DeepVesselNet: Vessel Segmentation, Centerline Prediction, and Bifurcation Detection in 3-D Angiographic Volumes

Giles Tetteh*, Velizar Efremov, Nils D. Forkert, Matthias Schneider, Jan Kirschke, Bruno Weber, Claus Zimmer, Marie Piraud, and Björn H. Menze,

(This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.)

Abstract—We present DeepVesselNet, an architecture tailored to the challenges to be addressed when extracting vessel networks and corresponding features in 3-D angiographic volume using deep learning. We discuss the problems of low execution speed and high memory requirements associated with full 3-D convolutional networks, high class imbalance arising from low percentage (less than 3%) of vessel voxels, and unavailability of accurately annotated training data - and offer solutions that are the building blocks of DeepVesselNet.

First, we formulate 2-D orthogonal cross-hair filters which make use of 3-D context information at a reduced computational burden. Second, we introduce a class balancing cross-entropy score with false positive rate correction to handle the high class imbalance and high false positive rate problems associated with existing loss functions. Finally, we generate synthetic dataset using a computational angiogenesis model, capable of generating vascular networks under physiological constraints on local network structure and topology, and use these data for transfer learning.

DeepVesselNet is optimized for segmenting and analyzing vessels, and we test the performance on a range of angiographic volumes including clinical Time-of-Flight MRA data of the human brain, as well as synchrotron radiation X-ray tomographic microscopy scans of the rat brain. Our experiments show that, by replacing 3-D filters with 2-D orthogonal cross-hair filters in our network, we achieve over 23% improvement in speed, lower memory footprint, lower network complexity which prevents over fitting and comparable and even sometimes higher accuracy. Our class balancing metric is crucial for training the network and pre-training with synthetic data helps in early convergence of the training process.

Index Terms—vessel segmentation, centerline prediction, bifurcation detection, deepvesselnet, cross-hair filters, class balancing, synthetic data.

I. INTRODUCTION

Angiography offers insights into blood flow and conditions of vascular network. Three dimensional volumetric angiography information can be obtained using magnetic resonance (MRA), ultrasound, or x-ray based technologies like computed tomography (CT). A common first step in analyzing these data is vessel segmentation. Still, moving from raw angiography images to vessel segmentation alone might not provide enough information for clinical use, and other vessel features like centerline, diameter, or bifurcations of the vessels are also needed to accurately extract information about the vascular network, for example, to characterize its structural properties of flow pattern. In this work, we present a deep learning approach, called DeepVesselNet, to perform vessel segmentation, centerline prediction, and bifurcation detection tasks. DeepVesselNet deals with challenges that result from speed and memory requirements, unbalanced class labels, and the difficulty of obtaining well-annotated data for curvilinear volumetric structures by addressing the following three key limitations.

a) Fast cross-hair filters: Processing 3-D medical volumes poses a memory consumption and speed challenge. Using 3-D CNNs leads to drastic increase in number of parameters and computations compared to 2-D CNNs. At the same time, applying a 2-D CNN in a slice-wise fashion discards valuable 3-D context information that is crucial for tracking curvilinear structures in 3-D. Inspired by the ideas of [1]–[3] who proposed separable filters and using of intersecting 2-D planes, we demonstrate the use of cross-hair filters from three intersecting 2-D filters, which helps to avoid the memory and speed problems of classical 3-D networks, while at the same time making use of 3-D information in volumetric data. Unlike the existing ideas where 2-D planes are extracted at a pre-processing stage and used as input channels (see discussion in Sec. III-A), our cross-hair filters (see Sec. III-A) are implemented on a layer level which help retain the 3-D information throughout the network.

b) Extreme class balancing: The vessel, centerline and bifurcation prediction tasks is characterized by high class imbalances. Vessels account for less than 3% of the total voxels in a patient volume, centerlines represent a fraction of the segmented vessels, and visible bifurcations are in the hundreds at best – even when dealing with volumes with 10^6 and more voxels. This bias towards the background class is a common problem in medical data [4]–[6]. Unfortunately current class balancing loss functions for training CNNs turns out to be numerically unstable in extreme cases as ours. We...
offer a solution to this ‘extreme class imbalance’ problem by introducing a new loss function (see Sec. III-B) that we demonstrate to work well with our vascular features of interest.

c) Transfer learning from synthetic data: Manually annotating vessels, centerlines, and bifurcations requires many hours of work and expertise. To this end, we demonstrate the successful use of simulation based frameworks [7], [8] that can be used for generating synthetic data with accurate labels (see Sec. III-C) for pre-training our networks, rendering the training of our supervised classification algorithm feasible.

II. Prior Work and Open Challenges

a) Vessel Segmentation: Vessel enhancement and segmentation is a longstanding task in medical image analysis (see reviews by [9], [10]). The range of methods employed for vessel segmentation reflect the development of image processing during the past decades, some examples including region growing techniques [11], active contours [12], statistical and shape models [13]–[16], particle filtering [17]–[19] and path tracing [20]. All of these examples are interactive, starting from a set of seed label as root and propagating towards the branches. Other approaches aim at an unsupervised enhancement of vascular structures: the multi scale second order local structure of an image (Hessian) was examined by [21] with the purpose of developing a vessel enhancement filter. A measure of vessel-likeliness is then obtained as a function of all eigenvalues of the Hessian. A novel curvilinear structure detector, called Optimally Oriented Flux (OOF) was proposed by [22] to find an optimal axis on which image gradients are projected to compute the image gradient flux. OOF has a lower computational load than the calculation of the Hessian matrix proposed in [21]. A level-set segmentation approach with vesselness-dependent anisotropic energy weights is presented and evaluated in [23], [24] for 3-D time-of-flight (TOF) MRA. Phellan and Forkert [25] presented a comparative analysis of the accuracy gains in vessel segmentation generated by the use of nine vessel enhancement algorithms on time-of-flight MRA that included multi scale vesselness algorithms, diffusion-based filters, and filters that enhance tubular shapes. A machine learning approach was followed by [26], combining joint 3-D vessel segmentation and centerline extraction using oblique Hough forest with steerable filters. In a similar fashion, [27] used deep artificial neural network as a pixel classifier to automatically segment neuronal structures in stacks of electron microscopy images, a task somewhat similar to vessel segmentation. One example using deep learning architecture is, [28] who used a deep convolutional neural network to automatically segment the vessels of the brain in TOF MRA by extracting manually annotated bi-dimensional image patches in the axial, coronal, and sagittal directions as an input to the training process.

b) Centerline Prediction: Identifying the center of a vessel is relevant for calculating the vessel diameter, but also for extracting the ‘skeleton’ of a vessel when extracting the vascular network (see Fig. 1). The vessels’ skeleton and center can be found by post-processing a previously generated vessel segmentation or deal with centerline extraction in raw images with high vessel contrast. A method based on morphological operations is developed by [29] which performs erosion using 2 × 2 neighborhoods of a pixel to determine if a pixel is a centerline candidate. Active contour models are applied in [30] as well as path planning and distance transforms for extracting centerline in vessels, and [31] proposed a geodesic or minimal path technique. A morphology-guided level set model is used in [32] to performed centerline extraction by learning the structural patterns of a tubular-like object, and estimating the centerline of a tubular object as the path with minimal cost with respect to outward flux in gray level images. Zheng et al. [33] adopted vesselness filters to predict the location of the centerline, while [34] used Houghs transforms. Hough random forest with local image filters is designed in [7], [26] to predict the centerline, and trained on centerline data previously extracted using one of the level set approaches. The application of deep learning to the extraction of vessel centerline has not been explored. One reason may be the lack of annotated data necessary to train deep architectures that is hard to obtain especially in 3-D datasets.

c) Bifurcation Detection: Vessel bifurcation refers to the point on a vessel centerline where the vessel splits into two or more smaller vessels (see Fig. 1). Bifurcations represent the nodes of the vascular network and knowing their locations is important both for network extraction and for studying its properties [35]. They represent structures that can easily be used as landmarks in image registration, but also indicate the locations of modified blood flow velocity and pressure within the network itself [36]. Bifurcations are hard to detect in volumetric data – as they are rare point-like features varying in size and shape significantly. Similar to centerline extraction, the detection of bifurcations often happens by post-processing a previously generated vessels segmentation or by searching a previously extracted vessel graph. A two staged deep learning architecture is proposed in [37] for detecting carotid artery bifurcations as a specific landmark in volumetric CT data by first training a shallow network for predicting candidate regions followed by a sparse deep network for final prediction. A three stage algorithm for bifurcation detection is proposed in [36] for digital eye fundus images, a 2-D task, and their approach included image enhancement, clustering,
and searching the graph for bifurcations. The direct predicting of the location of bifurcations in a full volumetric data set is a task which has not been attempted yet. Bifurcations – and the connections in between them – are the nodes and, hence, the building blocks of a vascular network. To this end, we aim at a direct prediction of their location to offer new alternatives to a plain search of the potentially inaccurate network graph.

### d) Convolutional Neural Networks for Angiography

**Analysis:** The use of Convolutional Neural Networks (CNNs) for image segmentation has seen rapid progress during the past years. Originally aiming at predicting global labels or scores, CNNs are employed in [27] for pixel-wise classification using as input the raw intensity values of a square window centered on the pixel of interest. Unfortunately, these patch-wise architectures suffer from replication of information in memory, which limits the amount of training data that can be used during training and affects speed during testing.

The problems were addressed in [38], [39] through inverting the downsampling and convolutional operations in CNN with upsampling and deconvolutional operators leading to a Fully Convolutional Neural Network (FCNN) which is trained end-to-end for semantic segmentation. Based on this idea different variants of state-of-the-art FCNN have been presented for medical image segmentation [5], [40]–[45]. Most of these architectures were either employed on 2-D images or extended to 3-D volumes in a manner which leads to loss of 3-D object information or to high memory needs. For example, [5] applied the U-NET architecture proposed by [39] on 3-D data in a 2-D slice-wise fashion (so called 2.5D CNNs), which does not account for inter-slice information. At the same time, full 3-D architectures, such as the V-Net by [41], come with the cost of increasing the number of parameters to be learned during training by at least a factor of three over 2-D architectures of similar depth.

### III. METHODOLOGY

#### A. Cross-hair Filters Formulation

![Fig. 2. Graphical representation of cross-hair filters for 3-D convolutional operation. Left: A classical 3-D convolution with filter \( M \). Right: Cross-hair 3-D convolutional with 2-D filter stack \( M_i, M_j, M_k \).](image)

In this section, we discuss the formulation of the 3-D convolutional operator, which utilizes cross-hair filters to improve speed and memory usage while maintaining accuracy. Let \( I \) be a 3-D volume, \( M \) a 3-D convolutional kernel of shape \((k_x, k_y, k_z)\), and \( * \) be a convolutional operator. We define \( * \) as:

\[
I * M = A = [a_{ijk}]; \quad a_{ijk} = \sum_{r=1}^{k_x} \sum_{s=1}^{k_y} \sum_{t=1}^{k_z} I(r,s,t) M(r,s,t); \quad (1)
\]

\[
R = i + r - \left(1 + \frac{k_x}{2}\right), \quad S = j + s - \left(1 + \frac{k_y}{2}\right), \quad T = k + t - \left(1 + \frac{k_z}{2}\right),
\]

where \([x]\) is the greatest integer less or equal to \(x\). From Eq. (1), we see that a classical 3-D convolution involves \(k_x k_y k_z\) multiplications and \(k_x k_y k_z - 1\) additions for each voxel of the resulting image.

For a 3 × 3 × 3 kernel, we have 27 multiplications and 26 additions per voxel. Changing the kernel size to \(5 \times 5 \times 5\) increases the complexity to 125 multiplications and 124 additions per voxel. This then scales up with the dimension of the input image. For example, a volume of size \(128 \times 128 \times 128\) and a \(5 \times 5 \times 5\) kernel results in about \(262 \times 10^6\) multiplications and \(260 \times 10^6\) additions. To handle this increased computational complexity, we approximate the standard 3-D convolution operation by

\[
a_{ijk} = \sum_{x=1}^{k_x} \sum_{y=1}^{k_y} \sum_{z=1}^{k_z} I(x,y,z) M^i(x,y,z) + \sum_{r=1}^{k_x} \sum_{t=1}^{k_z} I(r,s,t) M^j(r,s,t) + \sum_{s=1}^{k_y} \sum_{t=1}^{k_z} I(r,s,t) M^k(r,s,t); \quad (2)
\]

where \(M^i, M^j, M^k\) are 2-D cross-hair filters. Using cross-hair filters results in \((k_x k_z + k_x k_y + k_x k_y + k_x k_y)\) multiplications and \((k_y k_z + k_x k_z + k_x k_y - 1)\) additions.

If we let \(k_{m1}, k_{m2}, k_{m3}\) be the sizes of the kernel \(M\) such that \(k_{m1} \geq k_{m2} \geq k_{m3}\), we can show that

\[
k_y k_z + k_x k_z + k_x k_x \leq 3(k_{m1} k_{m2}) \leq k_x k_y k_z, \quad (3)
\]

where strict inequality holds for all \(k_{m3} > 3\). Eq. (3) shows a better scaling in speed and also in memory since the filters sizes in (1) and (2) are affected by the same inequality. With the approximation in (2), and using the same example as above (volume of size \(128 \times 128 \times 128\) and a \(5 \times 5 \times 5\) kernel), we now need less than \(158 \times 10^6\) multiplications and \(156 \times 10^6\) additions to compute the convolution leading to a reduction in computation by more than \(100 \times 10^6\) multiplications and additions when compared to a classical 3-D convolution.

Increasing the volume or kernel size, further increases the gap between the computational complexity of (1) and (2). Moreover, we will see later from our experiments that (2) still retains essential 3-D context information needed for the classification task.

**a) Efficient Implementation:** In Eq. (2), we presented our 2-D crosshair filters. However, applying (2) independently for each voxel (as defined in Eq. (2)) leads to a redundant use of memory. More precisely, voxels close to each other share some neighbourhod information and making multiple copies of it is not memory efficient. We now present an efficient implementation, which we develop and use in our experiments (Fig. 3). Consider \(I\) as defined in Eq. (1) and let
us extract the sagittal, coronal, and axial planes as \( I^s, I^c, \) and \( I^a \) respectively. By application of Eqs (1), and (2), we have a final implementation as follows:

\[
I \circ M = A = \alpha A^c + \beta A^a + \gamma A^s,
\]

\[
A^c = I^c \ast M^1, \quad A^a = I^a \ast M^1, \quad A^s = I^s \ast M^1,
\]

where \( \ast \) refers to a 2-D convolution along the first and second axes of the left hand side matrix over all slices in the third axis and \( \circ \) refers to our crosshair filter operation. This implementation is efficient in the sense that it makes use of one volume at a time instead of copies of the volume in memory where voxels share the same neighbourhood. In other words, we still have only one volume in memory but rather rotate the kernels to match the slices in the different orientaions. This lowers the memory requirements during training and inference, allowing to train on more data with little memory.

b) 2.5-D Networks vs. 3-D Networks with Cross-hair Filters: We end the presentation of cross-hair filters by discussing the difference between existing 2.5-D networks and our proposed cross-hair filters. Given a 3-D task (e.g. vessel segmentation in 3-D volume) a 2.5-D based network handles the task by considering one 2-D slice at a time. More precisely, the network takes a 2-D slice as input and classifies all pixels in this slice. This is repeated for each slice in the volume and the final results from the slices are fused again to form the 3-D result. Other 2.5-D methods include a pre-processing stage where several 2-D planes are extracted and used as input channels to the 2-D network [2], [3]. On the architecture level, 2.5-D networks are 2-D networks with a preprocessing method for extracting 2-D slices and a postprocessing method for fusing 2-D results into a 3-D volume. We note that the predictions of 2.5-D networks are solely based on 2-D context information. Example of 2.5-D networks is the implementation of U-Net in [5] used for liver and lesion segmentation tasks in CT volumetric dataset and the network architecture of [44] for annotation of lumbar vertebrae.

On the other hand, 3-D networks based on our proposed cross-hair filters take the whole 3-D volume as input and at each layer in the network, we apply the convolutional operator discussed in Section III-A. Therefore, our filters make use of 3-D context information at each convolutional layer and do not require specific preprocessing or post processing. Our proposed method differs from classical 3-D networks in the sense that it uses less parameters and memory since it does not use full 3-D convolutions. However, it is worth noting that our filters scale exactly the same as 2.5-D (i.e. in only two directions) with respect to changes in filter and volume sizes. More precisely, given a square or cubic filter of size \( k \), we have \( k^2 \) parameters in a 2.5-D network and \( 3k^2 \) in our cross-hair filter based network. Increasing the filter size by a factor of \( r \) will scale up as \( k + r \) quadratically in both situations (i.e. \((k + r)^2 \) for 2.5-D and \(3(k + r)^2\) in cross-hair filter case) as compared to full 3-D networks where the parameter size scales as a cube of \( k + r \).

Unlike the existing 2.5-D ideas where 2-D planes are extracted at a pre-processing stage and used as input channels to a 2-D network architecture, our cross-hair filters are implemented on a layer level which help retain the 3-D information throughout the network making it a preferred option when detecting curvilinear objects in 3-D.

B. Extreme Class Balancing with Stable Weights

Often in medical image analysis, the object of interest (e.g. vessel, tumor etc.) accounts for a minority of the total voxels of the image. Fig. 4 shows that the objects of interest in the datasets used in this work account for less than 2.5% of the voxels (the different datasets are described in Section IV-A). To remedy this problem, [46] proposed a biased sampling loss function for training multi scale convolutional neural networks with extreme class imbalance. However, this function introduces additional trade-off parameters and then samples twice more for positive cost-sensitive fine-tuning, and vice versa, for negative cost-sensitive fine-tuning. Based on this, [47] proposed a class-balancing cross entropy loss function of the form

\[
\mathcal{L}(W) = -\beta \sum_{j \in Y^+} \log P(y_j = 1|X; W) - (1 - \beta) \sum_{j \in Y^-} \log P(y_j = 0|X; W),
\]

where \( W \) denotes the standard set of parameters of the network, which are trained with backpropagation and \( \beta \) and
$1 - \beta$ are the class weighting multipliers, which are calculated as $\beta = \frac{|Y|}{|Y^c|}$, $1 - \beta = \frac{|Y^c|}{|Y|}$. $P(\cdot)$’s are the probabilities from the final layer of the network, and $Y_+^c$ and $Y_-$ are the set of positive and negative class labels respectively. This idea, which is to give more weight to the cost associated with the class with the lowest count, has been used in other recent works [5], [40], [42], [43]. However, our experiments (in Section B-0c) show that the above loss function raises two main challenges:

$$\mathcal{L}_1(W) = \mathcal{L}_1(W) + \mathcal{L}_2(W)$$

$$\mathcal{L}_1(W) = -\frac{1}{|Y_+|} \sum_{j \in Y_+} \log P(y_j = 1|X; W)$$

$$\mathcal{L}_2(W) = -\frac{\gamma_1}{|Y_-|} \sum_{j \in Y_-} \log P(y_j = 0|X; W)$$

$\L_1$ is a more numerically stable version of Eq. 6 since it computes the voxel-wise, cost which scales well with the size of the dataset or batch. But the ratio of $\beta$ to $1 - \beta$ is maintained as desired. $\L_2$ (FP Rate Correction) is introduced to penalize the network for false predictions. However, we do not want to give false positive ($Y_+$) and false negatives ($Y_-$) the same weight as total predictions ($Y_+, Y_-$), since we will end up with a loss function without any class balancing because the weights will offset each other. Therefore, we introduce $\gamma_1$ and $\gamma_2$, which depend on the mean absolute distance of the wrong predicted probabilities to 0.5 (the value can be changed to suit the task). This allows us to penalize false predictions, which are very far from the central point (0.5). The false predictions ($Y_+, Y_-$) are obtained through a 0.5 probability threshold. Experimental results from application of FP rate correction can be found in Section B-0c.

### C. Synthetic Data Generation and Preparation

To generate synthetic data, we follow the method of [7] which considers the mutual interplay of arterial oxygen ($O_2$) supply and vascular endothelial growth factor (VEGF) secreted by ischemic cells to achieve physiologically plausible results. Each vessel segment is modeled as a rigid cylindrical tube with radius $r$ and length $l$. It is represented by a single directed edge connecting two nodes. Semantically, this gives rise to four different types of nodes, namely root, leaf, bifurcation, and inter nodes. Each node is uniquely identified by the 3-D coordinate $\vec{P} = (x, y, z)^T$. Combining this with connectivity information, fully captures the geometry of the approximated vasculature. Radius of parent bifurcation branch $r_p$, and the radius of left ($r_l$) and right ($r_r$) daughter branches are related by a bifurcation law (also known as Murray’s law) given by $r_{f}^2 = r_l^2 + r_r^2$, where $\gamma$ is the bifurcation exponent. Further constraints:

$$\cos(\phi_l) = \frac{r_p^2 + r_r^2 - r_l^2}{2r_p r_r}, \quad \cos(\phi_r) = \frac{r_p^2 + r_l^2 - r_r^2}{2r_p r_l}, \quad (8)$$

are placed on the bifurcation angles of the left ($\phi_l$) and right ($\phi_r$) respectively, which geometrically corresponds to the optimal position of the branching point $\vec{P}_b$ with respect to a minimum volume principle [7]. The tree generation model and the bifurcation configuration is shown in Fig. 5. In the arterial tree generation experiment, the parameters in Table 1 of [7] are used. We use the default (underlined) values for all model parameters and generate 136 volumes.

The output of the generation process is a tree with information on the 3-D position of the nodes, their type (root, bifracion, inter, leaf), and connectivity information, which
includes the edge \( E_{ij} \) between two nodes \( N_i \) and \( N_j \), and its radius \( R_{ij} \). We used this information to construct the actual 3-D volume by modeling each vessel segment as a cylinder in 3-D space. Vessel intensities are randomly chosen in the interval \([128, 255]\) and non-vessel intensities are chosen from the interval \([0 \text{ – } 100]\). Gaussian noise is then applied to the generated volume randomly changing the mean (i.e. in the range \([-5, 5]\)) and the standard deviation (i.e. in the range \([-15, 30]\)) for each volume.

D. Network Architecture and Implementations

We test the performance of DeepVesselNet discussed in sections III-A, III-B, and III-C through two main implementations:

a) DeepVesselNet-FCN (DVN-FCN): We construct a Fully Convolutional Network FCN with four convolutional layers and a sigmoid classification layer. In this implementation, we do not use any down-sampling layer and we carry out the convolutions in a way that the output image is of the same size as the input image by zero-padding. The removal of the down-sampling layer is motivated by the fact that the tasks (vessel segmentation, centerline prediction, and bifurcation detection) involve fine detailed objects and down-sampling has an effect of averaging over voxels which causes these fine details to be lost. With this network implementation, we have a very simple 5-layer fully-convolutional network, which takes a volume of arbitrary size and outputs a segmentation map of the same size. For the network structure and a description of the parameters, see Fig. 6.

b) DeepVesselNet-VNet (DVN-VNet): As a proof of principle, we take the V-Net architecture proposed by [41] (see Fig. A.6 in the Appendix) and replace all 3-D convolutions with our proposed cross-hair filters discussed in section III-A. The aim is to test the improvement in memory requirements as well as the speed up that our cross-hair implementation provides over the original architecture proposed in [41]. We also use it to evaluate whether speed and memory consumption have a significant effect on prediction accuracy.

c) Network Configuration, Initialization, and Training:

We use the above described architecture to implement three binary networks for vessel segmentation, centerline prediction, and bifurcation detection. Network parameters are randomly initialized, according to the method proposed in [48], by sampling from a uniform distribution in the interval \((-\frac{1}{\sqrt{k_xk_yk_z}}, \frac{1}{\sqrt{k_xk_yk_z}})\) where \((k_x \times k_y \times k_z)\) is the size of the given kernel in a particular layer. For each volume, we extract non-overlapping boxes of size \((64 \times 64 \times 64)\) covering the whole volume and then feed them through the network for the fine-tuning of parameters. After this, we then train the network using a stochastic gradient descent without regularization. During pre-training, we use a learning rate of 0.01 and decay of 0.99, which is applied after every 200 iterations. For fine-tuning, we use a learning rate of 0.001 and a decay of 0.99 applied after every 200 iterations. We implement our algorithm using the THEANO [49] Python framework and train on a machine with 64GB of RAM and Nvidia TITAN X 12GB GPU.

IV. EXPERIMENTS, RESULTS AND DISCUSSION

A. Datasets

In this work, we use three different datasets to train and test the networks. In all three data sets, the test cases are kept apart from the training data and are used only for testing purposes.

a) Synthetic Dataset: Training convolutional networks from scratch typically requires significant amounts of training data. However, assembling a properly labeled dataset of 3-D curvilinear structures, such as vessels and vessel features, takes a lot of human effort and time, which turns out to be the bottleneck for most medical applications. To overcome this problem, we generate synthetic data based on the method proposed in [7]. A brief description of this process has already been presented in Section III-C. We initialize the processes with different random seeds and scale the resulting vessel sizes in voxels to match the sizes of vessels in clinical datasets. After this, we add different levels of Gaussian noise, as mentioned earlier in Section III-C, to increase the randomness and to make it more realistic. We generate 136 volumes of size \(325 \times 304 \times 600\) with corresponding labels for vessel segmentation, centerlines, and bifurcation detection. We then select twenty volumes out of the 136 as a test set for the pre-training phase and use the remaining volumes for pre-training in the various tasks at hand. An example of the synthetic dataset can be found in Fig. 7(c).

b) Clinical MRA Time-of-Flight (TOF) Dataset: To fine-tune and test our network architectures on real data, we obtain 40 volumes of clinical TOF MRA of the human brain, 20 of which are fully annotated and the remaining 20 partially annotated using the method proposed by [23]. Each volume has a size of \(580 \times 640 \times 136\) and spatial resolution of \(0.3125\text{mm} \times 0.3125\text{mm} \times 0.6\text{mm}\) on the coronal, sagittal, and axial axes respectively. We select 15 out of the 20 fully annotated volumes for testing and use the remaining five as a validation set. We also correct the 20 partially annotated volumes by manually verifying some of the background and foreground voxels. This leads to three labels, which are true foreground (verified foreground), true background (verified background), and the third class, which represent the remaining voxels not verified. After this, we use the true foreground and background labels to fine-tune our network after pre-training with the synthetic dataset. This approach helps in
avoiding any uncertainty with respect to using the partially annotated data for fine-tuning of the network. A sample of volume from the TOF MRA dataset can be found in Fig. 7(a).

c) Synchrotron Radiation X-ray Tomographic Microscopy (SRXTM): A 3-D volume of size \(2048 \times 2048 \times 2740\) and spacial resolution \(0.7\, \text{mm} \times 0.7\, \text{mm} \times 0.7\, \text{mm}\) is obtained from synchrotron radiation X-ray tomographic microscopy of a rat brain. From this large volume, we extract a dataset of 20 non-overlapping volumes of size \(256 \times 256 \times 256\), which were segmented using the method proposed by [26] and use them to fine-tune the network. To create a test set, we manually annotate 52 slices in 4 other volumes (208 slices in total). Detailed description of the SRXTM data can be found in [50], and a sample volume is presented in Fig. 7(b).

B. Data Preprocessing with Intensity Projection

In our experiments, we use different datasets that come from different sources and acquisition modalities. Therefore, we need to normalize their intensity ranges and contrast. We test the following preprocessing strategies to achieve homogeneity between the datasets.

First, the original intensities are normalized to the range \([0, 1]\) using \(f(x) = \frac{x - \min(X)}{\max(X) - \min(X)}\) where \(x\) is the pixel intensity and \(X\) denotes the range of all intensities in the volume. The second strategy involves clipping the intensity values by \(g(x) = \{c, x > c; x, x \leq c\}\) and then normalizing the intensities by \(f(x)\). Our experiments show that a value of \(c = 190\) is optimal for the intensity range of the datasets. The final strategy builds on the second preprocessing strategy by clipping by \(g(x)\), normalizing by \(f(x)\), and then projecting the resulting intensities by the function \(q(x) = x^p\). In our experiments, we test quadratic and cubic projections (i.e. \(p = 2\) and \(p = 3\), respectively).

The visual effect of the strategies discussed above can be found in Fig. A.1 in the Appendix and it shows the improvement in contrast that is achieved through the preprocessing strategies. Results from experiments with these preprocessing strategies are given in Section B-0a.

C. Evaluating DeepVesselNet Performance

Prior to evaluating the performance of DeepVesselNet, we conducted a series of experiments to test the components of DeepVesselNet which includes the preprocessing strategies, fast cross-hair filters, the FP rate correction, and pre-training on synthetic data. These experiments and their results are discussed in appendix B. In this subsection, we retain the best training strategy from the above described experiments and assess the performance of our proposed network architecture with other available methods mainly on the vessel segmentation task. As a further validation of our methodology we handle centerline prediction, and bifurcation detection using the proposed architectures. Given a good vessel segmentation, centerline prediction and bifurcation detection tasks can best be handled by applying vessel skeletonization as a post-processing step. Our aim in applying our architecture to handle these tasks is not to show superiority over the existing vessel skeletonization methods but it is to serve as a further verification of the effects of our described methodology. We therefore discuss mainly results on vessel segmentation and briefly mention the results from centerline and bifurcation task. The full result and discussion of these experiments are therefore given in the appendix section C.

a) Experiment 1: Vessel Segmentation: We pre-train DeepVesselNet-(FCN and VNet) architectures (from Figs. 6 and A.6) on synthetic volumes for vessel segmentation and evaluate its performance on TOF MRA volumes. We then fine-tune the networks with additional clinical TOF MRA data, repeating the evaluation. Table 1 reports results of these tests,
Fig. 7. Sample of datasets used in our experiments with the corresponding ground truth segmentations.

together with performances of competing methods. We obtain a Dice score of 0.81 for DeepVesselNet-FCN and 0.80 for DeepVesselNet-VNet before, and 0.86 (DeepVesselNet-FCN) as well as 0.84 (DeepVesselNet-VNet) after fine tuning. Generating vessel segmentations takes less than 13s per volume using DeepVesselNet-FCN and 20s for DeepVesselNet-VNet. Table I also reports results from the methods of [26] (V-Net) and [23] both of which are outperformed by DeepVesselNet-FCN both in terms of speed (execution time) and Dice score. Comparing DeepVesselNet-VNet and original V-Net on the MRA data, we find a small advantage for the latter in terms of Dice score which cannot be considered as significant (0.8425 and 0.8497 respectively with sample standard error of 0.0066 and T-test significance probability of 0.1950). However, DeepVesselNet-VNet has the advantage of being six seconds faster (about 23% improvement) during prediction, a time difference that will scale up with volume size and filter sizes, and our results show that cross-hair filters can be used in DeepVesselNet at a little to no cost in terms of vessel segmentation accuracy.

b) Experiment 2: Centerline Prediction: For centerline prediction, we train DeepVesselNet on the synthetic dataset and test it on synthetic as well as clinical datasets. The network uses the probabilistic segmentation masks from experiment 1 as an input (together with the same training and test splits described in experiment 1) and uses the vessel predictions to restrict predicted centerlines to be within vessel regions. We obtain a Dice score of 0.80 for DeepVesselNet-FCN, outperforming all other methods by a margin of more than 5% in Dice score, these methods include Schneider et al [26] which is based on morphological operations for vessel skeletonization (the state of the art). DeepVesselNet-FCN is also better than the competing methods in terms of execution time. Fig. 8 shows a qualitative example of centerline prediction with DeepVesselNet-FCN. Full quantitative results are in Table I.

Table I

| Dataset     | Method       | Precision | Recall | Dice  | Ex. time |
|-------------|--------------|-----------|--------|-------|----------|
| Synthetic   | DVN-FCN      | 99.84     | 99.87  | 99.86 | 13 s     |
|             | DVN-VNet     | 99.54     | 99.59  | 99.56 | 17 s     |
|             | V-Net        | 99.48     | 99.50  | 99.49 | 23 s     |
|             | Schneider et al. | 99.47 | 99.56  | 99.52 | n/a      |
| TOF MRA     | DVN-FCN (F)  | 86.44     | 86.93  | 86.68 | 13 s     |
|             | DVN-FCN (P)  | 82.76     | 80.25  | 81.48 | 13 s     |
|             | DVN-VNet (F) | 85.00     | 83.51  | 84.25 | 20 s     |
|             | DVN-VNet (P) | 83.32     | 77.12  | 80.10 | 20 s     |
|             | V-Net (F)    | 84.34     | 85.62  | 84.97 | 26 s     |
|             | V-Net (P)    | 82.41     | 75.82  | 78.98 | 26 s     |
|             | Schneider et al. | 84.81 | 82.15  | 83.46 | 100 min  |
|             | Forkert et al. | 84.99  | 73.00  | 78.57 | n/a      |
| SRXTM       | DVN-FCN      | 96.72     | 95.82  | 96.27 | 4 s      |
|             | DVN-VNet     | 95.83     | 96.18  | 96.01 | 7 s      |
|             | V-Net        | 95.25     | 95.84  | 95.55 | 11 s     |
|             | Schneider et al. | 95.15 | 91.51  | 93.30 | 23 min   |

| Task        | Method       | Precision | Recall | Dice  | Ex. time |
|-------------|--------------|-----------|--------|-------|----------|
| Centerline  | DVN-FCN      | 77.63     | 82.35  | 79.92 | 13 s     |
|             | V-Net        | 76.41     | 73.30  | 74.82 | 23 s     |
| Bifurcation | DVN-FCN      | 78.80     | 92.97  | 85.30 | 13 s     |
|             | Schneider et al. | 77.18 | 85.08  | 80.94 | n/a      |
in the appendix.

Fig. 8. Centerline prediction on synthetic test data using DeepVesselNet-FCN (centerline in green)

c) Experiment 3: Bifurcation Detection: For a quantitative evaluation of DeepVesselNet in bifurcation detection, we use synthetically generated data, and adopt a two-input-channels strategy. We use the vessel segmentations from experiments 1 as one input channel and the centerline predictions from Experiment 2 as a second input channel relying on the same training and test splits as in the previous experiments. Results from Table I and Fig. A.5 show that DeepVesselNet-FCN performs better than the other architectures in 5 out of 6 metrics with a detection Dice score of 85% and again showing superiority in execution time. DeepVesselNet-FCN again performs better than Schneider et al. [26] which is based on first reconstructing the underlying network and then assigning bifurcation points based on node connectivity. Fig. 9 shows a qualitative example of bifurcation detection with DeepVesselNet-FCN. Full qualitative results are in Table VI in the appendix.

Fig. 9. Bifurcation detection on synthetic test data using DeepVesselNet-FCN (bifurcations in green)

V. SUMMARY AND CONCLUSIONS

We present DeepVesselNet, an architecture tailored to the challenges of extracting vessel networks and network features using deep learning. Our experiments show that the cross-hair filters, which is one of the components of DeepVesselNet, performs comparably well as 3-D filters and, at the same time, improves significantly both speed and memory usage, easing an upscaling to larger data sets. Another component of DeepVesselNet, the introduction of new weights and the FP rate correction in the class balancing loss function helps in maintaining a good balance between precision and recall during training. This turns out to be crucial for preventing over and under-segmentation problems, which are common problems in vessel segmentation. Finally, we successfully demonstrated that transfer learning of DeepVesselNet through pre-training on synthetically generated data improves segmentation and detection results, especially in situations where obtaining manually annotated data is a challenge.

As future work, we will generalize DeepVesselNet to multiclass segmentation, handling vessel segmentation, centerline prediction, and bifurcation detection simultaneously, rather than in three subsequent binary tasks. We also expect that network architectures tailored to our three hierarchically nested classes will improve the performance of the DeepVesselNet, for example, in a single, but hierarchical approach starting from a base network for vessel segmentation, additional layers for centerline prediction, and a final set of layers for bifurcation detection. The current implementation (cross-hair layers, the networks, cost function), and future extensions of DeepVesselNet, will be made publicly available on Github.

REFERENCES

[1] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, “Learning separable filters,” in The IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), June 2013.
[2] H. R. Roth, L. Lu, A. Seff, K. M. Cherry, J. Hoffman, S. Wang, J. Liu, E. Turkbey, and R. M. Summers, “A new 2.5d representation for lymph node detection using random sets of deep convolutional neural network observations,” in Med. Image Comput. and Computer-Assisted Intervention – MICCAI 2014. Springer Int. Publishing, 2014, pp. 520–527.
[3] S. Liu, D. Zhang, Y. Song, H. Peng, and W. Cai, “Triple-crossing 2.5d convolutional neural network for detecting neuronal arbours in 3d microscopic images,” in Mach. Learn. in Med. Imaging. Springer Int. Publishing, 2017, pp. 185–193.
[4] J. W. Grzymala-Busse, L. K. Goodwin, W. J. Grzymala-Busse, and X. Zheng, “An approach to imbalanced data sets based on changing rule strength,” in Rough-Neural Computing: Techniques for Computing with Words, S. K. Pal, L. Polkowski, and A. Skowron, Eds. Berlin, Heidelberg: Springer, 2004, pp. 543–553.
[5] P. F. Christ, F. Ettlinger, F. Grün, M. E. A. Elshaera, J. Lipkova, S. Schlecht, F. Ahmaddy, S. Tatavarty, M. Bickel, P. Bilic, M. Rempfler, F. Hofmann, M. D. Anastasi, S.-A. Ahmadi, G. Kaisiss, J. Holch, W. Sommer, R. Braren, V. Heinemann, and B. Menze, “Automatic liver and tumor segmentation of ct and mri volumes using cascaded fully convolutional neural networks,” ArXiv e-prints, Feb. 2017.
[6] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, “Learning from class-imbalanced data: Review of methods and applications,” Expert Systems with Applications, vol. 73, pp. 220 – 239, 2017.
[7] M. Schneider, J. Reichold, B. Weber, G. Székely, and S. Hirsch, “Tissue metabolism driven arterial tree generation,” Med. Image Anal., pp. 1397–1414, 2012.
[8] D. Szczepan and G. Székely, “Simulating vascular systems in arbitrary anatomies,” in Med. Image Comput. and Computer-Assisted Intervention – MICCAI. 8th Int. Conf., Palm Springs, CA, USA, October 26–29, 2005, Proc., Berlin, Heidelberg: Springer, 2005, pp. 641–648.
[9] C. Kirbas and F. Quek, “A review of vessel extraction techniques and algorithms.” ACM Comput. Surv., vol. 36, pp. 81–121, 2004.
Large-Scale Annotation of Biomedical Data and Expert Label Synthesis. Springer Int. Publishing, 2017, pp. 39–46.

B. Shagufa, S. A. Khan, A. Hassan, and A. Rashid, “Blood vessel segmentation and centerline extraction based on multilayered thresholding in ct images,” Proc. of the 2nd Int. Conf. on Intelligent Systems and Image Processing, pp. 426–432, 2014.

M. M. Maddah, A. Afrati, and H. Soltanian, “Snake modeling and distance transform approach to vascular center line extraction and quantification,” Computerized Med. Imag. and Graphics, vol. 27 (6), pp. 503–512, 2003.

D. Chen and L. D. Cohen, “Piecewise geodesics for vessel centerline extraction and boundary delineation with application to retina segmentation,” in Scale Space and Variational Methods in Computer Vision: 5th Int. Conf. , SVM 2015, Lege-Cap Ferrret, France, May 31 - June 4, 2015. Springer Int. Publishing, 2015, pp. 270–281.

A. Santamaria-Pang, C. M. Colbert, P. Saggau, and I. A. Kakadiaris, “Automatic centerline extraction of irregular tubular structures using probability volumes from multiphoton imaging,” in Med. Image Comput. and Computer-Assisted Intervention – MICCAI 2007: 10th Int. Conf., Brisbane, Australia, October 29 - November 2, 2007, Proc., Part II. Berlin, Heidelberg: Springer, 2007, pp. 486–494.

Y. Zheng, J. Shen, H. Tek, and G. Funka-Lea, “Model-driven centerline extraction for severely occluded major coronary arteries,” in Mach. Learn. in Medical Imaging: Third Int. Workshop, MLMI 2012, Held in Conjunction with MICCAI: Nice, France, Revised Selected Papers. Berlin, Heidelberg: Springer, 2012, pp. 10–18.

M. M. G. Macedo, C. Mekkaoui, and M. P. Jackowski, “Vessel centerline tracking in cta and mra images using hough transform,” in Progress in Pattern Recognition, Image Anal., Computer Vision, and Applications: 15th Iberoamerican Congress on Pattern Recognition, CIARP 2010, Sao Paulo, Brazil, November 8-11, 2010. Proc. Berlin, Heidelberg: Springer, 2010, pp. 295–302.

M. Rempfler, M. Schneider, G. D. Ielaquda, X. Xiao, S. R. Stock, J. Klohs, G. Székely, B. Andres, and B. H. Menze, “Reconstructing cerebrovascular networks under local physiological constraints by integer programming,” Med. Image Anal., pp. 86–94, 2015.

T. Chaichana, Z. Sun, M. Barrett-Baxendale, and A. Nagar, “Automatic location of blood vessel bifurcations in digital eye fundus images,” in Proc. of Sixth Int. Conf. on Soft Computing for Problem Solving: SocProS 2016, Volume 2. Singapore: Springer Singapore, 2017, pp. 332–342.

Y. Zheng, D. Lui, B. Georgescu, H. Nguyen, and D. Comaniciu, “3d deep learning for efficient and robust landmark detection in volumetric data,” in Med. Image Comput. and Computer-Assisted Intervention – MICCAI: N. Navab, J. Hornegger, W. M. Wells, and A. Frangi, Eds. Berlin, Heidelberg: Springer, 2015, pp. 565–572.

J. Long, E. Shellhammer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” CoRR, abs/1411.4038, 2014.

O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” Med. Image Comput. and Computer-Assisted Intervention (MICCAI), vol. 9351, pp. 234–241, 2015.

K. K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool, “Deep retinal image understanding,” in Med. Image Comput. and Computer-Assisted Intervention – MICCAI 2016: 19th Int. Conf., Athens, Greece, October 17-21, 2016, Proc., Part II. Springer Int. Publishing, 2016, pp. 140–148.

F. Milletari, N. Navab, and S. Ahmadi, “V-net: Fully convolutional neural networks for volumetric medical image segmentation,” in Fourth Int. Conf. on 3D Vision, Cont. T. Pfister, H. Hoppe, M. Alexa, and H. P. A. Gruber, Eds. Berlin, Heidelberg: Springer, 2016, pp. 565–571.

I. Nogues, L. Lu, X. Wang, H. Roth, G. Bertasius, N. Lay, J. Shi, Y. Tsehay, and R. M. Summers, “Automatic lymph node cluster segmentation using holistically-nested neural networks and structured optimization in ct images,” in Med. Image Comput. and Computer-Assisted Intervention – MICCAI 2016: 19th Int. Conf., Athens, Greece, October 17-21, 2016, Proc., Part II. Springer Int. Publishing, 2016, pp. 388–397.

H. R. Roth, L. Lu, A. Farag, A. Sohn, and R. M. Summers, “Spatial aggregation of holistically-nested networks for automated pancreas segmentation,” in Med. Image Comput. and Computer-Assisted Intervention – MICCAI 2016: 19th Int. Conf., Athens, Greece, October 17-21, 2016, Proc., Part II. Springer Int. Publishing, 2016, pp. 451–459.

A. Sekuboyina, A. Valentinitsch, J. Kirschke, and B. H. Menze, “A localsegmentation algorithm for multi-label segmentation of lumbar vertebrae using deep nets,” arXiv, vol. 1703.04347, 2017.

G. Tetteh, M. Rempfler, C. Zimmer, and B. H. Menze, “Deep-fext: Deep feature extraction for vessel segmentation and centerline prediction,” in
we observe that the intensity projections (quadratic and cubic) shows the outcome of these experiments. From the results, is to the synthetic dataset after the preprocessing. Table II enables us to measure how similar our clinical MRA dataset vessels in the clinical MRA dataset after applying each of the segmentation and then use this trained network to segment DeepVesselNet-FCN solely on our synthetic data for vessel considering the synthetic dataset as ideal. Therefore, we train proposed in Section IV-B. We carry out these experiments a and the preprocess intensities in b. From the images above we could the preprocessing discussed preprocessing schemes. Intensities in c. and d. have been further normalized between [0,1]. From the images above we could the preprocessing strategies has improved the image contrast between the original intensities in realistic vascular networks," Journal of Cerebral Blood Flow & Metabolism, vol. 29, no. 8, pp. 1429–1443, 2009.

**APPENDIX A**

**EFFECT OF PREPROCESSING STRATEGIES**

- a. Original intensities
- b. Clipped image
- c. Projected intensities $f(x) = x^2$
- d. Projected intensities $f(x) = x^3$

Fig. A.1. Sample of slide from TOF MRA dataset after application of the discussed preprocessing schemes. Intensities in c. and d. have been further normalized between [0,1]. From the images above we could the preprocessing strategies has improved the image contrast between the original intensities in a and the preprocess intensities in b, c and d.

**APPENDIX B**

**EVALUATING THE DEEPVESSELNET COMPONENTS**

- a) **Experiment 1: Testing Preprocessing Strategies:** In a first experiment we test the different preprocessing strategies proposed in Section IV-B. We carry out these experiments considering the synthetic dataset as ideal. Therefore, we train DeepVesselNet-FCN solely on our synthetic data for vessel segmentation and then use this trained network to segment vessels in the clinical MRA dataset after applying each of the four preprocessing strategies discussed in Section IV-B. This enables us to measure how similar our clinical MRA dataset is to the synthetic dataset after the preprocessing. Table II shows the outcome of these experiments. From the results, we observe that the intensity projections (quadratic and cubic) improve the results (a lot in terms of Dice score). Quadratic intensity projection performs slightly better than the cubic intensity projection. Therefore, we use the quadratic intensity projection in all following experiments.

**TABLE II**

| Preprocessing method         | Precision | Recall | Dice | Ex. time |
|------------------------------|-----------|--------|------|----------|
| Original Intensities         | 0.0112    | 1.0    | 0.0221 |          |
| Clipped Intensities          | 0.3117    | 0.9379 | 0.4679 |          |
| Quadratic Projection         | 0.8276    | 0.8025 | 0.8149 |          |
| Cubic Projection             | 0.8995    | 0.7033 | 0.7894 |          |

b) **Experiment 2: Fast Cross-hair Filters:** To investigate the usefulness of the 3-D context information extracted by the cross-hair filters of DeepVesselNet, we construct a similar network as in Fig. 6 (DeepVesselNet-FCN) but replace the cross-hair filters by 2-D filters. We train this 2.5-D network on 2-D slices (as described in Section III-A) and then fuse the final slice-wise results to obtain the final volumetric predictions. We perform this experiment using the synthetic dataset for centerline prediction with the training and test splits discussed in Section IV-A and measure the performance of the two networks in terms of execution time per volume (Ex. time) and in terms of accuracy by the Dice score. Results are reported in Table III. We observe a benefit of 2.5-D network in terms of run time (6s vs. 13s) when compared to the 3-D network but at a cost of accuracy in terms of Dice Score (0.70 vs. 0.80), which can be explained by lack of 3-D context information.

**TABLE III**

| Method                      | Precision | Recall | Dice    | Ex. time |
|-----------------------------|-----------|--------|---------|----------|
| 3-D (Cross-hair filters)    | 0.7763    | 0.8235 | 0.7992  | 13 s     |
| 2.5-D (2-D on slices)       | 0.6580    | 0.7472 | 0.7017  | 6 s      |

c) **Experiment 3: The FP Rate Correction Loss Function ($L_1 + L_2$):** To test the effect of FP rate correction loss function discussed in section III-B, we train DeepVesselNet-FCN architecture on a sub-sample of four clinical MRA volumes from scratch, with and without FP rate correction described in Eq. (7). We train for 5000 iterations and record the ratio of precision to recall every 5 iterations using a threshold of 0.5 on the probability maps. A plot of the precision - recall ratio during training without FP rate correction ($L_1$ Only) and with FP rate correction ($L_1 + L_2$) is presented in Fig. A.2. The results of this experiments suggest that training with both factors in the loss function, as proposed in Section III-B keeps a better balance between precision and recall (i.e. a ratio closer to 1.0) than without the second factor. A balanced precision-recall ratio implies that the training process is not bias towards
the background or the foreground. This helps prevent over-segmentation, which is caused by the introduction of the class balancing.

![Plot of precision - recall ratio against training iteration](image)

**Fig. A.2.** Precision - recall ratio during training, with FP rate correction and without FP rate correction in the loss function, on four selected clinical MRA volumes. max. avg Dice refers to the final training Dice value on these four volumes. A balanced precision-recall ratio (i.e. close to 1) implies that the training process is not bias towards the background or the foreground.

*d) Experiment 4: Pre-training on Synthetic Data:* We assess the usefulness of transfer learning with synthetic data by comparing the training convergence speed, and various other scores that we obtain when we pre-train DeepVesselNet-FCN on synthetic data and fine-tune on the clinical MRA dataset, compared to training DeepVesselNet-FCN from scratch on the clinical MRA. For this experiment, we only consider the vessel segmentation task, as no annotated clinical data is available for centerline and bifurcation tasks. Results of this experiment are reported in Table [IV]. We achieve a Dice score of 0.8639 for training from scratch without pre-training on synthetic data and 0.8668 when pre-training on synthetic data. This shows that training from scratch or pre-training on synthetic data does not make a big difference regarding the accuracy of the results. However, training from scratch requires about 600 iterations more than pre-training on synthetic data for the network to converge (i.e. 50% more longer).

### APPENDIX C

**FULL RESULTS AND DISCUSSION ON CENTERLINE AND BIFURCATION DETECTION TASKS**

*a) Experiment 2: Centerline Prediction:* For centerline prediction, we train DeepVesselNet on the synthetic dataset and test it on synthetic as well as clinical datasets. The network uses the probabilistic segmentation masks from experiment 1 as an input (together with the same training and test splits described in experiment 1) and uses the vessel predictions to restrict predicted centerlines to be within vessel regions. Qualitative results are presented in Figs. [8] and [A.3] together with quantitative scores in Table [V]. We obtain a Dice score of 0.80 for DeepVesselNet-FCN, outperforming all other methods by a margin of more than 5%. Here we note that DeepVesselNet-FCN shows a higher precision (0.78 vs 0.48) than the morphological operations based method of Schneider et al. [26] which is one of the standard method for centerline prediction. The results also show that DeepVesselNet-FCN is better than the competing methods in terms of execution time. Again, DeepVesselNet-VNet performs slightly worse than V-Net in terms of the Dice score (0.67 vs. 0.75). However, DeepVesselNet-VNet has an advantage in speed of execution (17s vs. 23s) as expected.

**TABLE V**

| Method         | Precision | Recall | Dice  | Ex. Time |
|----------------|-----------|--------|-------|----------|
| DVN-FCN        | 0.7763    | 0.8235 | 0.7992| 13 s     |
| DVN-VNet       | 0.6515    | 0.6887 | 0.6696| 17 s     |
| V-Net          | 0.7641    | 0.7330 | 0.7482| 23 s     |
| Schneider et al. | 0.4807    | 0.8603 | 0.6168| n/a      |

**b) Experiment 3: Bifurcation Detection:** For a quantitative evaluation of DeepVesselNet in bifurcation detection, we use synthetically generated data, and adopt a two-input-channels strategy. We use the vessel segmentations from experiments 1 as one input channel and the centerline predictions from Experiment 2 as a second input channel relying on the same training and test splits as in the previous experiments.

**APPENDIX C FULL RESULTS AND DISCUSSION ON CENTERLINE AND BIFURCATION DETECTION TASKS**

*a) Experiment 2: Centerline Prediction:* For centerline prediction, we train DeepVesselNet on the synthetic dataset and test it on synthetic as well as clinical datasets. The network uses the probabilistic segmentation masks from experiment 1 as an input (together with the same training and test splits described in experiment 1) and uses the vessel predictions to restrict predicted centerlines to be within vessel regions. Qualitative results are presented in Figs. [8] and [A.3] together with quantitative scores in Table [V]. We obtain a Dice score of 0.80 for DeepVesselNet-FCN, outperforming all other methods by a margin of more than 5%. Here we note that DeepVesselNet-FCN shows a higher precision (0.78 vs 0.48) than the morphological operations based method of Schneider et al. [26] which is one of the standard method for centerline prediction. The results also show that DeepVesselNet-FCN is better than the competing methods in terms of execution time. Again, DeepVesselNet-VNet performs slightly worse than V-Net in terms of the Dice score (0.67 vs. 0.75). However, DeepVesselNet-VNet has an advantage in speed of execution (17s vs. 23s) as expected.

**TABLE V**

| Method         | Precision | Recall | Dice  | Ex. Time |
|----------------|-----------|--------|-------|----------|
| DVN-FCN        | 0.7763    | 0.8235 | 0.7992| 13 s     |
| DVN-VNet       | 0.6515    | 0.6887 | 0.6696| 17 s     |
| V-Net          | 0.7641    | 0.7330 | 0.7482| 23 s     |
| Schneider et al. | 0.4807    | 0.8603 | 0.6168| n/a      |

**b) Experiment 3: Bifurcation Detection:** For a quantitative evaluation of DeepVesselNet in bifurcation detection, we use synthetically generated data, and adopt a two-input-channels strategy. We use the vessel segmentations from experiments 1 as one input channel and the centerline predictions from Experiment 2 as a second input channel relying on the same training and test splits as in the previous experiments.
In our predictions we aim at localizing a cubic region of size $5 \times 5 \times 5$ around the bifurcation points, which are contained within the vessel segmentation. We evaluate the results based on a hit-or-miss criterion: a bifurcation point in the ground truth is counted as hit if a region of a cube of size $5 \times 5 \times 5$ centered on this point overlaps with the prediction, and counted as a miss otherwise; a hit is considered as true positive (TP) and a miss is considered as false negative (FN); a positive label in the prediction is counted as false positive (FP) if a cube of size $5 \times 5 \times 5$ centered on this point contains no bifurcation point in the ground truth. Qualitative results on synthetic and clinical MRA TOF are shown in Figs. 9 and A.4, respectively. Results for [26] are obtained by first extracting the vessel network and then all nodes with two or more splits are treated as bifurcations – this being one of the standard methods for bifurcation extraction. In Fig. A.5, we present the precision-recall curve obtained by varying the threshold for converting probability maps, from the networks, into binary predictions. Results from Table VI and Fig. A.5 show that DeepVesselNet-FCN performs better than the other architectures in 5 out of 6 metrics. In our experiments, it became evident that V-Net tends to over-fit, possibly suffering from its high number of parameters to be determined from the rather few bifurcations present in our training data. This may explain why results for V-Net are worse than all other methods, also suggesting that in cases where little training data is available, the DeepVesselNet-FCN architecture may be the preferable.

Table VI

| Method      | Precision | Recall | Det. | Mean Err | Std Err | Ex. Time |
|-------------|-----------|--------|------|----------|---------|----------|
| DVN-FCN     | 0.7880    | 0.9297 | 86.87| 0.2090   | 0.6671  | 13 s     |
| DVN-VNet    | 0.4680    | 0.5670 | 84.21| 1.6533   | 0.9645  | 17 s     |
| V-Net       | 0.2550    | 0.6871 | 70.29| 1.2434   | 1.3857  | 23 s     |
| Schneider et al | 0.7718  | 0.8508 | 84.30| 0.1529   | 0.7074  | n/a      |

Fig. A.4. Bifurcation detection on clinical MRA test data using DeepVesselNet-FCN (bifurcations in green). At regions where a lot of vessels intersect, the network predicts it as a big bifurcation, this can be seen in the circled regions in zoomed images (a, b, and c).

Fig. A.5. Precision-recall curve of the results from the bifurcation detection task obtained by varying the threshold for converting probabilities maps into binary predictions.
APPENDIX D
THE VNET NETWORK ARCHITECTURE

Fig. A.6. Our DeepVesselNet-VNet architecture implementation. We replace full 3-D filters in the V-Net architecture proposed by [41] with cross-hair filters and use it for comparison against the proposed DeepVesselNet-FCN.