Brain tumor Segmentation with variations of 3DUNets

S. SHAYAN DANESHVAR and S. REZA RAZAVI

In the past few years, deep learning has been used tremendously in medical imaging problems, including brain tumor segmentation. Brain tumor segmentation is useful for doctors to quicken the process of labeling brain tumors, which is a time-consuming task. In this paper, we implement three different convolutional neural networks based on UNet and report their segmentation results, namely attention 3D UNet, Residual 3DUNet, and vanilla 3DUNet. We also use the segmentation results to predict the survival days of the patients using a vanilla fully-connected neural network. The project's code is available at github.com/shayandaneshvar/braTS-2020

CCS Concepts: • **Computing methodologies** → **Image segmentation**.

Additional Key Words and Phrases: brain tumor segmentation, deep learning, convolutional neural networks, 3DUNet

ACM Reference Format:

S. Shayan Daneshvar and S. Reza Razavi. 2023. Brain tumor Segmentation with variations of 3DUNets. In . ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION AND RELATED WORKS

According to the National Brain Tumor Society, there has been an increase in the number of cases of brain lesions in recent years. It is estimated that approximately 700,000 people in the United States are living with brain tumors, and this figure is expected to rise by the end of 2020 [3]. While it is true that, in comparison to other forms of cancer such as breast or lung cancer, the affliction of a brain tumor is not more prevalent, it is nevertheless a malignancy that commands much apprehension, as it stands as the tenth leading cause of death on a global scale. [4]. According to statistics obtained from the United States, an astonishing 18,020 adults are projected to lose their lives to brain cancer [6]. Furthermore, the specific brain lesion associated with this illness can trigger a whirlwind of harmful consequences on the patient's cognitive abilities, resulting in a chain of ruinous after-effects that may affect not just the brain but also other crucial organs.

Brain tumors can be categorized as malignant or benign, with the former being cancerous and quickly spreading to other brain parts. However, both types of tumors can result in dysfunction or life-threatening symptoms for the human body due to their growth in the rigid space of the brain. Magnetic Resonance Imaging (MRI) and resection surgery are currently the most common means of diagnosing and treating brain tumors. However, accurately marking the tumor region is a priority for neurosurgeons. Unfortunately, manual labeling is a laborious and time-consuming task for doctors, and replicating segmentation results can be difficult due to practical operation factors.

The adoption of Deep Learning (DL) in medical imaging is a widely acknowledged phenomenon, thanks to its remarkable ability to learn complex representations from raw data, thus rendering human engineering and domain expertise for feature extraction unnecessary. Many studies have confirmed the success of DL-based applications in brain tumor segmentation. Deep Convolutional Neural Networks (DCNN) have been demonstrated to be highly effective in both natural and medical image segmentation tasks, including those related to brain tumor segmentation [1, 8, 10].

Manuscript submitted to ACM

However, the dependence of DCNN methods on large amounts of training data with expert annotations represents a significant challenge, especially when sourcing such data from multiple institutions. Conversely, the U-Net network structure, which is extensively utilized, comprises a contracting path designed to capture context and a symmetric expanding path that enables accurate localization and segmentation of the entire input image [13]. The MSD2018 [2, 14] and BraTS2020 [4, 5, 11] databases have provided a challenging platform for brain tumor segmentation by AI techniques and attracted enormous attention and interest from researchers in this field. The Multimodal Brain tumor Segmentation Challenge (BraTS) dataset, which includes multi-institutional multimodal MRI scans of glioblastoma and lower-grade glioma, has attracted many researchers to submit their fully automatic brain tumor segmentation algorithms and received significant results.

The UNet, [13] was the first high-impact encoder-decoder structure that was widely employed for medical image segmentation. In one study, a 3DUNet-based deep learning model was trained using brain-wise normalization and patching strategies for brain tumor segmentation. Additionally, the study extracted numerical features from predicted tumor labels and used them for predicting overall survival days [15]. In another study, a 3D UNet model was developed with adaptations in the training and testing strategies, network structures, and model parameters for brain tumor segmentation. The ensemble of multiple models trained with different hyper-parameters was used to improve performance. This study also developed a linear model based on features extracted from segmentation and other clinical features to predict patient overall survival [7]. Other variations of U-Net are also used as a tool for image segmentation. In [12], the authors proposed a novel attention UNet architecture incorporating attention gates into standard CNN architectures such as UNet to be evaluated on two large CT abdominal datasets for multi-class image segmentation and consistently improved prediction performance while preserving computational efficiency.

In this project, we propose a solution based on deep learning for brain tumor segmentation. Specifically, we plan to use three different 3D UNet models, including residual, attention, and vanilla 3D UNet [17], for tumor segmentation. For our database, we will use the BraTS dataset.

In addition to tumor segmentation, we will use the tumor volume in segmented images and age information provided in the dataset to estimate overall survival days. Data of patients whose resection status was GTR (aka gross total Resection – these are the patients who underwent surgery) will be used for training in the second part. We will use a fully connected neural network with at least two hidden layers to predict survival rates. By combining deep learning-based segmentation and survival estimation, we aim to improve brain tumor diagnosis and treatment accuracy and efficiency, ultimately benefiting patients and medical professionals.

2 METHODS

In this section, we talk about various methods that we used, and in the next section, we justify why we used them.

2.1 Dataset

(Shayan) For the dataset, we used the BraTS2020 dataset which contains 371 labeled train data and 129 unlabeled validation data. As the test data was not widely available to the public and the validation set lacked labels, we used the training set to create our new training and validation sets, dedicating 60% of the data to training and the remaining 40% to validation. Hence, we ended up with 221 training data items and 148 validation data items.

The dataset's images have the dimension of 240x240x155, but most of the space in all images is blank, thus we cut the images to some extent to reduce the blank space and achieved images of size 128x128x128. Also, the dataset has 4 different sets of inputs, namely native (T1), post-contrast T1-weighted (T1CE), T2-weighted (T2), and T2 Fluid

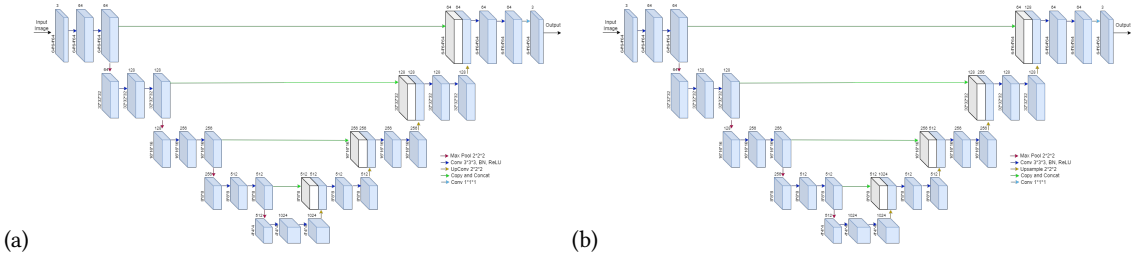
Attenuated Inversion Recovery (FLAIR) images, which the second one is the same as the first one but with better contrast, so we did not use the first one as its data was already available in the second image types. The dataset has 3 different masks, namely necrotic and the non-enhancing tumor core (NCR/NET), edema (ED), and enhancing tumor (ET). They are presented with numbers 1,2,4 alongside 0 for background. We changed 4's to 3's to reduce the one-hot representation size of them, and removed the background from the dataset, as it can be inferred from the other labels and the fact that the background is not interesting by itself.

2.2 Segmentation Models' Architectures

We used three different deep-learning architectures in different settings, all of which are based on UNet[13], and are trained with similar data, losses, and environments.

All of these models were purely implemented with Pytorch and without the use of any specific resources available online, yet we tried to keep the models as close as possible to the descriptions given in their original papers unless mentioned otherwise.

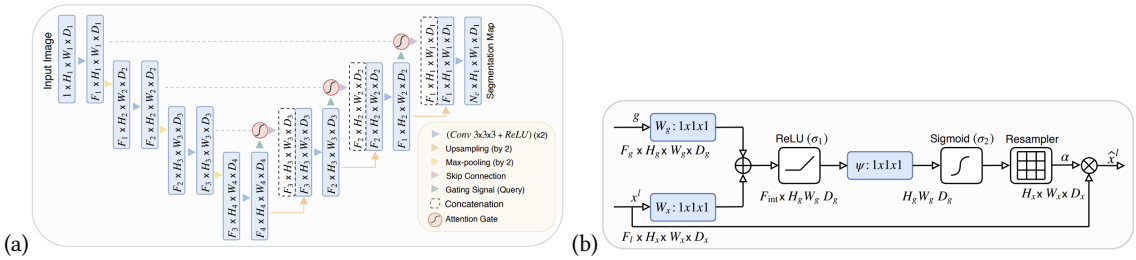
We trained each of the following models with two different loss functions for 100 epochs in batches of 4 and with Adam optimizer with a learning rate of 0.0001, leaving with 10 different models for segmentation.



(a) (b)

Fig. 1. Our 3DUNets Architectures (a) Vanilla 3DUNet (b) 3DUNet with Upsample instead of convolution transpose

2.2.1 Vanilla 3DUNets. **(Shayan)** UNet is a well-known CNN used for medical segmentation, comprised of an encoder and a decoder network connected via a bottleneck, taught in detail in class. We implemented the original 3DUNet [17], which uses convolution transpose layers instead of upsampling. We also implemented the same network but replaced the convolution transpose layers with upsampling which introduced some challenges, and interestingly increased the number of parameters. The architecture of both models is depicted in great detail in figure 1.

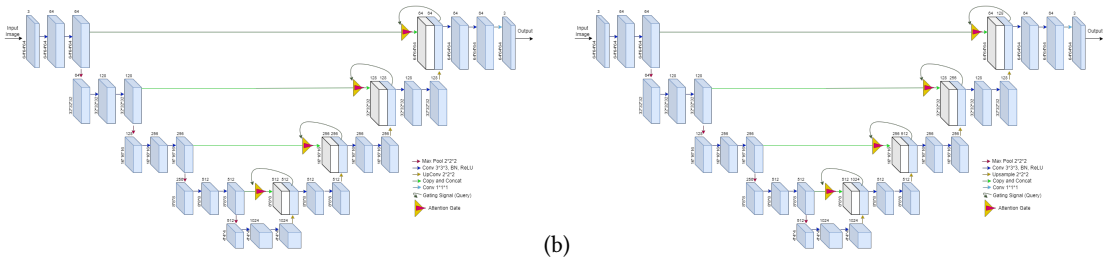


(a) (b)

Fig. 2. Original Attention UNet Figures [12] (a) The Attention UNet architecture (b) The Architecture of the attention block

2.2.2 Attention 3DUNets. **(Shayan)** In order to implement the attention mechanism, various options were available. We used the original Attention UNet [12] descriptions and changed them from 2D to 3D by replacing all 2D convolutions, batch normalization, and max pooling to their 3D variants. The reason I chose the original version was because of its popularity as well as the fact that it was the most sophisticated, and is considered to be a spatial attention. The other alternative which is far more trivial is to use channel attention which is easier to implement, yet does not make much of a difference in the end, and to me, it made no sense, as a filter's job is to find the best weights to be multiplied by pixels, and the filter itself will learn which channel is more important, to give it more weight.

The original Attention UNet which is shown in figure 2 replaces the raw skip connections with attention blocks, which take in G and X, which are the output of the coarser layer and the corresponding layer in the encoder, applies a point-wise 1*1*1 convolution on both (The query and the key in attention terminology), add them together, apply Relu, another pointwise convolution with a single filter, then sigmoid (this can be thought of the value) and finally upsample the output and multiply it to the skip connection. For the sake of novelty, I thought of the idea of self-attention taught in class, hence I tried to come up with something that somehow merged the idea of spatial and self-attention together. Hence, instead of taking G from a coarser scale before upscaling, I took it when upscaling was already applied, so in some sense, the attention is totally calculated from the current layer and the skip connection. This idea saved me from having to use a separate upsampler at the end of the attention block, making the network faster. The novel architecture of the whole module is depicted in figure 3. The Attention module however is similar to the original attention UNet 2, but without the upsampler which is applied at the end before the final multiplication.



(a) (b)

Fig. 3. My Attention UNets Architectures (a) With convolution transpose (b) With Upsample

To make the code more understandable, I leveraged object-oriented programming to inherit from the vanilla 3DUNet implemented previously. In this way, this is obvious that Attention 3DUNet is simply a 3DUNet that has Attention blocks on the skip connections, and the input from the encoder layers does not directly go to the corresponding decoder layers, instead a probability map is computed using both the current layer and the corresponding layer in the encoder, and then it is multiplied to the corresponding output of the layer in the encoder.

2.2.3 Deep Residual 3DUNets. **(Reza)** Going deeper would improve the performance of a multi-layer neural network, however, could hamper the training, and a degradation problem may occur [9]. To overcome these problems, He et al. [9] proposed the residual neural network to facilitate training and address the degradation problem. Residual UNets is a modified version of U-Net architecture that includes residual connections to better capture feature maps and enhance the network's performance. Figure 4 shows the difference between a plain and residual unit.

Our implementation for Deep Residual 3DUNet was based on its 2D implementation of Deep Residual UNets used in [16]. Deep Residual UNets combine the strengths of both U-Net and Residual UNets. This combination brings us two

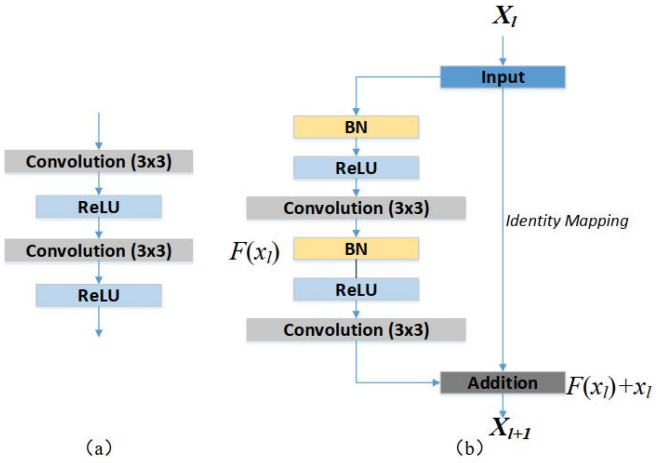


Fig. 4. Building blocks of neural networks [16]. (a) Plain neural unit used in U-Net and (b) residual unit with identity mapping used in the proposed residual UNets.

benefits: 1) the residual unit will ease training of the network; 2) the skip connections within a residual UNets and between low levels and high levels of the network will facilitate information propagation without degradation.

Zhang et al. [16] utilized a 7-level architecture of deep Residual UNets for road area extraction, as shown in figure 5. The network comprises of three parts: encoding, bridge, and decoding. The first part encodes the input image into compact representations. The last part recovers the representations to a pixel-wise categorization, i.e. semantic segmentation. The middle part serves as a bridge connecting the encoding and decoding paths. All of the three parts are built with residual units which consist of two 3 × 3 convolution blocks and an identity mapping. Each convolution block includes a BN layer, a ReLU activation layer, and a convolutional layer. The identity mapping connects the input and output of the unit.

The encoding path has three residual units. In each unit, instead of using a pooling operation to downsample the feature map size, a stride of 2 is applied to the first convolution block to reduce the feature map by half. Correspondingly, the decoding path composes of three residual units, too. Before each unit, there is an up-sampling of feature maps from lower level and a concatenation with the feature maps from the corresponding encoding path.

In order to adapt the architecture to the segmentation problem at hand, we modified it to be 3D and added an extra layer to each of the encode and decode sections, as this allowed for comparison with other models being used in the project.

2.3 Segmentation Metrics

We used four different metrics to evaluate our models, all of which are discussed in detail in the following:

2.3.1 Accuracy. **(Shayan)** The simplest of all is the accuracy metric, which is simply calculated by dividing the number of all correct predictions by the number of all predictions. i.e. the sum of true positives and true negatives divided by the sum of all true positives, true negatives, false positives, and false negatives.

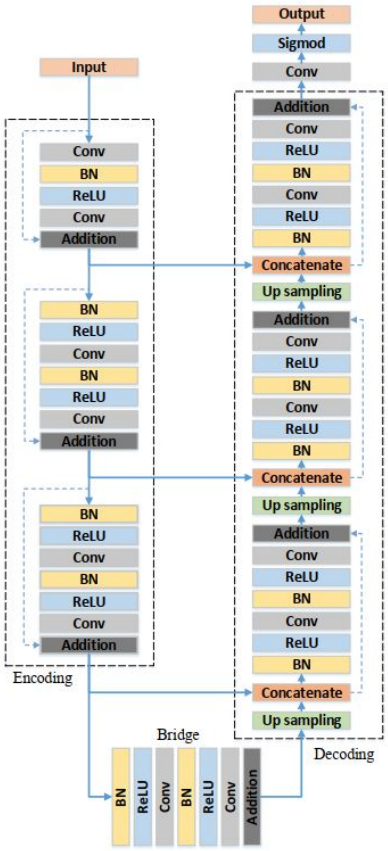


Fig. 5. The architecture of the proposed deep Residual UNets [16].

2.3.2 IoU. **(Shayan)** IoU or Intersection over Union is a relatively common metric for segmentation tasks and the most common for detection tasks. It is simply defined as the area of intersection between the prediction and ground truth over their union. It is less common for segmentation tasks as it is more important to achieve a high recall and a high precision on test data, and the Dice Coefficient does this better.

2.3.3 Dice Coefficient. aka F1 **(Shayan)** Dice Coefficient is closely related and positively correlated with IoU. It is simply defined as twice the sum of all predicted and all ground truth pixels/voxels divided by the sum of their union and intersection. Interestingly if we write the F1-score, which is the harmonic mean of precision and recall, using true positives, false positives, true negatives, and false negatives, we reach the formula for Dice Coefficient. This is the sole reason Dice is more prevalent and common for segmentation tasks.

2.3.4 AUC-ROC. **(Shayan)** AUC-ROC stands for the area under the receiver operating curve (ROC), which is well known yet it is not very common for segmentation and detection tasks as it requires keeping the results of all massive predictions for calculations, and statisticians argue that AUC-ROC only gives good utility for single-class classification

problems, hence we report the AUC-ROC for each segmentation mask, and not on all of them. AUC-ROC provides an aggregate measure of performance across all possible classification thresholds. The ROC curve is created by plotting the true positive rate (aka recall) against the false positive rate (aka 1-specificity) at various threshold settings.

2.4 Losses

We used a pure and a hybrid loss function for training these models, namely Dice, and BCE-Dice losses.

2.4.1 Dice Loss. **(Shayan)** Inspired by Dice Coefficient, Dice loss can be created by multiplying the Dice Coefficient by -1 and then optionally adding 1 to it, so that its value is between 0 and 1. Hence, the first loss function we used is simply defined as 1 - Dice Coefficient. It is known that the choice of Dice Loss is very tricky for training, and the model sometimes does not converge at all. But, it is usually preferred where there is a huge class imbalance.

2.4.2 BCE-Dice Loss. **(Shayan)** Binary Cross-Entropy is a natural choice for many segmentation tasks, yet it performs poorly where there is a huge class imbalance, such as medical image segmentation. Hence, we added BCE and Dice Loss together to see if models trained using this loss could beat the models trained with the pure dice loss.

2.5 Overall Survival Days Prediction

(Reza) In the second part of the project, we used the predicted tumor images for overall survival days prediction. For this purpose, We utilized a total of six features, which consisted of five ratios computed from predicted images and the age information of the subjects. The first three ratios were the volume of each tumor sub-region (NCR/NET, ED, and ET) to the size of the whole brain. The fourth and fifth ones were the ratio of the tumor core (including NCR/NET and ET) and the whole tumor (the combination of TC and ED) to the size of the whole brain, respectively. The combination of these six features has been demonstrated with better performance than other mixtures. To predict overall survival days, we utilized a fully connected neural network with two hidden layers, each containing 128 filters. We further improved the network's performance by implementing dropout and batch normalization techniques.

To predict the overall survival days, we only included patients who had undergone surgery with gross total resection (GTR) status. We used a three-category classification system to evaluate the model's performance. We defined survival days less than 300 as short survival, survival days between 300 and 450 as mid-survival, and survival days over 450 as long survival. We assessed the evaluation of the predictions using several metrics, including accuracy, precision, recall, F1 score, and the area under the receiver operating curve.

3 MAIN CHALLENGES

In this section, we discuss the challenges we faced in this project.

3.1 Dataset Pre-processing

(Shayan) There were 4 different images that we could have used as the input to the models. I decided to not use the T1 images to reduce the size of the input and speed up the training process. This was due to the fact that T1CE was the high-contrast image of T1, so using them wouldn't have made much of a difference. To reduce the size of the images, I observed various images and decided to cut the images as the majority of the voxels were blank. This helped reduce the size of the images and masks from 240x240x155 to 128x128x128. This was still too big to make it possible to train the models in batches of at least 4. Consequently, I decided to drop every other row in every axes to reduce the size of the images down to 64x64x64. Because of the final reduction in size, I decided to upsample the output of the networks

using trilinear interpolation for validation, so that we could get more accurate and closer to reality results. Interestingly calculating the metrics on the upsampled output and without it did not make too much of a difference. This is due to the fact that in segmentation tasks, any voxel is likely to be similar to the voxels around it or the average of the voxels around it, and trilinear interpolation calculates the new voxel's value based on the voxels around it.

3.2 Implementation

This section talks about the challenges we faced, and how we overcame them.

3.2.1 The Upsample for Vanilla 3DUNet and Attention 3DUNet. **(Shayan)** Initially I implemented the models with convolution transpose, which makes the whole model very symmetric and is the more common in UNet papers. To make things interesting, I decided to implement the models with Upsample to see if it performs just as well, yet Upsample simply uses simple interpolation techniques such as trilinear which I used; meaning that there are no filters similar to transpose convolution so we can get the desired number of channels. Two ideas came to me: 1) add a 1*1*1 convolution after each upsample with the desired number of filters to make the number of decoder channels similar to the vanilla model, or 2) simply concatenate the bigger output, which is possible as concatenative skip connections are being used. I chose the second option as the first option would have totally changed the architecture of UNet, in a way that it had a lot more layers, while the second option would have just changed the number of channels after every concatenation with the skip connection. Another reason I chose the second option was that I did not find any source to implement a UNet with Upsample that way, and still call it UNet.

As I leveraged object-oriented programming, the upsample problem simply was solved for the attention model, but the attention module required a bit of tweaking in terms of the number of parameters. To be specific, the attention module now was getting upsampled data that was supposed to be the same as the skip connection, hence the only thing I needed to handle was the number of channels being fed to the attention blocks, and multiply them by 2.

3.2.2 Identity Mapping Section of Residual 3DUNet. **(Reza)** The identity mapping section of Residual 3DUNet connects the input and output of each unit. As we mentioned before, instead of using a pooling operation to downsample the feature map size for each residual unit, a stride of 2 is applied to the first convolution block [16]. Because of that, it is not feasible to directly connect the input and output of each residual unit. After conducting an online search, we discovered that a possible solution to this issue involves implementing a 1×1 convolution layer with a stride of 1 and zero padding.

3.2.3 Unavailability of dataset. As we stated in the dataset section since the test data was not publicly available and the validation set did not have labels, we utilized the training set to form new training and validation sets, allocating 60% of the data for training and 40% for validation.

3.2.4 New to Pytorch and Colab. **(Reza)** After not having used Python for several years, taking your course motivated me to start using Python again and to familiarize myself with Colab. Although it was initially challenging for me, I managed to adapt. Assignments 3 and 4 were the first instances where I used PyTorch and implemented a deep learning model on my own, and through devoting more time to learning PyTorch, I was able to figure it out. Nonetheless, implementing this project was also challenging, and Shayan really helped in guiding me through the implementation of my part of the project.

3.3 Training

(Shayan) Vanilla 3DUNet could be trained on RTX 3070 Ti laptop GPU, yet all the other models including 3DUNet with Upsample was too big to be trained on the GPU in the same number of batches, Hence all of the models were moved to google collab to be trained.

Each model took about 4.5 hours to get trained on google collab, and as we were using the free plan, we had to create multiple gmails, to train all the models in time.

3.4 The figures

(Shayan) As the attention UNet that I implemented was new to some extent, I had to draw an intuitive visualization of it to include in the report. I tried various tools and finally realized that the best and fastest tool to create simple diagrams for neural networks is draw.io. I tried some other tools such as neutron, and less well-known tools that generate the architecture based on the final .pt file that contains the model, but their outputs were either too simple or contained too much unnecessary information. The 3DUNet 1 and Attention UNet 3 figures are drawn from scratch.

4 RESULTS AND DISCUSSIONS

In this section, we show the final results of the models in terms of metrics and discuss them.

4.1 Segmentation

For the segmentation results, we first take a look at the training loss history and then see how each of the models performs using different metrics, and see if we can make any sense of it.

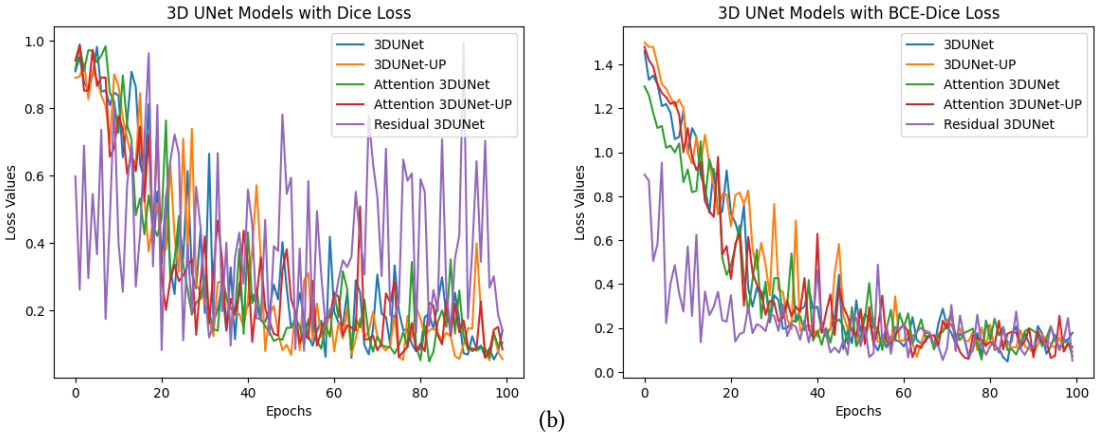


Fig. 6. Training Loss Changes Over Epochs (a) Models Trained with Dice Loss (b) Models Trained with BCE-Dice Loss

4.1.1 Training Loss. **(Shayan)** The changes in the loss value of the models as seen in figure 6 oscillates too much, meaning that a slightly smaller learning rate might have been more useful for training, and achieving better results. Furthermore, these oscillations are better controlled when the bce-dice loss is used, meaning that bce-dice loss is more stable and guides the model to convergence better. Moreover, if we take a closer look at the fluctuations between different models in both of the losses, we see that the upsampler version of every model fluctuates slightly more than

its counterpart with the convolution transpose modules, this can simply be due to the fact that the upsampler versions have more parameters, so when an update is applied to the weights, the change in the loss value is more pronounced.

(Reza) The figure 6 indicates that the residual 3DUNet initially exhibits a smaller loss compared to the other models. However, towards the end of the training, the loss values of the other models become slightly lower. This suggests that while the residual 3DUNet shows promising early convergence, the other models gradually catch up and achieve comparable or slightly better performance in terms of minimizing the loss function. In terms of training models with dice loss, the figure reveals that the residual 3DUNet exhibits fluctuations in its loss values, indicating instability and lack of consistent learning. This discrepancy suggests that the residual 3DUNet trained with Dice loss may face challenges in effectively optimizing the model parameters and achieving stable learning patterns compared to the other models, yet we did not train the model with different learning rates, so they might have been able to converge better if a lower learning rate were used for these two models, especially when trained with the pure dice loss.

Model	Accuracy (%) TC | ET | WT	Avg. IoU (%) TC | ET | WT	Dice/F-1 (%) TC | ET | WT	AUC-ROC(%) TC | ET | WT
3DUNet DICE	99.28|99.68|99.05	65.35|55.46|76.78	75.11|67.10|85.31	88 | 96 | 98
3DUNet BCE-DICE	99.36|99.70|99.11	68.85|58.63|78.09	78.43|69.83|86.20	97 | 98 | 98
3DUNet-U DICE	99.37|99.69|99.06	69.11|57.77|76.46	78.62|68.89|84.80	92 | 96 | 96
3DUNet-U BCE-DICE	99.33|99.69|99.09	67.87|57.71|77.47	77.87|68.91|85.76	97 | 98 | 98
Attention3DUNet DICE	99.36|99.69|99.08	69.10|58.25|77.52	78.53|69.32|85.83	94 | 98 | 96
Attention3DUNet BCE-DICE	99.31|99.69|99.07	67.01|57.83|77.18	76.67|69.08|85.30	99 | 99 | 99
Attention3DUNet-U DICE	99.34|99.69|99.08	68.66|56.70|77.08	78.50|67.93|85.44	93 | 95 | 96
Attention3DUNet-U BCE-DICE	99.34|99.68|99.06	68.62|56.59|76.90	78.31|67.69|85.26	96 | 97 | 98
Residual3DUNet DICE	98.06|99.14|97.43	0.00 | 0.00 | 44.53	0.00 | 0.00 | 59.61	29 | 24 | 71
Residual3DUNet BCE-DICE	99.29|99.67|99.01	64.82|55.84|75.22	75.36|67.37|84.06	98 | 99 | 99

Table 1. All Segmentation Models Evaluation Results

4.1.2 Vanilla and Attention 3DUNet Results. **(Shayan)** It is well-known that IoU and Dice are positively correlated, and if between two models one has a higher dice score, it will have a higher IoU score, and vice-versa. Hence, there's no point in looking at both the IoU and Dice scores, yet we did put it in the report as it was a requirement for projects that decided to do binary segmentation. From the table 3 it can be seen that AUC-ROC for Attention3DUNet with BCE-Dice loss is higher than all the other models, while its dice and IoU even though are high and better than average, but they are not the best. This is due to the fact that AUC-ROC uses true negatives, while the others do not use true negatives in their formula. But, what this means is that the introduction of an attention mechanism does not have much effect on detecting the positive class, but it certainly has a good effect on detecting the negative class, and not misclassifying the uninteresting voxels, which is very important in medical imaging. Nonetheless, it is hard to say which model is best for different labels as different models are best in terms of dice/IoU score, and another model in terms of AUC-ROC, but the attention 3D UNet with BCE-dice loss might be preferred for real-world use-cases as it will perform better in all thresholds, as its AUC-ROC suggests.

Moreover, it is immediately realized that all of the models that used BCE-Dice loss instead of the pure Dice loss perform better in terms of AUC-ROC, meaning that BCE-Dice helps the model differentiate better between the negative and positive classes.

Finally, with and without using upsample instead of convolution transpose there's not much of a consistent difference, as it had a positive effect on 3DUNets with Dice loss and a negative on all other models. Hence, it is better to use convolution transpose as it prevents later layers to have bigger inputs, and therefore more parameters in the model.

Results:

- *Among all models, Attention3DUNet BCE-DICE beats other models in terms of AUC-ROC, with close values of Dice scores to the best ones*
- *The attention mechanism helps the model to look at the interesting part, and hence not mislabel the negative classes*
- *BCE-Dice loss is more stable and helps the models achieve a higher AUC-ROC*
- *Using Upsample instead of convolution transpose does not make much of an impact, thus it is better not to use them as they lead to more parameters even though they themselves do not have any learnable parameters, compared to convolution transpose*

4.1.3 Residual 3DUNet Results. **(Reza)** The comparison between the two models, Residual 3DUNet trained with Dice loss and Residual 3DUNet trained with BCE-Dice loss, reveals clear distinctions in their performance. The model trained with BCE-Dice loss exhibits superior results across multiple evaluation metrics, underscoring its effectiveness in predicting survival outcomes and accurately segmenting tumor sub-regions.

In terms of accuracy, the BCE-Dice model outperforms the Dice model with significantly higher scores for all three tumor sub-regions. The accuracy rates of 99.29%, 99.67%, and 99.01% for the TC, ET, and WT groups respectively demonstrate the model's ability to classify and predict the presence of tumor regions more accurately.

The BCE-Dice model outperforms the Dice model in both IOU and Dice scores, indicating superior segmentation accuracy and similarity to ground truth tumor regions. With IOU scores of 64.82, 55.84, and 75.22 for the TC, ET, and WT groups, and Dice scores of 75.36, 67.37, and 84.06 for the same groups, the BCE-Dice model showcases its ability to accurately capture tumor boundaries and intricate details.

Additionally, the AUC-ROC values further support the superior performance of the BCE-Dice model. The AUC-ROC is a measure of the model's ability to distinguish between positive and negative cases, with higher values indicating better discrimination. With AUC-ROC values of 98, 99, and 99 for the TC, ET, and WT groups respectively, the BCE-Dice model showcases its effectiveness in accurately predicting survival outcomes.

These comprehensive findings suggest that the incorporation of BCE-Dice loss in the training process significantly enhances the Residual 3DUNet model's performance.

4.2 Prediction

(Reza) To predict the overall survival days, we only included patients who had GTR status. This resulted in a training set of 71 subjects, while the validation set consisted of 46 subjects. Our model was trained using the Adam optimizer and cross-entropy loss function, with a batch size of 8 and an initial learning rate of 1e-4. We chose 500 epochs as the number of training iterations in our configuration. It is important to note that during the training phase, we utilized the actual masks provided in the dataset to train our prediction model. However, for the validation phase, specifically in calculating the volumes of various tumor subregions, we employed the predicted mask generated by our best-performing segmentation model, which was the attention 3DUNet (this model demonstrated superior performance compared to the other models).

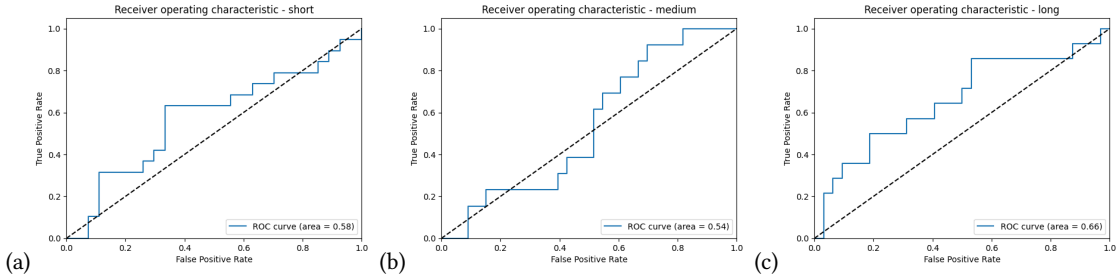
The results of our prediction are presented in Table 2, where we report the accuracy, precision, recall, F1-score, and AUC-ROC for the overall survival days prediction task. To calculate these measures, as mentioned earlier, we classified

survival days into three categories: short-survival, mid-survival, and long-survival. Based on the results presented in the table, it can be observed that the model's overall accuracy was moderate, with a higher precision value for the mid-survival category and higher recall values for the long-survival category. The F1 score also showed higher values for the mid-survival category, indicating a better balance between precision and recall. The AUC-ROC values of 0.58, 0.54, and 0.66 for the short, mid, and long survival categories, respectively, suggest that the model's performance in predicting long-term survival is better than its performance in predicting short and mid-term survival. However, the overall AUC-ROC values for all categories are relatively low, indicating that the model's predictive power could be improved.

Prediction Results					
Categories	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC-ROC (%)
short-survival	56.5	74.3	70.3	72.2	0.58
mid-survival	60.9	77.8	73.7	75.7	0.54
long-survival	58.7	61.4	93.1	74.0	0.66

Table 2. The results of our prediction model.

After calculating the AUC-ROC, we visualized the ROC curves for different categories and included them in figure 7. Additionally, we monitored the loss over the epochs and created a figure to visualize the training process of the model. Figure 8 shows how the model's performance changed over time and how it adapted to the data.



(a) (b) (c)

Fig. 7. The ROC curve and AUC-ROC for (a) short-survival, (b) mid-survival, and (c) long-survival categories.

5 CONCLUSION AND FUTURE WORK

(Shayan) For the first phase of the project, we developed 5 models each trained with two different loss functions. We realized that the residual 3DUNet does not perform well, and might not even train, and that Attention Mechanism helps the model concentrate on the difference between the positive and negative classes. We also realized that there's not much of a difference between upsample and convolution transpose, yet in our UNets upsample led to more learnable parameters because of the channel preservation effect and thus we concluded that for UNets it is better to use convolution transpose.

As a next step, we can implement the attention mechanism similar to the original paper to see if it performs better, plus we can try and compare other types of UNet such as UNet++, Fractal UNets, etc.

(Reza) Also, it will be worth the time to train the residual 3DUNets with smaller learning rates to make sure that the learning rate was the issue and not something else, especially for the one trained with pure dice loss.

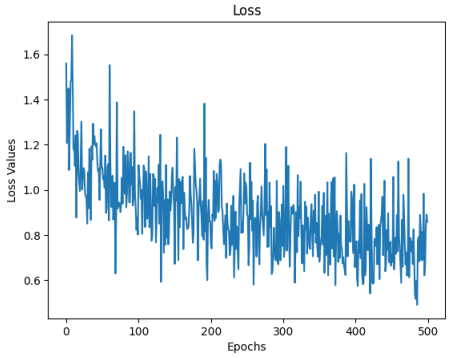


Fig. 8. The loss over the epochs

(Reza) For the second part of this project, we developed a model that utilizes the volumes of different tumor sub-regions, as well as age information, to predict the survival days of patients who underwent surgery for gross total resection. As discussed earlier, the dataset utilized in our study consisted of fewer subjects compared to the original BraTS2020 dataset, and we also used a smaller training dataset in the prediction part of the project in comparison to other studies. To further enhance the results of our predictions, future studies could focus on expanding the dataset with more subjects, exploring different types of models for the prediction part, and incorporating additional clinical features such as molecular and genetic types.

REFERENCES

[1] Zeynettin Akkus, Alfiia Galimzianova, Assaf Hoogi, D. Rubin, and Bradley James Erickson. 2017. Deep Learning for Brain MRI Segmentation: State of the Art and Future Directions. *Journal of Digital Imaging* 30 (2017), 449 – 459.

[2] Michela Antonelli, Annika Reinke, Spyridon Bakas, Keyvan Farahani, AnnetteKopp-Schneider, Bennett A. Landman, Geert J. S. Litjens, Bjoern H Menze, Olaf Ronneberger, Ronald M.Summers, Bram van Ginneken, Michel Bilello, Patrick Bilic, Patrick Ferdinand Christ, Richard K. G. Do, Marc J. Gollub, Stephan Heckers, Henkjan J. Huisman, William R. Jarnagin, Maureen McHugo, Sandy Napel, Jennifer S. Goli Pernicka, Kawal S. Rhode, Catalina Tobon-Gomez, Eugene Vorontsov, James Alastair Meakin, Sébastien Ourselin, Manuel Wiesenfarth, Pablo Arbeláez, Byeonguk Bae, Sihong Chen, Laura Alexandra Daza, Jian-Jun Feng, Baochun He, Fabian Isensee, Yuanfeng Ji, Fucang Jia, Namkug Kim, Ildoo Kim, Dorit Merhof, Akshay Pai, Beomhee Park, Mathias Perslev, Ramin Rezaiifar, Oliver Rippel, Ignacio Sarasua, Wei Shen, Jaemin Son, Christian Wachinger, Liansheng Wang, Yan Wang, Yingda Xia, Daguang Xu, Zhanwei Xu, Yefeng Zheng, Amber L. Simpson, Lena Maier-Hein, and Manuel Jorge Cardoso. 2021. The Medical Segmentation Decathlon. *Nature Communications* 13 (2021).

[3] Milica M. Bada and Marko Barjaktarović. 2020. Classification of Brain Tumors from MRI Images Using a Convolutional Neural Network. *Applied Sciences* (2020).

[4] Akbari H. Sotiras A. et al. Bakas, S. 2017. Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features. *Scientific data 4.1, 170117* (2017).

[5] Spyridon Bakas, Mauricio Reyes, András Jakab, Stefan Bauer, Markus Rempfler, Alessandro Crimi, Russell Takeshi Shinohara, Christoph Berger, Sung Min Ha, Martin Rozycki, Marcel Prastawa, Esther Alberts, Jana Lipková, John B. Freymann, Justin S. Kirby, Michel Bilello, Hassan M. Fathallah-Shaykh, Roland Wiest, Jan Stefan Kirschke, Benedikt Wiestler, Rivka R. Colen, Aikaterini Kotrotsou, Pamela J. LaMontagne, Daniel S. Marcus, Mikhail Milchenko, Arash Nazeri, Marc-André Weber, Abhishek Mahajan, Ujjwal Baid, Dongjin Kwon, Manu Agarwal, Mahbubul Alam, Alberto Albiol, Antonio Albiol, Alex Varghese, Tran Anh Tuan, Tal Arbel, Aaron Avery, B. Pranjal, Subhashis Banerjee, Thomas Batchelder, Nematollah K. Batmanghelich, Enzo Battistella, Martin Bendszus, Eze Benson, José Bernal, George Biros, Mariano Cabezas, Siddhartha Chandra, Yi-Ju Chang, and et al. 2018. Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge. *ArXiv* abs/1811.02629 (2018).

[6] Angona Biswas and Md. Saiful Islam. 2021. Brain Tumor Types Classification using K-means Clustering and ANN Approach. In *2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*. 654–658. https://doi.org/10.1109/ICREST51555.2021.9331115

[7] Xue Feng, Nicholas J Tustison, Sohil H Patel, and Craig H Meyer. 2020. Brain tumor segmentation using an ensemble of 3d u-nets and overall survival prediction using radiomic features. *Frontiers in computational neuroscience* 14 (2020), 25.

[8] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron C. Courville, Yoshua Bengio, Christopher Joseph Pal, Pierre-Marc Jodoin, and H. Larochelle. 2015. Brain tumor segmentation with Deep Neural Networks. *Medical Image Analysis* 35 (2015), 18–31.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[10] Sajid Iqbal, Muhammad Usman Ghani, Tanzila Saba, and Amjad Rehman. 2018. Brain tumor segmentation in multi-spectral MRI using convolutional neural networks (CNN). *Microscopy Research and Technique* 81 (2018), 419 – 427.

[11] Bjoern H Menze, András Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin S. Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, Levente Lanczi, Elizabeth R. Gerstner, Marc-André Weber, Tal Arbel, Brian B. Avants, Nicholas Ayache, Patricia Buendia, D. Louis Collins, Nicolas Cordier, Jason J. Corso, Antonio Criminisi, Tilak Das, Hervé Delingette, Çagatay Demiralp, Christopher R. Durst, Michel Dojat, Senan Doyle, Joana Festa, Florence Forbes, Ezequiel Geremia, Ben Glocker, Polina Golland, Xiaotao Guo, Andaç Hamamci, Khan M. Iftekharuddin, Rajesh Jena, Nigel M. John, Ender Konukoglu, Danial Lashkari, José Antonio Mariz, Raphael Meier, Sérgio Pereira, Doina Precup, Stephen John Price, Tammy Riklin-Raviv, Syed M. S. Reza, Michael T. Ryan, Duygu Sarikaya, Lawrence H. Schwartz, Hoo-Chang Shin, Jamie Shotton, Carlos Alberto Silva, Nuno Sousa, Nagesh K. Subbanna, Gábor Székely, Thomas J. Taylor, Owen M. Thomas, N. Tustison, Gözde B. Ünal, Flor Vasseur, Max Wintermark, Dong Hye Ye, Liang Zhao, Binsheng Zhao, Darko Zikic, Marcel Prastawa, Mauricio Reyes, and Koenraad Van Leemput. 2015. The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE Transactions on Medical Imaging* 34 (2015), 1993–2024.

[12] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. 2018. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999* (2018).

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (Eds.). Springer International Publishing, Cham, 234–241.

[14] Amber L. Simpson, Michela Antonelli, Spyridon Bakas, Michel Bilello, Keyvan Farahani, Bram van Ginneken, Annette Kopp-Schneider, Bennett A. Landman, Geert J. S. Litjens, Bjoern H Menze, Olaf Ronneberger, Ronald M. Summers, Patrick Bilic, Patrick Ferdinand Christ, Richard Kinh Gian Do, Marc J. Gollub, Jennifer Golia-Pernicka, Stephan Heckers, William R. Jarnagin, Maureen McHugo, Sandy Napel, Eugene Vorontsov, Lena Maier-Hein, and M. Jorge Cardoso. 2019. A large annotated medical image dataset for the development and evaluation of segmentation algorithms. *ArXiv* abs/1902.09063 (2019).

[15] Feifan Wang, Runzhou Jiang, Liqin Zheng, Chun Meng, and Bharat Biswal. 2020. 3d u-net based brain tumor segmentation and survival days prediction. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 5th International Workshop, BrainLes 2019, Held in Conjunction with MICCAI 2019, Shenzhen, China, October 17, 2019, Revised Selected Papers, Part I*. Springer, 131–141.

[16] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. 2018. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters* 15, 5 (2018), 749–753.

[17] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 2016. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*.

A MODELS' PARAMETERS

Model	Number of Trainable Parameters
3DUNet	90,302,147
3DUNet-U	94,130,947
Attention3DUNet	91,005,199
Attention3DUNet-U	95,182,159
Residual3DUNet	95,882,563

Table 3. Number of Trainable Parameters in Each Model

B VISUALIZATIONS

As the report has already surpassed the maximum number of pages required, and the visualizations take up a lot of space, and we have shown them in the presentation/talk video, we did not put any of those pictures here. However, they can be seen and accessed alongside the models' ROC curves, which their AUC was reported in this report, here in the repository.

Received 11 May 2023