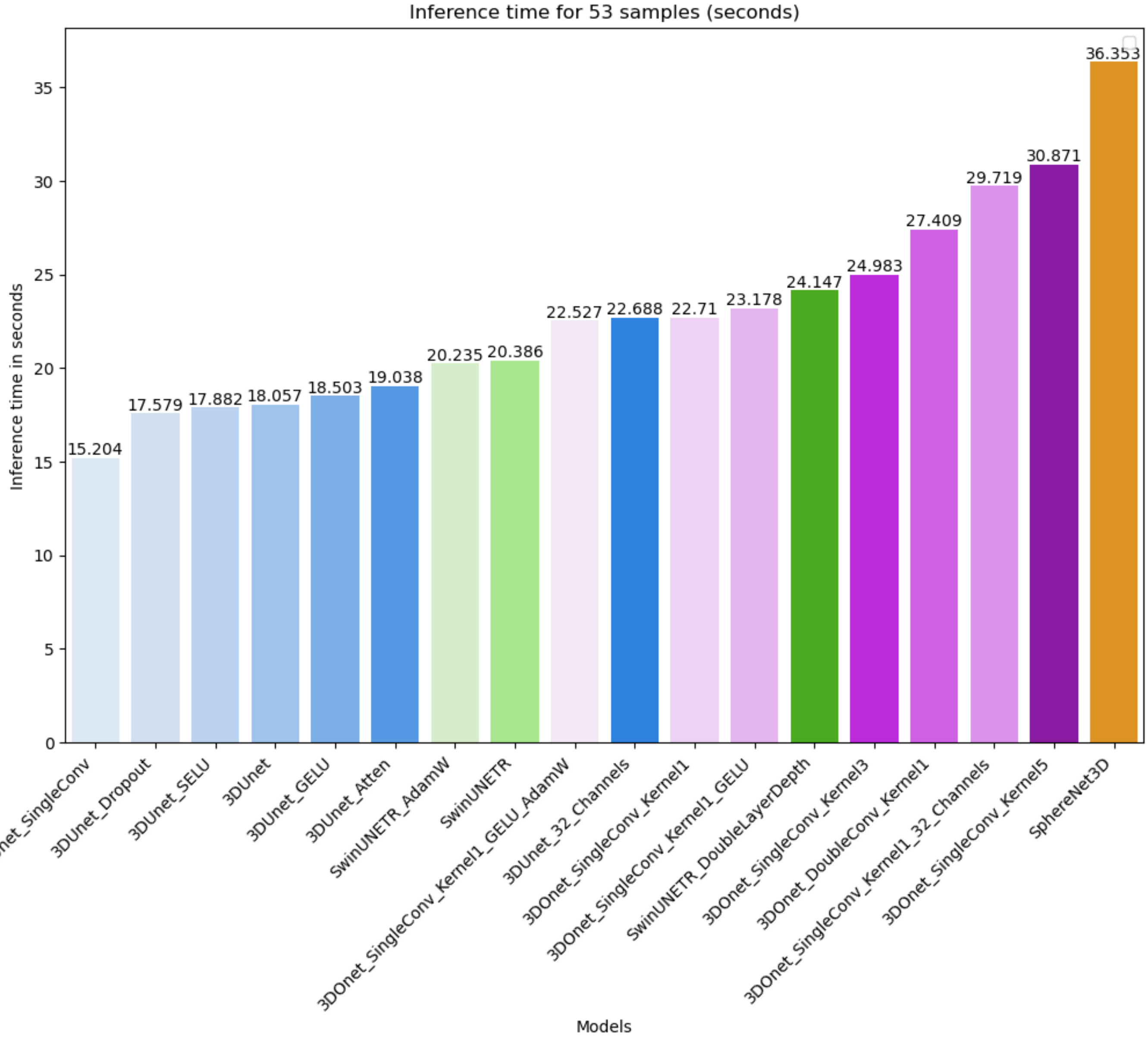


Figure 13: Inference time for all models in seconds

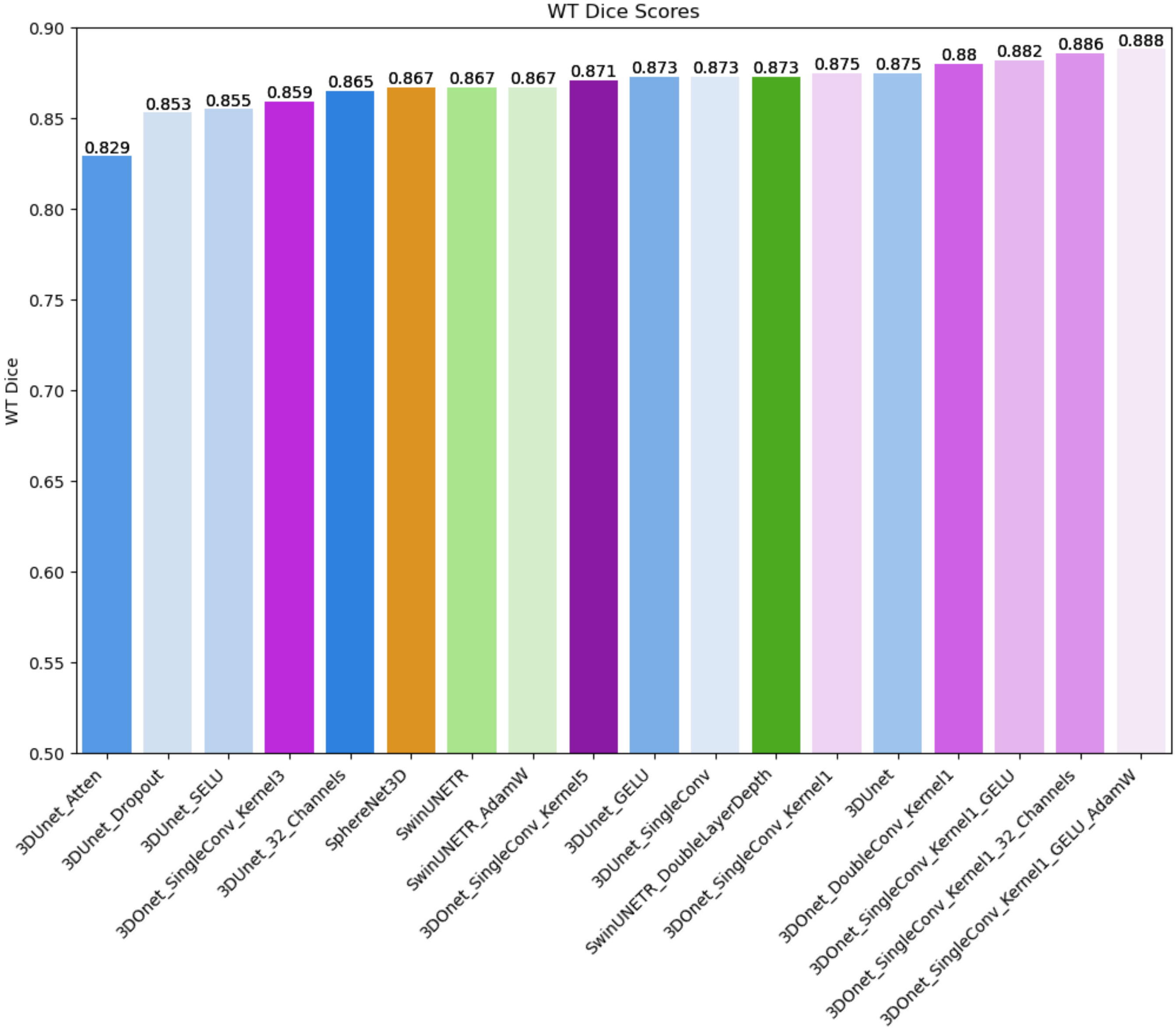


Figure 14: WT dice scores for all models

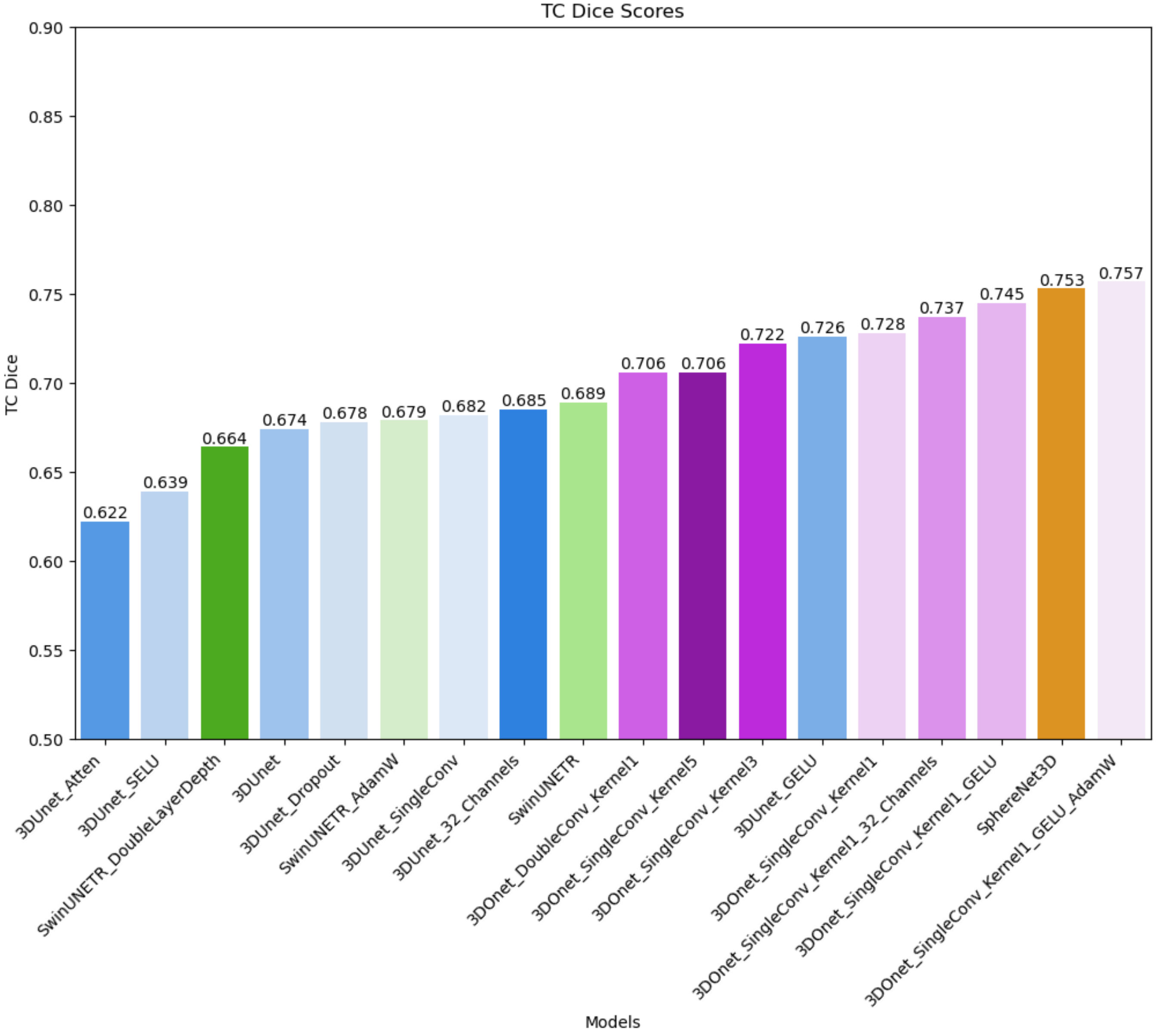


Figure 15: TC dice scores for all models

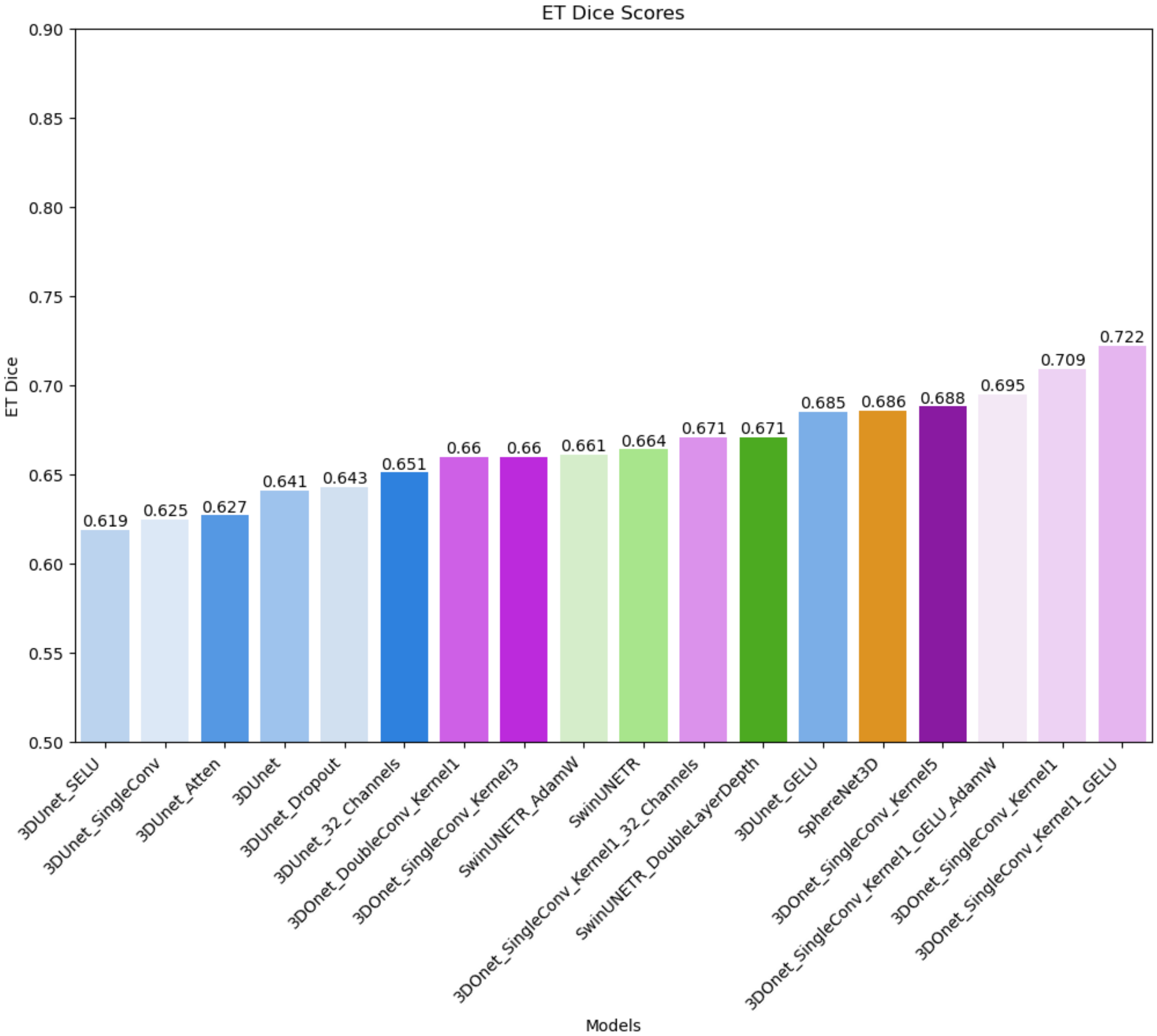


Figure 16: ET dice scores for all models

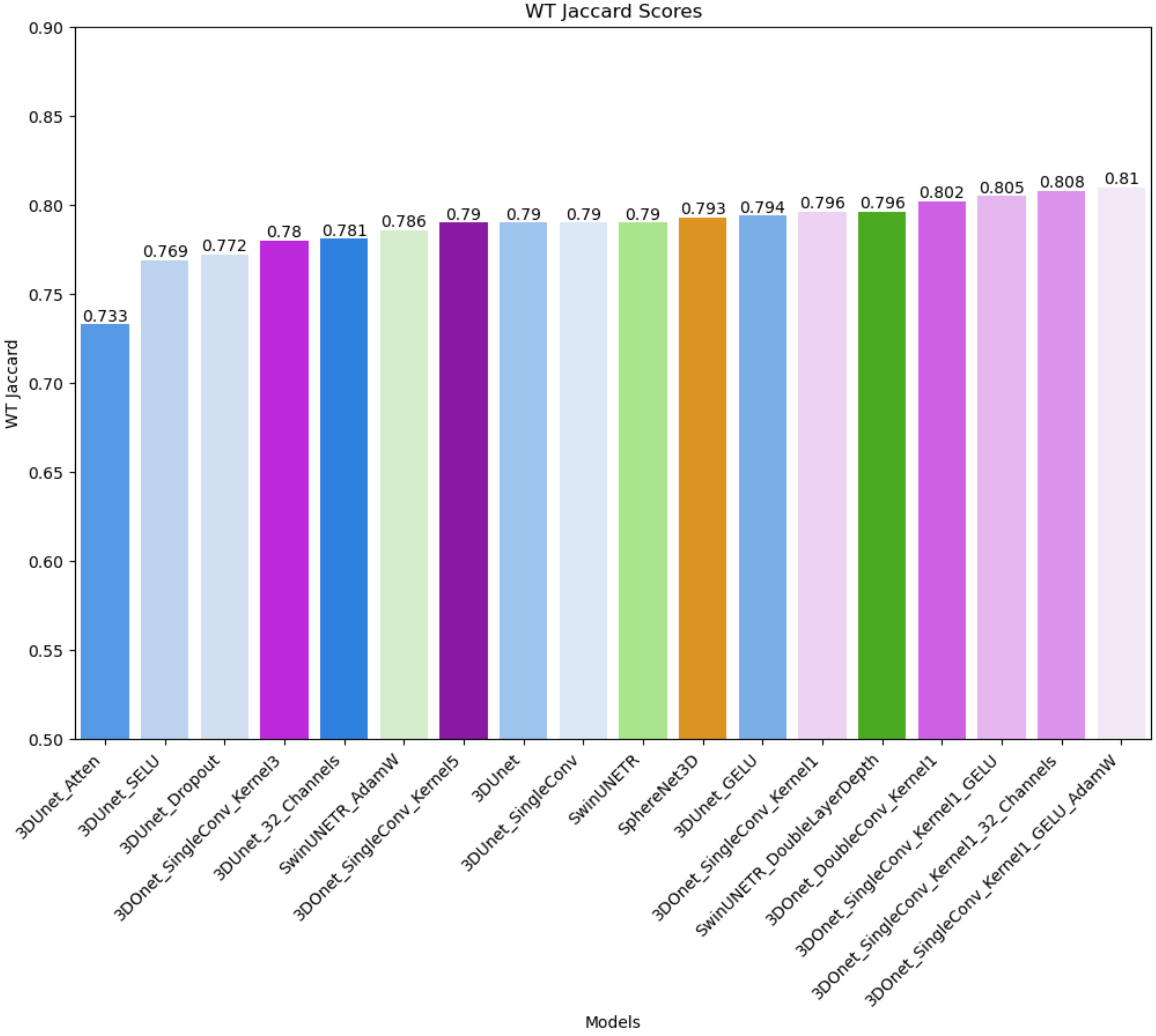


Figure 17: WT jaccard scores for all models

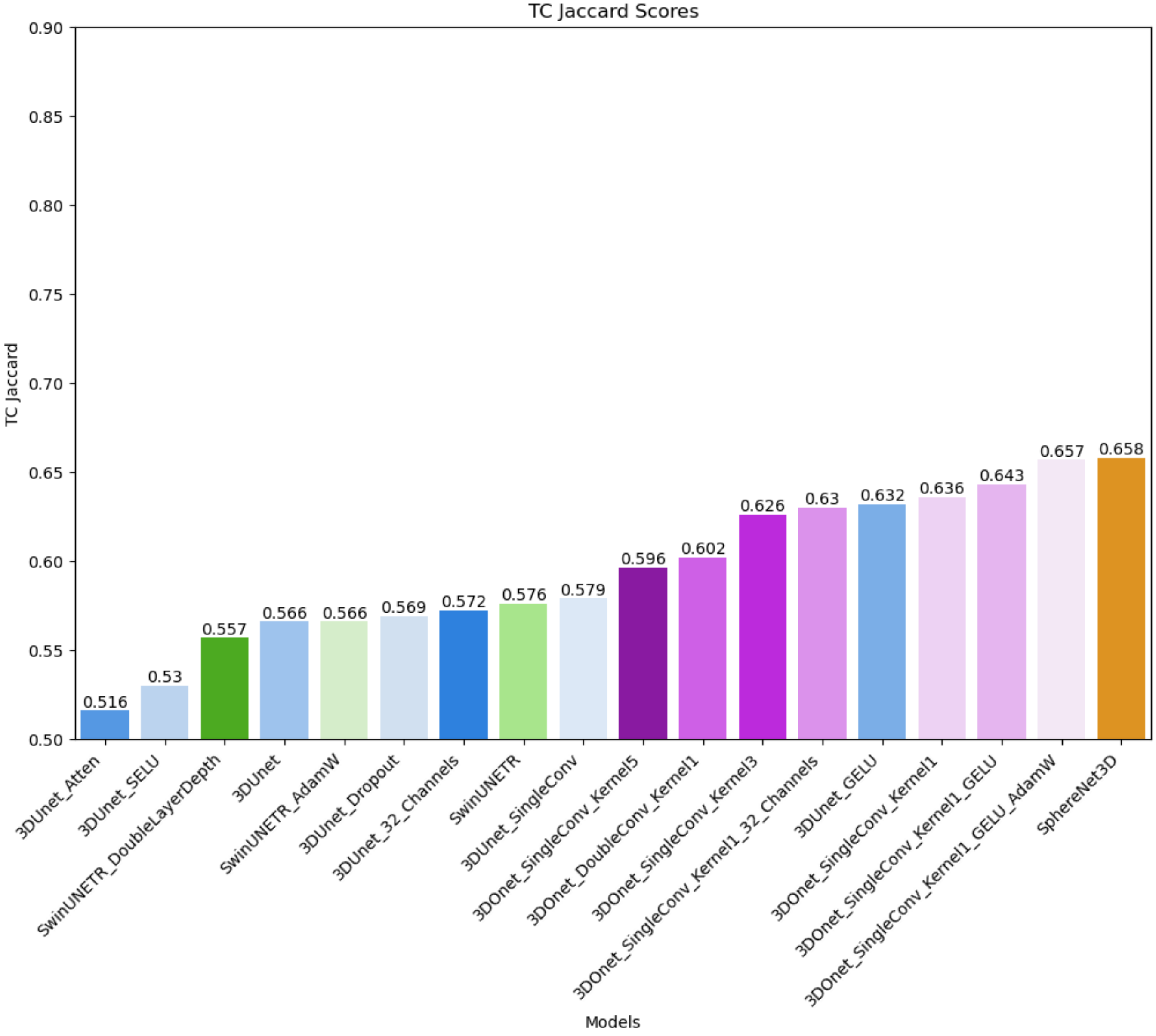


Figure 18: TC jaccard scores for all models

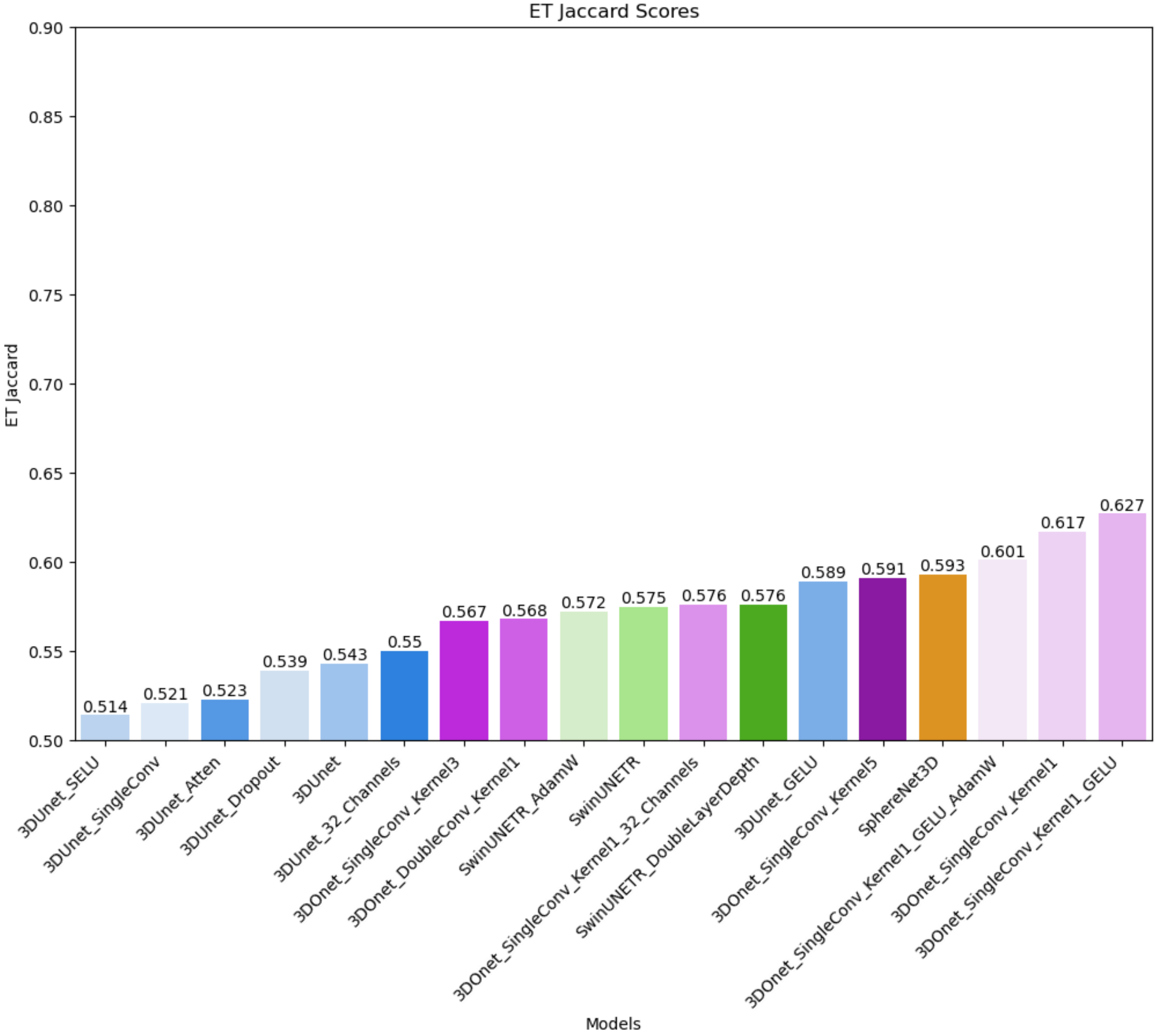


Figure 19: ET jaccard scores for all models

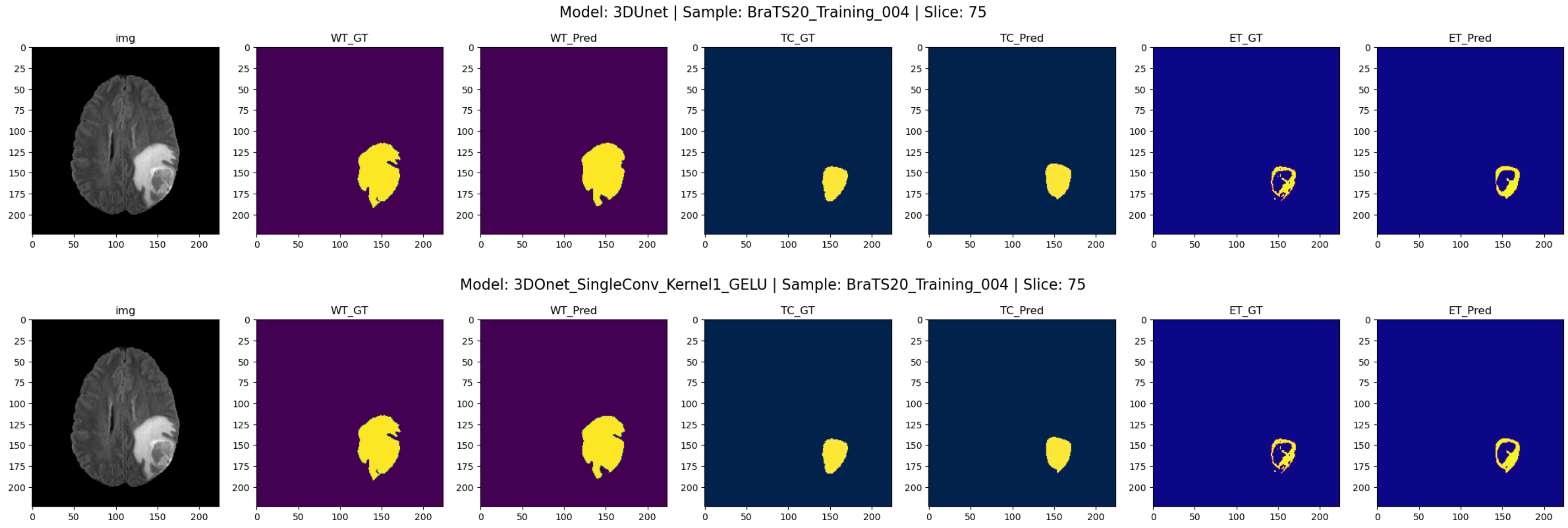

Model: 3DUnet | Sample: BraTS20_Training_004 | Slice: 75
img
WT_GT
WT_Pred
TC_GT
TC_Pred
ET_GT
ET_Pred
Model: 3DOnet_SingleConv_Kernel1_GELU | Sample: BraTS20_Training_004 | Slice: 75
img
WT_GT
WT_Pred
TC_GT
TC_Pred
ET_GT
ET_Pred
0
25
50
75
100
125
150
175
200
0
50
100
150
200

Figure 20: Sample model prediction against ground truth by `3DUnet` (top) and `3DOnet_SingleConv_Kernel1_GELU_AdamW` (bottom)