Multimodal Brain Tumor Segmentation with Fully Convolutional Neural Networks

Xiaofeng Shi[a], Zilin Xu[a], Ziyun Du[a]

[a]*The Department of Mathematics and Computer Science, Emory University, GA*

Abstract

Tumor Segmentation, which is an essential procedure for brain cancer treatment, means identifying whether the pixels in MRI scans are located in the tumor regions or the background regions. In this project, we propose a novel Fully Convolutional Network (FCN) model, the patch-based 3D FCN, to automatically segment tumors from 4-modal 3D MRI scans. In our framework, we first downsample the 3D patches from original scans and then train an end-to-end 3D FCN to learn the pixel-wise classification of the small patch. To overcome the class-imbalance problem of the MRI data, our model is trained on a class-balanced patch dataset and fine-tuned on a class-imbalanced patch set, which has the similar data distribution as the testing data. Experiments on the BraTS 2015 dataset reveal that our model can produce competitive segmentation results on most patient cases compared to the state of the art, but fail to perform well on the rest several cases.

Keywords: tumor segmentation, fully convolutional nerual network, deep learning.

1. Introduction

In 2017, nearly 80,000 new cases of primary brain tumors are expected to be diagnosed and approximately one-third of these tumors are malignant [1]. According to Central Brain Tumor Registry of the United States [2], with proper treatment, 5-Year Relative Survival Rate is more than 50% for most kinds of brain tumor. The first stage of the therapy process is to localize the precise location of tumor and figure out its

Email addresses: xiaofeng.shi@emory.edu (Xiaofeng Shi), zilin.xu@emory.edu (Zilin Xu), ziyun.du@emory.edu (Ziyun Du)

shape for diagnose. However, even with the (Magnetic Resonance Imaging) MRI 3D scans, it is still expensive and challenging to obtain the pixel-level identification of the irregularly shaped tumor regions. Thus, in our project, we propose to automatically segment the tumors from MRI scans with deep learning methods.

Tumor segmentation is actually a semantic segmentation task in Computer Vision. Basically, the main goal of semantic segmentation is to classify the texture of each pixel in the image [3]. Specifically, in the tumor segmentation task, each pixel of the 3D scan of the brain is required to be classified as foreground pixel (tumor pixel) or background pixel (non-tumor pixel).

The traditional way to address the tumor segmentation problem is using the artificial feature extractors to extract the pixel-level features and training classifiers to learn these features [4]. Basically, a number of artificial filters are designed to learn the spatial information from a selected window. The outputs of the filters are assigned to the center point of the window as the feature vector of that pixel. Then the extractor window is slid pixel by pixel to produce the feature representation for each pixel of the image. After that, a classifier is trained on the pixel samples to learn the pixel features and predict the categories for all the pixels in testing data. For the tumor segmentation task, the commonly used artificial features include Harr-like features [5], HOG features [6] [7] and LBP features [8], while Random forest [9] and support vector machines [10] are the most frequently-used machine learning pipelines to classify the pixels. These traditional segmentation methods can perform well with only a small training dataset; however, they are rather expensive for prediction. In addition, the segmentation performance of these methods highly relies on the design of feature filters, which requires much expertise in MRI images and Computer Vision.

In recent years, the parameters of the filters for image feature extraction can be automatically learned on big data with the Convolutional Neural Networks (CNNs) [11]. Therefore, instead of the manually designed the filters, the patch-based CNN models can be used to extract the pixel-level features (or labels) for tumor segmentation [12]. Similar to the traditional methods, the output of the CNN is assigned to the center pixel of the patch. However, this method is still inefficient at the prediction stage, since the CNN patch filter is required to be slid pixel by pixel to produce the entire segmenta-

tion results. To solve this problem, the Fully Convolutional Network [13] [14] was proposed to learn the end to end mapping of the image. In this network, the output image size is exactly the same as the original input. Hence, the FCN framework is a good solution for semantic segmentation. Most recently, the FCN models have been successfully applied to the tumor segmentation challenges [15] and produced state of the art segmentation results with respect to both accuracy and efficiency.

Inspired by the FCN deep learning framework, we propose to implement our own FCN model to solve the tumor segmentation problem in our project. Specifically, we design and implement a novel FCN architecture, the patch-based 3D FCN, to identify the tumor regions from 4-modal 3D MRI brain scans. In our framework, we propose to downsample the patches to construct a class balanced dataset and train the parameters of the neural network on this dataset. In addition, we also downsample a class-imbalanced patch set with uniform sampling and fine-tune our neural network on this dataset to improve its performance on the imbalanced real data. Experiments show that our model can produce highly competitive segmentation results on most of the testing data, while it fails to perform well on the rest. In the discussion section, we will analysis the reasons for the failure cases and propose possible solutions to further enhance the performance.

2. Proposed Method

2.1. FCN frameworks for Tumor Segmentation

In our project, we consider three FCN frameworks to address tumor segmentation. Inspired by the full-size end-to-end network architecture for 2D image segmentation [13], we first consider training a full-size 3D FCN, which is fed with the entire 4-modal 3D MRI scans with the size of $155 \times 240 \times 240$ (It represents the shape of MRI images in the BraTs 2015 dataset. See the Experiments section for more details.) at the input layer and outputs the 5 one-hot predictions with the same image size at the output layer. The outline of this 3D End-to-End FCN model is shown in Figure 1.

In this architecture, however, the network needs to be very deep to downsample the image and restore it to the original size. Thus, the computational complexity would be

Figure 1: The framework of full-size end-to-end 3D FCN. Input: the entire $155 \times 240 \times 240$ 3D 4-modal MRI scans. Output: 5 one-hot $155 \times 240 \times 240$ predictions.

much higher than that our machine can afford. In addition, we did not have sufficient well-annotated training samples to train this deep neural network. Therefore, we did not adopt this framework in our project.

Our second proposal is to train a 2D full-size end-to-end FCN to segment the vertical 2D slices of the original 3D scans, namely, the input of the network are the 4-modal 2D MRI slices with the size of 240×240 and the output are the 5 one-hot 2D predicted feature maps with the same size (shown in Figure 2). Since the convolutional kernels are now 2D kernels, the computational complexity is much smaller than the previous model. In addition, now we can extract more than 155 2D slices (with image augmentation) for each patient. Thus, the annotated samples are sufficient for training. We implemented and trained this 2D framework, however, the performance was limited. The training Dice score [4] for each batch with the size of 80 is barely around 0.4. The poor performance is mainly caused by the class imbalance problem of the training data. Only a small proportion of the vertical slices actually contain the tumors, while the tumor sizes are relatively small compared to the 240×240 image. Additionally, through cutting the 3D image into 2D slices, this method naturally damages the horizontal spatial information of the data.

Hence, we propose to sample 3D patches from the original scans to construct a balanced data set and train a patch-based 3D FCN to learn this patches, as shown in Figure 3. The idea of this learning model is similar to a combination of the pixel-wise CNN with sliding window and the End-to-End FCN. At the prediction stage, we can simply slide the patch window with the stride of the patch size and the FCN model can

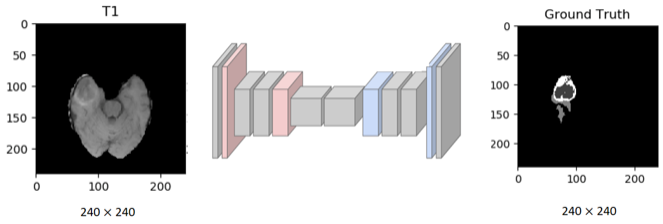

Figure 2: The framework of full-size end-to-end 2D FCN. Input: the 240×240 4-modal vertical slices of the MRI scans. Output: 5 one-hot 240×240 predictions.

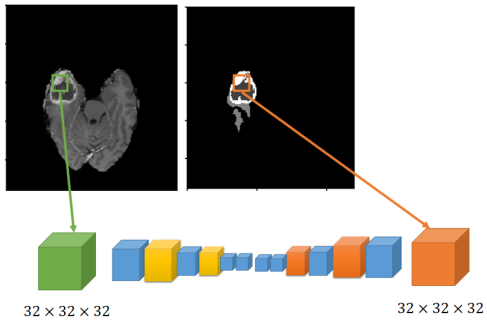

Figure 3: The framework of patch-based end-to-end 3D FCN. Input: the $32 \times 32 \times 32$ 4-modal patch of the MRI scans. Output: 5 one-hot $32 \times 32 \times 32$ predictions.

make predictions for all the pixels in the same patch simultaneously. Thus, the model is much more efficient than the pixel-wise CNN for prediction; while the network does not need to be deep to learn the small patches, which makes training executable in our machines. The next section introduces the architecture of the patch-based 3D FCN model, which is our final plan to solve the tumor segmentation challenge.

2.2. The architecture of the patch-based 3D FCN

The proposed patch-based 3D FCN model contains several levels of convolutional components (as shown in Figure 4), each of which is comprised of two convolutional layers and one Max pooling layer, and transposed convolutional components (as shown in Figure 5), each of which is comprised of one unpooling layer, one transposed con-

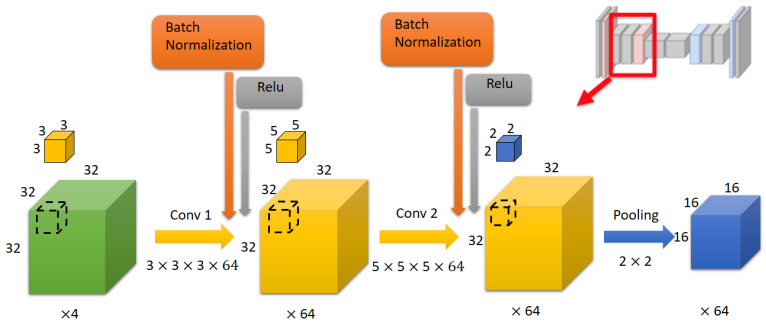

Figure 4: The architecture of the first convolutional component in the patch-based 3D FCN. This component contains two convolutional layers and one max pooling layers. Following each convolutional layer are the Batch Normalization layer and the Relu layer. The depth of the intermediate feature maps is 64 (this hyper-parameter is set according to the performance of our machines). Other convolutional components have the similar structure as this component, but the kernel sizes are different.

volutional layer and two convolutional layers. Following each convolutional layer and transposed convolutional layer are the Batch Normalization layer [16] and the Relu layer.

The entire architecture of the model is shown in Figure 6. At the input layer, the model adopts the sampled $32 \times 32 \times 32$ 4-modal patches. Through the convolutional-pooling pathway, the patch is downsampled to the $4 \times 4 \times 4$ feature maps. Then the feature maps are restored to the size of $32 \times 32 \times 32$ through the transposed convolutional components. At the output layer, 5 1×1 convolutional kernels are used to transform the feature maps so that the depth of the feature maps is 5, which represents the 5 one-hot categories. Then a softmax activation function is used to learn the probability of each category for each pixel.

However, one problem of this network architecture is that the resolution of the image is damaged by the max-pooling layers, leading to the loss of details in the restored feature maps. Therefore, we introduce several cross-layer connections in the network (shown as the red dotted arrows in Figure 6) to preserve the image resolution.

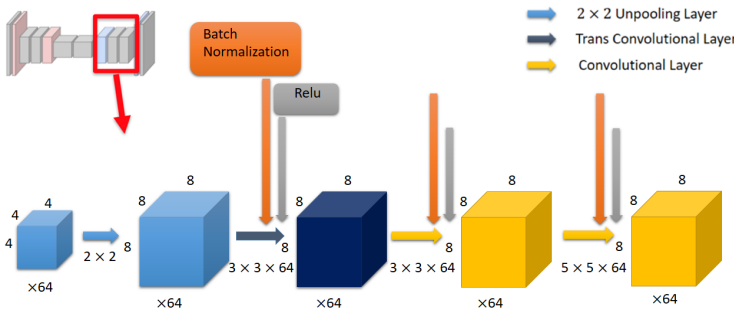

Figure 5: The architecture of the first transposed convolutional component in the patch-based 3D FCN. This component contains one unpooling layer, one transposed convolutional layer, and two convolutional layers. Following each convolutional layer and transposed convolutional layer are the Batch Normalization layer and the Relu layer. The depth of the intermediate feature maps is 64. The kernel sizes of the convolutional layers are symmetrical to the previous convolutional components.

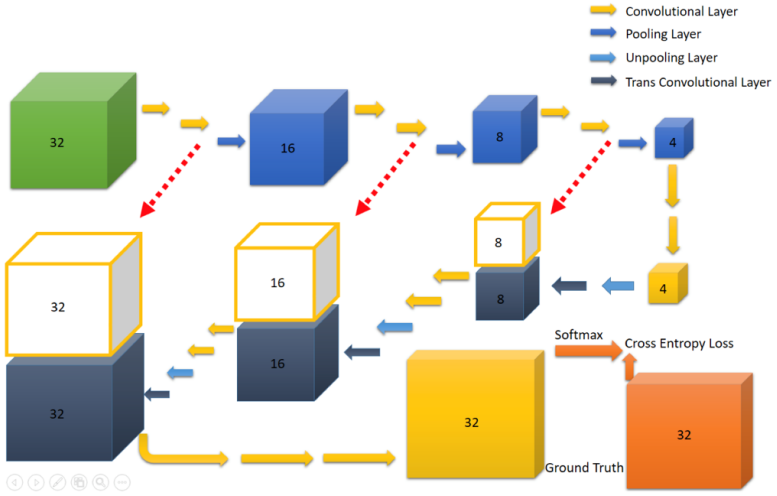

Figure 6: The overall architecture of the patch-based 3D FCN. The red dotted arrows represent the cross-layer connections, namely, we copy the feature maps in the previous layer and concatenate them with the feature maps in current layer to preserve resolution.

2.3. Loss Function

We use the pixel-wise cross entropy loss as the objective of this deep neural network. Additionally, as suggested in [3], we introduce weights for different categories to reduce the impact of imbalanced classes. Moreover, we add the L2 penalty of all the network weight parameters as a regularization. Thus, the loss function is defined as this:

$$L(X) = \sum_{c \in C}(-w_c \sum_{i,j,l \in F} logE(y_{ijk} = c; \theta)) + \lambda L2(W_{nn}) \quad (1)$$

where c represents the category, w_c represents the weight ratio, i, j, k represents the coordination of the pixel, y_{ijk} represents the binary prediction of each pixel for each category, and $L2(W_{nn})$ represents the L2 penalty.

2.4. Prediction and Bagging of models

In the prediction stage, we slide the trained patch-FCN model patch by patch (with the stride of the patch size) on the entire MRI image (with the channel depth of 4) to make a prediction for the whole image. Compared to the traditional feature extraction methods and CNN models, the computational complexity of concatenating the predictions of patch 3D FCN to a final output 3D map is 32^3 times less than sliding the filters pixel by pixel.

In this framework, the size of the input patch is a hyper-parameter. In our implementation, we defined the input size of the patch as $32 \times 32 \times 32$. A smaller patch size would make the model be more focused on the details, while a larger patch size could learn more global information but also introduce more computational complexity.

Through learning on the selected patches, the trained FCN is only sensitive to the local spatial features within the patch, such as local edges and shapes, but cannot recognize the spatial features of the entire image. In order to make the prediction results more robust, we propose to train multiple models with different network architectures and hyper-parameter settings (such as input patch size, dropout rate, depth of the feature maps) and random states (such as parameter initialization, sampling, sequence of the inputs), then bagging the multiple predictions by computing the mode class for

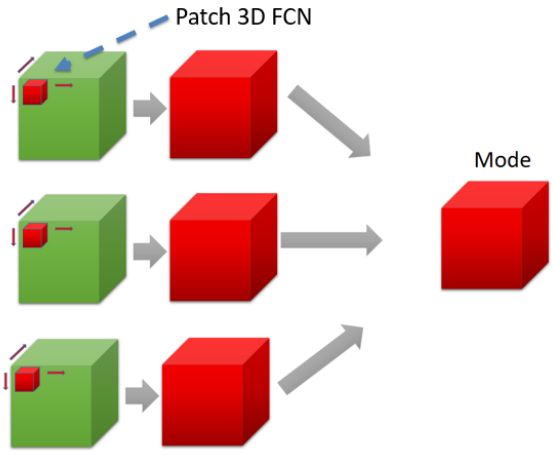

Figure 7: Slide the patch-FCN window patch by patch on the entire MRI image (with the channel depth of 4) to make prediction for the whole image. Then integrate the predictions of multiple FCN models with Bagging method to smooth the output.

each pixel. The bagging strategy can be considered as a smoothing method across parallelized predictions for each pixel. Thus the variance of the model can be reduced. Constrained by the computational resources, we only trained 3 variants of the proposed $32 \times 32 \times 32$ model with random parameter initializations, input sequences and sampling strategies.

3. Experiments

3.1. Dataset

In our project, we used the Brain Tumor Segmentation Challenge (BraTS) 2015 dataset [4] for training and evaluation. This dataset includes 220 High-Grade Glioma (HGG) patients and 54 Low-Grade Glioma (LGG) patients in total. For each patient, the dataset provides the 4-modal (T1, T1-contrast, T2, and FLAIR) 3D MRI scans with the size of $155 \times 240 \times 240$ and the corresponding segmentation ground truth. In the ground truth, the MRI pixels are labeled with five categories: background, necrotic core, edema, enhancing core, non-enhancing core. An example of vertical slides of multimodal brain scans for a brain and the ground truth is shown in Figure 8.

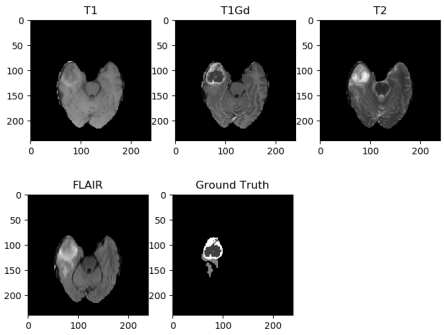

Figure 8: An example of vertical slides of multimodal brain scans for a brain and the ground truth.

In our project, we are mainly focusing on the foreground (tumor regions) and background (anything else) segmentation. Therefore, for evaluation, we sum the 4 types of tumors together as the positive class. The rest background is considered as the negative class. The performance of the segmentation results is evaluated by the Dice score matric, as shown in formula 2.

$$Dice(P,T) = \frac{P_1 \wedge T_1}{(|P_1| + |T_1|)/2} \tag{2}$$

where P_1 and P_0 represent the the subset of voxels predicted as positives and negatives respectively in model predictions, while T_1 and T_0 represent similar subsets in ground truth (as shown in Figure 9).

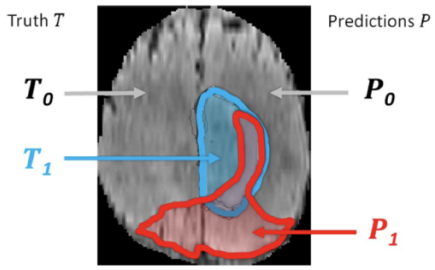

Figure 9: Evaluation of dice score [4].

3.2. Data Preprocessing

We splitted the dataset into the training set with 200 (73.0%) patient cases, and the testing set with 74(27.0%) patient cases. All the image pixels are standardized into 0 mean and 1 variance (for each scan).

3.3. Patch Sampling

The BraTs data is a naturally imbalanced data, with normally less than 1% of the pixels are non-tumor pixels. Training the neural network with uniformly sampled patch set would result in predicting everything negative. Thus, we constructed a balanced training patch set (Figure 10) by putting more emphasis on the positive pixels, namely, we sampled the patches to make the center point of the patch has 50% probability to be located in the tumor regions (If the goal is to predict the specific tumor type for the pixel, we will need to sample the data to make the training set balanced on each category). In addition, we also sampled a group of imbalanced patches for fine-tuning the network with uniform sampling.

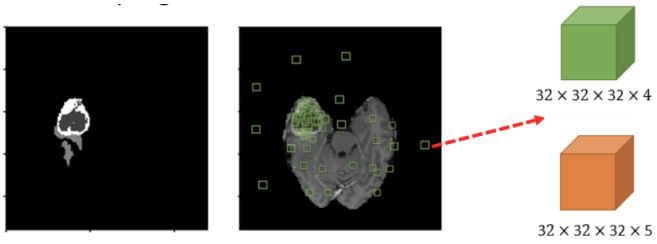

Figure 10: Sampling the patches to construct a balanced training set.

3.4. Training

Through sampling, we constructed a balanced patch set for training. However, the real pixel class distribution in prediction is still highly imbalanced. Thus, the statistical parameters of the training data, such as the empirical mean and variance of Batch Normalization, are greatly different from the testing data. In this case, the model with more than 0.9 averaged dice score on training batches would still produce invalid segmentation results on testing data. Therefore, we adopted three different steps for training the network to enable its transfer learning capacity on testing data:

1. Toggle on batch normalization, train model on balanced dataset to learn the empirical mean and variance of Batch Normalization.
2. Toggle off batch normalization, train model on balanced dataset to make the network be more focused on learning the weights with fixed Batch Normalization parameters and statistics.
3. Keep batch normalization off, fine-tune model on imbalanced dataset, which has the similar distribution as the testing data.

For other hyper-parameters, we set the dropout rate as 0.6, the parameter λ for L2 regularization as 0.0001, and the batch size as 28 (mainly considering the memory capacity of our GPU). In addition, we used the Adam optimizer [17] for Stochastic Optimization, with the initial learning rate 0.0001.

3.5. Results

3.5.1. Evaluation on Training data

Figure 11 shows the batch accuracy and dice score in the training process. A prediction is made on the training batch before feeding the batch to the network. Since the batch has never been trained before, this can be considered as the evaluation on validation data. We tuned our network based on the Dice score of the training batch. From the figure, we can see after we turned off the batch normalization at around 550th iteration, the dice score grows quickly to higher than 0.9 for the balanced training batches.

However, after we started training on the imbalanced data, although the accuracy grows higher, the dice score shakes dramatically (the 0 Dice score is caused by the 0 prediction of the positive class). This is because the distribution of data changes greatly. But this imbalanced data set is more similar to the testing set. The red line shows the state-of-the-art [15] averaged dice score for all the patients in the validation dataset, while the green line shows the best result on testing data in the BraTS 2015 benchmark [4]. These two baselines are actually not comparable with our output, because our dice in this figure is calculated on the batch data for training instead of the entire image, but they reflect a reference of performance by which we can decide whether our network has been well-trained.

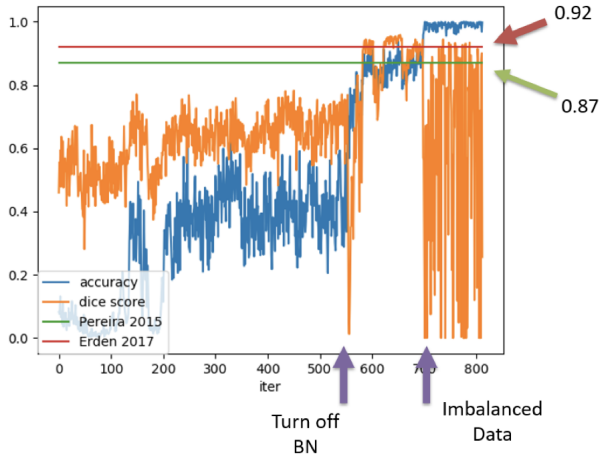

Figure 11: Dice Score and accuracy on Training batches. The horizontal axis represents the number of iterations. Before turning batch normalization on, the dice score is vibrating around 0.7; afterwards, it raises to a higher level around 0.9. However, after starting training on the imbalanced data, the dice score shakes dramatically. The red line represents state-of-the-art [15] dice score, and the green one is the highest score for BraTS competition [4].

Figure 12 shows the effect of Batch Normalization. We can see without batch normalization the dice score decreases to 0 quickly and the model predicts everything negative. In addition to accelerating the training, Batch Normalization worked as a regularization in our framework, which prevents the neural network from overfitting to the imbalanced data (even for balanced patch set, the positive and negative pixels are still highly imbalanced).

3.5.2. Evaluation on testing data

We totally trained 3 models with random initialization and input sequence. Figure 13 shows the evaluation statistics of the first trained model on the testing dataset, and Figure 14 shows the accurate dice score for each patient. The averaged dice score for the first model on testing data is 0.73. The red line in Figure 13 and the green line in Figure 14 represent the best performance in BraTS 2015 Challenge [4]. From the figures, we can see our model performs well for part of the patient cases, with the dice score higher than 0.8. However, the variance of the model is relatively high. For

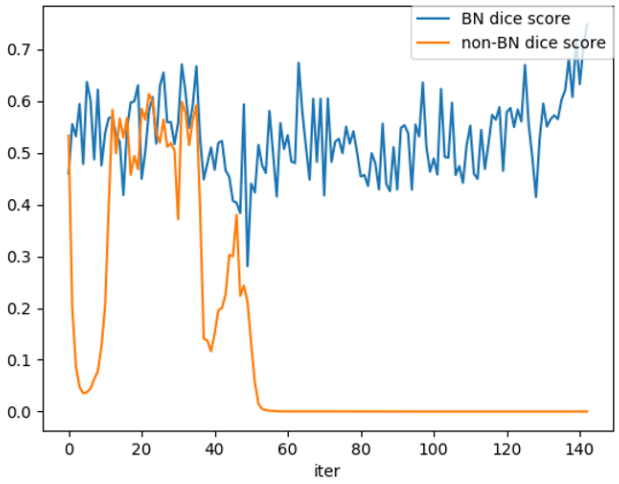

Figure 12: The effect of Batch Normalization. The horizontal axis represents the number of iterations, while the vertical axis represents the training batch dice score.

example, the model fails to recognize the tumor regions for a few patients, with the dice score lower than 0.4.

Figure 15 shows a successful segmentation for a patient in testing data, with the total dice score of 0.93. While Figure 16 shows a failure case, with a total dice score of 0.64 (the dice score for the selected slice is 0). From the figures, we can find the misclassifications mainly concentrate in the boundary regions. We suppose the reason for the invalid results is that the model cannot recognize the global information of the MRI scans with patch-based method.

Figure 17 shows the performance of bagging methods. By integrating the three models together, the prediction results are smoothed. Hence, the variance of the model is reduced. Unfortunately, limited by the computational resoures, we did not train more models for ensemble learning. However, we suppose the performance could be further enhanced by introducing more randomness and bagging more models.

4. Conclusion and Discussion

In this project, we proposed the patch-based 3D FCN model to solve the tumor segmentation problem. Experimental results reveal that our proposed model can produce

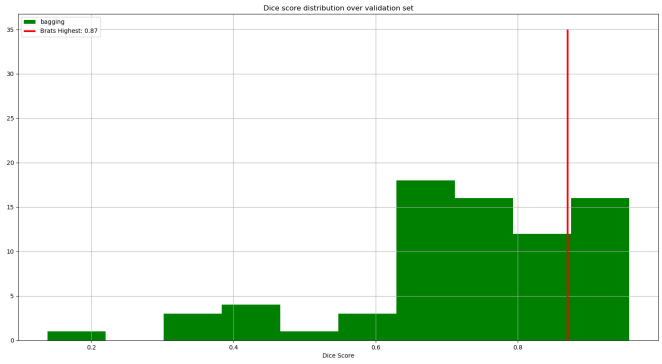

Figure 13: The performance statistics on testing data. The horizontal axis represents the dice score. The red line shows the best performance [4] in BraTS 2015 Challenge.

highly competitive segmentation results on part of the testing dataset, while fails to perform well on the rest. The performance shows that our model has a relatively low bias and a high variance. We propose to integrate multiple models that are trained with different hyper-parameter settings and random stages to smooth the outlier outputs and reduce the variance. However, limited by the computational resources, we did not have a chance to integrate more models in this project.

We suppose the poor performance on the failure testing cases is caused by the loss of global spatial information with the patch-based method. In fact, many wrongly classified pixels typically lie at the boundary of the brain and non-brain regions (see Figure 16), which can be easily detected by the human. These misclassified pixels can be corrected if the global information is learned by the neural network.

In our future work, we plan to introduce learning the localization of the tumor region before segmentation. Inspired by a recent deep learning model, the Mask R-CNN [18], we plan to adopt a CNN model with two components: the first CNN component learns a bounding box that localizes the tumors, and the second FCN or R-CNN component makes pixel-wise classification within the bounding box. In this way, more global information of the MRI scans could be learned in the localization part, while the positive and negative pixels are more balanced in the semantic segmentation part.

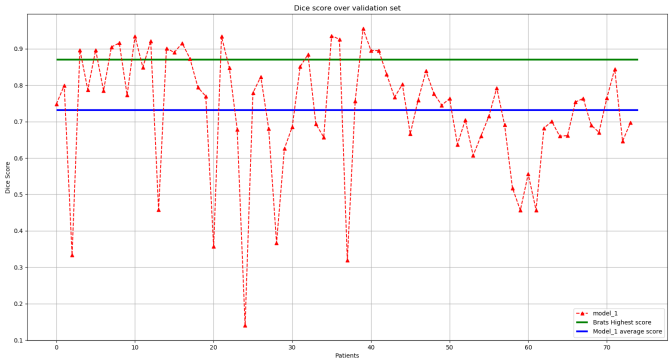

Figure 14: The accurate dice score for each patient in testing data. The horizontal axis represents the patient ID. The green line shows the best performance [4] in BraTS 2015 Challenge. The blue line shows our averaged dice score.

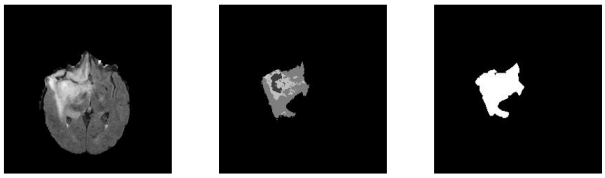

Figure 15: A successful segmentation case. Left: MRI slice in Flair channel. Middle: Ground Truth. Right: Prediction result.

Acknowledgement

The code can be downloaded from https://github.com/LAB1/Patch-based-3D-FCN-for-BraTS-Tumor-Segmentation.git.

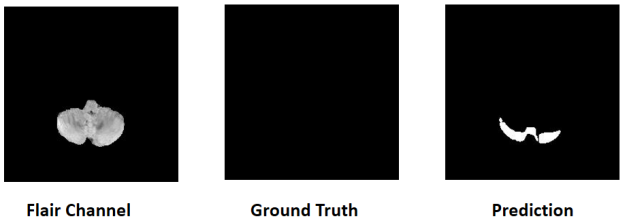

Figure 16: A failure segmentation case. Left: MRI slice in Flair channel. Middle: Ground Truth. Right: Prediction result.

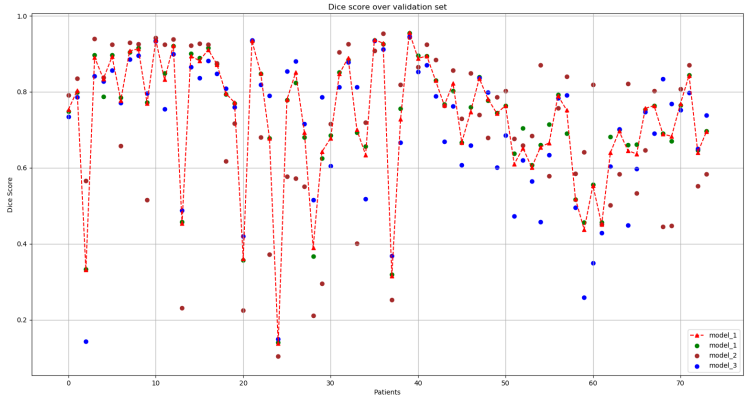

Figure 17: Bagging of the three models. The outliers are smoothed with bagging.

References

[1] ABTA, Brain tumor statistics, http://www.abta.org/about-us/news/brain-tumor-statistics/ (2017).

[2] ACS, Survival rates for selected adult brain and spinal cord tumors, https://www.cancer.org/cancer/brain-spinal-cord-tumors-adults/detection-diagnosis-staging/survival-rates.html (2017).

[3] C. Farabet, C. Couprie, L. Najman, Y. LeCun, Learning hierarchical features for

scene labeling, IEEE transactions on pattern analysis and machine intelligence 35 (8) (2013) 1915–1929.

[4] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, et al., The multimodal brain tumor image segmentation benchmark (brats), IEEE transactions on medical imaging 34 (10) (2015) 1993–2024.

[5] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, Vol. 1, IEEE, 2001, pp. I–I.

[6] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, Vol. 1, IEEE, 2005, pp. 886–893.

[7] N. Nabizadeh, M. Kubat, Brain tumors detection and segmentation in mr images: Gabor wavelet vs. statistical features, Computers & Electrical Engineering 45 (2015) 286–301.

[8] T. Ojala, M. Pietikainen, T. Maenpaa, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, IEEE Transactions on pattern analysis and machine intelligence 24 (7) (2002) 971–987.

[9] D. Zikic, B. Glocker, E. Konukoglu, A. Criminisi, C. Demiralp, J. Shotton, O. M. Thomas, T. Das, R. Jena, S. J. Price, Decision forests for tissue-specific segmentation of high-grade gliomas in multi-channel mr, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2012, pp. 369–376.

[10] S. Bauer, L.-P. Nolte, M. Reyes, Fully automatic segmentation of brain tumor images using support vector machine classification in combination with hierarchical conditional random field regularization, in: International Conference on Medi-

cal Image Computing and Computer-Assisted Intervention, Springer, 2011, pp. 354–361.

[11] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[12] M. Havaei, A. Davy, D. Warde-Farley, A. Biard, A. Courville, Y. Bengio, C. Pal, P.-M. Jodoin, H. Larochelle, Brain tumor segmentation with deep neural networks, Medical image analysis 35 (2017) 18–31.

[13] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.

[14] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2015, pp. 234–241.

[15] A. Casamitjana, S. Puch, A. Aduriz, V. Vilaplana, 3d convolutional neural networks for brain tumor segmentation: A comparison of multi-resolution architectures, in: International Workshop on Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries, Springer, 2016, pp. 150–161.

[16] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, 2015, pp. 448–456.

[17] D. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[18] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, arXiv preprint arXiv:1703.06870.