##### A Separate Three-dimensional Convolution Neural Network Architecture for Three-dimensional Medical Image Segmentation

Shidu Dong

In order to exploit three-dimensional (3D) context information to improve 3D medical image segmentation, we propose a separate 3D convolution neural network (CNN) architecture. First, a two-dimensional (2D) CNN is utilized to extract 2D features of each slice in the  plane of 3D medical images. Second, one-dimensional (1D) features reassembled from the 2D features in the -axis are inputted to a 1D-CNN and then are classified feature-wise. The analysis shows that the separate 3D-CNN has lower time complexity, fewer parameters and less memory space requirements than the other 3D-CNN with a similar structure. As an example, we extend the deep convolutional encoder-decoder architecture (SegNet) to the separate 3D-SegNet for brain tumor image segmentation. We also propose a method based on priority queues and the dice loss function to tackle the severe class imbalance for medical image segmentation. Experimental results show: (1) the separate 3D-SegNet extended from the 2D-SegNet can improve brain tumor image segmentation, (2) the proposed imbalance tackling method can speed up the training convergence and reduce the negative impact of the imbalance, (3) the separate 3D-SegNet with the proposed imbalance tackling method offers comparable performance with the existing state-of-the-art 3D-CNNs and experts for brain tumor image segmentation.

Deep learning, image semantic segmentation, class imbalance, medical image.

## 1 Introduction

Mdical image segmentation provides valuable information for medical diagnosis and is very important for planning treatment strategies, monitoring disease progression and predicting patient outcome [9]. In the last decade, many automatic segmentation methods have been proposed. Some of them use hand-designed features, such as local histograms, Gabor, and dense scale invariant feature transform descriptors (SIFT). These features are extracted first and then given to a classifier, such as support vector machine (SVM) and random decision forest, whose training procedure does not change the features [6]. Many methods also utilize conditional random field (CRF) for post-processing to refine the automatic segmentation results [10].

In recent years, deep convolution neural networks (CNN), which have shown powerful ability to learn hierarchy task-adapted features from in-domain data, have seen huge success in image classification [11, 19, 21], object detection and image semantic segmentation [17, 15, 3]. Shelhamer et al. proposed fully convolutional networks for semantic segmentation (FCN) [17]. Badrinarayanan et al. presented a deep convolutional encoder-decoder architecture for scene segmentation (SegNet) with a symmetrical encoder-decoder architecture [3]. Ronneberger et al. presented convolutional networks for biomedical image segmentation (U-Net) [16]. These networks are trained end-to-end and pixel-to-pixel. CNNs are also utilized to segment brain tumor image. Havaei et al. proposed two path and cascade CNN architectures for brain tumor image segmentation [6]. Recently, Wang et al. proposed three networks to hierarchically segment the whole tumor (WNet), tumor core (TNet) and enhanced tumor core (ENet) sequentially [22]. However, these methods based on two-dimensional (2D) CNN, which only extract the features in a slice, ignore the context information among adjacent slices, which can obviously be utilized to improve the performance. Fortunately, some three-dimensional (3D) CNN architectures are proposed to take advantage of 3D context information. Kamnitsas et al. proposed a 3D-CNN with a dual pathway and 11 layers [9] for brain tumor and ischemic stroke lesion image segmentation. Milletari et al. proposed fully convolutional neural networks with five compression and decompression paths and  pixel patch for volumetric prostate image segmentation (V-Net) [14]. Li et al. proposed a 3D-CNN with 7 convolution layers, 12 dilated convolution (atrous convolution) layers, and  pixel patch for granular image segmentation.

As a result of the enormous volume of 3D medical images and limited GPU memory, it is difficult to directly extend the advanced 2D-CNN image segmentation methods, such as FCN and SegNet, to three dimensions to exploit the 3D context for 3D medical image segmentation. We propose a separate 3D-CNN (S3D-CNN) by applying a combination of a 2D-CNN with a one-dimensional (1D) CNN to cope with the problem. The 2D-CNN is utilized to capture the 2D context features, and the 1D-CNN is applied to extract the context features among the slices along the -axis. As an example, we extend 2D-SegNet to the separate 3D-SegNet (S3D-SegNet) for brain tumor image segmentation.

One of the challenges of medical image segmentation is the severe class imbalance. The methods based on deep learning also cannot escape from the negative impact of the class imbalance [8]. Resampling, which balances the class by undersampling the majority class or oversampling the minority class [8, 7], is utilized by some methods based on patch such as DeepMedic to reduce the negative impact [9]. However, it has limiting effects to our method as well as the other methods, such as FCN, SegNet, and their variants, because they take the whole 2D-images as the inputs. The weighted loss function, which assigns higher misclassification costs to the minority classes than to the majority [8, 18] may be another option. But many pixels belonging to the majority class are encouraged to be classified as the minority to pursue the lowest cost, which may cause a lower dice score. The dice score is an important evaluation measure for the image segmentation. Recently, the dice loss function and its variants are utilized to tackle this issue [14, 20]. Inspired by these methods, we develop an imbalance tackling method based on the priority queue and dice loss function. The priority queue of each tumor class is created to choose an equal number of samples for mini-batch training, which can ensure relative class imbalance. In addition, the dice loss function is also applied to reduce the negative impact further.

The contributions of this paper are four folds: [1)]

1. We propose a separate 3D-CNN scheme for the 3D medical image segmentation by applying a 2D-CNN in the  plane and then a 1D-CNN in the -axis.

2. We present an imbalance tackling method based on the priority queue and dice loss function for the image semantic segmentation.

3. We analyze and compare the time, space, and parameter requirements and receptive fields of S3D-CNN and 3D-CNN with the similar structure.

4. As an example, we extend 2D-SegNet to S3D-SegNet for brain tumor image segmentation.

## 2 Architecture

### 2.1 S3D-CNN Architecture

As being illustrated in Fig. 1, the S3D-CNN contains 2D-CNN and 1D-CNN parts. First, slices in the  plane of 3D medical images are fed into the 2D-CNN to extract their 2D features which certainly holds the 2D context information in their receptive field. Second, the 2D features are reassembled into the 1D features in the -axis. Third, the 1D features are inputted to the 1D-CNN to take advantage of the 3D context and then are classified feature-wise.

Figure 1: An illustration of the separate 3D-CNN architecture

Figure 2: An illustration of the upsampling for 2D-SegNet [3]

Figure 3: An illustration of the 2D-SegNet [3].

Obviously, any 2D-CNN can easily be extended to an S3D-CNN to exploit the 3D context information. As an example, we extend the SegNet (2D-SegNet) [3] to the separate three dimensional SegNet (S3D-SegNet) for brain tumor image segmentation. Different from [3], in this paper, the 2D-SegNet only includes four encoder-decoder layers as being depicted in Fig. 3 and Table 1, considering of the resolution of brain MRI images used in experiments. In Fig. 3, Conv + Batch Normalization + Relu layer (Conv+ for short) in encoder performs convolution with  (kernel height and width) kernel, followed by batch normalization and an element-wise Relu (.i.e max(0, x)). Then the  max-pooling with stride 2 (non-overlapping window) is performed. The resulting outputs contain features subsampled by a factor 2 and max-pooling indices (MaxIndices for short) implying the location of the maximum feature value in each pooling window [3]. Max-pooling pursues a broad context for each feature and MaxIndices are to capture the edge information [3]. Each decoder holds an upsampling layer and 2-3 Conv+ layers. Upsampling fills the output feature maps with its input features at the position implied by the memorized MaxIndices from the corresponding encoder (see Fig. 2). The last decoder layer outputs features containing 64 channels for each slice. The outputs of the softmax are applied to train the network.

Table 1: The Shapes of the Convolution Kernels of 2D-SegNet

To capture the context information across slices, we use 1D-CNN to process the output features of the 2D-CNN along the -axis. Inspired by SegNet [3], we provide 1D-SegNet given as Fig. 4 and Table 2. The 1D-SegNet contains three encoder-decoder layers. In Fig. 4, 1D Conv + Batch Normalization + Relu layer (1D Conv+ for short) in an encoder layer performs 1D convolution, followed by batch normalization and Relu. Each encoder has 2-3 1D Conv+ layers and a 1D max-pooling layer which can figure out the features and MaxIndices by the following equations





where  denotes the  feature of the  max-pooling layer,  indicates the input features, and  denotes its length. Each decoder has 1D upsampling layer, and 2-3 1D-Conv+ (see Fig. 4 ). Upsampling fills the output feature maps with its input features at the position implied by MaxIndices by the following equation



The outputs of the last decoder (Decoder 1) concatenated with the 1D-features of the input are pushed to a convolution layer with  kernel and a softmax layer to classify each pixel independently.

Training procedure contains three stages. First, like [3], the 2D-SegNet is trained end-to-end with 2D slices in the  plane and their corresponding labels in the training dataset. Second, all slices in the training data set are fed to the 2D-SegNet to extract their features. Third, the 1D-SegNet is trained end-to-end with the 1D-features in the -axis and their corresponding labels.

Figure 4: An Illustration of the 1D-SegNet

Table 2: The Shapes of the Convolution Kernel of 1D-SegNet

### 2.2 The Analysis of S3D-CNN

For easy comparison, we suppose: (1) the S3D-CNN and the 3D-CNN as a reference have the similar layer structures, (2) the kernel’s height , width , and depth  of each layer of the S3D-CNN are the same as those of 3D-CNN and they are constant, (3) the height , width , and slice number  of the input data size of each layer of the S3D-CNN are the same as those of the 3D-CNN.

#### 2.2.1 Time Complexity Analysis

The speed of a program is critical in some application. The real running time of training or testing for given data are reported in some works for comparability. But it is unfair to take the real running time as the metric of the algorithm time complexity because many factors such as a version of GPU or CPU, the deep learning frameworks, and even the programmer’s skills can impact the real running time of a program. So I advocate using the number of elementary operations performed by the algorithm as the metric just like the time complexity analysis in computer science.

The time complexity of inference is more important than that of training because the inference process will be executed many times after being trained. Though training needs repeating the forward inference and backpropagation to calculate the gradients many times, given training data, optimization algorithm, and iteration times, the training time can be proportional to the inference time.

Convolution, which is the most computation consuming operations in CNN, has only multiplication and addition operations. The number of multiplications performed by 3D convolution of the  layer is given by

,

where , , , and  denote the height, width, slice number, and channel number of the input data, respectively. , , and  denote the height, width, and depth of the kernel, respectively.  denotes the channel number of the output data. In general, . And the number of the additions is .

Batch normalization has multiplication, addition, subtraction, and division operations. The number of multiplications needed by batch normalization is written as

,

and the amount of the additions, subtractions and divisions approach to .

Relu, max-pooling, and upsampling only have comparison operation which always needs fewer clock cycles than multiplication. The number of comparisons performed by Relu, max-pooling or upsampling is estimated by

.

The number of operations performed by softmax can be considered as a constant, because each pixel needs it only once and almost every network has a similar softmax layer.

Smaller kernel found many advantages for CNN [19], so in general, the . To capture more features, in general . Comparison with , the number of the operations performed by batch normalization, Relu, max-pooling, and upsampling are small. The total number of the operations needed by the network is mainly determined by the number of the multiplications performed by the convolutions. So the complexity of the 3D-CNN can be given by

 (1)

where . Obviously, 2D-CNN can be seen as a special case of 3D-CNN. Let , the complexity of the 2D-CNN can be given by



And similarly, let , the complexity of the 1D-CNN can be written as



Therefore, the time complexity of the S3D-CNN can be formulated as



 (2)

From (1) and (??), when ,  and , we get . Especially, if , , ,  and  are constant, and , we get



#### 2.2.2 The Analysis of Minimum Memory Space Required by Training Process

In each iteration of training, the output data of the previous layer can’t be released, due to being utilized to compute gradient in backpropagation. Since input of the  layer are output of the  layer, we can only calculate the memory space requirement of the output. The memory space required by the output of 3D convolution of the  layer can be given by



Thus, the total space requirement of the 3D convolution can be approximated by



The batch normalization and Relu layers need as much memory space as the convolution layer, and max-pooling /upsampling need as much memory space as the first convolution layer of the next encoder/decoder. Every network has only one softmax layer, and the memory space requirements of the parameters and their gradients are small, compared with those of the data and features. Therefore, the memory space needed by them can be ignored, and the total space requirement of 3D-CNN is given by

 (3)

where c is constant. Distinctly, when there is only one sample in mini-batch, the memory space requirement is minimum. 3D-CNN has to take  continuous slices of the same subject as inputs, so the minimum memory space required by 3D-CNN is given by Eq.3. 2D-CNN can have a slice in the  plane in mini-batch, thus its minimum memory space requirement can be given by



1D-CNN can have only a 1D-feature in -axis in mini-batch,so its minimum memory space requirement can be given by



In general, , so the minimum space requirement of S3D-CNN can be determined by S2D-CNN

 (4)

From (3) and (4), we can see that 3D-CNN needs approximately  times minimum memory space than S3D-CNN. As a result of the limited memory capacity of GPU, it is maybe difficult for 3D-CNN to take the whole 3D medical image as input. The samples from the same subject may arise severe class imbalance in mini-batch training for the deficiency in the samples of some classes. Some 3D-CNN methods use patch to cope with those problems. For example, V-Net [14], High-Res Net [12], and DeepMedic [9] use , , and  pixel patch respectively. However, the size of patch may limit the size of the receptive field of 3D-CNN.

Requiring as much minimum memory space as 2D-CNN and less minimum memory space than 3D-CNN , S3D-CNN has the following advantages: (1) it can reduce the class imbalance in mini-batch training by sampling more images belonging to different patients to contain different class pixels, (2) it is also easier to increase the receptive field by inserting more layers, (3) all advanced 2D-CNN network can be easily extended to S3D-CNN to take advantage of 3D context information for improving 3D medical image segmentation.

#### 2.2.3 Receptive Field Analysis

The neighborhood of voxels in the input that influence the activation of a neuron is its receptive field [9], whose size determines how much the network can exploit the context information near the current pixel. The receptive field of the  3D convolution layer can be given by [9]



where  denotes the receptive field of the  layer (if the previous layer is max-pooling , ), and  and  denote the size and stride of the kernel of the  layer, respectively. In general, .

Similarly, the receptive field of the  max-pooling layer can be given by



where  denotes the stride of the  max-pooling layer. In general .

Let 3D-CNN be  layers, the receptive field of 3D-CNN can be estimated by the following recurrence equation

 (5)

The equation (5) is valid for 2D-CNN and 1D-CNN. In the decoding part, the increased resolution by upsampling may offset the extended receptive field by the convolution. So the receptive field of S3D-CNN is given by

 (6)

The receptive field of the S3D-CNN is the same as the 3D-CNN with a similar structure. However the receptive field of S3D-CNN can be increased by inserting some convolution or max-pooling layers in the 2D-CNN or 1D-CNN easily. While due to the limited memory of GPU, it is difficult for the 3D-CNN to be inserted more layers. With  patch, only 8 convolution layers and no max-pooling, the receptive field of DeepMedic is . Another pathway with the down-sampled image by a factor 3 has to be added to extend the receptive field to . From the Eq. (6), Table 1, and Table 2, we can figure out that the receptive field of the S3D-SegNet is . The receptive field of the 1D-SegNet in the -axis is 42 rather than 86 due to having 3 rather than 4 encoder-decoder layers like 2D-SegNet.

#### 2.2.4 The Parameter Number Analysis

Obviously, the parameters number of the  3D convolution layer can be given by



Batch normalization has two parameters which can be ignored. The total parameter number of the 3D-CNN can be written as:



the total parameter number of the 2D-CNN and 1D-CNN can be written as:





and the total parameter number of the S3D-CNN can be formulated as



If , we get



With fewer parameters than 3D-CNN, S3D-CNN can be trained easier.

## 3 Imbalance

One of the challenges for medical image segmentation is the severe class imbalance. We surveyed the pixel and slice distribution of the different classes for the images in the training set of multimodal brain tumor segmentation challenge 2017 (Brats 2017 for short). From the first and second rows of the table, we can see the severe pixel distribution imbalance: the pixel number of the class 0 (L0, i.e, background, health, and others) is more 260 times than that of the class 4 (L4, i.e, Necrotic and Non-enhancing solid core). And from the third and fourth rows, we can see the slice imbalance: every slice contains pixels of the class 0 (L0), while only 16.68 percent of slices of LGG includes pixels of the class 1 (L1).

Table 3: The Pixel and Slice Percentages of the Different Classes for the HGG and LGG Images

### 3.1 The Class Imbalance of Slice Distribution

We proposed a sampling method based on the priority queue to cope with the slice class imbalance. Except the class 0, each class has a queue storing the indices of its samples sorted by the weight determined by Eq. (7). In mini-batch training, each queue prepares an equal number of samples, which ensures the relative class balance for mini-batch training. Algorithm 1 outputs the index list  of the slice image for the class 0 and the minimum-oriented priority queue  for each of the other classes. Algorithm 2 prepares images and the corresponding labels for mini-batch training with  and  and updates the priority value  by Eq. (9) with the training loss and access frequency. We use the similar priority queue for 1D-CNN.

\begin{algorithm}[H]

\caption{Priority Queue}

**\label{AlgCreateQueue}**

\textbf{Input:}

All label maps in training set. $X\_{p,s}$ and $L\_{p,s}$ denote the $s^{th}$ slice image of the $p^{th}$ patient and the corresponding label map, respectively.

\textbf{Output:}

$LL\_0$, $Q\_{1..K}$, $f$, and $R$. $f$ and $R$ denote the access frequency of the slice in the training and the priority weight value of the sample in the priority queue, respectively.

%\REQUIRE in

% \ENSURE out

\begin{algorithmic}[1]

\State $L\_0 \leftarrow \Phi$, $Q\_{1..K}\leftarrow \Phi$

\For{ each $L\_{p,s}$ }

\State Count the pixel number of each class $l$ in $L\_{p,s}$: $C\_{p,s,l}$

\For{$l \leftarrow 1, \ldots,K$}

\If {$C\_{p,s,l}>0$}

\State Compute the priority value $R$ of $X\_{p,s}$ in $Q\_l$

\begin{equation}

\**label{eq:InitPriorityValue}**

W \leftarrow R \leftarrow 30+log\left(\frac{\sum\_{k \neq l} C\_{p,s,k}}{C^2\_{p,s,l}}\right)

\end{equation}

\State Push $[W,R,p,s,f\leftarrow 0]$ into queue $Q\_l$ in an ascending order of $W$

\EndIf

\EndFor

\If {$C\_{p,s,0}>0$ and $\sum\_{l>0}C\_{p,s,l}=0$}

\State $LL\_0 \leftarrow LL\_0\cup[p,s]$

\EndIf

\EndFor

\end{algorithmic}

\end{algorithm}

### 3.2 The Class Imbalance of Pixel Distribution

In spite of dealing with slices or 1D features imbalance with the priority queue, there exists the class imbalance of pixel distribution in a slice or z-direction, due to 2D-CNN or 1D-CNN taking the whole slice or z-axis as input in training. We use the following dice loss function to deal with the problem.

 (8)

where  is the output of the softmax layer, and  is the ground truth which are  matrices.  and  denote the pixel number and class number respectively.  indicates the probability of pixel  belonging to the class  and  denotes that the pixel  is labeled as the class  by experts ( denotes it is labeled as the others).

\renewcommand{\algorithmicensure}{\textbf{Output:}}

\textbf{Output:}

The parameters of 2D-SegNet

%\REQUIRE in

% \ENSURE out

\begin{algorithmic}[1]

\For{ each step }

\State $Images\leftarrow\Phi$,$Labels\leftarrow \Phi,idx \leftarrow 0$

\State $len \leftarrow $ length of $LL\_0$

\State $[p,s] \leftarrow LL\_0\left[(idx+1) \ \ mod \ \ len \right] $

\State $Images \leftarrow Images \union X\_{p,s}$

\State $Labels \leftarrow Labels \union L\_{p,s}$

\State $TopL \leftarrow \Phi $

\For {$f \leftarrow 1, \ldots,fn$}

\For {$l \leftarrow 1, \ldots,K$}

\State Get top element (the minimum weight value )

\State $[R,p,s,f] \leftarrow Pop(Q\_l)$

\State $f \leftarrow f+1$

\State $Images \leftarrow Images \union X\_{p,s}$

\State $Labels \leftarrow Labels \union L\_{p,s}$

\State Push($TopL$,$[R,p,s,l,f]$)

\EndFor

\EndFor

\State Training 2DSegNet

\State $loss \leftarrow $train$(Images,Labels)$

\For{$t \in TopL$}

\State $[R,p,s,l,f] \leftarrow t$

\State Update weight value

\begin{equation}

\**label{eq:updateRate}**

W \leftarrow R\times \max(1-loss,0.01)\times f

\end{equation}

\State Push $[W,R,p,s,i]$ into queue $Q\_l$ in the ascending order of $W$

\EndFor

\EndFor

\end{algorithmic}

\end{algorithm}

## 4 Experiment

### 4.1 Benchmark Dataset and Training Parameters

We use the dataset of Multimodal Brain Tumor Segmentation Challenge 2017 (BRATS 2017) to evaluate our methods. The training data set consists of 210 cases of high-grade gliomas (HGG) and 75 patients of low-grade gliomas (LGG). The validation datasets include 46 subjects without being labeled and graded. Each case has four MRI sequences i.e, FLAIR, T1, T1-contrast, and T2. Each sequence consists of 155 slices, each of which has  resolution. All images in training set are annotated by experts. Due to ambiguities in individual tumor structure definitions, the variability in manual segmentation of inter-expert is a bit high. The mean dices of the expert’s manual segmentation of the whole tumor (including all four tumor structures, WT for short), the tumor core region (including enhancing, non-enhacing core and necrotic structures, TC for short), and the active tumor region (enhacing structure, ET for short) are 85, 75, and 74 respectively. Their standard deviation are 8, 24, and 13 respectively [13]. The scores are obtained by comparing individual raters against the consensus segmentation. These scores provide an estimate of an upper limit for the performance of any algorithmic segmentation [13]. Especially the mean and standard deviation of the expert’s manual segmentation of the tumor core region for LGG are 67 and 28 respectively. It means that it is more difficult to segment the tumor core region of LGG images. Different from Brats 2013, only three classes are labeled, .i.e, the necrotic (class 1), the peritumoral edema (class 2) and the enhancing tumor (class 4). Because of only images in the training dataset being labeled, we randomly select 10 percent of the samples as the testing set. So, we just use 199 HGG and 69 LGG cases as real training set to train the models in our all experiments.

2D-SegNet are trained end-to-end with the slices and the corresponding labels prepared by Algorithm 3.1 and 3.2 from real training dataset. Its mini-batch size is 16. 1D-SegNet are trained end-to-end with the 1D-features in -axis and the corresponding labels chosen by the similar algorithms to Algorithm 3.1 and 3.2 . Its mini-batch size is 1680. We use Adam optimization algorithm with an initial learning rate  and exponential decay rates  [1].

### 4.2 Performance Scores

Let  denote the binary map of the class  of the  pixel obtained by the segmentation algorithm, and  denote the corresponding experts’ consensus truth. So we can calculate the well-known Dice Score [13]



where  is the logical AND operator. We also can gain sensitivity (True Positive Rate, TPR, Recall) and specificity (True Negative Rate, TNR)[13]

 (10)



The dice score, sensitivity, and specificity are measures of pixel-wise overlap of the segmented regions[13]. The hausdorff distance is the measure of surface distance between the algorithm segmentation results  and ground truth  [13]



### 4.3 Experimental Results on the Testing Dataset

We randomly select 10 percent of subjects from the training dataset of brain tumor segmentation challenge 2017 as the testing dataset. It contains 21 cases of HGG and 7 cases of LGG never used in training process. The source code of DeepMedic is downloaded from Kamnitsask’s Github. DeepMedic are trained in 35 epochs with the default parameters and training dataset images. All methods did not utilize post-processing, such as conditional random field (CRF). 2D-SegNet with the weighted loss function (2D-SegNet-WL) are trained in 120,000 iteration (about 52 epochs). 2D-SegNet and 1D-SegNet of S3D-SegNet are trained in 60K and 30K (about 24 epochs and 10 epochs), respectively. The Table 3 illustrates the performance scores of S3D-SegNet with priority queues and the dice loss function (S3D-SegNet), DeepMedic, and 2D-SegNet-WL. The tables 4 and 5 are the average confusion matrices of S3D-SegNet, DeepMedic and 2D-SegNet-WL for the test dataset from HGG and LGG, respectively. The element in row  column  of the confusion matrices indicates the percentage of pixel assigned to class  which belong to the class . From the Table 3, we can see that: (1) S3D-SegNet gained comparable dice scores for HGG with DeepMedic, (2) S3D-SegNet obtained higher average dice scores for LGG and the whole testing set than DeepMedic and 2D-SegNet-WL, (3) S3D-SegNet offered better Hausdorff score for HGG, LGG and the whole testing set than DeepMedic and 2D-SegNet-WL. Especially, the dice scores of the class 1 and 2 of S3D-SegNet are 7-8 percent higher than those of DeepMedic for LGG image segmentation. It is considered more difficult to segment images of LGG than HGG. Although DeepMedic and 2D-SegNet-WL gain higher sensitivity scores, from the confusion matrices, we can see that they classify more pixels of health or background (the class 0) into the other tumor classes than S3D-SegNet. It may be not worth the cost because cutting the more health area of the brain may disable more functions of the brain. Since only 39 per million of pixels (35 pixels per patient) in LGG belong to Class 4, the scores of class 4 in LGG are very low. From Tabel 5, we can see that many pixels belonging to the class 1 (necrotic) are falsely classified into the class 2 (edema), which cause the low dice scores of LGG.

Fig. 6 illustrates the segmentation examples of expert’s manual segmentation, S3D-SegNet, DeepMedic, and 2D-SegNet-WL. From the Fig. 6, we can see that S3D-SegNet gain better segmentation results than DeepMedic and 2D-SegNet-WL. Although DeepMedic and 2D-SegNet-WL gain higher sensitivity scores than S3D-SegNet, without 3D-CRF as post-processing, the segmentation results of DeepMedic contains many isolated edemas which should be health pixels, and the edemas segmented by 2D-SegNet-WL are larger than those by expert’s manual segmentation. From Eq. 10, we can see that sensitivity score is unable to measure the error that the pixels of the class 0 are falsely classified into the other tumor classes.

Performance Score Comparision on Testing DataSet

Table 4: Confusion Matrices of S3D-SegNet, DeepMedic, and 2D-SegNet-WL for HGG

Table 5: Confusion Matrices of S3D-SegNet, DeepMedic, and 2D-SegNet-WL for LGG

Table 6: Segmentation Examples of Expert’s Manual Segmentation, S3D-SegNet, DeepMedic, and 2D-SegNet-WL

### 4.4 The Performance Comparison of the Variants of S3D-SegNet

In this experiment, we use the validation datasets of Brats2017 to evaluate the proposed method and their variants. The performance scores of the validation datasets are from the official website [4, 5] by uploading the segmentation results of the methods( notes: the scores of the class 0, i.e., health and background pixels isn’t given). In Table 4.4, RS and PQ denotes the methods using random sampling and priority queue to choose samples for the mini-batch training, respectively. And WL and DL imply the methods with weighted loss function and dice loss functions, respectively. The weights of class 0, 1, 2, and 4 are 0.00247, 3.94, 0.302, and 1 respectively, which are figured out with the ratio between pixel numbers of classes [3] and their median. From the Table 4.4, we can see that: (1) the dice scores of all methods based on S3D-SegNet are better than those based on 2D-SegNet, due to exploiting 3D-context information. (2) The dice and the hausdorff scores of all methods based on priority queue are better than those based on the random sampling, due to reducing the slice imbalance. (3) The dice and the hausdorff scores of all methods based on dice loss function are better than those based on the weighted loss function.

Table 7: The Performance comparison of the Variants of S3D-SegNet and 2D-SegNet

### 4.5 Performance Comparison between the Methods Based on Priority Queue Sampling and Random Sampling

Fig. 5 illustrates the dice scores of 2D-SegNet with the priority queue (PrioQueue for short) and the random sampling (RandSamp for short) at 5K, 10K, 15K, 20K, 25K, 30K, 35K, and 40K iteration which given the mini-batch size and training data set size approximately correspond to 2, 4, 6, 8, 10, 12, 14, and 16 epochs. All scores are from the official evaluation website of Brats2017 by uploading the segmentation results of validation dataset. From the Fig. 5, we can see that 2D-SegNet based on the priority queue offers higher average dice scores at each iteration and faster training convergence than 2D-SegNet based on random sampling.

Figure 5: The Performance and Convergency Speed Comparison of 2D-SegNet with the Priority Queue and the Random Sampling

### 4.6 The Performance Comparison of S3D-SegNet and the Other 3D-CNN

Table 4.6 illustrates the performance comparison of S3D-SegNet, DeepMedic [9], Dense VNet, and High-Res Net[12]. The performance scores of S3D-SegNet are from the official evaluation website of Brats2017 by uploading the segmentation results of validation dataset. The performance score of DeepMedic, Dense VNet[2], and High-Res Net (including small default, large and their ensemble, i.e, Majority Vote) from the Brats 2017 paper of Zach Eaton-Rosen, et al. [2]. From Table 4.6, we can see that the S3D-SegNet gained the comparable dice scores with DeepMedic, Dense VNet, and High-Res Net and obtained better hausdorff scores for ET and WT than these methods. Although its sensitivity scores are lower than DeepMedic, as mentioned previously, DeepMedic without 3D-CRF as post-processing classify many health pixels as tumor pixels. Especially, the average dice scores of WT and TC of S3D-SegNet are higher than those of expert’s manual segmentation (WT: , TC: ) which are considered as an upper limit for the performance of any algorithmic segmentation [13].

Table 8: The Performance Comparison of S3D-SegNet and the other 3D-CNN

## 5 Conclusion

We presented S3D-CNN, a separate 3D convolution network architecture for 3D medical image segmentation. The primary motivation behind S3D-CNN was the need to design an efficient network taking advantage of 3D context information for improving 3D medical image segmentation. We analyzed the time complexity, minimum memory space, and parameter number requirements and receptive field of both S3D-CNN and the 3D-CNN with a similar structure. We also proposed a method based on the priority queue and dice loss function to cope with the severe class imbalance. The experimental results shows that 3D-SegNet extended from 2D-SegNet with the proposed method and the imbalance tackling method based on the priority queue can improve 3D brain tumor image segmentation.