Optimized U-Net for Brain Tumor Segmentation

Michał Futrega, Alexandre Milesi, Michał Marcinkiewicz, Pablo Ribalta

NVIDIA, Santa Clara CA 95051, USA
{mfutrega,alexandrem,michalm,pribalta}@nvidia.com

Abstract. We propose an optimized U-Net architecture for a brain tumor segmentation task in the BraTS21 challenge. To find the optimal model architecture and the learning schedule, we have run an extensive ablation study to test: deep supervision loss, Focal loss, decoder attention, drop block, and residual connections. Additionally, we have searched for the optimal depth of the U-Net encoder, number of convolutional channels and post-processing strategy. Our method won the validation phase and took third place in the test phase. We have open-sourced the code to reproduce our BraTS21 submission at the NVIDIA Deep Learning Examples GitHub Repository[1].

Keywords: U-Net · Brain Tumor Segmentation · Deep Learning · MRI

1 Introduction

One of the most challenging problems in medical image processing is automatic brain tumor segmentation. Obtaining a computational model capable of surpassing a trained human-level performance would provide valuable assistance to clinicians and would enable a more precise, reliable, and standardized approach to disease detection, treatment planning and monitoring. Gliomas are the most common type of brain tumors in humans [1]. Their accurate segmentation is a challenging medical image analysis task due to their variable shape and appearance in multi-modal magnetic resonance imaging (MRI). Manual segmentation of such brain tumors requires a great deal of medical expertise, is time-consuming, and prone to human error. Moreover, the manual process lacks consistency and reproducibility, which negatively affects the results and can ultimately lead to incorrect prognosis and treatment.

The rapid progress in development of deep learning (DL) algorithms shows great potential for application of deep neural networks (DNNs) in computer-aided automatic or semi-automatic methods for medical data analysis. The drastic improvements of convolutional neural networks (CNNs) resulted in models being able to approach or surpass the human level performance in plethora of applications, such as image classification [2] or microscope image segmentation [3], among many others. DL-based models are great candidates for brain tumor segmentation, as long as sufficient amount of training data is supplied. The Brain

[1] https://github.com/NVIDIA/DeepLearningExamples/blob/master/PyTorch/Segmentation/nnUNet/notebooks/BraTS21.ipynb

Tumor Segmentation Challenge (BraTS) provides a large, high-quality dataset consisting of multi-modal MRI brain scans with corresponding segmentation masks [4,5,6,7,8].

The state-of-the-art models in brain tumor segmentation are based on the encoder-decoder architectures, with U-Net [9] being the most popular for medical image segmentation, based on the citations number. In recent years, U-Net-like architectures were among top submissions to the BraTS challenge. For instance, in 2018, Myronenko *et al.,* modified a U-Net model by adding a variational autoencoder branch for regularization [10]. In 2019, Jiang *et al.,* employed a two-stage U-Net pipeline to segment the substructures of brain tumors from coarse to fine [11]. In 2020, Isensee *et al.,* applied the nnU-Net framework [12] with specific BraTS designed modifications regarding data post-processing, region-based training, data augmentation, and minor modifications to the nnU-Net pipeline [13].

Those achievements prove that well-designed U-Net based architectures have the ability to perform very well on tasks such as brain tumor segmentation. In order to design a competitive solution for challenges like BraTS21, both optimal neural network architecture and training schedule has to be selected. However, there exists a plethora of U-Net variants, for example: Attention U-Net [14], Residual U-Net [15], Dense U-Net [16], Inception U-Net [17], U-Net++ [18], SegResNetVAE [10] or UNETR [19], just to name a few. A wide range of U-Net architectures makes the selection of the optimal one a difficult task. Furthermore, once the neural network architecture is selected, designing a proper training schedule is critical for getting optimal performance. Designing a training schedule is associated with selecting optimal components, such as a loss function, data augmentation strategy, learning rate and its schedule, number of epochs to train, and many more. Also, it is not trivial to decide which model extensions to add, for example, deep-supervision [20] or drop-block [21].

The fact that datasets for medical image segmentation are small (usually around 100 examples), and there is no common benchmark for measuring improvements of different architecture tweaks, often makes such comparisons unreliable. However, the dataset released for BraTS21 provides 2,040 examples (respectively 1251, 219, 570 examples in the training, validation, and test set), which makes it the largest dataset for medical image segmentation at the moment, and a perfect candidate to measure performance improvements for different U-Net variants.

In this paper, we have run extensive ablation studies to select both an optimal U-Net variant and training schedule for the BraTS21 challenge. We have tested U-Net[9], Attention U-Net [14], Residual U-Net [15], SegResNetVAE [10] and UNETR [19] for U-Net variants, and experimented with Deep Supervision [20], Drop-Block [21], and different loss functions (combinations of Cross Entropy, Focal, and Dice). Furthermore, we have optimized our model further by increasing the encoder depth, adding one-hot-encoding channel for the foreground voxels to the input data, and increasing the number of convolutional filters.

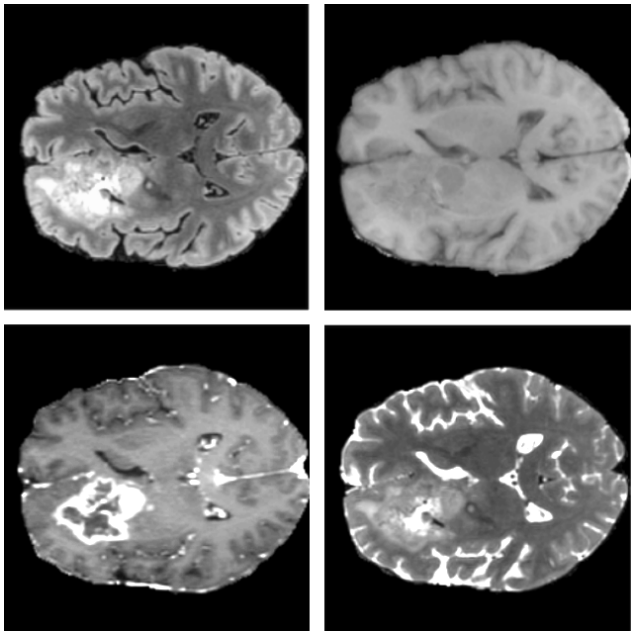

Fig. 1. Example with ID 00000 from the BraTS21 training dataset. Each subplot presents a different modality. From top left to bottom right: FLAIR, T1, T1Gd T2.

2 Method

2.1 Data

The training dataset provided for the BraTS21 challenge [4,5,6,7,8] consists of $1,251$ brain MRI scans along with segmentation annotations of tumorous regions. The 3D volumes were skull-stripped and resampled to 1 mm^3 isotropic resolution, with dimensions of $(240, 240, 155)$ voxels. For each example, four modalities were given: native (T1), post-contrast T1-weighted (T1Gd), T2-weighted (T2), and T2 Fluid Attenuated Inversion Recovery (T2-FLAIR). Example images of each modality are presented on Fig. 1. Segmentation labels were annotated manually by one to four experts. Annotations consist of four classes: enhancing tumor (ET), peritumoral edematous tissue (ED), necrotic tumor core (NCR), and background (voxels that are not part of the tumor).

2.2 Data Preprocessing and Augmentations

Each example of the BraTS21 dataset consists of four NIfTI [22] files with different MRI modalities. As a first step of data pre-processing, all four modalities were stacked such that each example has a shape of $(4, 240, 240, 155)$ (input tensor is in the (C, H, W, D) layout, where C-channels, H-height, W-width and

D-depth). Then redundant background voxels (with voxel value zero) on the borders of each volume were cropped, as they do not provide any useful information and can be ignored by the neural network. Subsequently, for each example, the mean and the standard deviation were computed within the non-zero region for each channel separately. All volumes were normalized by first subtracting the mean and then divided by the standard deviation. The background voxels were not normalized so that their value remained at zero. To distinguish between background voxels and normalized voxels which have values close to zero, an additional input channel was created with one-hot encoding for foreground voxels and stacked with the input data.

Data augmentation is a technique that alleviates the overfitting problem by artificially extending a dataset during the training phase. To make our method more robust, the following data augmentations were used during training phase:

1. **Biased crop**: From the input volume, a patch of dimensions $(5, 128, 128, 128)$ was randomly cropped. Additionally, with probability of 0.4 the patch selected via random biased crop is guaranteed that some foreground voxels (with positive class in the ground truth) are present in the cropped region.
2. **Zoom**: With probability of 0.15, a random value is sampled uniformly from $(1.0, 1.4)$ and image size is resized to its original size times the sampled value with the cubic interpolation, while the ground truth with the nearest neighbour interpolation.
3. **Flips**: With probability of 0.5, for each x, y, z axis independently, volume was flipped along that axis.
4. **Gaussian Noise**: With probability of 0.15, random Gaussian noise with mean zero and standard deviation sampled uniformly from $(0, 0.33)$ is sampled for each voxel and added to the input volume.
5. **Gaussian Blur**: With probability of 0.15, Gaussian blurring with standard deviation of the Gaussian Kernel sampled uniformly from $(0.5, 1.5)$ is applied to the input volume.
6. **Brightness**: With probability of 0.15, a random value is sampled uniformly from $(0.7, 1.3)$ and then input volume voxels are multiplied by it.
7. **Contrast**: With probability of 0.15, a random value is sampled uniformly from $(0.65, 1.5)$ and then input volume voxels are multiplied by it and clipped to their original value range.

2.3 Model architecture

In order to select the most optimal neural network architecture, we have run ablation studies for the following models: U-Net [9], Attention U-Net [14], Residual U-Net [15], SegResNetVAE [10] and UNETR [19]. Below, we present a short description of each model.

U-Net [9] architecture (shown in the Fig. 2) is characterised by a symmetric U-shape, and can be divided into two parts, i.e., encoder and decoder. The first part is the contracting path (encoder) which is transforming the input volume into lower dimensional space. The encoder has a modular structure consisting of repeating convolution blocks. Each block has two smaller blocks of transformations (dark and light blue blocks on the Fig. 2). The first smaller block is reducing the spatial dimensions of the input feature map by a factor of two via convolutional layer with kernels 3x3x3 and stride 2x2x2, then instance normalization and Leaky ReLU activation with negative slope if 0.01 are applied (dark blue block). Next feature map is transformed with almost the same set of operations except that the convolutional layer has stride 1x1x1 (light blue).

After the spatial dimensions of the feature map are transformed to the size of 2x2x2, then the decoder part starts. The decoder also has a modular structure, but its goal is to increase the spatial dimensions by reducing the encoder feature map. The block in the decoder is built from three smaller blocks. The first one is transposed convolution with kernels 2x2x2 and stride 2x2x2, which is increasing the spatial dimensions of the feature map by a factor of two. Then upsampled feature map is concatenated with encoder feature map from the equivalent spatial level and then transformed by two identical blocks with convolutional layer with kernels 3x3x3 and stride 1x1x1, instance normalization and Leaky ReLU activation with negative slope if 0.01 are applied (light blue). Additionally, deep-supervision (Subsection 2.4) can be used, which is computing loss functions for outputs from lower decoder levels.

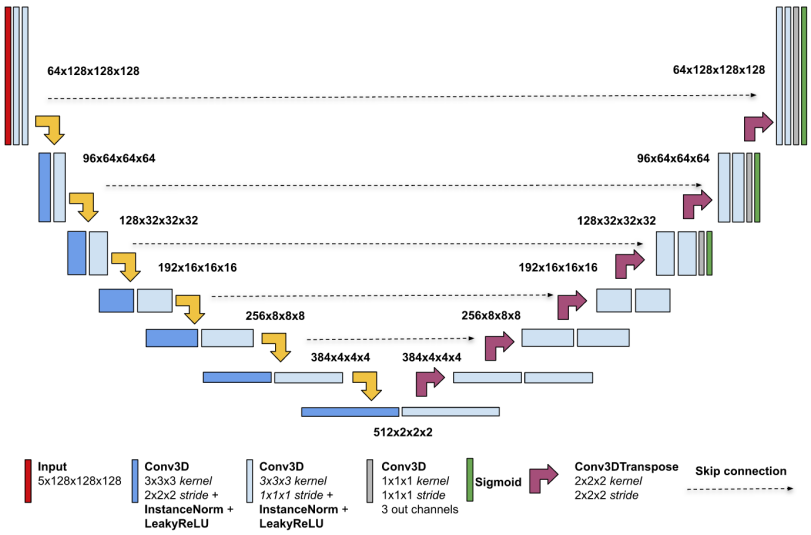

Fig. 2. U-Net architecture. The encoder is transforming the input by reducing its spatial dimensions, and then the decoder is upsampling it back to the original input shape. Additional two output heads are used for deep supervision loss (green bars).

SegResNetVAE [10] is a residual U-Net with autoencoder (shown in the Fig. 3) regularization that has won the BraTS 2018 challenge and modifies U-Net by designing new architecture for encoder blocks and by adding a variational autoencoder (VAE) [23] branch in the decoder, which reconstructs the input and has a regularization effect.

The encoder part uses ResNet like blocks, where each block consists of two convolutions with group normalization and ReLU activation, followed by additive identity skip connection. The decoder structure is similar to the encoder, but only with a single block per each spatial level. Each decoder block begins with reducing the number of channels by a factor of 2 (with 1x1x1 convolution) and doubling the spatial dimension (using 3D bilinear), followed by an addition with encoder feature map from the equivalent spatial level.

In the VAE branch in the decoder, first the feature map from the bottleneck is reduced into a low dimensional space of 256 (128 to represent mean, and 128 to represent std). Then, a sample is drawn from the Gaussian distribution with the given mean and std, and reconstructed into the input image dimensions following the same architecture as the decoder.

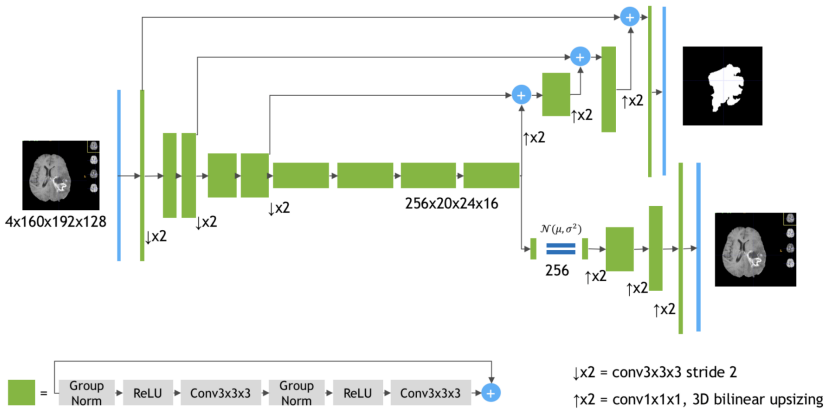

Fig. 3. SegResNetVAE architecture. Each green block is a ResNet-like block with the group normalization. The VAE branch reconstructs the input image into itself, and is used only during training to regularize the shared encoder. Image from [10].

UNETR [19] architecture (shown in the Fig. 4) is a generalization of Vision Transformer (ViT) [24] to the 3D convolutions—it replaces the 3D convolutions in the encoder with multi-head self-attention [25]. To convert a 3D input volume into an input for a multi-head self-attention it is divided into a sequence of uniform non-overlapping patches (with 16x16x16 shape) and projected into an embedding space (with 768 dimensions) using a linear layer, and added with a positional embedding. Such input is then transformed by a multi-head self-attention encoder.

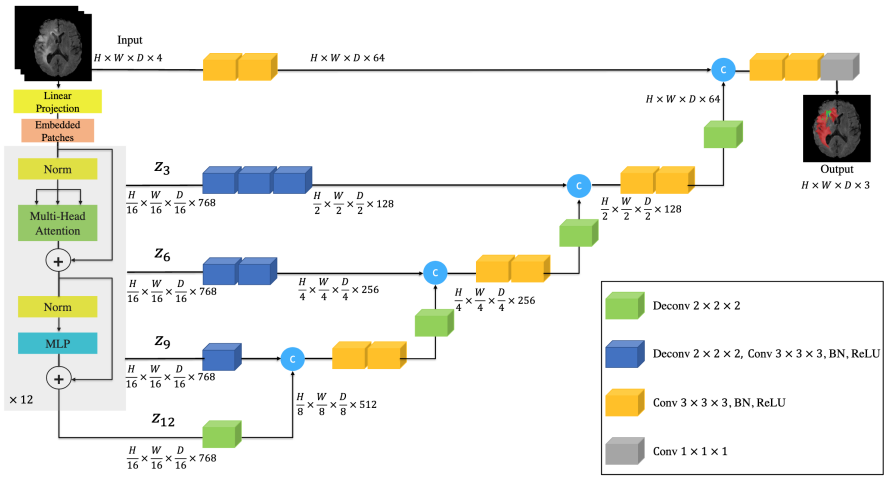

Fig. 4. UNETR architecture. Instead of using 3D convolution in the encoder, UNETR is transforming the input volume via multi-head self-attention blocks known from the Transformer model. Image from [19].

Attention U-Net [14] is extending base U-Net by adding an attention gate (shown in the Fig. 5) in the decoder part. Attention gate is transforming the feature map from the encoder before the concatenation in the decoder block. It learns which regions of the encoder feature map are the most important, considering the context of the feature map from the previous decoder block. This is achieved by multiplication of the encoder feature map with the weights computed by the attention gate. The weight values are in the (0, 1) range and represent the attention level that the neural network is paying to a given pixel.

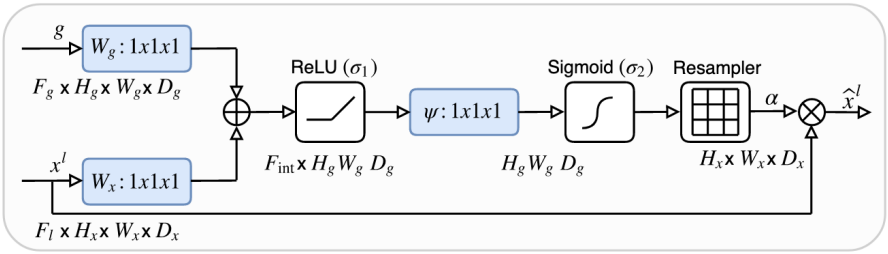

Fig. 5. The architecture of the attention gate. Input features (x^l) are multiplied by attention weights (α). To compute α, input features (x^l), and feature map from corresponding encoder level are first transformed by 1x1x1 convolution, and the summed. Next, ReLU activation and another 1x1x1 convolution are applied. Finally, attention weights are upsampled with trilinear interpolation. Image from [14].

Residual U-Net [15] is inspired by a ResNet model [15] where residual connections were proposed. Adding residual connections is helping with training a deep neural network due to better gradient flow. The only difference between basic U-Net and Residual U-Net is the computation within a convolutional block, which is shown in the Fig. 6.

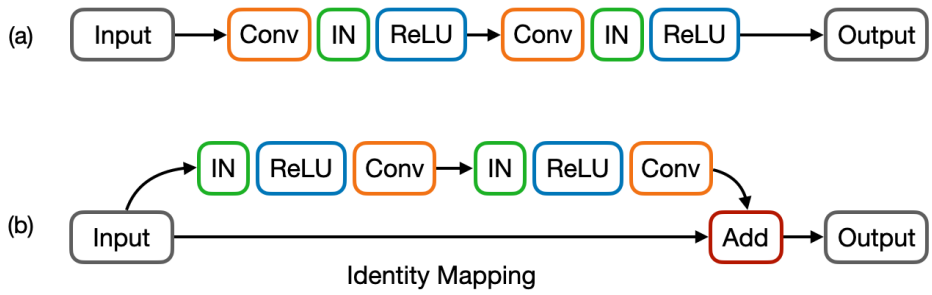

Fig. 6. Difference between blocks in basic U-Net (a) and Residual U-Net (b). Conv block (orange) corresponds to a convolutional layer with 3x3x3 kernels, IN (green) is instance normalization and ReLU (blue) is a Rectified Linear Unit activation.

Based on our experiments (the detailed results are shown in subsection 3.3), a basic U-Net achieves the best results, and was selected for further exploration. The next optimization was adjusting the encoder depth and optimal selection of the convolution channels. As a baseline, a default U-Net architecture from the nnU-Net framework was used, i.e., the depth of the network was 6, and the convolution channels at each encoder level were: $32, 64, 128, 256, 320, 320$. Our experiments have demonstrated that increasing the depth of the encoder to 7, and modifying the number of channels to: $64, 96, 128, 192, 256, 384, 512$, further improves the baseline score.

2.4 Loss function

Based on the *nnU-Net for Brain Tumor Segmentation* [13] paper, the classes present in the label were converted to the three partially overlapping regions: whole tumor (WT) representing classes 1, 2, 4; tumor core (TC) representing classes 1, 4; and enhancing tumor (ET) representing the class 4. The contest leaderboard is computed based on those overlapping regions instead of classes present in the labels. It is beneficial to construct the loss function based on classes used for ranking calculation, thus we designed the output feature map to have three channels (one per class) which at the very end are transformed via the sigmoid activation.

Each region was optimized separately with a sum of binary cross-entropy or Focal loss [26] (with gamma parameter set to 2) with the Dice loss [27]. For Dice loss, its batched variant was used, i.e., Dice loss was computed over all samples in the batch instead of averaging the Dice loss over each sample separately.

Deep supervision [20] is a technique that helps with a better gradient flow by computing loss function on different decoder levels. In this work, we added two additional output heads, marked by green bars on Fig. 2. To compute the deep supervision loss, labels were first downsampled using nearest neighbor interpolation to the $(64, 64, 64)$ and $(32, 32, 32)$ spatial shapes such that they match the shapes of additional outputs. For labels y_i and predictions p_i for $i = 1, 2, 3$, where $i = 1$ corresponds to the last output head, $i = 2$ is the output head on the penultimate decoder level and $i = 3$ is before the penultimate, final loss function is computed as follows:

$$\mathcal{L}(y_1, y_2, y_3, p_1, p_2, p_3) = \mathcal{L}(y_1, p_1) + \frac{1}{2}\mathcal{L}(y_2, p_2) + \frac{1}{4}\mathcal{L}(y_3, p_3). \tag{1}$$

2.5 Inference

During inference, the input volume can have arbitrary size, instead of the fixed patch size $(128, 128, 128)$ as during the training phase. Thus, we used a sliding window inference [2], where the window has the same size as the training patch, i.e., $(128, 128, 128)$ and adjacent windows overlap by half the size of a patch. The predictions on the overlapping regions are then averaged with Gaussian importance weighting, such that the weights of the center voxels have higher importance, as in the original nnU-Net paper [12].

One of the known ways to improve robustness of predictions is to apply test time augmentations. During inference, we have created eight versions of the input volume, such that each version corresponds to one of eight possible flips along the x, y, z axis combination. Then we run inference for each version of the input volume and transform the predictions back to the original input volume orientation by applying the same flips to predictions as were used for the input volume. Finally, the probabilities from all predictions were averaged.

By optimizing the three overlapping regions (ET, TC, WT) we had to convert them back to the original classes (NCR, ED, ET). The strategy for transforming classes back to the original one is the following: if the WT probability for a given voxel is less than 0.45 then its class is set to 0 (background), otherwise if the probability for TC is less than 0.4 the voxel class is 2 (ED), and finally if probability for ET is less than 0.45 voxel has class 1 (NCR), or otherwise 4 (ET).

Furthermore, we applied the following post-processing strategy: find ET connected components, for components smaller than 16 voxels with mean probability smaller than 0.9, replace their class to NCR (such that voxels are still considered part of the tumor core), next if there is overall less than 73 voxels with ET and their mean probability is smaller than 0.9 replace all ET voxels to NCR. With such post-processing we avoided the edge case where the model predicted a few voxels with enhancing tumor, but there were not any in the ground truth. Such post-processing was beneficial to the final score as if there were no enhancing tumor voxels in the label, then the Dice score for zero false positive prediction was 1, and 0 otherwise.

[2] MONAI sliding window implementation was used.

That methodology was tested on the validations sets from the 5-fold cross-validation. Hyperparameters were selected to yield the highest score combined on all the folds. The threshold value was selected via a grid search method with a step of 0.05 in the range (0.3, 0.7). Similarly, the number of voxels was searched in the range (0, 100) and selected by maximizing score on the 5-fold cross-validation.

3 Results

3.1 Implementation

Our solution is written in PyTorch [28] and extends NVIDIA's implementation of the nnU-Net. The code is publicly available on the NVIDIA Deep Learning Examples GitHub repository [3]. Proposed solution is using the NVIDIA NGC PyTorch 21.07 Docker container [4] which allows for the full encapsulation of dependencies, reproducible runs, as well as easy deployment on any system. All training and inference runs were performed with use of Mixed Precision [29], which speeds-up the model and reduces the GPU memory consumption. Experiments were run on NVIDIA DGX A100 (8×A100 80 GB) system.[5]

3.2 Training schedule

Each experiment was trained for 1,000 epochs using the Adam optimizer [30] with three different learning rates: 0.0005, 0.0007, 0.0009 and a weight decay equal to 0.0001. Additionally, during the first 1000 steps, we used a linear warm-up of the learning rate, starting from 0 and increasing it to the target value, and then it was decreased with a cosine annealing scheduler [31]. The weights for 3D convolutions were initialized with Kaiming initialization [32].

For model evaluation, we used 5-fold cross validation and compared the average of the highest Dice score reached on each of the 5-folds. The evaluation on the validation set was run after every epoch. For each fold, we have stored the two checkpoints with the highest mean Dice score on the validation set reached during the training phase. Then during the inference phase, we ensembled the predictions from stored checkpoints by averaging the probabilities.

3.3 Experiments

To select the model architecture, we experimented with three U-Net variants: baseline U-Net [9] which architecture follows the nnU-Net [12] architecture heuristic, UNETR [19] which replaces the U-Net encoder with a Vision Transformer (ViT) [24] generalization for the 3D convolutions, and U-Net with autoencoder

[3] https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/Segmentation/nnUNet

[4] https://ngc.nvidia.com/catalog/containers/nvidia:pytorch

[5] https://www.nvidia.com/en-us/data-center/a100

regularization (SegResNetVAE) which extends U-Net architecture with variational autoencoder (VAE) [23] branch for input reconstruction in the decoder.

Model	U-Net	UNETR	SegResNetVAE
Fold 0	**0.9087**	0.9044	0.9086
Fold 1	**0.9100**	0.8976	0.9090
Fold 2	**0.9162**	0.9051	0.9140
Fold 3	**0.9238**	0.9111	0.9219
Fold 4	**0.9061**	0.8971	0.9053
Mean Dice	**0.9130**	0.9031	0.9118

Table 1. Averaged Dice scores of ET, TC, WT classes for each 5-folds comparing the baseline U-Net, UNETR, SegResNetVAE models.

Presented results in the Table 1 have shown that baseline U-Net achieves the highest score. Although the score of SegResNetVAE is similar to the plain U-Net, the training time is three times longer compared to U-Net, because of the additional VAE branch. Thus, we decided to select U-Net architecture for further exploration.

In the next phase of experiments we tested various U-Net architecture tweaks: decoder attention [14], deep supervision [20], residual connections [15] and drop block [21]. Additionally, we have experimented with the modified loss function with Focal loss [26] instead of cross-entropy, so that the loss function was Focal+Dice.

The experimental results presented in the Table 2 have shown that the only extension which significantly improves the 5-fold average Dice score over the baseline U-Net (0.9130) was the deep supervision (0.9149).

Model	baseline	Attention	DS	Residual	DB	Focal
Fold 0	0.9087	0.9091	**0.9111**	0.9087	0.9096	0.9094
Fold 1	0.9100	0.9110	**0.9115**	0.9103	0.9114	0.9026
Fold 2	0.9162	0.9157	**0.9175**	**0.9175**	0.9159	0.9146
Fold 3	0.9238	0.9232	**0.9268**	0.9233	0.9241	0.9229
Fold 4	0.9061	0.9061	**0.9074**	0.9070	0.9071	0.9072
Mean Dice	0.9130	0.9130	**0.9149**	0.9134	0.9136	0.9133

Table 2. Averaged Dice scores of ET, TC, WT classes for each 5-folds comparing the decoder attention (Attention), deep supervision (DS), residual connections (Residual), drop block (DB) and Focal loss (Focal).

Finally, for the U-Net with deep supervision, we tested the modification of the U-Net encoder. The baseline U-Net architecture follows the architecture heuristic from the nnU-Net [12] framework for which the depth of the network was 6, and the convolution channels at each encoder level were: $32, 64, 128, 256, 320, 320$. We experimented with an encoder of depth 7, modified the number of channels to: $64, 96, 128, 192, 256, 384, 512$, and checked the input volume with an additional channel with one-hot encoding for foreground voxels.

Model	DS	Deeper	Channels	One-hot	D+C+O
Fold 0	0.9111	**0.9118**	0.9107	0.9109	**0.9118**
Fold 1	0.9115	0.9140	0.9135	0.9132	**0.9141**
Fold 2	0.9175	0.9170	0.9173	0.9174	**0.9176**
Fold 3	**0.9268**	0.9256	0.9265	0.9263	**0.9268**
Fold 4	0.9074	**0.9079**	0.9072	0.9075	0.9076
Mean Dice	0.9149	0.9152	0.9150	0.9050	**0.9156**

Table 3. Averaged Dice scores of ET, TC, WT classes for each 5-folds comparing the deep supervision (DS), deeper U-Net encoder, modified number of convolution channels, additional input channel with one-hot encoding for foreground voxels, and all modification applied together (D+C+O) i.e., deeper U-Net with changed number of convolution channels and one-hot encoding channel for foreground voxels.

The results in the Table 3 have shown that applying each of the modifications separately is slightly improving the score over baseline U-Net with deep supervision (0.9149), however if using all modifications together then the score is further improved (0.9156).

Finally, we experimented with a post-processing strategy. It is known from previous BraTS editions that removing small regions with enhanced tumor can be beneficial to the final score. It is so because if there is no enhancing tumor in the label, then the Dice score for zero false positive prediction is 1, and 0 otherwise. The best strategy we found for our 5-fold cross-validation is the following: find ET connected components, for components smaller than 16 voxels with mean probability smaller than 0.9, replace their class to NCR, next if there is overall less than 73 voxels with ET and their mean probability is smaller than 0.9 replace all ET voxels to NCR.

Post-processing	without	with
Fold 0	0.9118	**0.9132**
Fold 1	0.9141	**0.9142**
Fold 2	0.9176	**0.9189**
Fold 3	**0.9268**	**0.9268**
Fold 4	0.9076	**0.9086**
Mean Dice	0.9156	**0.9163**

Table 4. Averaged Dice scores of ET, TC, WT classes for each 5-folds without and with post-processing.

The best model, i.e., deeper U-Net with deep supervision, modified number of channels and additional input one-hot encoding channel for foreground voxels was the winner of the challenge validation phase. The detailed scores are shown in the Table 5.

1	2	3	4	5	6	7	8	9
0.267	0.272	0.287	0.289	0.298	0.305	0.306	0.312	0.316

Table 5. Top 9 normalized statistical ranking scores for BraTS21 validation phase.

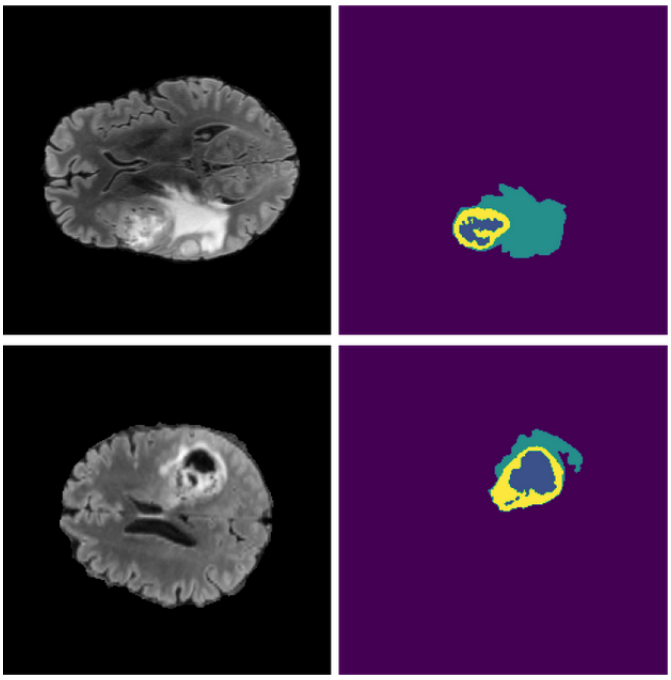

Fig. 7. Predictions on the challenge validation dataset. On the left column FLAIR modality is visualized while on the right model predictions where the meaning of colors is the following: purple - background, blue - NCR, turquoise - ED, yellow - ET.

4 Conclusions

We have experimented with various U-Net variants (basic U-Net [9], UNETR [19], SegResNetVAE [10], Residual U-Net [15], and Attention U-Net [14]), architecture modifications and training schedule tweaks like: deep supervision [20], drop block [21], and Focal loss [26]. Based on our experiments, U-Net with deep supervision yields the best results which can be further improved by adding an additional input channel with one-hot encoding for foreground, increasing encoder depth together with a number of convolutional channels and designing a post-processing strategy.

References

1. Goodenberger, M.L., Jenkins, R.B.: Genetics of adult glioma. Cancer Genetics **205** (12 2012). https://doi.org/10.1016/j.cancergen.2012.10.009
2. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision **115** (12 2015). https://doi.org/10.1007/s11263-015-0816-y

3. Zeng, T., Wu, B., Ji, S.: DeepEM3D: approaching human-level performance on 3D anisotropic EM image segmentation. Bioinformatics **33**(16), 2555–2562 (03 2017). https://doi.org/10.1093/bioinformatics/btx188, https://doi.org/10.1093/bioinformatics/btx188
4. Baid, U., et al: The RSNA-ASNR-MICCAI BraTS 2021 Benchmark on Brain Tumor Segmentation and Radiogenomic Classification (2021)
5. Menze, B.H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., et al.: The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). IEEE Transactions on Medical Imaging **34**(10), 1993–2024 (2015). https://doi.org/10.1109/TMI.2014.2377694
6. Bakas, S., Akbari, H., Sotiras, A., Bilello, M., Rozycki, M., Kirby, J., et al.: Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features. Scientific Data **4** (09 2017). https://doi.org/10.1038/sdata.2017.117
7. Bakas, S., Akbari, H., Sotiras, A., Bilello, M., Rozycki, M., Kirby, J., et al.: Segmentation Labels and Radiomic Features for the Pre-operative Scans of the TCGA-GBM collection (07 2017). https://doi.org/10.7937/K9/TCIA.2017.KLXWJJ1Q
8. Bakas, S., Akbari, H., Sotiras, A., Bilello, M., Rozycki, M., Kirby, J., et al.: Segmentation Labels and Radiomic Features for the Pre-operative Scans of the TCGA-GBM collection (07 2017). https://doi.org/10.7937/K9/TCIA.2017.GJQ7R0EF
9. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. pp. 234–241. Springer International Publishing, Cham (2015)
10. Myronenko, A.: 3D MRI Brain Tumor Segmentation Using Autoencoder Regularization. In: Crimi, A., Bakas, S., Kuijf, H., Keyvan, F., Reyes, M., van Walsum, T. (eds.) Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries. pp. 311–320. Springer International Publishing, Cham (2019)
11. Jiang, Z., Ding, C., Liu, M., Tao, D.: Two-Stage Cascaded U-Net: 1st Place Solution to BraTS Challenge 2019 Segmentation Task. In: Crimi, A., Bakas, S. (eds.) Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries. pp. 231–241. Springer International Publishing, Cham (2020)
12. Isensee, F., Jäger, P.F., Kohl, S.A., Petersen, J., Maier-Hein, K.H.: nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. pp. 1–9. Nature Methods (2020)
13. Isensee, F., Jäger, P.F., Full, P.M., Vollmuth, P., Maier-Hein, K.H.: nnU-Net for Brain Tumor Segmentation. In: Crimi, A., Bakas, S. (eds.) Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries. pp. 118–132. Springer International Publishing, Cham (2021)
14. Oktay, O., Schlemper, J., Folgoc, L.L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N.Y., Kainz, B., Glocker, B., Rueckert, D.: Attention u-net: Learning where to look for the pancreas (2018)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015)
16. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks (2016)
17. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Deep residual learning for image recognition (2014)
18. Zhou, Z., Siddiquee, M.M.R., Tajbakhsh, N., Liang, J.: Unet++: A nested u-net architecture for medical image segmentation (2018)

19. Hatamizadeh, A., Yang, D., Roth, H., Xu, D.: UNETR: Transformers for 3D Medical Image Segmentation (2021)
20. Zhu, Q., Du, B., Turkbey, B., Choyke, P.L.., Yan, P.: Deeply-supervised cnn for prostate segmentation (2017)
21. Ghiasi, G., Lin, T.Y., Le, Q.V.: Dropblock: A regularization method for convolutional networks (2018)
22. Cox, R., Ashburner, J., Breman, H., Fissell, K., Haselgrove, C., Holmes, C., Lancaster, J., Rex, D., Smith, S., Woodward, J., Strother, S.: A (sort of) new image data format standard: NiFTI-1. vol. 22 (01 2004)
23. Kingma, D.P., Welling, M.: Auto-encoding variational bayes (2014)
24. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale (2021)
25. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2017)
26. Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár: Focal Loss for Dense Object Detection. International Conference on Computer Vision (ICCV) (2017)
27. Fausto Milletari, Nassir Navab, Seyed-Ahmad Ahmadi: V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. International Conference on 3D Vision (3DV) (2016)
28. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
29. Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., Wu, H.: Mixed precision training (2018)
30. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2017)
31. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts (2017)
32. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification (2015)
33. Antonelli, M., Reinke, A., Bakas, S., Farahani, K., AnnetteKopp-Schneider, Landman, B.A., Litjens, G., Menze, B., Ronneberger, O., Summers, R.M., van Ginneken, B., Bilello, M., Bilic, P., Christ, P.F., Do, R.K.G., Gollub, M.J., Heckers, S.H., Huisman, H., Jarnagin, W.R., McHugo, M.K., Napel, S., Pernicka, J.S.G., Rhode, K., Tobon-Gomez, C., Vorontsov, E., Huisman, H., Meakin, J.A., Ourselin, S., Wiesenfarth, M., Arbelaez, P., Bae, B., Chen, S., Daza, L., Feng, J., He, B., Isensee, F., Ji, Y., Jia, F., Kim, N., Kim, I., Merhof, D., Pai, A., Park, B., Perslev, M., Rezaiifar, R., Rippel, O., Sarasua, I., Shen, W., Son, J., Wachinger, C., Wang, L., Wang, Y., Xia, Y., Xu, D., Xu, Z., Zheng, Y., Simpson, A.L., Maier-Hein, L., Cardoso, M.J.: The Medical Segmentation Decathlon (2021)
34. NVIDIA nnU-Net implementation. https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/Segmentation/nnUNet, accessed: 2021-09-30