Chapter 10
Anisotropic Hybrid Network for Cross-Dimension Transferable Feature Learning in 3D Medical Images

Siqi Liu, Daguang Xu, S. Kevin Zhou, Sasa Grbic, Weidong Cai and Dorin Comaniciu

Abstract While deep convolutional neural networks (CNN) have been successfully applied for 2D image analysis, it is still challenging to apply them to 3D anisotropic volumes, especially when the within-slice resolution is much higher than the between-slice resolution and when the amount of 3D volumes is relatively small. On one hand, direct learning of CNN with 3D convolution kernels suffers from the lack of data and likely ends up with poor generalization; insufficient GPU memory limits the model size or representational power. On the other hand, applying 2D CNN with generalizable features to 2D slices ignores between-slice information. Coupling 2D network with LSTM to further handle the between-slice information is not optimal due to the difficulty in LSTM learning. To overcome the above challenges, 3D anisotropic hybrid network (AH-Net) transfers convolutional features learned from 2D images to 3D anisotropic volumes. Such a transfer inherits the desired strong generalization capability for within-slice information while naturally exploiting between-slice information for more effective modeling. We show the effectiveness of the 3D AH-Net on two example medical image analysis applications, namely, lesion detection from a digital breast tomosynthesis volume, and liver, and liver tumor segmentation from a computed tomography volume.

S. Liu (✉) · S. Grbic · D. Comaniciu
Digital Services, Digital Technology and Innovation, Siemens Healthineers, Princeton, NJ, USA
e-mail: lsqshr@gmail.com

D. Xu
NVIDIA Corporation, Santa Clara, USA

S. K. Zhou
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China

W. Cai
School of Information Technologies, University of Sydney, Darlington, NSW, Australia

© Springer Nature Switzerland AG 2019

L. Lu et al. (eds.), *Deep Learning and Convolutional Neural Networks for Medical Imaging and Clinical Informatics*, Advances in Computer Vision and Pattern Recognition,
https://doi.org/10.1007/978-3-030-13969-8_10

10.1 Introduction

3D volumetric images (or volumes) are widely used for clinical diagnosis, surgical planning, and biomedical research. The 3D context information provided by such volumetric images are important for visualizing and analyzing the object of interest. However, given the added dimension, it is more time-consuming and sometimes harder to interpret 3D volumes than 2D images by machines. Many previous studies use convolutional neural networks (CNN) to extract the representation of structural patterns of interests in human or animal body tissues.

Due to the special imaging settings, many imaging modalities come with anisotropic voxels, meaning not all the three dimensions have equal resolutions. For example, in the 3D volumes of digital breast tomosynthesis (DBT), and sometimes computed tomography (CT), the image resolution in xy plane/slice (or within-slice resolution) is more than ten times higher than that of the z resolution (or between-slice resolution). Thus, the xy slices preserve much more information than the z dimension. In DBT images, only the spatial information within the xy-plane can be guaranteed. However, the 3D context between xy slices, even with a slight misalignment, still carries meaningful information for analysis (Fig. 10.1).

Directly applying 3D CNN to such images remains a challenging task due to the following reasons: (1) It may be hard for a small $3 \times 3 \times 3$ kernel to learn useful features from anisotropic voxels, because of the different information density along each dimension. (2) Theoretically more features are needed in 3D networks compared to 2D networks. The capability of 3D networks is bounded by the GPU memory, constraining both the width and depth of the networks. (3) Unlike 2D computer vision tasks which nowadays can make use of the backbone networks pretrained

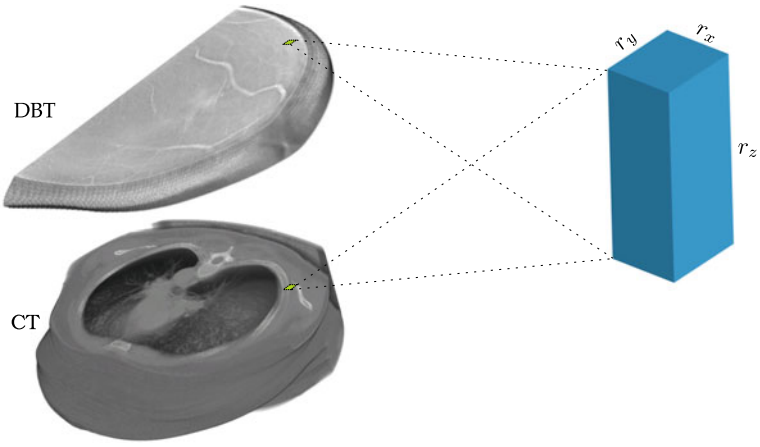

Fig. 10.1 The example anisotropic volumes of DBT and CT are shown in the left column. Such volumes contain voxels with much higher within-slice resolution $r_x \times r_y$ than the between-slice resolution r_z

using millions of 2D images [21], 3D tasks mostly have to train from scratch, and hence suffer from the lack of large 3D datasets. In addition, the high data variations make the 3D networks harder to be trained. Also, 3D CNNs trained on such small image datasets with relatively small 3D context are hard to generalize to unseen data.

Besides the traditional 3D networks built with $1 \times 1 \times 1$ and $3 \times 3 \times 3$ kernels, there are other methods for learning representations from anisotropic voxels. Some studies process 2D slices separately with 2D networks [14]. To make better use of the 3D context, more than one image slice is used as the input for 2D networks [12, 24]. The 2D slices can also be viewed sequentially by combining a fully convolutional network (FCN) architecture with convolutional LSTM to view the adjacent image slices as a time series to distill the 3D context from a sequence of abstracted 2D context [5]. There are also a few studies using anisotropic convolutional kernels to distribute more learning capability on the xy-plane than on the z-axis [2, 11, 22].

In this chapter, we present the 3D anisotropic hybrid network (AH-Net) [15] to learn informative features from images with anisotropic resolution. To obtain the 3D AH-Net, we first train a 2D fully convolutional ResNet [17] which is initialized with pretrained weights and uses multiple 2D image slices as inputs. The feature encoder of such a 2D network is then transformed into a 3D network by extending the 2D kernel with one added dimension. Then we add a feature decoder subnetwork to extract the 3D context. The feature decoder consists of anisotropic convolutional blocks with $3 \times 3 \times 1$ and $1 \times 1 \times 3$ convolutions. Different anisotropic convolutional blocks are combined with dense connections [8]. Similar to U-Net [20], we use skip connections between the feature encoder and the decoder. A pyramid volumetric pooling module [25] is stacked at the end of the network before the final output layer for extracting multiscale features. Since AH-Net can make use of 2D networks pretrained with large 2D general image datasets such as ImageNet [21], it is easier to train as well as to generalize. The anisotropic convolutional blocks enable it to exploit the 3D context. With end-to-end inference as a 3D network, AH-Net runs much faster than the conventional multichannel 2D networks regarding the GPU time required for processing each 3D volume.

10.2 Related Work

It is hard for conventional 3D neural networks with isotropic $3 \times 3 \times 3$ kernels to extract robust representations from 3D volumes with anisotropic resolution. The most intuitive approach is to resample the images to isotropic resolutions [16]. This would work when the difference between the three dimensions are small, and the spatial information between different slices is accurate. When the z resolution is much smaller than the xy resolution, the majority of voxels added by image resampling are redundant, thus introducing unnecessary extra computational cost. It may also result in a loss of information if downsampling happens in the xy-direction.

Instead of using 3D networks, some studies deal with the voxel anisotropy using 2D networks. DeepEM3D-Net [24] has only two 3D convolution layers to integrate

3D information in the early stages and performs 2D convolution for the rest of the following layers in an FCN. The input to DeepEM3D-Net is a stack of 2D image slices. The resultant 3D segmentation is obtained by concatenating the 2D output slices. HDenseNet [12] applies 2D networks on all image slices at first. Then a 3D DenseUNet is applied to the concatenated 3D output volume to obtain the final result. Different from our proposed network, HDenseNet does not have shared convolutions between the 2D and 3D networks. Also, we use anisotropic 3D convolutional blocks to replace the isotropic 3D convolutions.

A bidirectional convolutional LSTM (BDC-LSTM) and an FCN model are combined to view slices as a time series [5]. BDC-LSTM is trained to exploit the 3D contexts by applying a series of 2D convolutions on the xy-plane in a recurrent fashion to interpret 3D contexts while propagating contextual information in the z-direction. The FCN model is used for extracting the initial 2D feature maps which are used as the inputs to BDC-LSTM. The final output is obtained from the BDC-LSTM model with a softmax layer. Though the idea of fusing the 2D features to maintain the between-slice consistency is similar to our proposed method, we believe this can be achieved with stacked anisotropic convolution blocks, which are easier to train and to generalize than the convolutional LSTM.

Some studies use 3D convolutional kernels with anisotropic sizes to distribute more learning capability to the xy-plane. For example, $9 \times 9 \times 5$ convolutions are used in [2]. However, large convolution kernels would bring higher computational cost. Two more recent studies [11, 18, 22] use small kernels to simulate the large anisotropic kernels. The convolution modules in [11] starts with a $3 \times 1 \times 1$ convolution, followed by two $3 \times 3 \times 3$ convolutions. Similar to our work, all the isotropic convolutions are replaced by $3 \times 3 \times 1$ and $1 \times 1 \times 3$ convolutions in [18, 22]. Several possible designs of combining the $3 \times 3 \times 1$ and $1 \times 1 \times 3$ kernels are discussed in a recent paper [18] that focuses on video learning. Our network is different to the ones in [18, 22] since we use the anisotropic 3D convolutions only in the feature decoder while the encoder is locked with pretrained weights transferred from a 2D network. It allows the proposed AH-Net to use any 2D fully convolutional networks pretrained on large-scale datasets for initializing the encoder network. In [23], the authors show that the network with pretrained network could be significantly helpful to train 3D models for volumetric segmentation.

10.3 Anisotropic Hybrid Network

The AH-Net consists of a feature encoder and a feature decoder. The encoder, transformed from a 2D network, is designed for extracting the deep representations from 2D slices with high resolution. The decoder built with densely connected blocks of anisotropic convolutions is responsible for exploiting the 3D context and maintaining the between-slice consistency. The network training is performed in two stages: the encoder is learned, then the 3D decoder is added and fine-tuned with the encoder parameters locked. To perform end-to-end hard-voxel mining, we use the focal loss (FL) originally designed for object detection [13].

10.3.1 Learning a Multichannel 2D Feature Encoder

We train a 2D multichannel global convolutional network (MC-GCN) similar to the architecture proposed in [17] to extract the 2D within-slice features at different resolutions, as shown in Fig. 10.2. In this chapter, we choose the ResNet50 model [7] as the backbone network which is initialized by pretraining with the ImageNet images [21], although other pretrained networks would work similarly. The network is then fine-tuned with 2D image slices extracted from the 3D volumes. The input

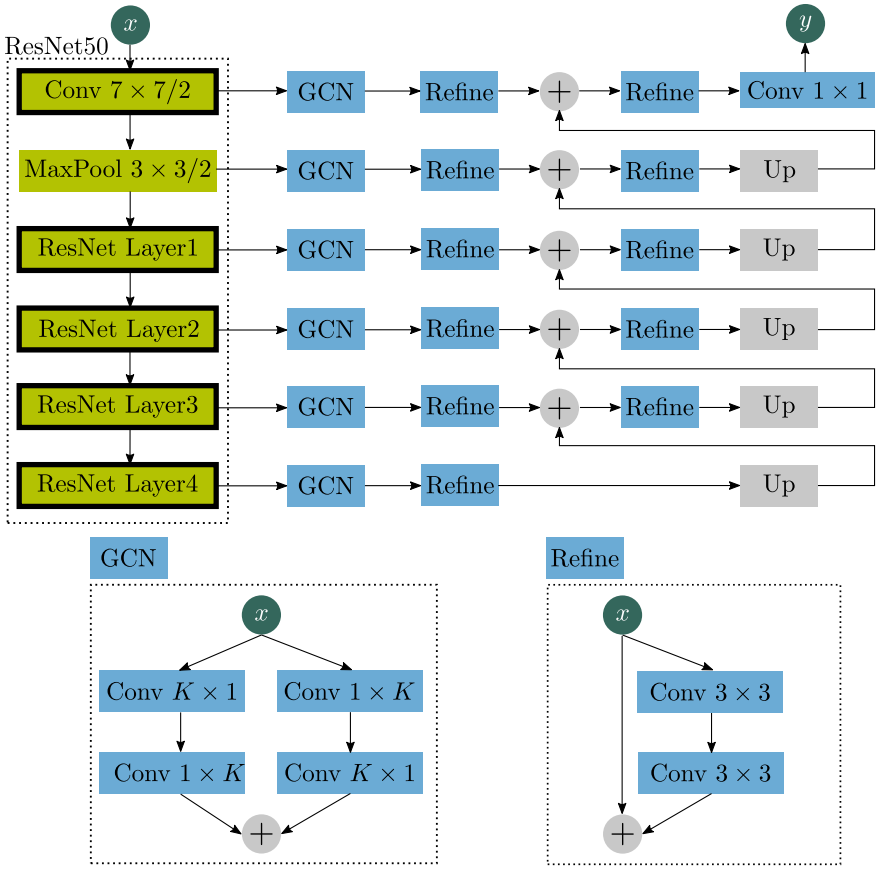

Fig. 10.2 The network architecture for pretraining the 2D encoder network multichannel global convolutional neural network (MC-GCN). The ResNet50 is used as the backbone network, initialized with ImageNet images. The global convolutional network modules and refinement modules [17] are added to the encoder network to increase the receptive field during the pretraining as well as to increase the output response map to the original resolution. Conv $K \times K/S$ represents a convolution layer with the kernel size K and the stride size S in each dimension. The upsampling module (Up) consists of a Conv 1×1 projection layer and a bilinear upsampling layer

to this network is three neighboring slices (treated as RGB channels). Thus, the entire architecture of the ResNet50 remains unchanged. The multichannel 2D input could enable the 2D network to fuse the between-slice context at an early stage. A decoder is added to accompany the encoder to upscale the response map to the original resolution. We choose the decoder architecture with the global convolutional networks (GCN) and refinement blocks [17]. The GCN module simulates a large $K \times K$ convolutional kernel by decomposing it into two 1D kernels ($1 \times K$ and $K \times 1$). Two branches containing the 1D kernels permuted in different orders are merged by summation. The output of each GCN module contains the same number of output maps as the final outputs. The large kernels simulated by GCNs ensure that the network has a large receptive field at each feature resolution. Each refinement block contains two 3×3 convolutions with a ReLU activation in the middle. The input of each refinement block is also added to its output to form a residual connection. At the end of each encoder resolution level, the features are fed into GCN modules with the kernel sizes of 63, 31, 15, 9, 7, 5, respectively. The output features are fed into a refinement block and summed with the features upsampled from a lower resolution level. The summed features are fed into another refinement block and upsampled with a 1×1 convolution and a bilinear upsampling layer. The final output has the same resolution as the image input. The decoder has only a small number of parameters with little computational cost. The lightweight decoder makes the encoder features easier to be transferred to the 3D AH-Net since majority of the feature learning relies on the encoder network.

10.3.2 Transferring the Learned 2D Net to 3D AH-Net

The architecture of the proposed 3D anisotropic hybrid network (AH-Net) is shown in Fig. 10.3. After the 2D MC-GCN network converges, we extract the parameters of its encoder and transfer them to the corresponding encoder layers of AH-Net. The decoder part of the 2D MC-GCN is discarded and instead, we design a new decoder for the AH-Net that consists of multiple levels of densely connected blocks, followed by a pyramid volumetric pooling module. The parameters of the new decoder are randomly initialized. The input and output of AH-Net are now 3D patches, similar to other conventional 3D CNN. The transformation of convolution tensors from 2D to 3D is illustrated in Fig. 10.4, which aims to perform 2D convolutions on 3D volumes slice by slice in the encoder part of AH-Net.

10.3.2.1 Notations

A 2D convolutional tensor is denoted by $T^i_{n\times m\times h\times w}$, where n, m, h, and w, respectively, represent the number of output channels, the number of input channels, the height, and width of the ith convolution layer. Similarly, a 3D weight tensor is denoted by $T^i_{n\times m\times h\times w\times d}$ where d is the filter depth. We use $P^{(b,a,c,d)}(T_{a\times b\times c\times d})$ to denote

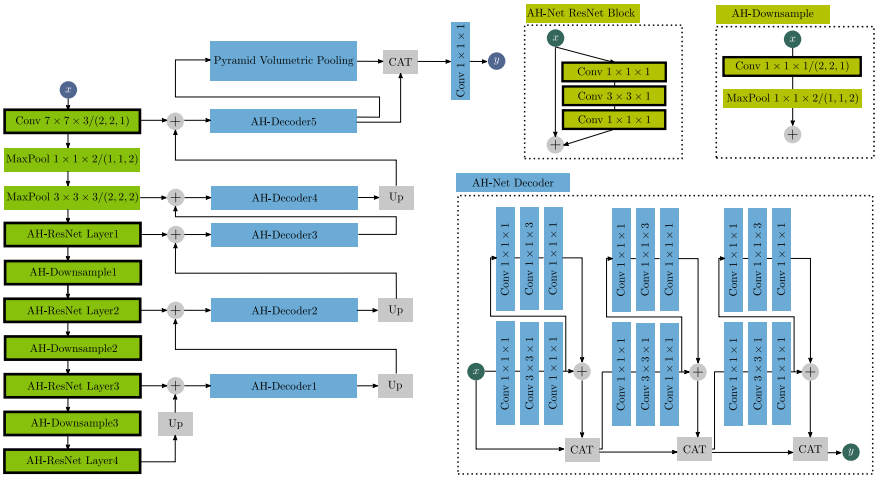

Fig. 10.3 The architecture of 3D AH-Net. The feature encoder with AH-ResNet blocks is transferred from the pretrained 2D network with $1 \times 1 \times 1$ and $3 \times 3 \times 1$ convolutions. The features are then processed with the AH-Net decoders which are designed with $3 \times 3 \times 1$ and $1 \times 1 \times 3$ convolutional blocks. Feature summation is used instead of concatenation as in [4] to support more feature maps with less memory consumption. The pyramid pooling [25] is used for extracting the multiscale feature responses. We hide the batch normalization [9] and ReLu layers for brevity. The weights of the blocks with black borders are transformed from the 2D MC-GCN

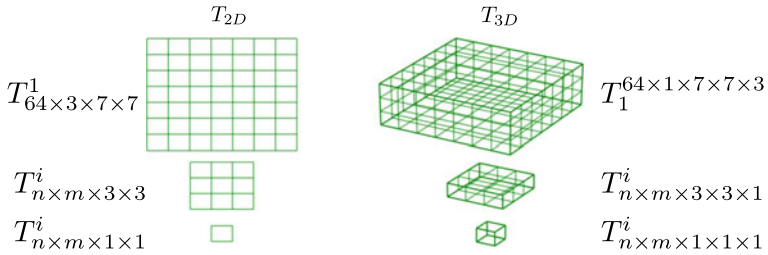

Fig. 10.4 Transforming the 2D convolutional weight tensor T^{2D} to 3D T^{3D}, where m and n are the number of features and channels of a layer, respectively. The first layer weight tensor $T^1_{64\times3\times7\times7}$ is transformed to $T^1_{64\times1\times7\times7\times3}$. The other convolutional kernels are transformed by adding an extra dimension

the dimension permutation of a tensor $T_{a\times b\times c\times d}$, resulting in a new tensor $T_{b\times a\times c\times d}$ with the first and second dimensions switched. $P^{(a,*,b,c,d)}(T_{a\times b\times c\times d})$ adds an identity dimension between the first and second dimensions of the tensor $T_{a\times b\times c\times d}$ and gives $T_{a\times1\times b\times c\times d}$. We define a convolutional layer as Conv $K_x \times K_y \times K_z/(S_x, S_y, S_z)$, where K_x, K_y, and K_z are the kernel sizes; S_x, S_y, and S_z are the stride step size in each direction. Max pooling layers are denoted by MaxPool $K_x \times K_y \times K_z/(S_x, S_y, S_z)$. The stride is omitted when a layer has a stride size of one in all dimensions.

10.3.2.2 Input Layer Transform

The input layer of the 2D MC-GCN contains a convolutional weight tensor $T^1_{64\times3\times7\times7}$ inherited from its ResNet50 backbone network. The 2D convolutional tensor $T^1_{64\times3\times7\times7}$ is transformed into 3D as

$$P^{(1,*,3,4,2)}(T^1_{64\times3\times7\times7}) = T^1_{64\times1\times7\times7\times3} \tag{10.1}$$

in order to form a 3D convolution kernel that convolves three neighboring slices. To keep the output consistent with the 2D network, we only apply stride 2 convolutions on the xy-plane and stride 1 on the third dimension. This results in the input layer Conv $7 \times 7 \times 3/(2, 2, 1)$. To downsample the z-dimension, we use a MaxPool $1 \times 1 \times 2/(1, 1, 2)$ to fuse every pair of the neighboring slices. An additional MaxPool $3 \times 3 \times 3/(2, 2, 2)$ is used to keep the feature resolution consistent with the 2D network.

10.3.2.3 ResNet Block Transform

All the 2D convolutional tensors $T^i_{n\times m\times1\times1}$ and $T^i_{n\times m\times3\times3}$ in the ResNet50 encoder are transformed as

$$P^{(1,2,3,4,*)}(T^i_{n\times m\times1\times1}) = T^i_{n\times m\times1\times1\times1} \tag{10.2}$$

and

$$P^{(1,2,3,4,*)}(T^i_{n\times m\times3\times3}) = T^i_{n\times m\times3\times3\times1}. \tag{10.3}$$

In this way, all the ResNet Conv $3 \times 3 \times 1$ blocks as shown in Fig. 10.3 only perform 2D slice-wise convolutions on the 3D volume within the xy-plane. The original downsampling between ResNet blocks is performed with Conv $1 \times 1/(2, 2)$. However, in a 3D volume, a Conv $1 \times 1 \times 1/(2, 2, 2)$ skips a slice for every step on the z-dimension. This would miss important information when the image only has a small number of slices along the z-dimension, especially for detection tasks. We therefore use a Conv $1 \times 1 \times 1/(2, 2, 1)$ followed by a MaxPool $1 \times 1 \times 2/(1, 1, 2)$ to downsample the 3D feature maps between the ResNet blocks as shown in the AH-Downsample block in Fig. 10.3. This MaxPooling simply takes the maximum response along the z-direction between two neighboring slices. Unlike the previous studies that avoided downsampling along the z-direction [11], we find it important for allowing the use of large and deep networks on 3D data with limited GPU memory.

10.3.3 Anisotropic Hybrid Decoder

Accompanying to the transformed encoder, an anisotropic 3D decoder subnetwork is added to exploit the 3D anisotropic image context. In the decoder, anisotropic convolutional blocks with Conv $1 \times 1 \times 1$, Conv $3 \times 3 \times 1$, and Conv $1 \times 1 \times 3$ are used. The features are passed into an xy bottleneck block at first with a Conv $3 \times 3 \times 1$ surrounded by two layers of Conv $1 \times 1 \times 1$. The output is then forwarded to another bottleneck block with a Conv $1 \times 1 \times 3$ in the middle and summed with itself before forwarding to the next block. This anisotropic convolution block decomposes a 3D convolution into 2D and 1D convolutions. It receives the inputs from the previous layers using a 2D convolution at first, preserving the detailed 2D features. Conv $1 \times 1 \times 3$ mainly fuses the within-slice features to keep the z-dimension output consistent.

Three anisotropic convolutional blocks are connected as the densely connected neural network [8] using feature concatenation for each resolution of encoded features. Similar to LinkNet [4], the features received from each resolution of the encoder are first projected to match the number of features of the higher encoder feature resolution using a Conv $1 \times 1 \times 1$. They are then upsampled using the 3D tri-linear interpolation and summed with the encoder features from a higher resolution. The summed features are forwarded to the decoder blocks in the next resolution.

At the end of the decoder network, we add a pyramid volumetric pooling module [25] to obtain multi-scaled features. The output features of the last decoder block are first downsampled using four different Maxpooling layers, namely, MaxPool $64 \times 64 \times 1$, MaxPool $32 \times 32 \times 1$, MaxPool $16 \times 16 \times 1$, and MaxPool $8 \times 8 \times 1$ to obtain a feature map pyramid. Conv $1 \times 1 \times 1$ layers are used to project each resolution in the feature pyramid to a single response channel. The response channels are then interpolated to the original size and concatenated with the features before downsampling. The final outputs are obtained by applying a Conv $1 \times 1 \times 1$ projection layer on the concatenated features.

Training AH-Net using the same learning rate on both the pretrained encoder and the randomly initialized decoder would make the network difficult to optimize. To train the 3D AH-Net, all the transferred parameters are locked at first. Only the decoder parameters are fine-tuned in the optimization. All the parameters can be then fine-tuned altogether afterward to the entire AH-Net jointly. Though it is optional to unlock all the parameters for fine-tuning afterward, we did not observe better performance.

10.4 Experimental Results

To demonstrate the efficacy and efficiency of the proposed 3D AH-net, we conduct two experiments, namely, lesion detection from a digital breast tomosynthesis (DBT) volume and liver tumor segmentation from a computed tomography (CT) volume. We

use ADAM [10] to optimize all the compared networks with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We use the initial learning-rate 0.0005 to fine-tune the 2D multichannel GCN. Then, the learning rate is increased to 0.001 to fine-tune the AH-Net after the 2D network is transferred. We find that 3D networks need a larger learning-rate to converge within a reasonable amount of time. All the networks are implemented in Pytorch (http://pytorch.org).

10.4.1 Breast Lesion Detection from DBT

We use an in-house database containing 2809 3D DBT volumes acquired from 12 different sites globally. DBT is an advanced form of mammography, which uses low-dose X-rays to image the breast. Different from 2D mammography that superimposes 3D information into one 2D image, DBT creates 3D pictures of the breast tissue, and hence allows radiologists to read these pictures and detect breast cancer more easily, especially in dense breast tissues. The xy-plane of DBT images has a high spatial resolution of 0.085 mm $\times$ 0.085 mm which is much larger than the z-dimension of 1 mm. The structures in the z-dimension are not only is compressed during the imaging process, but the 3D volumetric information also has large variations due to imaging artifacts.

We have experienced radiologists annotate and validate the lesions in DBT volumes, which might contain zero to several lesions. Each lesion is approximately annotated with a 3D bounding box. To train the proposed networks as lesion detection networks, we generate 3D multivariant Gaussian heatmaps that have the same sizes as the original images as

$$f(\mathbf{p}) = \sum_{\mu_\mathbf{i}, \Sigma_\mathbf{i}} \frac{\mathbf{exp}(-\frac{\mathbf{1}}{\mathbf{2}}(\mathbf{p} - \mu_\mathbf{i})^\mathbf{T} \Sigma_\mathbf{i}(\mathbf{p} - \mu_\mathbf{i}))}{\sqrt{\mathbf{det}(\mathbf{2}\pi\, \Sigma_\mathbf{i})}}, \tag{10.4}$$

where $\mathbf{p}$ is a 3D coordinate x, y, z; $\mu_\mathbf{i}$ is the center coordinate of each lesion 3D bounding box; and $\Sigma_\mathbf{i}$ is the covariant matrix of the i-th Gaussian determined by the height, width, and depth of the 3D bounding box. Please note that we do not directly predict the bounding box coordinates as the general object detection methods such as Faster RCNN [19] because it is sometimes challenging to define the exact boundary of a breast lesion. Also, the voxel-wise confidence maps of lesion presence could be more helpful for clinical decision support than bounding boxes.

We randomly split the database into the training and the testing sets as described in Table 10.1. A volume or a 3D patch is considered positive if at least one lesion is annotated by the radiologist. We ensure the images from the same patient could only be found either in the training or the testing set. For training, we extract 256 $\times$ 256 $\times$ 32 3D patches. 70% of the training patches are sampled as positives with at least one lesion included, considering the balance between the voxels within and without

Table 10.1 The numbers of volumes (#Volumes), lesion-positive volumes (#Positive) and lesions (#Lesions) in the evaluated DBT dataset

	#Volumes	#Positives	#Lesions
Train	2678	1111	1375
Test	131	58	72

Table 10.2 The number of convolutional layers (#Conv Layers) and model float parameters (#Parameters), respectively, in 2D-UNet, 3D-UNet, ResNet50, GCN, and AH-Net. ResNet50 is shown here as a reference to be compared with GCN with a simple decoder added

Network	#Conv Layers	#Parameters
2D-UNet	15	28,254,528
3D-UNet	15	5,298,768
*ResNet50	53	23,507,904
GCN	94	23,576,758
AH-Net	123	27,085,500

Table 10.3 The GPU inference time (ms) of different networks on a $384 \times 256 \times 64$ volume computed by averaging 1000 inferences with a NVIDIA GTX 1080Ti

	2D U-Net	3D U-Net	MC-GCN	3D AH-Net
ms	699.3	2.3	775.2	17.7

a breast lesion. The patches are sampled online asynchronously with the network training to form the mini-batches.

Along with the proposed networks, we also train 2D and 3D U-Nets with the identical architecture and parameters [3, 20] as two baseline comparisons. The 2D U-Net is also trained with input having three input channels. The 3D U-Net is trained with the same patch sampling strategies as the AH-Net. All the networks are trained till convergence then the L2 loss function is replaced with the Focal Loss [13] for hard-voxel mining. The number of convolutional layers and parameters is shown in Table 10.2. Using 2D networks, such as the MC-GCN and the 2D U-Net, to process 3D volumes involves repeatedly feeding duplicated images slices. Thus, they could be slower than the 3D networks when they are used for processing 3D volumes. We measure the GPU inference time of four networks by forwarding a 3D DBT volume of size $384 \times 256 \times 64$ 1000 times on an NVIDIA GTX 1080Ti GPU. The time spent on operations such as volume slicing is not included in the timing. The mean GPU time (ms) is shown in Table 10.3. The GPU inference of AH-Net is 43 times faster than MC-GCN though AH-Net has more parameters. The speed gain could be brought mostly by avoiding repetitive convolutions on the same slices required by multichannel 2D networks.

Non-maximal suppression is performed on the network output map to obtain the lesion locations. The network responses at the local maximal voxels are considered

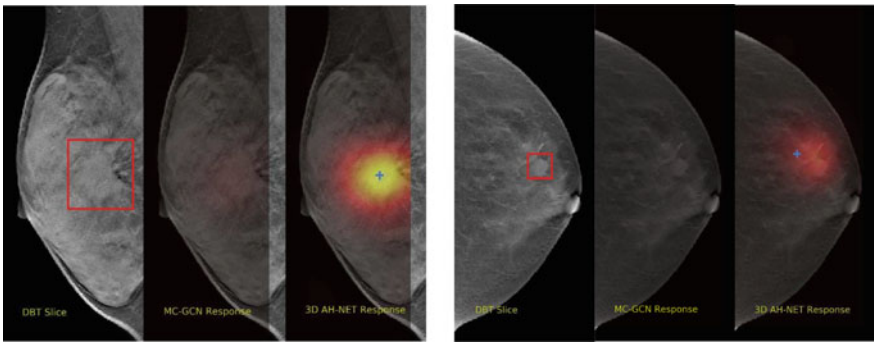

Fig. 10.5 Two example cases that AH-Net could detect the lesions that MC-GCN missed using the identical encoder weights

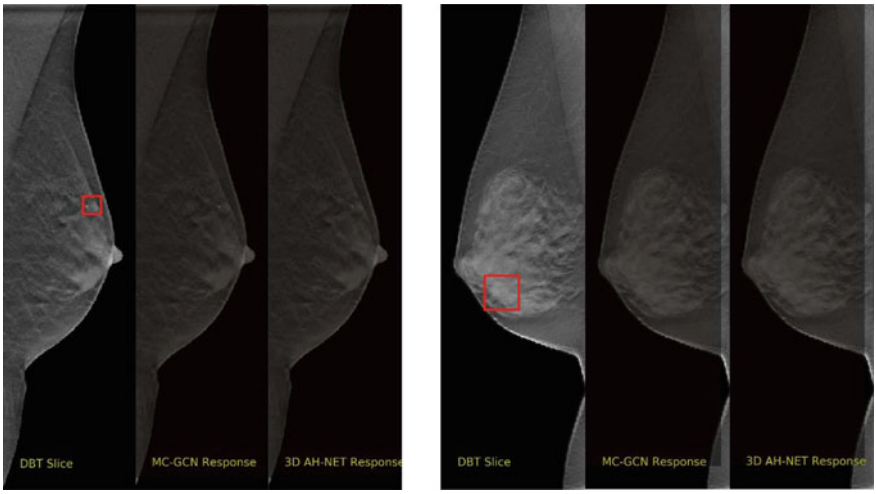

Fig. 10.6 Two example breast lesions that neither MC-GCN nor AH-Net was able to detect

as the confidence scores of the cancerous findings. Figure 10.5 shows some visual comparisons of the network's output on two example cases that AH-Net could detect the lesions missed by MC-GCN. Figure 10.6 shows two example cases with lesions surrounded by dense breast tissues that neither MC-GCN nor AH-Net was able to detect.

By altering a threshold to filter the response values, we can control the balance between the false positive rate (FPR) and true positive rate (TPR). The lesion detected by the network is considered a true positive finding if the maximal point resides in a 3D bounding box annotated by the radiologist. Similarly, if a bounding box contains a maximal point, we consider it is detected by the network. The maximal points are otherwise considered as false positive findings. We evaluate the lesion detection performance by plotting the free response operating characteristic (FROC)

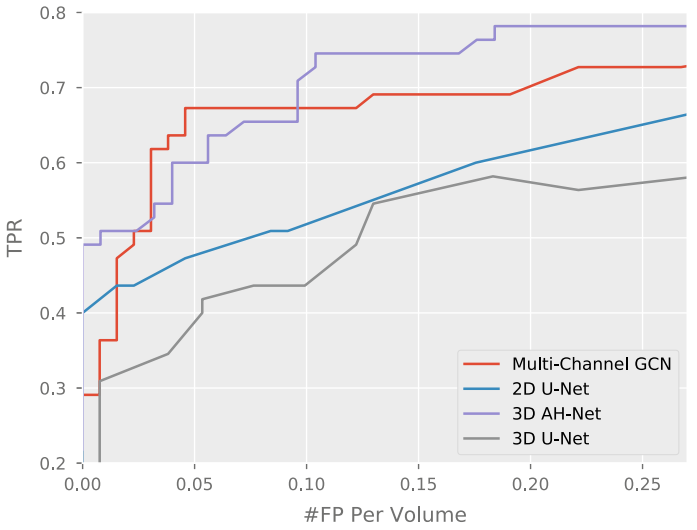

Fig. 10.7 The free response operating characteristic (FROC) curves regarding the lesion detection performance

curves, which measures the true positive rate (TPR) against the number of false positive (#FP) allowed per volume. TPR represents the percentage of lesions that have been successfully detected by the network. FPR represents the percentage of lesions that the network predicted that are false positives. As shown in Fig. 10.7, the proposed AH-Net outperforms both the 2D and 3D U-Net with large margins. Compared to the performance of the 2D network (multichannel GCN), the 3D AH-Net generates higher TPR for a majority of thresholds, except the region around 0.05 per volume false positives. It is noticeable that AH-Net also obtains nearly 50% TPR even when only 0.01 false positive findings are allowed per volume. Interestingly, the performance of 3D-UNet is slightly worse than that of 2D-UNet, though the DBT volumes have three dimensions. This might be caused by the anisotropic resolution of DBT images and the limited number of parameters constrained by the GPU memory. The FROC numbers are summarized in Table 10.4.

Table 10.4 The quantitative metrics of the compared networks on the DBT dataset. True positive rate (TPR) sampled at five different numbers of false positive (FP) findings allowed are shown in the first five columns

	FP = 0.01	FP = 0.05	FP = 0.10	FP = 0.15	FP = 0.20	FP = 0.25
2D U-Net	0.4238	0.4767	0.5181	0.5723	0.6166	0.6506
3D U-Net	0.2448	0.3877	0.4381	0.5592	0.5738	0.5733
GCN	0.3385	**0.6727**	0.6727	0.6909	0.7018	0.7272
AH-Net	**0.4931**	0.6000	**0.7272**	**0.7454**	**0.7818**	**0.7818**

10.4.2 Liver and Liver Tumor Segmentation from CT

The second evaluation dataset was obtained from the liver lesion segmentation challenge in MICCAI 2017 (lits-challenge.com), which contains 131 training and 70 testing 3D contrast-enhanced abdominal CT scans. Liver lesions are one of the commonest cancer worldwide. It is estimated that 28 920 people will die of liver lesion and 40 710 new cases will be diagnosed in 2017 [1]. Automatic segmentation of liver and lesion is challenging due to the heterogeneous and diffusive appearance of both liver and lesions. Also, the number, shape, and location of the lesions vary a lot among different volumes. The data and ground-truth masks were provided by various clinical sites around the world. The ground-truth masks contain both liver and lesion labels. Most CT scans consist of anisotropic resolution: the between-slice resolution ranges from 0.45 to 6.0 mm while the within-slice resolution varies from 0.55 to 1.0 mm. All scans cover the abdominal regions but may extend to head and feet. Other than the liver lesion, other diseases may also exist in these data, which further increases the task difficulty.

In preprocessing, the abdominal regions are truncated from the CT scans using the liver center biomarker detected by a reinforcement learning based algorithm [6]. While this step makes the network concentrate on the targeting region, its accuracy is not critical as we choose a relatively large crop region which usually ranges from the middle of the lung to the top of the pelvis. The image intensity is truncated to the range of $[-125, 225]$ HU based on the intensity distribution of liver and lesion in the training data. Due to the limited number of training data, we applied random rotation (within ± 20 degree in the xy-plane), random scaling (within ± 0.2 in all directions), and random mirror (within xy-plane) to reduce overfitting.

We first train the MC-GCN with pretrained ResNet50 as the backbone network. The input size of stacked 2D slices is 512×512 with three channels. After convergence, the weights of the encoder part of MC-GCN are transformed to the corre-

Table 10.5 The liver lesion segmentation (LITS) challenge results with the dice global (DG) and dice per case (DPC). Please refer to the challenge leaderboard for the latest results (lits-challenge.com/#results)

	Lesion		Liver	
Method	DG	DPC	DG	DPC
leHealth	0.794	**0.702**	0.964	0.961
H-DenseNet [12]	0.829	0.686	0.965	0.961
hans.meine	0.796	0.676	0.963	0.960
medical	0.783	0.661	0.951	0.951
deepX	0.820	0.657	0.967	**0.963**
superAI	0.814	0.674	–	–
GCN	0.788	0.593	0.963	0.951
3D AH-Net	**0.834**	0.634	**0.970**	**0.963**

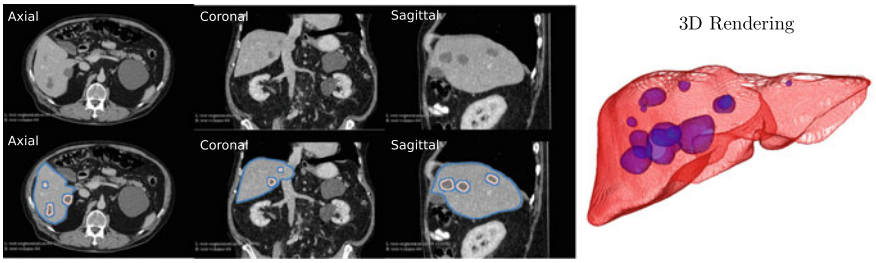

Fig. 10.8 The example liver lesion segmentation results from 3D AH-Net. The segmented contours of liver (blue) and liver lesion (pink) are overlaid on three slices viewed from different orientations (Axial, Coronal and Sagittal). The segmentations are rendered in 3D on the right

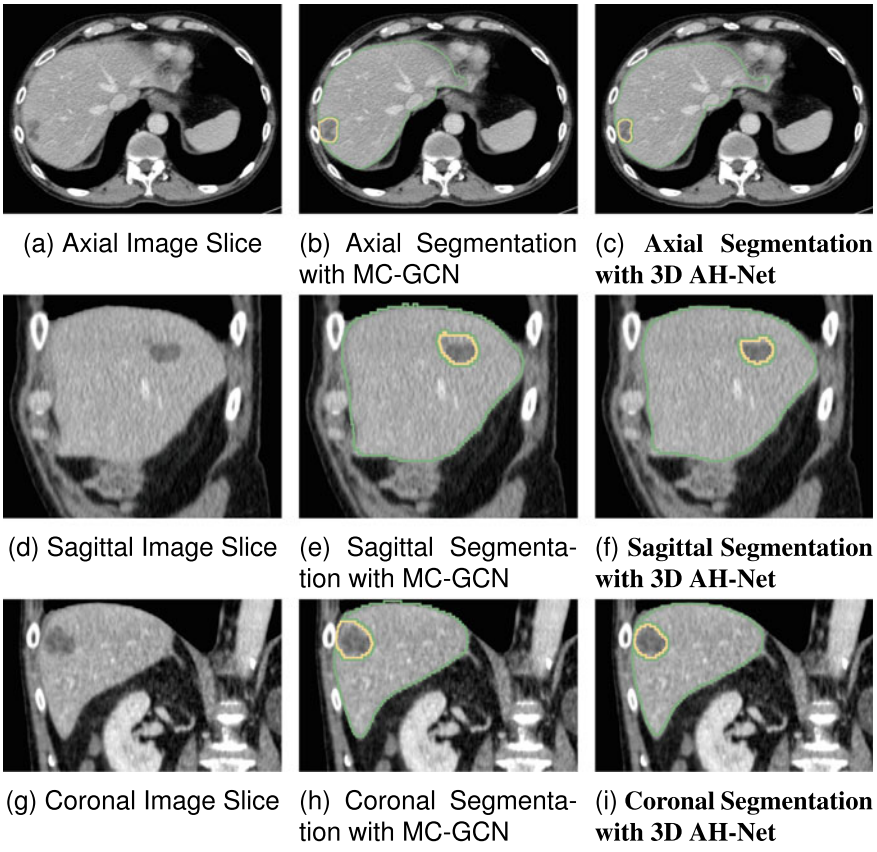

Fig. 10.9 Multi-view slices of the example test CT volume 1 of the LITS challenge

sponding layers of a 3D AH-Net, which is then fine-tuned using 3D patches with size $192 \times 192 \times 64$. The weights of other layers are randomly initialized. In the training of both networks, the cross-entropy loss is used at the beginning until convergence, which is then replaced by the Focal Loss for hard-voxel mining [13].

The performance of AH-Net is listed in Table 10.5, together with other six top-ranked submissions retrieved from the LITS challenge leaderboard. These submissions employ various types of neural network architectures: 2D, 3D, 2D–3D hybrid, and model fusion. Two evaluation metrics are adopted: (1) Dice Global (DG) which is the dice score combining all the volumes into one; (2) dice per case (DPC) which is the average of the dice scores of every single case. The dice score between two masks is defined as $DICE(A,B) = 2|A \cap B|/(|A| + |B|)$. Our results achieve the state-of-the-art performance in three of the four metrics, including the dice global score of the lesions, dice global, and dice per case score of the livers, which prove the effective-

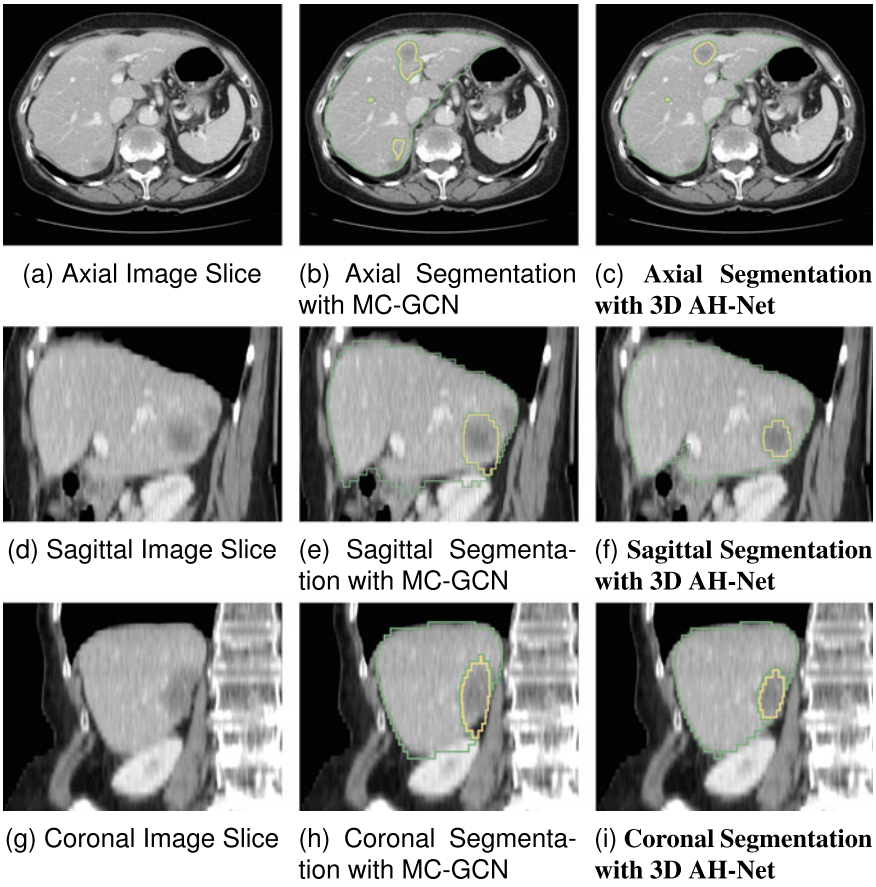

Fig. 10.10 Multi-view slices of the example test CT volume 2 of the LITS challenge

ness of AH-Net for segmenting 3D images with diverse anisotropic resolution. Some example visual results are shown in Fig. 10.8. In Figs. 10.9 and 10.10, we visually compare the results from MC-GCN and AH-Net on two different volumes acquired from the LITS challenge. AH-Net generated less false positive areas in the upper and the lower boundaries of both the lesion and liver.

10.5 Conclusion

In this chapter, we propose the 3D anisotropic hybrid network (3D AH-Net) which is capable of transferring the convolutional features of 2D images to 3D volumes with anisotropic resolution. By evaluating the proposed methods on both a large-scale in-house DBT dataset and a highly competitive open challenge dataset of CT segmentation, we show our network could obtain the state-of-the-art results. AH-Net generalizes better than the traditional 3D networks, such as 3D U-Net [3] due to the features transferred from a 2D network and the anisotropic convolution blocks. The GPU inference of AH-Net is also much faster than piling the results from a 2D network. Though AH-Net is designed for anisotropic volumes, we believe it could also be applied to volumes with resolution closed to being isotropic, such as CT and MRI.

Disclaimer: This feature is based on research, and is not commercially available. Due to regulatory reasons, its future availability cannot be guaranteed.

References

1. American Cancer Society (2017) Cancer facts and figures 2017. American Cancer Society
2. Brosch T, Tang LYW, Yoo Y, Li DKB, Traboulsee A, Tam R (2016) Deep 3D convolutional encoder networks with shortcuts for multiscale feature integration applied to multiple sclerosis lesion segmentation. IEEE Trans Med Imaging 35(5):1229–1239
3. Çiçek Ö, Abdulkadir A, Lienkamp SS, Brox T, Ronneberger O (2016) 3D U-Net: learning dense volumetric segmentation from sparse annotation. arXiv e-prints arXiv:1606.06650
4. Chaurasia A, Culurciello E (2017) LinkNet: exploiting encoder representations for efficient semantic segmentation. arXiv e-prints arXiv:1707.03718
5. Chen J, Yang L, Zhang Y, Alber M, Chen DZ (2016) Combining fully convolutional and recurrent neural networks for 3D biomedical image segmentation. arXiv e-prints arXiv:1609.01006
6. Ghesu FC, Georgescu B, Grbic S, Maier AK, Hornegger J, Comaniciu D (2017) Robust multi-scale anatomical landmark detection in incomplete 3d-ct data. In: MICCAI
7. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. arXiv e-prints arXiv:1512.03385
8. Huang G, Liu Z, van der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 2261–2269
9. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv e-prints arXiv:1502.03167

10. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv e-prints arXiv:1412.6980
11. Lee K, Zung J, Li P, Jain V, Seung HS (2017) Superhuman accuracy on the SNEMI3D connectomics challenge. arXiv e-prints arXiv:1706.00120
12. Li X, Chen H, Qi X, Dou Q, Fu CW, Heng PA (2017) H-DenseUNet: hybrid densely connected unet for liver and tumor segmentation from CT volumes. arXiv e-prints arXiv:1709.07330
13. Lin TY, Goyal P, Girshick R, He K, Dollár P (2017) Focal loss for dense object detection. arXiv e-prints arXiv:1708.02002
14. Liu F, Zhou Z, Jang H, Samsonov A, Zhao G, Kijowski R (2017) Deep convolutional neural network and 3D deformable approach for tissue segmentation in musculoskeletal magnetic resonance imaging. Magn Reson Med 79(4):2379–2391
15. Liu S, Xu D, Zhou SK, Pauly O, Grbic S, Mertelmeier T, Wicklein J, Jerebko A, Cai W, Comaniciu D (2018) 3d anisotropic hybrid network: Transferring convolutional features from 2d images to 3d anisotropic volumes. In: Frangi AF, Schnabel JA, Davatzikos C, Alberola-López C, Fichtinger G (eds) Medical Image Computing and Computer Assisted Intervention - MICCAI 2018. Springer International Publishing, Cham, pp 851–858
16. Moeskops P, Viergever MA, Mendrik AM, de Vries LS, Benders MJNL, Isgum I (2016) Automatic segmentation of MR brain images with a convolutional neural network. IEEE Trans Med Imaging 35(5):1252–1261
17. Peng C, Zhang X, Yu G, Luo G, Sun J (2017) Large kernel matters improve semantic segmentation by global convolutional network. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 1743–1751
18. Qiu Z, Yao T, Mei T (2017) Learning spatio-temporal representation with pseudo-3d residual networks. In: 2017 IEEE international conference on computer vision (ICCV), pp 5534–5542
19. Ren S, He K, Girshick RB, Sun J (2015) Faster r-cnn: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39:1137–1149
20. Ronneberger O, Fischer P, Brox T (2015) U-net: convolutional networks for biomedical image segmentation. In: MICCAI
21. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2015) ImageNet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252
22. Wang G, Li W, Ourselin S, Vercauteren T (2017) Automatic brain tumor segmentation using cascaded anisotropic convolutional neural networks. In: BrainLes workshop at MICCAI 2017
23. Xia Y, Liu F, Yang D, Cai J, Yu L, Zhu Z, Xu D, Yuille A, Roth H (2018) 3D semi-supervised learning with uncertainty-aware multi-view co-training. arXiv e-prints arXiv:1811.12506
24. Zeng T, Wu B, Ji S (2017) DeepEM3D: approaching human-level performance on 3D anisotropic EM image segmentation. Bioinformatics 33(16):2555–2562
25. Zhao H, Shi J, Qi X, Wang X, Jia J (2017) Pyramid scene parsing network. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 6230–6239