Research Article

A Multiscale-Based Adjustable Convolutional Neural Network for Multiple Organ Segmentation

Zhiqiang Tian¹, Jingyi Song¹, Chenyang Zhang¹, Xiaohui Tian¹, Zhong Shi^{2,3}, and Xiaofu Yu^{2,3}

¹School of Software Engineering, Xi'an Jiaotong University, 710049, China

²Institute of Cancer and Basic Medicine (ICBM), Chinese Academy of Sciences, 310022, China

³Cancer Hospital of the University of Chinese Academy of Sciences, 310022, China

Correspondence should be addressed to Zhiqiang Tian; zhiqiangtian@xjtu.edu.cn

Received 7 February 2020; Revised 16 June 2020; Accepted 21 July 2020; Published 3 August 2020

Academic Editor: Yin Zhang

Copyright © 2020 Zhiqiang Tian et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Accurate segmentation of organs-at-risk (OARs) in computed tomography (CT) is the key to planning treatment in radiation therapy (RT). Manually delineating OARs over hundreds of images of a typical CT scan can be time-consuming and error-prone. Deep convolutional neural networks with specific structures like U-Net have been proven effective for medical image segmentation. In this work, we propose an end-to-end deep neural network for multiorgan segmentation with higher accuracy and lower complexity. Compared with several state-of-the-art methods, the proposed accuracy-complexity adjustment module (ACAM) can increase segmentation accuracy and reduce the model complexity and memory usage simultaneously. An attention-based multiscale aggregation module (MAM) is also proposed for further improvement. Experiment results on chest CT datasets show that the proposed network achieves competitive Dice similarity coefficient results with fewer float-point operations (FLOPs) for multiple organs, which outperforms several state-of-the-art methods.

1. Introduction

Radiation therapy is the main clinical method of treating various cancers [30]. It can be seen as a trade-off between sending maximum dose to the target-volume (TV) and minimum dose to the OARs [1, 31]. Accurate segmentation of OARs contributes to the protection of normal organs during treatment [32]. Manually delineating OARs slice-by-slice in CT scans requires expertise and lots of time. Adjacent soft tissue with low contrast and noise blurs the boundaries of organs, which may also lead to delineation errors. Therefore, automatic segmentation has become a research hotspot [33]. Such a framework could help radiologists segment OARs more accurately with much less time.

Deep convolutional models have shown state-of-the-art performance in image segmentation especially after fully convolutional networks (FCN) [2] were proposed. In FCN, fully convolutional layers are substituted for fully connected

layers in a classic classification network to reserve spatial information, which makes FCN a reasonable choice for dense prediction. Based on FCN, Badrinarayanan et al. [3] introduced a concept of encoders and decoders and upsampled the low-level inputs in turn to the original resolution, which achieves impressive performance in scene understanding applications. U-Net [4] adopted the similar encode and decode paths and used several skip connections between them to combine the low-level and high-level features. Thanks to this carefully designed structure, it achieves considerable success in medical image segmentation tasks. In addition, great efforts have been subsequently made to improve the performance of U-Net [5, 6].

Targets in images usually have various sizes. It is crucial to let a network “see” objects of different scales. What kind of multiscale information to use and how to integrate information are the purpose of this work. PSPNet [7] merged convolutional features from different region-based context

aggregations. DeepLabs [8–11] similarly introduced a spatial pyramid pooling with different dilated kernels. An attention mechanism was first proposed to model long-range dependencies in machine translation and has been successfully transferred to computer vision tasks. It allocates more weights to the potentially interesting regions and suppresses the less informative ones. DANet [12] built a dual attention module to simultaneously model channel-wise and spatial-wise semantics. Oktay et al. [5] proposed an attention gate in the skip connection layer of the U-Net to teach the decoder where to “look.” Selective Kernel [13] (SK) took advantage of the attention to guide the fusion of multiscale information. It embeds multiple lightweight SE [14] attentions to dynamically select the receptive field size of each neuron in convolutional layers, which is also consistent with the neuroscience cognition that the visual cortical neuron adaptively adjusts the size of its receptive field according to stimulus. In such case, the learned attention maps play the role of a stimulus.

Aside from accuracy, computation complexity is another factor to measure model performance. A widely used evaluation metric of computation complexity is the number of float-point operations, also known as FLOPs [15]. Substantial efforts have been made to reduce FLOPs of convolutional networks. MobileNets [16, 17] adopted depth-wise separable convolution, which splits a standard convolution into a depth-wise and a point-wise convolution. It can greatly reduce the computation cost. ShuffleNets [18, 19] further integrated group convolution and reduced channel redundancy in point-wise connectivity. These methods focus on sparse connection between channels. In contrast, Octave Convolution [20] (OctConv) focused on the spatial dimension and argued that potential redundancy may exist. Each location independently stores its own features, ignoring the possibility that there could be similar information between adjacent locations. OctConv presented a multifrequency processing algorithm, which splits feature maps into high and low frequency to reduce spatial redundancy. Through sharing information with adjacent neurons, the spatial resolution of the low-frequency parts can be reduced, which helps to save computation and memory cost.

A highly generalized CNN should take the computing power and memory consumption into consideration or otherwise may fail to run on the limited computation conditions, especially the training stage [21]. Several strategies have been proposed to alleviate this issue. A most straightforward method is to reduce batch size during training. Heavy models like V-Net [22] and No-New-Net [23] adopted a small batch size of two. However, reducing batch size would make the network hard to converge. In particular, when the batch size is reduced to one, the Batch Normalization [24] that is important for stable gradient propagation becomes invalid. Another possible way to reduce computation is to compress the width and depth of the network and image resolution, which is under the risk of accuracy decrease. Researches have shown that the increase of any of these three elements can improve model performance (the width [25], the depth [26], the image size [27], and all three aspects [28]). In this paper, we propose an end-to-end network with symmetrical

encode and decode paths to automatically segment multiple organs. The model complexity of the proposed network can be flexibly adjusted to meet the computation requirements without accuracy sacrifice. Neither the batch size nor the model capacity needs to be changed. Concretely, we introduce an accuracy-complexity adjustment module (ACAM) inspired by Chen et al. [20] throughout the encoder and decoder. A hyperparameter is adopted to control the compression of spatial redundancy of feature maps. To further enrich feature representations and capture information of organs in different sizes, we add a nonlinear multiscale aggregation module (MAM) after the encoder. The branches of the module have different rates of dilations, which is inspired by Li et al. [13].

Our contributions are summarized as follows:

- (i) We introduce an accuracy-complexity adjustment module throughout the encoder and decoder to increase the segmentation accuracy and reduce the model complexity and memory usage simultaneously
- (ii) We present an attention-based multiscale aggregation module after the encoder to enrich feature representation for further boosting the segmentation accuracy
- (iii) The proposed network achieves competitive results, which outperform several state-of-the-art methods on chest CT segmentation datasets

The rest of this paper is organized as follows. In Section 2, the proposed method is described in detail. In Section 3, experimental results of our method and the state-of-the-art methods are presented. Finally, we state the conclusion in Section 4.

2. Materials and Methods

Our network is a U-shaped architecture, which contains two symmetrical encode and decode paths [4]. Skip connections are used to transfer the features from the encoder directly to the decoder. ACAMs are applied all-through the encoder and decoder. MAM is added after the encoder to further enrich feature representation and boost segmentation performance. Details of the network architecture can be seen in Figure 1. The numbers of channels of all layers can be seen in Table 1.

2.1. Accuracy-Complexity Adjustment Module. Since images can be divided into high frequency with details and low frequency with global layout, Chen et al. [20] infer that feature maps can also be divided into high and low frequency with different spatial contexts. Following this idea, we present an ACAM and apply it in every convolutional layer to get high- and low-frequency feature maps. The width and the height of the low frequency are reduced to half size, and the number of channels is controlled by a parameter α to meet the computation resource.

To be specific, an input feature map $X \in \mathcal{R}^{c \times h \times w}$, where $h \times w$ denotes the spatial resolution and c the number of

FIGURE 1: The architecture of the proposed network. The accuracy-complexity adjustment module (ACAM) is applied all-through the encoder and decoder. $w \times h$ and c represent the resolution and channels of the feature map, respectively. α is a hyperparameter controlling the ratio of the low frequency along the channel dimension. The multiscale aggregation module (MAM) is added after the encoder to enrich feature representation.

TABLE 1: The numbers of channels of all layers.

Layer name	Input channels		Output channels	
	High frequency	Low frequency	High frequency	Low frequency
InConv	1	—	$64(1-\alpha)$	64α
DownConv1	$64(1-\alpha)$	64α	$128(1-\alpha)$	128α
DownConv2	$128(1-\alpha)$	128α	$256(1-\alpha)$	256α
DownConv3	$256(1-\alpha)$	256α	$512(1-\alpha)$	512α
DownConv4	$512(1-\alpha)$	512α	$512(1-\alpha)$	512α
MAM	$512(1-\alpha)$	512α	$512(1-\alpha)$	512α
UpConv1	$1024(1-\alpha)$	1024α	$256(1-\alpha)$	256α
UpConv2	$512(1-\alpha)$	512α	$128(1-\alpha)$	128α
UpConv3	$256(1-\alpha)$	256α	$64(1-\alpha)$	64α
UpConv4	$128(1-\alpha)$	128α	$64(1-\alpha)$	64α
OutConv	$64(1-\alpha)$	64α	num of classes	—

channels, can be factorized into $X = \{X^H, X^L\}$ along the channel dimension via α . $X^H \in \mathcal{R}^{(1-\alpha)c \times h \times w}$ and $X^L \in \mathcal{R}^{\alpha c \times h/2 \times w/2}$ represent high and low frequency, respectively, which is shown in Figure 2. These two frequency feature maps extract information independently through intrafrequency update and communicate with each other through

interfrequency exchange. An output feature map is represented as $Y = \{Y^H, Y^L\}$, which can be defined as follows:

$$Y^H = f^{H \rightarrow H}(X^H) + f^{L \rightarrow H}(X^L), \quad (1)$$

$$Y^L = f^{L \rightarrow L}(X^L) + f^{H \rightarrow L}(X^H), \quad (2)$$

where $f^{H \rightarrow H}$ and $f^{L \rightarrow L}$ denote intraupdates, while $f^{L \rightarrow H}$ and $f^{H \rightarrow L}$ denote interexchanges. The intra- and interfrequency communications help to strengthen information propagation between channels, which brings potential improvements.

To compute the output feature map Y , a standard convolution kernel with weight W of $k \times k$ size is split into four parts $W = \{W^{H \rightarrow H}, W^{L \rightarrow H}, W^{L \rightarrow L}, W^{H \rightarrow L}\}$ via α shown in Figure 3. For the intrapart, a vanilla convolution is the only requirement, while for the interpart, resolution matching (i.e., downsampling or upsampling) should be performed before or after convolution. Here, we use average pooling for downsampling and nearest interpolation for upsampling, which is shown in the bottom left corner of Figure 1. Then, Equation (1) can be rewritten as

$$Y^H = f(X^H; W^{H \rightarrow H}) + \text{interpolate}(f(X^L; W^{L \rightarrow H}), 2), \quad (3)$$

$$Y^L = f(X^L; W^{L \rightarrow L}) + f(\text{pool}(X^H, 2); W^{H \rightarrow L}), \quad (4)$$

where $f(X; W)$ denotes a convolution function with weight parameter W , $\text{interpolate}(*, m)$ denotes upsampling by a

FIGURE 2: Representation of high- and low-frequency feature maps. The ratio α is a hyperparameter of the ACAM to control the number of channels of high- and low-frequency feature maps. Concretely, $c_{\text{low}} = \alpha \times c$ and $c_{\text{high}} = (1 - \alpha) \times c$, where c is the total number of channels.

FIGURE 3: The $k \times k$ sized convolution kernel of accuracy-complexity adjustment module (ACAM).

factor of m , and $\text{pool}(*, n)$ denotes downsampling with kernel size $n \times n$ and stride n .

In ACAM, two operation strategies are used to improve the performance. First, the upsampling is performed after convolution in Y^H , and downsampling is performed before convolution in Y^L . Such an order makes convolutions operate on smaller feature maps, which can save computations. Second, using the same size of $k \times k$ kernels on the low frequency is equivalent to enlarging the receptive field size in the original pixel space, which introduces additional multi-scale information.

The setting of α can be divided into three stages. In the first convolutional layer, we set $\alpha_{\text{in}} = 0$ and $\alpha_{\text{out}} = \alpha$, in which case X^L is disabled. This stage is usually used at the beginning of the network, where an original CT image is split into high and low frequency. The high frequency is obtained by a convolution function, and the low frequency is obtained through a sequential downsampling and convolution operation. In the last convolutional layer, we set $\alpha_{\text{in}} = \alpha$ and $\alpha_{\text{out}} = 0$, in which case Y^L is disabled. This stage is usually used at the end of the network, mapping the feature map to the prediction mask. The low frequency is restored to high frequency

through a sequential convolution and an upsampling operation, which is added to the high-frequency part to get the final result. In all the middle convolutional layers, we keep $\alpha_{\text{in}} = \alpha_{\text{out}} = \alpha$ in ACAMs. The intermediate feature maps are obtained by Equation (3).

There are three kinds of operations in the ACAM, which are convolution, pooling, and interpolation. Compared with the convolution, the FLOPs of the pooling and interpolation are negligible. Therefore, we only compute the FLOPs of the convolution. For each convolution operation in ACAM, namely, $f^{H \rightarrow H}$, $f^{L \rightarrow H}$, $f^{L \rightarrow L}$, and $f^{H \rightarrow L}$, the theoretical FLOPs can be obtained by

$$\begin{aligned} \text{FLOPs}(f^{H \rightarrow H}) &= h \times w \times k^2 \times (1 - \alpha)^2 \times c^2, \\ \text{FLOPs}(f^{L \rightarrow H}) &= \frac{h}{2} \times \frac{w}{2} \times k^2 \times (1 - \alpha) \times \alpha \times c^2, \\ \text{FLOPs}(f^{L \rightarrow L}) &= \frac{h}{2} \times \frac{w}{2} \times k^2 \times \alpha^2 \times c^2, \\ \text{FLOPs}(f^{H \rightarrow L}) &= \frac{h}{2} \times \frac{w}{2} \times k^2 \times \alpha \times (1 - \alpha) \times c^2. \end{aligned} \quad (5)$$

The final FLOPs include four subconvolutions, which are merged as follows:

$$\text{FLOPs} = \left(1 - \frac{3}{4}\alpha(2 - \alpha)\right) \times h \times w \times k^2 \times c^2. \quad (6)$$

Therefore, by controlling the rate of low-frequency α , ACAM can save $(3/4)\alpha(2 - \alpha)$ computations compared with a vanilla convolution.

2.2. Multiscale Aggregation Module. Targets in medical images usually have different sizes (e.g., spinal cord and lungs in this segmentation task). The size of a specific organ also varies in different slices. Multiscale information should be taken into consideration during training. Therefore, the MAM is added after the encoder to enrich the feature representation.

Formally, let $X \in \mathbb{R}^{c \times h \times w}$ be an input feature map, where $h \times w$ denotes the spatial resolution and c the number of channels. The MAM with M branches is proposed for X to integrate multiscale information. Taking $M = 4$ as an example, the module structure is shown in Figure 4. The basic 3×3 sized convolution kernels corresponding to different branches have different dilated rates. The MAM generates M feature maps in parallel, namely, $U = [U_1, U_2, \dots, U_M]$. Remember that the purpose of MAM is to adaptively adjust the receptive field sizes of neurons in the next layer with multiscale information from these branches. Therefore, an element-wise fusion operation needs to be done first to integrate all the learned feature maps:

$$U = \sum_{b=1}^M U_b. \quad (7)$$

FIGURE 4: Multiscale aggregation module. For simplicity, we keep the number of channels c of $[U_1, U_2, \dots, U_M]$ the same as X .

Then, the channel-wise statistics of U , expressed as a tensor $s \in \mathcal{R}^c$, can be built through a global average pooling to embed spatial context of each channel:

$$s = \frac{1}{h \times w} \sum_{i=1}^h \sum_{j=1}^w U(i, j). \quad (8)$$

Then, s is passed through a bottleneck made by a fully connected layer with weight $W_1 \in \mathcal{R}^{c \times d}$ and transformed into a compressed tensor $z \in \mathcal{R}^d$, where d is

$$d = \max\left(\frac{c}{r}, L\right). \quad (9)$$

The ratio r relates to the degree of compression, which helps to limit model complexity and assist feature generalization. L denotes a safe margin, which is used to prevent too much loss of channel-wise information caused by overcompression.

In order to separately calculate the attentions of M branches, z is split into M independent tensors $[a_1, a_2, \dots, a_M]$ through fully connected layers with weight $W_2 \in \mathcal{R}^{M \times d \times c}$, decoding back to the original size. Then, an activation function $\sigma(\cdot)$ across c channels is used to build the corresponding attention maps $[\sigma(a_1), \sigma(a_2), \dots, \sigma(a_M)]$, which is a softmax function in this case. These attention maps emphasize the important channels and ignore the less important ones in branches, acting as gates to constraint the information propagation. The final output Y is obtained through a

TABLE 2: The effect of α on the performance. We calculate the percentage of FLOPs and memory relative to the baseline U-Net (the first row).

α	DSC (%)	FLOPs (G)	Memory (10^6)
Baseline ($\alpha = 0$)	94.34	123.93	1439
0.125	95.43	102.09 (82%)	1313 (91%)
0.25	95.41	83.26 (67%)	1188 (83%)
0.5	95.32	54.28 (44%)	938 (65%)
0.75	95.09	36.84 (30%)	687 (47%)
0.875	94.88	32.45 (26%)	562 (39%)
1.0	93.52	30.98 (25%)	365 (25%)

summation over channel-wise products of these attention weights and feature maps:

$$Y = \sum_{i=1}^M a_i \times U_i. \quad (10)$$

There are four operations in one MAM, which are convolutions, global average poolings, fully connected layers, and softmax. To analyze the model complexity, we compute FLOPs of each operation individually. There are $M + 1$ fully connected layers, which are all performed between one-dimensional tensors. Their total FLOPs are at most $(M + 1) \times c^2$, depending on the ratio parameter r , which is negligible compared with convolutions. We also ignore the FLOPs of softmax operation, which is performed M times on one-dimensional tensors. Element-wise additions and multiplies are conducted $2(M - 1)$ and M times, respectively. The sum of their FLOPs is $(3M - 2) \times h \times w \times c$, also much smaller than that of convolutions. Therefore, the main additional complexity is brought by the convolution performed early in each branch. To reduce the computation cost of the MAM, we put it in a bottleneck layer and compress the number of channels through point-wise convolutions.

3. Results and Discussion

3.1. Dataset and Metrics. An in-house dataset was used to evaluate the proposed method, which is provided by the Institute of Cancer and Basic Medicine (ICBM), Chinese Academy of Sciences, and Cancer Hospital of the University of Chinese Academy of Sciences. Manually labelled masks of the left lung, right lung, heart, and spinal cord of 36 patients were defined by two experienced radiation oncologists. The CT scans have 512×512 pixels in resolution. The pixel spacing varies from 0.95 mm to 1.37 mm, and all thicknesses are 5 mm. We randomly shuffled the dataset and split it into three subsets: training set, validation set, and test set. The training set accounts for 70% of the entire data set. The validation set accounts for 10%, and the test set accounts for 20%. No preprocessing was performed before training. A public dataset called SegTHOR is also used to test our method, which can be found in CodaLab. There are four organs in this dataset, esophagus, heart, trachea, and aorta. The CT scans have 512×512 pixels in resolution. The pixel

FIGURE 5: The relationships between DSC, FLOPs, memory, and α . Since the actual values of FLOPs and memory correspond to different scales, we only plot their percentages relative to the baseline.

spacing varies from 0.90 mm to 1.37 mm. The thicknesses vary from 2 mm to 3.7 mm. Image data and labels of 40 patients can be downloaded to train the network. Image data of 20 patients can be tested offline and uploaded onto the website to obtain the test results (<https://competitions.codalab.org/competitions/21145>). In the following sections, except for special declarations, we default to using the in-house dataset for experiments.

The proposed method was implemented by using a deep learning framework called Pytorch and trained on a single NVIDIA GTX 1080 Ti GPU with 11 GB of memory. The code of our complete method can be found online <https://github.com/Jennsoo/UNet-Accuracy-Complexity.git>. The network was trained with a stochastic gradient descend optimizer with an initial learning rate 0.03, momentum 0.9, and weight decay 0.0005. Poly learning rate policy was adopted to further decrease the learning rate by a factor of $(1 - (\text{iter}/\text{total_iters}))^{0.9}$. We trained our model using crossentropy loss with 100 epochs in total. Weights were initialized according to MSRA [29], a zero-mean Gaussian distribution with variance $2/n$, in which n is the number of input elements. The Dice similarity coefficient (DSC) is our evaluation metric, which is defined as follows: where X is the ground truth and Y is the prediction.

3.2. Guidance for Adjusting Accuracy and Complexity. The hyperparameter α in ACAM is the main factor to balance the accuracy and model complexity. To evaluate how α affects the DSC-FLOP trade-off in the segmentation task, we conduct an experiment with $\alpha \in \{0.0, 0.125, 0.25, 0.5, 0.75, 0.875, 1.0\}$ on our in-house dataset. The corresponding mean DSC, FLOPs, and memory are listed in Table 2. In particular, $\alpha = 0$ relates to the baseline, which means no low frequency is separated out. We observe that ACAMs notably improve the segmentation DSC, while having less computation complexity and memory cost. Such boost can be attributed to the multifrequency processing in ACAMs, which

TABLE 3: The effect of the number of branches. We calculate the increased percentage of FLOPs and memory relative to the baseline U-Net (the first row).

#Branches (dilation)	DSC (%)	FLOPs (G)	Memory (10^6)
Baseline	94.34	123.93	1439
2 (1, 2)	94.54	125.41 (+1.19%)	1454 (+1.04%)
3 (1, 2, 4)	94.71	126.02 (+1.69%)	1457 (+1.25%)
4 (1, 2, 4, 8)	95.02	126.62 (+2.17%)	1460 (+1.46%)

brings an extra multiscale complement. We plot the mean DSC, FLOPs, and memory in Figure 5. From the figure, we can see that the DSC first significantly increases and then slowly declines, while FLOPs and memory decrease monotonically. The network reaches its best DSC at $\alpha = 0.125$, about 1% DSC improvement, while decreasing 18% complexity and 9% memory cost. For higher α , the DSC slowly goes down from the peak but is still better than the baseline. An intuitive explanation is that the spatial compression of low-frequency feature maps does not lead to serious information loss. This confirms the argumentation that feature maps in convolutional layers do have spatial redundancy. Through compressing the feature maps, computation and memory cost can be effectively saved without sacrificing accuracy. This experiment provides guidance for α selection. The DSC of $\alpha = 0.25$ is almost the same with the DSC of $\alpha = 0.125$, but the FLOPs and memory of $\alpha = 0.25$ take more advantages. Therefore, we set $\alpha = 0.25$ for all the ACAMs.

3.3. The Influence of the Number of Branches in Multiscale Aggregation Module. Here, we conduct an experiment on the in-house dataset to explore the best choice of branches of the MAM. Concretely, we use kernels with different dilated rates to generate feature maps with different receptive field sizes. The basic kernel size is 3×3 with dilated rate 1. For each new branch, the dilated rate of the kernel is multiplied

FIGURE 6: The relationships between DSC, FLOPs, memory, and the number of branches M .

TABLE 4: The effect of the number of branches. We calculate the increased percentage of FLOPs and memory relative to the baseline U-Net (the first row).

	L-lung	R-lung	Heart	Sp-Co	Mean
Baseline	96.86	96.88	94.40	89.20	94.34
Baseline+MAM	97.53	97.27	94.97	90.31	95.02
Baseline+ACAMs	97.63	97.43	95.20	91.36	95.41
Baseline+ACAMs+MAM	97.76	97.52	95.27	92.24	95.70

FIGURE 7: The DSC of each single organ of the ablation experiment.

by a power of two. Note that a standard MAM requires at least two branches. In this paper, three designs of branches are presented, and its corresponding results are recorded in Table 3. It is worth mentioning that we have tried to add the MAM in every convolutional layer as a parallel module,

but the results were not satisfactory. We plot the mean DSC, FLOPs, and memory in Figure 6. No preprocessing was performed before training.

We observe that DSC is positively correlated with FLOPs and memory. The MAM with more branches performs

FIGURE 8: Qualitative results of the ablation experiment. The interesting areas on four images are framed out with white boxes.

better. A possible explanation is that more branches bring more multiscale information, which helps to capture a richer global context. Another observation is that the MAM is lightweight with negligible extra model complexity. Even if we use four branches, it will not consume much extra computation and memory resources. In all the following experiments, we use dilation of (1, 2, 4, 8) for the MAM.

3.4. Ablation Experiment. We investigate the performance of aggregating ACAMs and MAM in the proposed method. We list all the DSCs of every single organ of the in-house dataset in Table 4 and plot it in Figure 7. Aside from the

conclusion in Sections 3.2 and 3.3 that both ACAMs and MAM can boost the performance of the baseline, we can get two more observations when looking into the details. First, the results obtained by jointly applying ACAMs and MAM are better than using them alone, which indicates that there is a certain complementary relationship between the two modules. Figure 8 shows qualitative results of four images, validating this observation. Second, the DSCs of small organs get more improvement, especially the spinal cord. Our method improves the DSCs of the lungs and heart by about 0.8%, while increasing the DSC of the spinal cord up to 3%.

TABLE 5: Comparison results of DSC (%) of the proposed method and the state-of-the-art methods on the in-house dataset.

	L-lung	R-lung	Heart	Sp-Co	Mean
U-Net [4]	96.86	96.88	94.40	89.20	94.34
PSPNet [7]	96.83	96.70	94.12	83.70	92.84
DeepLab v3+ [11]	97.25	97.01	94.10	89.07	94.36
DANet [12]	93.24	93.74	85.68	59.76	83.11
Attention U-Net [5]	97.24	97.10	94.36	91.09	94.95
Nested U-Net [6]	96.69	96.76	94.51	89.91	94.47
Ours	97.76	97.52	95.27	92.24	95.70

FIGURE 9: The DSC of each single organ.

4. Comparison Experiment

Table 5 and Figure 9 present the comparison results on the in-house dataset with several state-of-the-art methods, which are U-Net [4], PSPNet [7], DeepLab v3+ [11], DANet [12], Attention U-Net [5], and Nested U-Net [6]. The implementation of these methods can be found online (U-Net <https://github.com/milesial/Pytorch-UNet>, PSPNet <https://github.com/hszhao/PSPNet>, DeepLab v3+ <https://github.com/jfzhang95/pytorch-deeplab-xception>, DANet <https://github.com/junfu1115/DANet>, and Attention U-Net and Nested U-Net <https://github.com/bigmb/Unet-Segmentation-Pytorch-Nest-of-Unets>). Although DANet achieves state-of-the-art performance on the Cityscape dataset, it is not applicable to our dataset. From the table, we can see that the DSCs of the proposed method on four organs are the best among state-of-the-art methods. This can be attributed to the combination of the ACAMs and MAM, which strengthens the feature extraction capabilities. We compare our method on each individual organ with other methods, which performs the best on it. Our network additionally increases the DSC of the left lung by 0.51% over DeepLab v3+, the right lung by 0.42% over Attention U-Net, the heart by 0.76% over Nested U-Net, and the spinal cord by 1.15% over Attention U-Net.

Figure 10 shows qualitative results of four methods, which are DeepLab v3+, Attention U-Net, Nested U-Net,

and our network. In the first image, the left lung looks very different from the common mirrored “C” shape. DeepLab v3+, Attention U-Net, and Nested U-Net are confused that they fail to recognize it, while our method gets a more accurate segmentation. In the second image, there is a thinner structure at the top of the left lung. Attention U-Net and Nested U-Net both ignore this structure and disrupt the continuity of the left lung during segmentation. DeepLab v3+ and our method perform well in such case. This may be due to multiscale modules applied in both methods, which is ASPP in the DeepLab v3+ and MAM in our method. In the third image, there is a noticeable small but deep depression in the left lung. Attention U-Net totally omits it and fills in this depression. DeepLab v3+ and Nested U-Net seem to observe this structure and “dig a hole” on the left lung. Our method performs the best and delineates the shape well. This further indicates that our method not only improves the segmentation of small targets but also is effective for the subtle structures. In addition, we randomly select a few cases and display the visualization results in Figure 11. In all these cases, our method has achieved satisfactory results, but there are still some defects on complex boundaries.

We observed that DANet [12] does not fit our dataset, even though it has achieved advanced results on other datasets. One possible reason is that the targets to segment in our cases have different sizes. The spinal cord is extremely small compared to other organs and background. Both the

FIGURE 10: Qualitative results of (a) Ground Truth, (b) DeepLab v3+, (c) Attention U-Net, (d) Nested U-Net, and (e) ours.

PSPNet and DeepLab v3+ in our comparison experiment introduce pyramid pooling modules. The U-Net-based methods also introduce multiscale components due to the skip connections between different levels. However, DANet only considers the long-range dependencies between pixels but ignores the target scale information, which may become a potential factor for small object segmentation errors.

In the SegTHOR dataset, our method ranks in 4 out of 19 methods (until 6/15/2020). The esophagus, trachea, and

aorta are small organs, while the heart is a relatively big organ. Except for the esophagus, our method generally achieves competitive results, which shows the effectiveness and generalization of our method. All the methods on the ranking perform the worst on the segmentation of the esophagus. We analyze the dataset and find that the esophagus is more special compared with the others, making it more challenging to segment. The esophagus is small in size and variable in shape. The boundary of the esophagus is blurred

FIGURE 11: Visualization results of four different cases.

TABLE 6: FLOPs (G) and memory (10^6) are calculated on CT image with size of 512×512 .

	FLOPs (G)	Memory (10^6)
U-Net [4]	123.93	1439
PSPNet [7]	262.13	2525
DANet [12]	275.42	2579
Attention U-Net [5]	266.25	2402
Nested U-Net [6]	552.01	2715
Ours	89.77	1221

due to its low contrast displayed in CT scans. Our method is not designed for such problems and is not robust enough to deal with such situations.

Model Complexity Analysis. Since the model complexity plays an important role in the design of our network, we compare the FLOPs and memory of our method with that of other state-of-the-art methods in Table 6. It is obvious that our method consumes the least computation and memory resources. The memory usage of our method is reduced nearly by 50%. The FLOPs of our method are reduced by an average of 67% except Nested U-Net. Nested U-Net has much more FLOPs due to its denser connections between convolutional layers. The reason is that the ACAMs embedded in our method compress the spatial redundancy of the feature maps, which can significantly reduce the computation cost.

5. Conclusions

In this paper, we propose an end-to-end method to segment multiple organs in CT scans. Benefiting from the compression of spatial redundancy in the applied accuracy-

complexity adjustment module, the model complexity can be reduced, while achieving higher accuracy. We also present the experimental results to provide guidance for the hyperparameter selection. A nonlinear multiscale aggregation module is added after the encoder to further enrich feature representation. The combination of these two modules in the proposed method achieves higher DSC and lower complexity than several state-of-the-art methods. Such an idea provides a direction of improving performance of high-complexity models like 3D U-Net, which will be our future work.

Data Availability

The in-house dataset used to support the findings of this study was supplied by the Chinese Academy of Sciences under license and so cannot be made freely available. Requests for access to these data should be made to zhiqiangtian@xjtu.edu.cn.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by NSFC under grant No. 61876148. This work was also supported in part by the Fundamental Research Funds for the Central Universities No. XJJ2018254 and China Postdoctoral Science Foundation No. 2018M631164.

References

- [1] R. Trullo, C. Petitjean, B. Dubray, and S. Ruan, "Multiorgan segmentation using distance-aware adversarial networks," *Journal of Medical Imaging*, vol. 6, no. 1, article 014001, 2019.
- [2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, Boston, MA, USA, 2015.
- [3] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: a deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
- [4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Munich, Germany, 2015.
- [5] O. Oktay, J. Schlemper, L. Le Folgoc et al., "Attention u-net: learning where to look for the pancreas," 2018, <https://arxiv.org/abs/1804.03999>.
- [6] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: a nested u-net architecture for medical image segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pp. 3–11, Springer, 2018.
- [7] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2881–2890, Honolulu, HI, USA, 2017.
- [8] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," 2014, <https://arxiv.org/abs/1412.7062>.
- [9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.
- [10] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017, <https://arxiv.org/abs/1706.05587>.
- [11] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 801–818, Munich, Germany, 2018.
- [12] J. Fu, J. Liu, H. Tian et al., "Dual attention network for scene segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3146–3154, Long Beach, CA, USA, 2019.
- [13] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 510–519, Long Beach, CA, USA, 2019.
- [14] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, Salt Lake City, UT, USA, 2018.
- [15] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," 2016, <https://arxiv.org/abs/1611.06440>.
- [16] A. G. Howard, M. Zhu, B. Chen et al., "Mobilenets: efficient convolutional neural networks for mobile vision applications," 2017, <https://arxiv.org/abs/1704.04861>.
- [17] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, Salt Lake City, UT, USA, 2018.
- [18] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: an extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, Salt Lake City, UT, USA, 2018.
- [19] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: practical guidelines for efficient cnn architecture design," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 116–131, Munich, Germany, 2018.
- [20] Y. Chen, H. Fan, B. Xu et al., "Drop an octave: reducing spatial redundancy in convolutional neural networks with octave convolution," 2019, <https://arxiv.org/abs/1904.05049>.
- [21] R. Brügger, C. F. Baumgartner, and E. Konukoglu, "A partially reversible u-net for memory-efficient volumetric image segmentation," 2019, <https://arxiv.org/abs/1906.06148>.
- [22] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: fully convolutional neural networks for volumetric medical image segmentation," in *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 565–571, Stanford, CA, USA, 2016.
- [23] F. Isensee, P. Kickingereder, W. Wick, M. Bendszus, and K. H. Maier-Hein, "No new-net," in *International MICCAI Brainlesion Workshop*, pp. 234–244, Granada, Spain, 2018.
- [24] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," 2015, <https://arxiv.org/abs/1502.03167>.
- [25] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, <https://arxiv.org/abs/1605.07146>.
- [26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, Las Vegas, NV, USA, 2016.
- [27] Y. Huang, Y. Cheng, A. Bapna et al., "Gpipe: efficient training of giant neural networks using pipeline parallelism," in *Advances in Neural Information Processing Systems*, pp. 103–112, Curran Associates, Inc., 2019.
- [28] M. Tan and Q. V. Le, "Efficientnet: rethinking model scaling for convolutional neural networks," 2019, <https://arxiv.org/abs/1905.11946>.
- [29] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, Santiago, Chile, 2015.
- [30] W. Owadally and J. Staffurth, "Principles of cancer treatment by radiotherapy," *Surgery (Oxford)*, vol. 33, no. 3, pp. 127–130, 2015.
- [31] A. L. Grosu, L. D. Sprague, and M. Molls, "Definition of target volume and organs at risk. Biological Target Volume," in *New Technologies in Radiation Oncology*, Springer, Berlin Heidelberg, 2006.

- [32] S. Scoccianti, B. Detti, D. Gadda et al., “Organs at risk in the brain and their dose-constraints in adults and in children: a radiation oncologist's guide for delineation in everyday practice,” *Radiotherapy and Oncology*, vol. 114, no. 2, pp. 230–238, 2015.
- [33] L. D. van Harten, J. M. H. Niothout, J. J. C. Verhoeff, J. M. Wolterink, and I. Išgum, “Automatic segmentation of organs at risk in thoracic CT scans by combining 2D and 3D convolutional neural networks,” in *Proceedings of the 2019 Challenge on Segmentation of THoracic Organs at Risk in CT Images (SegTHOR2019) co-located with the 16th International Symposium on Biomedical Imaging (ISBI)*, Venice, Italy, 2019.