

Synthesizing Filamentary Structured Images with GANs

He Zhao^{*†}, Huiqi Li^{*}, Li Cheng^{†‡}

^{*}Beijing Institute of Technology, China

[†]Bioinformatics Institute, A*STAR, Singapore

[‡]School of Computing, National University of Singapore, Singapore

Abstract—This paper aims at synthesizing filamentary structured images such as retinal fundus images and neuronal images, as follows: Given a ground-truth, to generate multiple realistic looking phantoms. A ground-truth could be a binary segmentation map containing the filamentary structured morphology, while the synthesized output image is of the same size as the ground-truth and has similar visual appearance to what have been presented in the training set. Our approach is inspired by the recent progresses in generative adversarial nets (GANs) as well as image style transfer. In particular, it is dedicated to our problem context with the following properties: Rather than large-scale dataset, it works well in the presence of as few as 10 training examples, which is common in medical image analysis; It is capable of synthesizing diverse images from the same ground-truth; Last and importantly, the synthetic images produced by our approach are demonstrated to be useful in boosting image analysis performance. Empirical examination over various benchmarks of fundus and neuronal images demonstrate the advantages of the proposed approach.

I. INTRODUCTION

A rather broad range of important biomedical images are filamentary structured in nature, including retinal fundus images, neuronal images, among many others. Take fundus images for example, topological and geometrical properties of vessel filamentary structures provide valuable clinical information in diagnosing diseases such as proliferative diabetic retinopathy, glaucoma, and hypertensive retinopathy [1]. Often, only a handful of annotated images are available, where the filamentary structures are delineated by domain experts through laboriously manual process — a typical situation that exists in many biomedical applications.

On the other hand, image analysis and image synthesis have long been regarded as tightly intertwined techniques, which include e.g. the research works in analysis-by-synthesis [2], as well as the recent progresses in human full-body and hand pose estimation problems [3], [4], where the synthesis processes or synthesized data play an important role in addressing the analysis tasks. In the meantime, although a large number of research activities have been engaged in filamentary structured image analysis (see e.g. the survey papers of retinal and neuronal image analyses [1], [5], [6]), relatively very little effort exists concerning synthesizing such images.

In this paper, we present a learning based approach for synthesizing filamentary structured biomedical images. Our

paper possesses the following major contributions: First, our synthesis model can be effectively learned in data-driven fashion from a relatively small sample size. For example, we have successfully constructed synthesis models for STARE [7] and DRIVE [8] fundus image benchmarks, where the corresponding training images are merely 10 and 20 images, respectively. Second, based on a single ground-truth input, our approach is capable of synthesizing multiple distinct images. This capacity of introducing diversity is important in biomedical data simulations. Third, the synthesized images are shown useful in improving image segmentation performance. In other words, a baseline supervised segmentation module is shown to improve its segmentation performance on real-world datasets when trained with additional images synthesized by our approach. Fourth, to our knowledge our approach¹ is among the first attempt toward data-driven synthesis of filamentary structured images, including fundus and neuronal images. More specifically, two variants of our proposed pipeline have been proposed and studied, which we refer to as *Fila-GAN* and *Fila-sGAN*, respectively. Extensive experiments on various applications and datasets demonstrate their ability in producing realistic looking images, as well as in boosting the performance of a baseline segmentation module.

II. RELATED WORKS

Synthesizing filamentary structured images: There have been several existing research activities in simulating filamentary structured images. In synthesizing retinal vascular structures, one notable application area is surgical simulations [9]. Others are driven by the practical demand in empirical evaluations of segmentation or tracing methods. A critical issue with retinal image analysis lies in the lack of ground-truth annotations, due to its expensive and laborious nature. This is further complicated by the inter- and intra- observer variabilities of expert human observers that are subjective and prone to annotation errors [10]. Synthesizing retinal phantoms [11], [12] may be useful in this context given its unique advantage of having complete and unambiguous knowledge of the corresponding ground-truths. In [11], the specific vascular morphology and texture are constructed from scratch based heavily on domain knowledge. The work of [12] instead

¹Note our implementation, together with the related results, are made publicly available at https://web.bii.a-star.edu.sg/archive/machine_learning/Projects/filaStructObjs/Synthesis/index.html.

aims to derive vessels and textures from real data, which is based on their morphological and textural statistics and by utilizing active shape contour and Kalman filter techniques. Neuronal image synthesis has also been studied with significant biological prior knowledge, where GENESIS, NEURON, and L-Neuron are probably the most well-known efforts. GENESIS [13] is a simulation environment for constructing realistic models of neurobiological systems. It was one of the first simulation systems specifically designed for modeling nervous systems; NEURON [14] is similarly developed as a simulation environment for modeling individual neurons and networks of neurons; L-Neuron [15] generates and studies anatomically accurate neuronal phantoms based on sets of recursive rules described by a Lindenmayer system or L-system. Different from the past efforts, our goal in this paper is to synthesize realistic-looking retinal phantoms in an end-to-end fashion with data-driven approach.

Image style transfer, filamentary structured image segmentation, and data augmentation: The problem of image style transfer has been studied in the past twenty years [16], [17]. Recently impressive results are obtained by Gatys et al. [18] through the successful application of deep learning techniques. It has been further improved by a number of works including [19], [20] with a preference toward being efficient and light-weight. To our knowledge, the proposed approach is the first such attempt to incorporate style transfer techniques in simulating filamentary structured images. Meanwhile, there has been vast literature on filamentary structured image segmentation, interested readers may refer to [21], [22] for more thorough reviews. In spite of these research efforts, it remains challenging to precisely segment 2D and 3D image-based filaments. This is evidenced by e.g. the recent BigNeuron initiative [23] that calls for innovations in addressing the demands from neuronal science community where a significant number of neuronal images have been routinely produced in wet labs, while there still lack sufficiently accurate tools to automatically segment the neurite structures. Our work may shed lights on the feasibility of utilizing the generated phantoms in improving segmentation performance. Furthermore, there have been research efforts to enrich the training dataset by means of data augmentation, such as cropping, flipping and rotating existing training images [24], [25] as well as applying small perturbations in the color or intensity imaging space [26]. In a sense, contrary to the aforementioned methods that mechanically memorizes the original image style or texture, our approach could be regarded as a more systematic way to enrich existing training examples with phantoms of similar style.

Generative adversarial networks: The advancement of deep learning techniques [27], [28], [29] have led to significant progress in generating photo-realistic images using techniques such as generative adversarial networks (GANs) [28]. It considers a two-player zero-sum game between two agents, a discriminator net and a generator net. The discriminator is to tell the real inputs apart from the faked ones, while the role of the generator is to fool the discriminator by synthesizing instances that appear to have come from the real data distribution. Subsequently, a number of GANs variants [30],

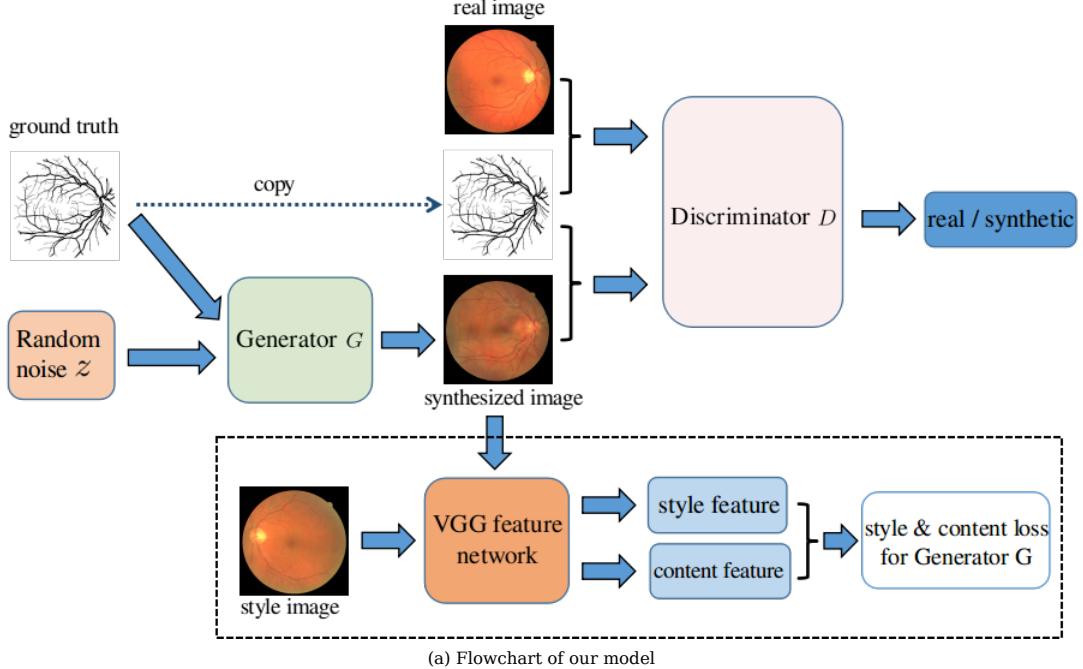
[31], [32], [33], [34] have been developed. Among them, DCGAN [30] introduces a set of constraints to stabilize the training dynamics between the generator and the discriminator. CGAN [31] facilitates the training of a synthesis model to generate images conditioned on some auxiliary information. LAPGAN [32] uses a cascade of convolutional networks within a Laplacian pyramid framework to generate the images from coarse to fine. InfoGAN [33] allows to learn disentangled representations in a completely unsupervised manner.

The closest work is perhaps that of [35], which also utilizes techniques similar to U-Net [36] to preserve global structural information during data generation process. Different from [35] which tends to generate only fixed output for a given input, our approach is able to produce unlimited number of individual phantoms from the same input. Moreover, instead of working with few hundreds to millions of training examples as of [35], our approach works with only tens of training images. Finally, our results are shown to boost the retinal segmentation performance when being included as additional training examples.

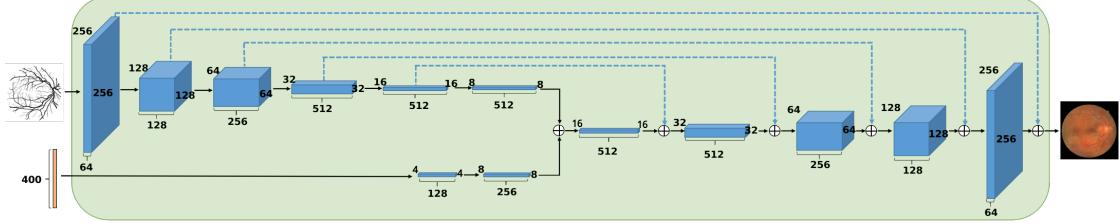
III. OUR APPROACH

In this work, we aim to learn a direct mapping from a segmentation (could be a ground-truth) back to a plausible raw filamentary structured image. More specifically, denote $\mathbf{x} \in \mathbb{R}^{W \times H \times 3}$ a RGB filamentary structured image, $\mathbf{y} \in \{0, 1\}^{W \times H}$ the corresponding segmentation (i.e. ground-truth annotation). By imitating the image formation processing, let $G_\theta : (\mathbf{y} \in \mathbb{R}^{W \times H}, \mathbf{z} \in \mathbb{R}^Z) \rightarrow \hat{\mathbf{x}} \in \mathbb{R}^{W \times H \times 3}$ denote the image generation function that takes a ground-truth binary image \mathbf{y} and a noise code \mathbf{z} as input, to produce a filamentary structured phantom $\hat{\mathbf{x}}$. Our goals are four-fold: (1) Learn the θ -parameterized function G from a usually very small training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$; (2) Be capable of exploring the underlying conditional image formation distribution $p(\mathbf{X}|\mathbf{y})$, where \mathbf{X} is a random variable denoting the feasible filamentary structured image realization conditioning on the particular realization \mathbf{y} . In other words, to sample plausible yet distinct RGB image instances from a same input \mathbf{y} , obtained by varying the noise code \mathbf{z} ; (3) These synthesized images are useful in boosting the performance of a supervised segmentation method when adding up to the training set; (4) Consider an interesting variant of our approach where a specific image style obtained from *one* additional input image \mathbf{x}_s can be directly transferred to the output phantom $\hat{\mathbf{x}}$. Here the style of \mathbf{x}_s could be very different from that of \mathbf{x} , and the corresponding contents (i.e. segmentations \mathbf{y}_s and \mathbf{y}) are usually unrelated. The aforementioned goals seems daunting: Given the intricate nature of the image formation process, G could be a rather sophisticated function, and the situation is exacerbated by the small n nature of the problem. Nonetheless, by resorting to the powerful deep learning framework of GANs, an end-to-end learning machine is proposed, as depicted in Fig. 1.

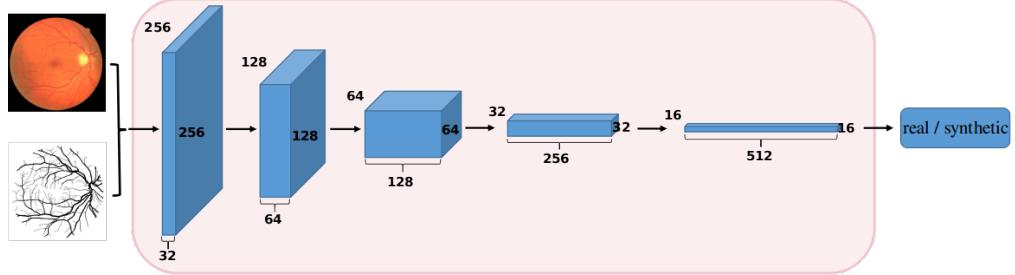
In addition to the generator G_θ , consider a discriminator function $D_\gamma : (\mathbf{X} \in \mathbb{R}^{W \times H \times 3}, \mathbf{y} \in \mathbb{R}^{W \times H}) \rightarrow d \in [0, 1]$, whose role is to tell apart synthesized phantom $\mathbf{X} := \hat{\mathbf{x}}$ (ideally $d \rightarrow 0$) from real filamentary structured image



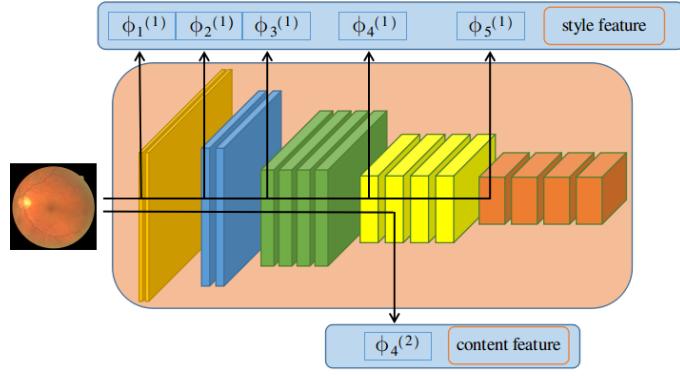
(a) Flowchart of our model



(b) Generator G



(c) Discriminator D



(d) VGG feature network

Fig. 1: (a) Flowchart of our approach, which contains the generator and the discriminator networks as detailed in (b) and (c). The dimensions of all layers are specified. The VGG feature networks are described in (d), where the top row indicates the specific layers (e.g. $\phi_1^{(1)}, \phi_2^{(1)}, \dots$) extracted as style features, while bottom row are the layers (e.g. $\phi_4^{(2)}$) used as content features. See text for details.

$X := \mathbf{x}$ (ideally $d \rightarrow 1$), as visually explained in Fig. 1(a). We follow the GANs idea for this two-player zero-sum game setting and consider the following optimization problem that characterizes the interplay between G and D :

$$\min_{\theta} \max_{\gamma} L(G_{\theta}, D_{\gamma}) = \mathbb{E}_{\mathbf{x}, \mathbf{y} \sim p(\mathbf{x}, \mathbf{y})} [\log D_{\gamma}(\mathbf{x}, \mathbf{y})] + \mathbb{E}_{\mathbf{y} \sim p(\mathbf{y}), \mathbf{z} \sim p(\mathbf{z})} [\log (1 - D_{\gamma}(G_{\theta}(\mathbf{y}, \mathbf{z}), \mathbf{y}))] + \lambda L_{\text{DEV}}(G_{\theta}), \quad (1)$$

with $\lambda > 0$ being a trade-off constant. Here the last term is introduced to ensure the synthesized image will not deviate significantly from the real image, and we consider a simple L1 loss

$$L_{\text{DEV}}(G_{\theta}) = \mathbb{E}_{\mathbf{x}, \mathbf{y} \sim p(\mathbf{x}, \mathbf{y})} [\|\mathbf{x} - G_{\theta}(\mathbf{y}, \mathbf{z})\|_1]. \quad (2)$$

In summary, during training generator G tries to synthesize realistic-looking images that can fool the discriminator D , by minimizing the objective function of Eq. (1). In practice, following the approximation scheme of [28], it is realized by minimizing a simpler form $-\log(D_{\gamma}(G_{\theta}(\mathbf{y}, \mathbf{z})))$, instead of the original $\log(1 - D_{\gamma}(G_{\theta}(\mathbf{y}, \mathbf{z})))$. To summarize, learning the generator G amounts to minimizing

$$L_G(G_{\theta}) = - \sum_i \log D_{\gamma}(G_{\theta}(\mathbf{y}_i, \mathbf{z}_i), \mathbf{y}_i) + \lambda \|\mathbf{x}_i - G_{\theta}(\mathbf{y}_i, \mathbf{z}_i)\|_1. \quad (3)$$

On the other hand, discriminator D attempts to properly separate the real images from synthesized ones by maximizing Eq. (1), while discriminator D is learned by maximizing

$$L_D(D_{\gamma}) = \sum_i \log D_{\gamma}(\mathbf{x}_i, \mathbf{y}_i) + \log (1 - D_{\gamma}(G_{\theta}(\mathbf{y}_i, \mathbf{z}_i), \mathbf{y}_i)). \quad (4)$$

Note the empirical sum is used in practice to approximate the expectation. The learning process is practically carried out by alternating between these two optimization operations that are similarly adopted by GANs and variants [28], [30], [35]. Unfortunately there does not exist a formal guarantee that this optimization procedure will reach Nash equilibrium. That being said, in practice we have observed convergence traits with reasonable image synthesis outputs, as also been demonstrated in the above GANs related works. The top part of Fig. 1(a) (i.e. excluding elements in the dashed box) illustrates the overall work flow of our approach so far, also referred to as *Fila-GAN*. In the following, we describe in detail the specific neural net architectures of our functions G and D .

A. Generator G and Discriminator D

The commonly used encoder-decoder strategy [37], [38], [39], [35] is adopted here, which allows the introduction of noise code in a natural manner. The encoder part acts as a feature extractor where the multiple layered structure captures local to more global data representations. The 400-dimensional random code \mathbf{z} is fully connected to the first layer, which is then reshaped. Note that unless otherwise specified, for all layers of G and D , we use kernel size 4 and stride 2 without any pooling layer. Meanwhile, in our context it is crucial for the generator function to respect the input filamentary structured morphology during image generating process. As such, the skip connections of U-Net [36] are

also considered here. That is, previous layer is duplicated by appending onto current layer in a mirrored fashion that skips odd-number of layers with the center coding layer as its origin. It is worth noting that when image size is small and the network model is shallow, the encoder-decoder framework may work well even without any skip connection. However, as we are working with large image size (e.g. 512×512) and deeper networks, training such model becomes problematic. This might be due to the effects of vanishing gradients over long paths in error back-propagation. The skip connection, by playing a similar role to the residual nets [40], allows the additional direct flow of error gradients from decoder layers to corresponding encoder layers. This facilitates the proper memorization of global and local shape contents as well as the corresponding textures encountered in the training set, and empirically helps to generate much better results. We follow the basic network architecture proposed in [30], to build the layers of the generator with multiple Convolution-BatchNorm-LeakyRelu components as in Fig. 1(b). The activation function of the output layer is *tanh* to squash the value between -1 and 1. On par with our generator, the same Convolution-BatchNorm-LeakyRelu building blocks are used in building our discriminator as in Fig. 1(c). Here the activation function of the output layer is instead *sigmoid*. The feature map sizes are halved after each convolutional layer (eg. The input size is 512×512 , after one convolutional layer the size becomes 256×256) while the number of filters (i.e. number of feature maps) are doubled from 32 all the way to 512.

Up to now, our approach considers to learn a generic representation from a (usually limited) set of training examples, which is then employed in generating filamentary structured phantoms, which is also referred to as *Fila-GAN*. Next, we consider a variant inspired by the recent style transfer techniques.

B. *Fila-sGAN*: the Style transfer variant

Inspired by the recent advance in image style transfer, here we consider an alternative variant of our approach as illustrated in Fig. 1(a) (now including the components in the dashed box): Given an input segmentation \mathbf{y} that delineates its filamentary structured content, the generated phantom $\hat{\mathbf{x}}$ is expected to possess the distinct style (i.e. texture) of a target style input \mathbf{x}_s , while still adhering to the content of \mathbf{y} that has been presented during training stage. This variant is also termed *Fila-sGAN*, to highlight the fact that the synthesized image is now based on a *particular* style representation provided by \mathbf{x}_s , instead of the generic representation as we have aimed for in our basic approach *Fila-GAN*. This is made possible by introducing a style image as an additional training input, i.e. a new filamentary structured image \mathbf{x}_s that comes with a different style or texture. Note in general \mathbf{x}_s has its own filamentary structure (aka segmentation), which is usually different from the other input argument \mathbf{y} , nonetheless this should not affect the synthesis performance of our approach. It is worth noting that this design make practical sense in biomedical imaging setting: On one hand very few images are annotated, while on the other hand, there could be abundant

un-annotated images out there that could serve as potential style inputs.

The training and testing processes of this variant largely remain the same as what we have described in our approach: The training is still carried out in batch mode over the n annotated training examples. The aforementioned generator and the discriminator functions still carry on. The differences lie on the objective function of Eq. (1), where the last term $\lambda L_{\text{DEV}}(G_\theta)$ is now replaced by a new cost term $L_{\text{ST}}(G_\theta)$ to be defined below. Without loss of generality, we follow the style transfer idea of [19], [20] to utilize the convolutional neural network (CNN) of VGG-19 [41] to extract features from its multiple layers. Accordingly to the network architecture of VGG-19, it can be represented as a series of five CNN blocks of the VGG net, with each block containing 2-4 consecutive convolution layers of the same size. For notation convenience, let Γ indexes a set of CNN blocks, and for a particular block $\gamma \in \Gamma$, its set of layers is represented by $\Lambda(\gamma)$ or simply Λ , with a layer index being $\lambda \in \Lambda$. Now, for a filamentary structured image \mathbf{X} (being either the real \mathbf{x} or phantom $\hat{\mathbf{x}}$), denote by $\phi_\gamma^{(\lambda)}(\mathbf{X})$ the λ -th corresponding layer of CNN block γ . The VGG-19 net is obtained by training on the ImageNet image classification task, with its further details being displayed in Fig. 1(d). In terms of the resulting optimization problem, it is realized in our approach by explicitly incorporating two perceptual loss components of [18], namely the style loss and the content loss, as well as a total variation loss, which are to be described next.

Style loss: The style loss is used to minimize the textural deviation between the target style \mathbf{x}_s and the phantom $\hat{\mathbf{x}}$. To achieve this, consider Γ_s indexing the set of CNN blocks, and for each block index $\gamma_s \in \Gamma_s$, its set of layers being represented by Λ_s . Now, define the λ_s -th layer of block γ_s , $\phi_{\gamma_s}^{(\lambda_s)}(\mathbf{X})$, where $\mathbf{X} = \mathbf{x}_s$ or $\mathbf{X} = \hat{\mathbf{x}}$. With a slight abuse of notation, denote by $|\lambda_s|$ the number of feature maps in current layer λ_s . Let i (or j) index the feature map of interest, and let k index an element of the current feature map i (or j). The corresponding feature information is characterized by the *Gram matrix* $\mathbb{G}_{\gamma_s}^{(\lambda_s)}(\mathbf{X}) \in \mathbb{R}^{|\lambda_s| \times |\lambda_s|}$, where each element $\mathbb{G}_{\gamma_s,i,j}^{(\lambda_s)}$ defines an inner product between the i -th and j -th feature maps in λ_s -th layer of γ_s -th block:

$$\mathbb{G}_{\gamma_s,i,j}^{(\lambda_s)} = \sum_k \phi_{\gamma_s,ik}^{(\lambda_s)} \phi_{\gamma_s,jk}^{(\lambda_s)}. \quad (5)$$

Now the style loss between \mathbf{x}_s and $\hat{\mathbf{x}}$ during training becomes

$$l_{\text{sty}}(G_\theta) = \sum_{\gamma_s \in \Gamma_s, \lambda_s \in \Lambda_s} \frac{\varpi_{\gamma_s}}{W_{\gamma_s} H_{\gamma_s}} \left\| \mathbb{G}_{\gamma_s}^{(\lambda_s)}(\mathbf{x}_s) - \mathbb{G}_{\gamma_s}^{(\lambda_s)}(\hat{\mathbf{x}}) \right\|_{\text{F}}^2, \quad (6)$$

where $\|\cdot\|_{\text{F}}$ is the matrix Frobenius norm, ϖ_{γ_s} denoting the weight of γ_s -th block *Gram matrix*. Note $\hat{\mathbf{x}} = G_\theta(\mathbf{y}, \mathbf{z})$ by definition.

Content loss: For content loss, consider the following notation: Let Γ_c index the set of CNN blocks, and for each block index $\gamma_c \in \Gamma_c$, its set of layers is denoted by Λ_c . As already stated, the synthesized phantom $\hat{\mathbf{x}}$ is expected to abide by the filamentary structure as prescribed in the real raw image \mathbf{x} of the segmentation input, which is carried out by encouraging them to match up, i.e. by minimizing the

following Frobenius norm of the difference between input and generated CNN features:

$$l_{\text{cont}}(G_\theta) = \sum_{\gamma_c \in \Gamma_c, \lambda_c \in \Lambda_c} \frac{1}{W_{\gamma_c} H_{\gamma_c}} \left\| \phi_{\gamma_c}^{(\lambda_c)}(\mathbf{x}) - \phi_{\gamma_c}^{(\lambda_c)}(\hat{\mathbf{x}}) \right\|_{\text{F}}^2. \quad (7)$$

Total variation loss: Furthermore, we consider encouraging spatial smoothness in the generated phantom by incorporating the following total variation loss:

$$l_{\text{tv}}(G_\theta) = \sum_{w,h} \left(\|\hat{x}_{w,h+1} - \hat{x}_{w,h}\|_2^2 + \|\hat{x}_{w+1,h} - \hat{x}_{w,h}\|_2^2 \right), \quad (8)$$

with $w, h \in W, H$, and $\hat{x}_{w,h}$ denotes the pixel value of given location in phantom image $\hat{\mathbf{x}}$.

The above-mentioned three loss functions together leads to $L_{\text{ST}}(G_\theta) = w_{\text{cont}} l_{\text{cont}} + w_{\text{sty}} l_{\text{sty}} + w_{\text{tv}} l_{\text{tv}}$. Thus we consider the above style transfer loss L_{ST} , instead of L_{DEV} as in Eq. (1). Accordingly, generator G now takes the objective function of the following form

$$L_G(G_\theta) = - \sum_i \log D_\gamma(G_\theta(\mathbf{y}_i, \mathbf{z})) + L_{\text{ST}}(G_\theta), \quad (9)$$

while the objective function of discriminator D of Eq. (4) remains unchanged. It is clear that in this variant, the style transfer contribution from the target style \mathbf{x}_s is obtained by back-propagation optimization of the above objective function, while the rest ingredients of our approach are kept as the same.

IV. EXPERIMENTAL DETAILS

Datasets and Preparation: Empirically our approach is examined on four standard benchmarks that covers a broad spectrum of filamentary structured images including both retinal blood vessels and neurons. They are DRIVE [8], STARE [7], high-res fundus or HRF [42], as well as 2D Neurons or NeuB1 [43]. The image sizes and the amount of training examples are also rather different across these datasets: DRIVE contains 20 training examples and 20 test images, with each of size 584×565 . STARE and HRF are two fundus image datasets with image sizes being 700×605 and $3,304 \times 2,336$ respectively, as well as split of training/testing images being 10/10 and 22/23 respectively. The NeuB1 dataset is considered here for microscopic neuronal images, which contains 112 images of size 512×512 . We also follow the standard train/test split of 37/75 as in [43].

To summarize, the image sizes of DRIVE, STARE, and NeuB1 are roughly similar, while HRF contains high resolution (and subsequently much larger size) images. Then in preprocessing, raw images of these first three datasets are all resized to a standard size of 512×512 , as follow: For DRIVE, as all images are of size 584×565 and contain a relatively large-size background area (determined if a pixel is outside of a prescribed circular-shaped mask), they are cropped into 565×565 sub-images centered around the original ones, that ensures all foreground pixels are still contained in the cropped images. They are further resized to 512×512 by bicubic interpolation; For STARE, as the images possess rather small

background margins outside of its foreground mask, they are directly resized to the same size of 512×512 by bicubic interpolation; Now for HRF, as its raw images are of much larger size of 3304×2336 , these images are all resized to $2,048 \times 2,048$ instead of the 512×512 template size we have used previously, in order to retain sufficient information of the original images. The pixel values of all input image signals are scaled to the same range of $[-1, 1]$, for each of the benchmarks. As a consequence, the learned generator in our approach will produce a phantom image of size 512×512 for DRIVE, STARE, and NeuB1, and size $2,048 \times 2,048$ for HRF, respectively. These images are subsequently upsampled to their original sizes. For any of the above-mentioned fundus image datasets, the final results are obtained by applying its prescribed circular-shaped mask, so only inside pixels are retained as foreground.

Model Architecture and internal parameters: Fig. 1(b-d) displays the architecture of our approach: In the generator and discriminator modules, each 3D box denotes a CNN layer consisting its feature maps, a directed edge usually represents a convolutional (or deconvolutional) operation with a filter size $w_f \times h_f \times l_f$. In this paper we consider $w_f = h_f = 4$ pixels with l_f self-manifested by the third dimension of its consecutive layer. Note Fig. 1(b-c) specify the intrinsic parameter values of our networks G and D , such as the size of each layers, and the length of the noise code $Z = 400$. In particular, in generator G , the \oplus sign together with the two directed edges pointing to it denote a concatenation operation. For example, the first \oplus sign shows a concatenation operated on a $8 \times 8 \times 512$ tensor and a $8 \times 8 \times 256$ tensor that produces a $8 \times 8 \times 768$ tensor; This is followed by a deconvolutional operation of filter size $4 \times 4 \times 512$ that produces the 3D box of size $16 \times 16 \times 512$.

Throughout the experiments in our approach, the following internal parameters are empirically considered: The TensorFlow deep learning library is used with training epochs being fixed to 100. To initialize the neural net weights of discriminator D and generator G (i.e. parameters θ, γ), a [-0.04, 0.04] truncated normal distribution of zero-mean and standard deviation of 0.02 is engaged. The weights θ of G is updated with mini-batch of size 1 using Adam optimizer [44]. Meanwhile, the vanilla stochastic gradient descent is employed for D to update γ . Learning rate is set to 0.0002 for the generator and 0.0001 for the discriminator during back-propagation training. λ in our *Fila-GAN* is set to 100. To balance the learning progress of G and D , we choose to update G twice then update D once during each learning iteration. During training, the noise code is sampled elementwise from zero-mean Gaussian with standard deviation 0.001; At testing run, it is sampled in the same manner but with a different standard deviation of 1. Empirically this is found useful in maintaining proper level of diversity for our small sample-size situation. We follow the practice of [30] and choose not to apply batch normalization to the output layer of G as well as to the input layer of D . Otherwise, batch normalization [45] is utilized right after each convolutional layer, as overall better model training behaviors have been observed for both G and D nets.

For our style transfer variant, *Fila-sGAN*, the VGG-19 nets are employed to produce the feature descriptors. Some of their

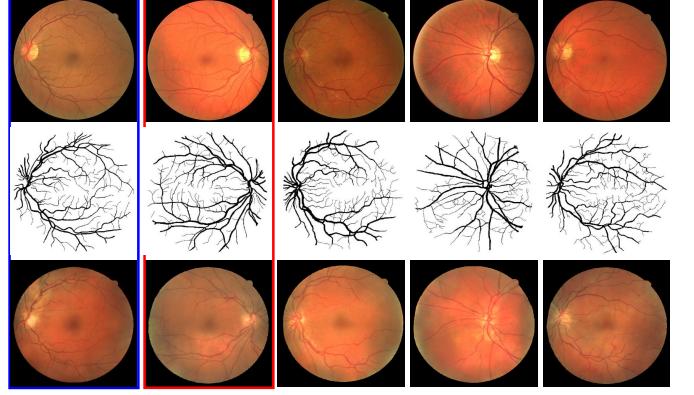


Fig. 2: Exemplar DRIVE phantoms generated by *Fila-GAN*. For each column, 1st row presents a real image, 2nd is the corresponding ground-truth, 3rd displays one generated phantom.

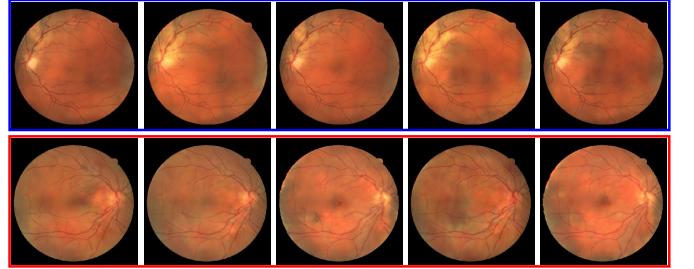


Fig. 3: The two rows in this figure display multiple synthesized results of the first two columns of Fig. 2 respectively, to showcase the ability of our approach in generating diverse outputs.

layers are used for extracting style/content features, which are illustrated in Fig. 1(d), and are specified as follows: The sets of block indices for style and content losses are $\Gamma_s = \{1, 2, 3, 4, 5\}$, and $\Gamma_c = \{4\}$, respectively. Besides, for any block $\gamma_s \in \Gamma_s$, its particular set of layer indices is $\Lambda_s(\gamma_s) = \{1\}$ for the style loss, and $\Lambda_c = \{2\}$ for the content loss. ϖ_{γ_s} is fixed to 0.2 over all blocks in Γ_s . Meantime, the weights of the three respective loss functions, namely $w_{\text{cont}}, w_{\text{sty}}, w_{\text{tv}}$, are 1, 10, and 100, accordingly.

Computation Time: All the experiments are carried out on a standard PC with Intel iCore 7 CPU and Titan-X GPU with 12GB memory. Our implementation of the proposed *Fila-GAN* and *Fila-sGAN* is in Python. Training time of *Fila-GAN* and *Fila-sGAN* on DRIVE takes around 108 minutes and 184 minutes, respectively. The average run-time speed on synthesizing a DRIVE-size image is 0.4471s.

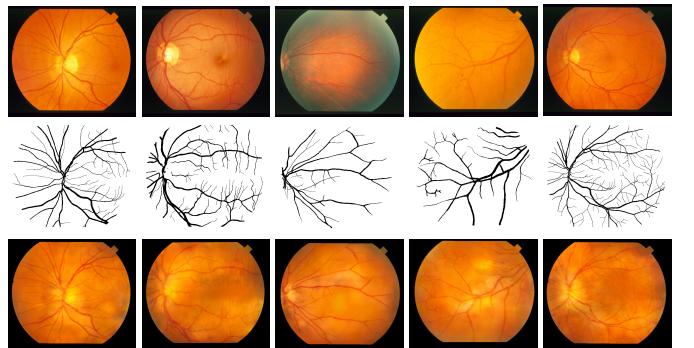


Fig. 4: Exemplar STARE phantoms generated by *Fila-GAN*. For each column, 1st row presents a real retinal image, 2nd is the corresponding ground-truth, while 3rd displays one generated phantom.

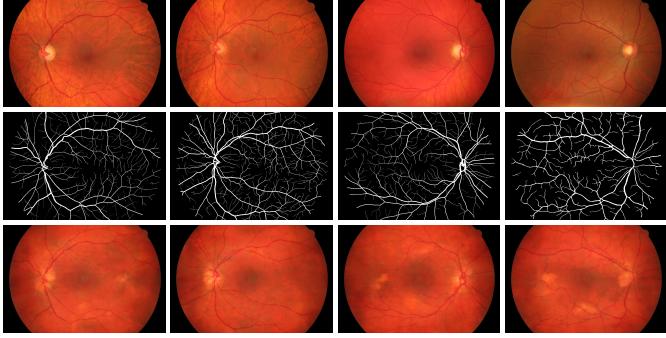


Fig. 5: Exemplar HRF phantoms generated by our *Fila-GAN* model. For each column, 1st row presents a real retinal image, 2nd is the corresponding ground-truth, while 3rd displays one generated phantom. A zoomed-in figure is presented in the supplementary file.

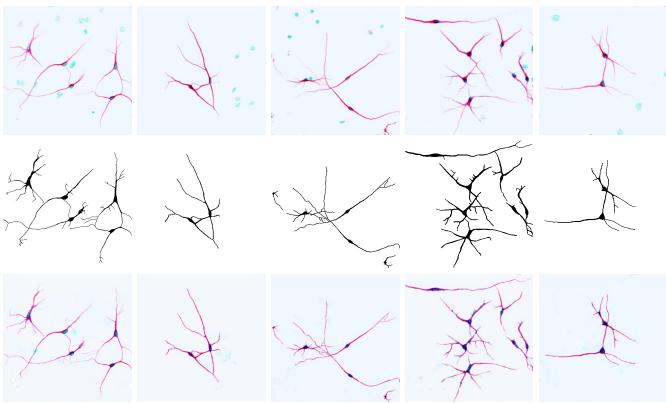


Fig. 6: Exemplar NeuB1 phantoms generated by our *Fila-GAN*. For each column, 1st row presents a real neuronal image, 2nd the corresponding ground-truth, while 3rd displays one generated phantom.

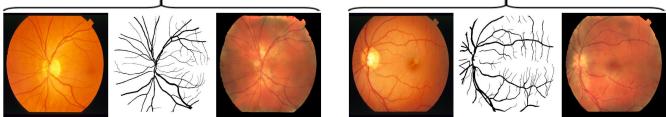


Fig. 7: Apply the *Fila-GAN* model trained on DRIVE training set (the same model used in Fig. 2) to STARE segmentation maps. Two random samples are selected, from left to right are real STARE image, ground-truth, and generated phantom, respectively.

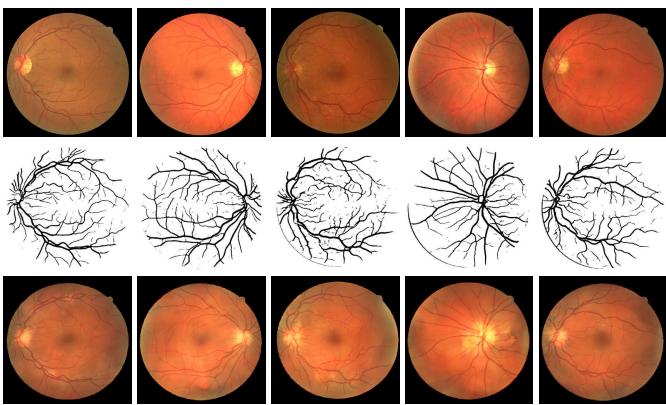


Fig. 8: Exemplar phantoms generated from segmentation predictions (instead of ground-truths) of a typical segmentation method trained on DRIVE. In each column, 1st row presents a real DRIVE image, 2nd the corresponding ground-truth, 3rd displays one generated phantom.

V. EMPIRICAL EXPERIMENTS

A. Visual results

Fig. 2 displays exemplar synthesized results of applying *Fila-GAN* on DRIVE, where the first row presents the real images, the second row refers to the corresponding ground-truth, and the last row displays the generated phantoms. It can be observed that the phantoms preserve the vascular morphology of the input (the second row in figure), while being able to present different yet realistic-looking texture appearances. It is interesting to note that the very bright colored areas are usually properly situated around the optical disk locations of the ground-truth images, which suggests that our phantom generation model could capture such intrinsic correlations without explicit human interventions for conveying such prior knowledge. Similar results can also be found in other datasets, e.g. STARE (Fig. 4), HRF (Fig. 5), and neuronal dataset NeuB1 (Fig. 6). To demonstrate the strength of *Fila-GAN* in generating multiple distinct syntheses from the same ground-truth, Fig. 3 further presents more synthesized results of the first and second ground-truth inputs as of Fig. 2, obtained by randomly drawing i.i.d. noise codes z . Clearly, for each of these ground-truth input, the results synthesized by *Fila-GAN* are visually different. It can be observed that *Fila-GAN* is relatively more powerful in emulating textural diversity and less in capturing illumination changes. In addition, this *Fila-GAN* model trained on DRIVE training set is applied to images from other datasets (here we consider the STARE dataset [7]), with exemplar results presented in Fig. 7: Two random samples are selected, from left to right are real image, ground-truth, and phantom, respectively. Not surprisingly, the two synthesized STARE phantoms bear DRIVE-like textures. Finally, to demonstrate that *Fila-GAN* works well even without ground-truth segmentation, an experiment is carried out to evaluate its behavior when a typical segmentation method is employed to produce binary segmentation maps. Note in fact any reasonable segmentation method should work, while we consider here a baseline convolutional neural nets segmentation method whose details are described in the supplementary file. As displayed in Fig. 8, the generated phantoms are visually also plausible. We note in the passing that promising results are also obtained on the BigNeuron 3D neuronal image stacks [6] as discussed in the supplementary file.

Now we switch gear to examine our style transfer variant, *Fila-sGAN*. Fig. 9 presents a gallery of collective results to the same set of DRIVE retinal images (shown in columns) when trained with different style images (shown in rows). Here the real images of hindsight are presented in the first row, followed by their corresponding ground-truths in the second row. From 3rd row onwards, the generated phantoms with different styles are presented. DRIVE, Kaggle², STARE denotes the three distinct style sources. It is observed that the texture style of each phantom synthesized by our *Fila-sGAN* is clearly controlled by its particular style input, which is especially pronounced for the Kaggle style images that are considerably different from the rest images. Meanwhile the

²Kaggle images are obtained from <https://www.kaggle.com/c/diabetic-retinopathy-detection>, a contest for diabetic retinopathy detection.

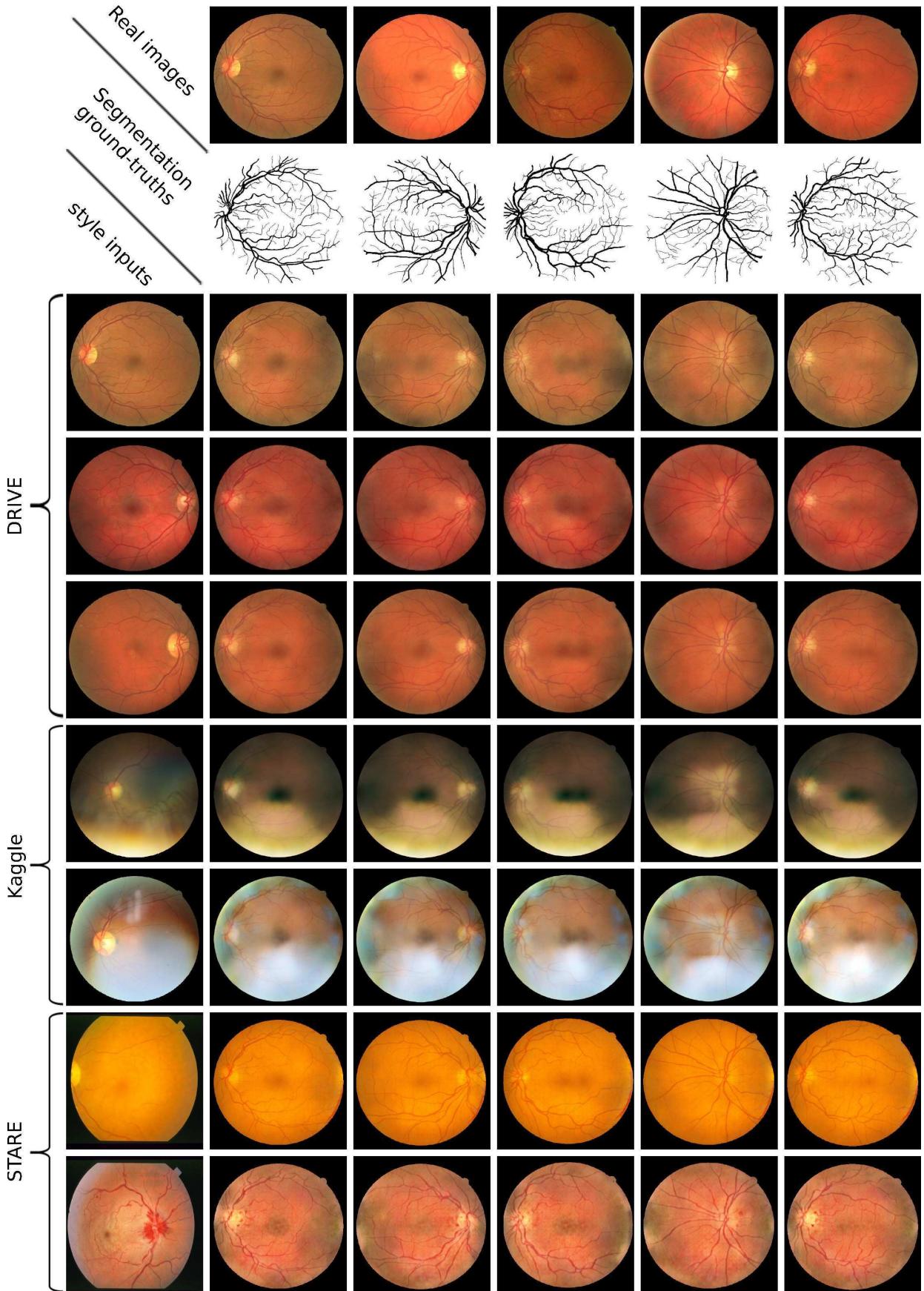


Fig. 9: Phantoms generated by *Fila-sGAN*. The first and second rows display the real DRIVE images in the hindsight, and corresponding ground-truths, respectively. From third row onwards, each row presents the phantoms synthesized from a specific style image shown in the first column. See text for details.

TABLE I: Quantitative results of evaluation scheme 1. Comparison of segmentation performance of a baseline segmentation method on the same DRIVE, STARE, HRF, NeuB1 test set while training on a different set of images (as depicted in the left-most column). Results are evaluated by average F1-score (%). See text for details.

	DRIVE	STARE	HRF	NeuB1
<i>synthetic images</i>	71.44	75.34	69.28	77.69
<i>real images</i>	79.15	78.11	78.68	83.91
<i>real + synthetic images</i>	80.33	79.02	79.50	85.06
$2 \times$ <i>real images</i>	80.70	79.66	79.77	85.40

respective filamentary structures are well preserved. Moreover, visually diverse results are again generated by varying the noise input z , as is presented in Fig.4 of the supplementary file. Visually *Fila-sGAN* is shown to be capable of producing realistic looking phantoms of very different styles.

B. Quantitative results

It is often difficult to quantitatively evaluate the quality of synthesized results, which is also mentioned in [35], [46]. We start by assuming to have access to any reasonable supervised segmentation method (here we consider the aforementioned baseline CNN segmentation method). In our context, we consider two relatively straightforward evaluation schemes: Scheme 1 is to examine the usefulness of these newly generated images in boosting the performance of the same segmentation method; Scheme 2 is through investigating the differences in segmentation results, when applying the same trained segmentation model on both synthesized and real retinal images.

Evaluation Scheme 1: As our baseline CNN segmentation method takes image patches as input, we start by preparing ready 800K image patches from real images in the training set which are evenly partitioned into the first 400K and the second 400K real image patches. Additionally we have another 400K image patches from the synthesized training images. This gives rise to 4 scenarios, which are: *synthetic images* where the segmentation baseline is trained on the 400K synthetic patches; *real images* where it is trained on the first set of 400K real image patches; *real + synthetic images* for training on the first set of 400K real patches plus the 400K synthetic patches; $2 \times$ *real images* where the training is on both sets of 400K real patches. As summarized in Table I, the segmentation model performs the worst when training only on synthetic images, which is to be expected. However, for example on DRIVE, compared with 79.15% F1-score when training on 400K real patches, the introduction of additional synthetic images, i.e. *real + synthetic images*, is demonstrated to improve the segmentation performance to 80.33%. As a side note, we observe that when replacing synthetic images by real ones, i.e. $2 \times$ *real images*, there is a further boost in segmentation performance to 80.70%. It suggests that overall synthesized images generated by *Fila-GAN* helps to improve segmentation performance; Meanwhile it is still preferable if more real data are available, which is to be expected. The same trend is consistently presented when working with all these different benchmarks. Further, similar empirical observations have also been made for our *Fila-sGAN* variant. Take the DRIVE benchmark for example,

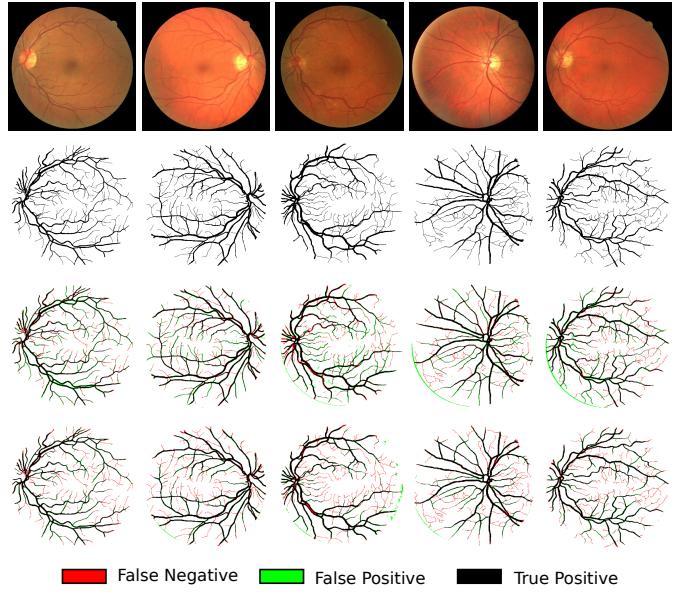


Fig. 10: visual comparison on segmentation results of real vs. synthetic images on DRIVE test set images (i.e. evaluation scheme 2). First row displays the real images, with the corresponding ground-truths shown in the second row. Third and fourth rows then present segmentation results of the real and synthesized phantom images, respectively. Note these phantom images are exactly the same images shown in Fig. 2. Here black refers to true positive (TP), green stands for false positive (FP), and red is false negative (FN).

as indicated in Table 1 of the supplementary file, similar patterns as presented in Table I for *Fila-GAN* also hold true for *Fila-sGAN*, where the segmentation results of *real + synthetic images* leads to a boost of performance to 80.49%. Finally, to put our segmentation results into context, we also cite the comparable average F1 score results of the state-of-the-art segmentation methods including SF-context distance [47] (78.86%), DRIU [48] (82.20%), among others on the DRIVE benchmark.

Evaluation Scheme 2: We follow the second evaluation scheme and inspect the segmentation results of real vs. generated phantoms when the same segmentation model trained on the real training set is employed. The segmentation method is firstly trained on the training set of the current benchmark in consideration, which is then applied on both real test set images and phantoms generated from the corresponding ground-truths. The widely used average F1-score is considered as our segmentation evaluation metric. Table 2 of the supplementary file summarizes quantitative results across the benchmark datasets of retinal and neuronal images. That quantitative performance on these two test sets (real vs. phantom) are indeed similar w.r.t. the same segmentation model. We attribute the relatively higher performance on the phantom images to their perhaps more homogeneous textural appearances. Visual comparison is also provided in Fig. 10. Given that the segmentation model is trained on the real training set, by and large the vascular structures are still well-segmented when applying to the phantom images. Compare with the segmentation results of real images, the phantoms tend to incur relatively less false positive pixels at the price of relatively more missing vessel pixels that are usually those tiny and thin filaments. It is also observed that segmentation results of different synthetic images based on the same segmentation

input seem to vary little, and the differences are mostly concentrated on these thin and tiny filaments.

VI. CONCLUSION

We propose a novel data-driven approach to synthesize filamentary structured images given a ground-truth input. The synthesized images are realistic-looking, and have been shown to boost image segmentation performance when used as additional training images. Moreover, the model is capable of learning from small training sets of as few as 10-20 examples. Future work includes investigation into related biomedical image datasets for the interplay of image synthesis, segmentation, and domain adaptation tasks.

ACKNOWLEDGMENT

HZ is supported by the CSC Chinese Government Scholarship. The project is partially supported by A*STAR JCO grants. We thank Xiaowei Zhang, Joe Wu, and Malyatha Shridharan for their help with this project.

REFERENCES

- [1] M. Abramoff, M. Garvin, and M. Sonka, “Retinal imaging and image analysis,” *IEEE Trans. Med. Imag.*, vol. 3, pp. 169–208, 2010.
- [2] U. Grenander, *Lectures in Pattern Theory I, II and III: Pattern Analysis, Pattern Synthesis and Regular Structures*. Springer-Verlag, 1976–1981.
- [3] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake, “Efficient human pose estimation from single depth images,” *IEEE Trans. PAMI*, vol. 35, no. 12, 2013.
- [4] C. Xu and L. Cheng, “Efficient hand pose estimation from a single depth image,” in *ICCV*, 2013.
- [5] M. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, A. Rudnicka, C. Owen, and S. Barman, “Blood vessel segmentation methodologies in retinal images - a survey,” *Comput. Methods Prog. Biomed.*, vol. 108, no. 1, pp. 407–433, 2012.
- [6] H. Peng, M. Hawrylycz, J. Roskams, S. Hill, N. Spruston, E. Meijering, and G. Ascoli, “Bigneuron: Large-scale 3d neuron reconstruction from optical microscopy images,” *Neuron*, vol. 87, no. 2, pp. 252–6, 2015.
- [7] A. Hoover, V. Kouznetsova, and M. Goldbaum, “Locating blood vessels in retinal images by piece-wise threshold probing of a matched filter response,” *IEEE Trans. Med. Imag.*, vol. 19, no. 3, pp. 203–10, 2000.
- [8] J. Staal, M. Abramoff, M. Niemeijer, M. Viergever, and B. van Ginneken, “Ridge based vessel segmentation in color images of the retina,” *IEEE TMI*, vol. 23, pp. 501–9, 2004.
- [9] M. Sagar, D. Bullivant, G. Mallinson, and P. Hunter, “A virtual environment and model of the eye for surgical simulation,” in *ACCIT*, 1994.
- [10] K. Fritzsche, A. Can, H. Shen, C. Tsai, J. Turner, H. Tanenbaum, C. Stewart, and B. Roysam, “Automated model based segmentation, tracing and analysis of retinal vasculature from digital fundus images,” in *State-of-The-Art Angiography, Applications and Plaque Imaging Using MR, CT, Ultrasound and X-rays*, J. Suri and S. Laxminarayan, Eds. Academic Press, 2003, pp. 225–298.
- [11] S. Fiorini, M. Biasi, L. Ballerini, E. Trucco, and A. Ruggeri, “Automatic generation of synthetic retinal fundus images,” in *Eurograph workshop*, 2014.
- [12] E. Menti, L. Bonaldi, L. Ballerini, F. Ruggeri, and E. Trucco, “Automatic generation of synthetic retinal fundus images: Vascular network,” in *Int. Workshop on Sim. and Syn. in Med. Imag.*, 2016.
- [13] J. Bower, H. Cornelis, and D. Beeman, “GENESIS, the GEneral NEural Simulation System,” in *Encyclopedia of Computational Neuroscience*, D. Jaeger and R. Jung, Eds. Springer, 2014, pp. 1–8.
- [14] N. Carnevale and M. Hines, *The NEURON Book*. Cambridge University Press, 2006.
- [15] G. Ascoli and J. Krichmar, “L-neuron: A modeling tool for the efficient generation and parsimonious description of dendritic morphology,” *Neurocomputing*, vol. 32–33, pp. 1003–11, 2000.
- [16] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin, “Image analogies,” in *SIGGRAPH*, 2001.
- [17] L. Cheng, S. Vishwanathan, and X. Zhang, “Consistent image analogies using semi-supervised learning,” in *CVPR*, 2008.
- [18] L. Gatys, A. Ecker, and M. Bethge, “A neural algorithm of artistic style,” *arXiv*, 2015.
- [19] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky, “Texture networks: Feed-forward synthesis of textures and stylized images,” in *ICML*, 2016.
- [20] J. Justin, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in *ECCV*, 2016.
- [21] C. Kirbas and F. Quek, “A review of vessel extraction techniques and algorithms,” *ACM Computing Surveys*, vol. 36, pp. 81–121, 2000.
- [22] D. Lesage, E. Angelini, I. Bloch, and G. Funka-Lea, “A review of 3D vessel lumen segmentation techniques: models, features and extraction schemes,” *Medical Image Analysis*, vol. 13, no. 6, pp. 819–45, 2009.
- [23] H. Peng, E. Meijering, and G. Ascoli, “From DIADEM to BigNeuron,” *Neuroinformatics*, vol. 13, no. 3, pp. 259–260, 2015.
- [24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *CVPR*, 2015.
- [25] D. Ciregan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” in *CVPR*, 2012.
- [26] A. Krizhevsky, I. Sutskever, and G. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012.
- [27] D. Kingma and M. Welling, “Auto-encoding variational bayes,” *ICLR*, 2014.
- [28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *NIPS*, 2014.
- [29] A. Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel recurrent neural networks,” in *ICML*, 2016.
- [30] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv*, 2015.
- [31] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” *arXiv*, 2014.
- [32] E. Denton, S. Chintala, and R. Fergus, “Deep generative image models using a Laplacian pyramid of adversarial networks,” in *NIPS*, 2015.
- [33] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “Infogan: Interpretable representation learning by information maximizing generative adversarial nets,” in *NIPS*, 2016.
- [34] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein GAN,” *arXiv*, 2017.
- [35] P. Isola, J. Zhu, T. Zhou, and A. Efros, “Image-to-image translation with conditional adversarial networks,” in *Arxiv*, 2016, pp. 1–16.
- [36] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *MICCAI*, 2015.
- [37] X. Wang and A. Gupta, “Generative image modeling using style and structure adversarial networks,” in *ECCV*, 2016.
- [38] X. Mao, C. Shen, and Y. Yang, “Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections,” in *NIPS*, 2016.
- [39] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. Efros, “Context encoders: Feature learning by inpainting,” in *CVPR*, 2016.
- [40] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016.
- [41] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv*, 2014.
- [42] T. Köhler, A. Budai, M. Kraus, J. Odstrčilík, G. Michelson, and J. Hornegger, “Automatic no-reference quality assessment for retinal fundus images using vessel segmentation,” in *IEEE Int. Sym. on Computer-Based Medical Systems*, 2013, pp. 95–100.
- [43] J. De, L. Cheng, X. Zhang, F. Lin, H. Li, K. Ong, W. Yu, Y. Yu, and S. Ahmed, “A graph-theoretical approach for tracing filamentary structures in neuronal and retinal images,” *IEEE Trans. Med. Imaging*, pp. 1–17, 2015.
- [44] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv*, 2014.
- [45] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv*, 2015.
- [46] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training GANs,” in *NIPS*, 2016.
- [47] G. Lin, X. Zhang, H. Zhao, H. Li, and L. Cheng, “Segment 2d and 3d filaments by learning structured and contextual features,” *IEEE Trans. TMI*, 2016.
- [48] K. Maninis, J. Pont-Tuset, P. Arbelaez, and L. V. Gool, “Deep retinal image understanding,” in *MICCAI*, 2016.