Medical Image Analysis 49 (2018) 14–26

Contents lists available at ScienceDirect

Medical Image Analysis

journal homepage: www.elsevier.com/locate/media

Synthesizing retinal and neuronal images with generative adversarial nets

He Zhao[a,b], Huiqi Li[a,*], Sebastian Maurer-Stroh[b], Li Cheng[b,c,*]

[a] Beijing Institute of Technology, China
[b] Bioinformatics Institute, A*STAR, Singapore
[c] Department of Electrical and Computer Engineering, University of Alberta, Canada

ARTICLE INFO

Article history:
Received 11 October 2017
Revised 8 June 2018
Accepted 3 July 2018
Available online 4 July 2018

Keywords:
Data-driven image synthesis
Retinal fundus image synthesis
Deep learning
Neuronal image synthesis

ABSTRACT

This paper aims at synthesizing multiple realistic-looking retinal (or neuronal) images from an unseen tubular structured annotation that contains the binary vessel (or neuronal) morphology. The generated phantoms are expected to preserve the same tubular structure, and resemble the visual appearance of the training images. Inspired by the recent progresses in generative adversarial nets (GANs) as well as image style transfer, our approach enjoys several advantages. It works well with a small training set with as few as 10 training examples, which is a common scenario in medical image analysis. Besides, it is capable of synthesizing diverse images from the same tubular structured annotation. Extensive experimental evaluations on various retinal fundus and neuronal imaging applications demonstrate the merits of the proposed approach.

1. Introduction

A broad range of biomedical images contain thin and long tubular-like foreground objects. They include tubular structured images of various modalities such as magnetic resonance angiography, x-ray angiography, retinal fundus images, as well as cellular neuronal images. Taking retinal fundus images as an example, topological and geometrical properties (Martinez-Perez et al., 2000) of the vessel structures provide valuable clinical information in diagnosing diseases such as proliferative diabetic retinopathy, glaucoma, and hypertensive retinopathy (Abramoff et al., 2010). Often, only a handful of annotated images are available, where the tubular structures are delineated by domain experts through a laborious manual process — a typical situation in many biomedical applications.

Image analysis and image synthesis have long been regarded as tightly intertwined techniques, for example, the research work in analysis-by-synthesis (Grenander, 1976; Gagalowicz and Philips, 2009), as well as the recent progress in human full-body and hand pose estimation problems (Shotton et al., 2013; Xu and Cheng, 2013), where the synthesis processes or synthetic data play an important role in addressing the analysis tasks. In the meantime, although a large number of research activities consider tubular structured image analysis (refer to the survey papers of neuronal and vascular image analyses Fraz et al., 2012; Peng et al., 2015a; Kirbas and Quek, 2000; Lesage et al., 2009; Scarpa et al., 2011; Annunziata et al., 2016), relatively very little effort has been put in synthesizing such images.

In this paper, we present a learning based approach for synthesizing retinal and neuronal images. Main contributions of our work are summarized as follows. (1) Our synthesis model can be effectively learned in a data-driven fashion from a relatively small sample size. For example, we have successfully constructed synthesis models for STARE (Hoover et al., 2000) and DRIVE (Staal et al., 2004) fundus image benchmarks, where the corresponding training images are merely 10 and 20 images, respectively. (2) Based on a single segmentation input, our approach is capable of synthesizing multiple images. This capacity of introducing diversity is important in biomedical data synthesis. (3) The proposed framework of *Tub-sGAN* is the first to incorporate style transfer into the GAN framework, to the best of our knowledge. It is worth noting that the synthesized images are shown useful in improving image segmentation performance. In other words, the segmentation performances of baseline supervised segmentation methods are improved when trained with additional images synthesized by our approach. Extensive experiments on various applications and datasets demonstrate the ability of our approach in producing realistic looking images, as well as in boosting the performance of

* Corresponding authors.
E-mail addresses: huiqili@bit.edu.cn (H. Li), licheng@bii.a-star.edu.sg (L. Cheng).

https://doi.org/10.1016/j.media.2018.07.001

the segmentation module. Our implementation, together with the related results, are made publicly available.[1]

2. Related work

2.1. Synthesizing retinal and neuronal images

One application of synthesizing retinal vessel structures is surgical simulations (Sagar et al., 1994), while others are driven by the practical demand in empirical evaluations of segmentation or tracing methods. A critical issue with retinal image analysis is the lack of annotated vessel structures, due to its expensive and laborious nature. This is further complicated by the inter-observer and intra-observer variabilities of human observers that are subjective and prone to annotation errors (Fritzsche et al., 2003; Trucco et al., 2013). Synthesizing retinal phantoms (Fiorini et al., 2014; Menti et al., 2016) can be useful in this aspect due to its unique advantage of containing complete and unambiguous segmentation annotations. Fiorini et al. (2014) focus on reconstructing the textural background from scratch that is heavily based on domain knowledge. The work of Menti et al. (2016) instead aims to derive vessels and textures from real data utilizing active shape contours and Kalman filter techniques. The results of these methods are reasonable, but these methods are complex, sensitive and heavily dependent on domain knowledge.

Neuronal image synthesis has also been studied based on prior biological knowledge, where GENESIS (Bower et al., 2014), NEURON (Carnevale and Hines, 2006), and L-Neuron (Ascoli and Krichmar, 2000) are probably the most well-known efforts. GENESIS is a simulation method for constructing realistic models of neurobiological systems. It was one of the first simulation systems specifically designed for modeling nervous systems. NEURON is developed similarly for modeling individual neurons and neuron networks. L-Neuron anatomically generates neuronal phantoms based on a Lindenmayer system (or L-system) that prescribes a set of recursive rules.

2.2. Image style transfer, tubular structured image segmentation, and data augmentation

The problem of image style transfer has been studied in the past twenty years as in *e.g.* Hertzmann et al. (2001) and Cheng et al. (2008). Different from generic texture synthesis that generates larger or new images with the same textural appearances, image style transfer aims at further altering the textural style to be similar to a specific reference style image. Recently, impressive results are obtained by Gatys et al. (2016). This is achieved by the successful application of convolutional neural networks (CNNs) to represent two complementary aspects of an image, namely, its content and its style (Tenenbaum and Freeman, 2000), where the pixels of the synthesized image output are updated iteratively to become texturally close to the reference style image. It has been further improved by Ulyanov et al. (2016) and Johnson et al. (2016), where feed-forward convolutional neural networks are introduced to produce a stylized image instead of updating the image pixels iteratively.

There have also been a good amount of methods developed on tubular structured image segmentation (Maninis et al., 2016; Gu et al., 2017). Interested readers can refer to Kirbas and Quek (2000); Lesage et al. (2009) for more thorough reviews. In spite of these research efforts, it remains challenging to precisely segment 2D and 3D image-based tubular structures. This is further evidenced by the recent BigNeuron effort (Peng et al., 2015b) that calls for innovations in addressing the demands from the neuronal science community. A significant number of neuronal images have been routinely produced in various wet labs, yet there are no sufficiently accurate tools available to automatically segment the neurite structures.

Furthermore, it has become a common practice to enrich the training dataset by means of data augmentation, such as cropping, flipping, and rotating existing training images (Szegedy et al., 2015; Ciregan et al., 2012) as well as applying small perturbations in color or intensity values (Krizhevsky et al., 2012).

2.3. Generative adversarial networks

The advancement of deep learning techniques (Kingma and Welling, 2014; Goodfellow et al., 2014; Oord et al., 2016) have led to significant progress in generating photo-realistic images using techniques such as generative adversarial networks (GANs) (Goodfellow et al., 2014). The GANs is considered as a minimax two-player game between a discriminator neural network function and a generator network function. Here the role of the discriminator is to identify the synthesized images out of the real ones, while the generator is to fool the discriminator by synthesizing instances as close to the real ones as possible.

A number of GANs variants have been developed for natural images, including Radford et al. (2015); Mirza and Osindero (2014); Denton et al. (2015); Chen et al. (2016); Arjovsky et al. (2017). Among them, DCGAN (Radford et al., 2015) introduces a set of constraints to stabilize the training dynamics between the generator and the discriminator. CGAN (Mirza and Osindero, 2014) facilitates the training of a synthesis model to generate images conditioned on class labels. LAPGAN (Denton et al., 2015) uses a cascade of CNNs within a Laplacian pyramid framework to generate the images from coarse to fine. InfoGAN (Chen et al., 2016) further allows to learn disentangled representations in a completely unsupervised manner. Moreover, the work of Isola et al. (2016) utilizes techniques similar to U-Net (Ronneberger et al., 2015) to preserve global structural information during its data generation process.

Recently, GANs techniques have also been introduced to synthesize biomedical images, such as CT and MRI images (Nie et al., 2017; Wolterink et al., 2017). Specifically, a retinal image synthesis method is developed in Costa et al. (2017). Following the technique of Isola et al. (2016), it works by training on a large amount of fundus images and the corresponding vessel segmentations. The work of (Costa et al., 2018) is a further development over (Costa et al., 2017) to remove the dependency on vessel tree inputs, which is achieved by adding an adversarial autoencoder. Their generated images are visually plausible, while sometimes the phantoms may exhibit unrealistic vessel morphologies. These methods (Isola et al., 2016; Costa et al., 2017; 2018) tend to generate only one fixed output for a given input. As a result, there is little diversity in the generated images. Our approach, on the other hand, is capable of producing multiple individual phantoms from the same input. A variant of our model is also proposed that can generate images with dedicated textural appearance resembling that of a target retinal fundus image. Moreover, instead of working with large amount of training pairs, the proposed approach can work well with only tens of training pairs. Finally, the proposed approach can boost the performance of retinal segmentation when the generated images are included as additional training examples.

3. Our approach

In this work, we aim to learn a direct mapping from a tubular structured annotation back to a plausible raw image. More specif-

[1] Downloadable at https://www.web.bii.a-star.edu.sg/archive/machine_learning/Projects/filaStructObjs/Synthesis/index.html.

ically, a RGB fundus or neuronal image is denoted by $\boldsymbol{x} \in \mathbb{R}^{W\times H\times 3}$, and $\boldsymbol{y} \in \{0, 1\}^{W\times H}$ refers to its corresponding tubular structured annotation. W and H are the width and height of the input image size, respectively. By imitating the image formation process, let $G_\theta : \left(\boldsymbol{y} \in \mathbb{R}^{W\times H}, \boldsymbol{z} \in \mathbb{R}^{Z}\right) \rightarrow \hat{\boldsymbol{x}} \in \mathbb{R}^{W\times H\times 3}$ denote the image generation function that takes a binary image of tubular structured annotation $\boldsymbol{y}$ and a noise code $\boldsymbol{z}$ as input, and produce as output a phantom $\hat{\boldsymbol{x}}$. The noise code vector $\boldsymbol{z}$ is to introduce the appearance diversity. In practice its length is set to 400.

Our goals are three-fold. (1) Learn the θ-parameterized function G from a small training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{n}$. (2) Be capable of exploring the underlying conditional image formation distribution $p(\boldsymbol{X}|\boldsymbol{y})$, where $\boldsymbol{x}$ is a random variable denoting a feasible fundus or neuronal image conditioning on the particular realization $\boldsymbol{y}$. A practical way in simulating such distribution is by sampling a noise code vector $\boldsymbol{z}$, which allows us to synthesize plausible yet distinct RGB image instances given the same input $\boldsymbol{y}$. (3) Consider an interesting variant of our approach where a specific image style obtained from *one* additional input image $\boldsymbol{x}_s$ can be directly transferred to the output phantom $\hat{\boldsymbol{x}}$. Here the style of $\boldsymbol{x}_s$ can be very different from that of $\boldsymbol{x}$, and the corresponding contents (i.e., tubular structure $\boldsymbol{y}_s$ and $\boldsymbol{y}$) are usually unrelated. The aforementioned goals seem daunting: given the intricate nature of the image formation process, G is usually a rather sophisticated function, while the problem is even harder with a small training set. Nonetheless, by resorting to the powerful deep learning framework of GANs, an end-to-end learning machine is proposed in this paper, as depicted in Fig. 1. The yellow colored area highlights the generator module that will be engaged during the testing stage, while the rest of the modules are employed only at the training stage. In what follows, we will give a detailed description of the two variants of our framework, *Tub-GAN* and *Tub-sGAN*.

3.1. Tub-GAN: the basic generative framework

The top part of Fig. 1(a) (*i.e.* excluding elements in the dashed box) illustrates the overall work flow of our approach *Tub-GAN*. In addition to the generator G_θ, consider a discriminator function $D_\beta : \left(\boldsymbol{x} \in \mathbb{R}^{W\times H\times 3}, \boldsymbol{y} \in \mathbb{R}^{W\times H}\right) \rightarrow d \in [0, 1]$, whose role is to tell apart synthetic phantom $\boldsymbol{X} := \hat{\boldsymbol{x}}$ (ideally $d \rightarrow 0$) from real tubular structured image $\boldsymbol{X} := \boldsymbol{x}$ (ideally $d \rightarrow 1$), as visually explained in Fig. 1(a). We follow the GANs approach for this minimax two-player game setting and consider the following optimization problem that characterizes the interplay between G and D:

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

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

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

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 Goodfellow et al. (2014), it is realized by minimizing a simpler form $-\log(D_\beta(G_\theta(\boldsymbol{y}, \boldsymbol{z})))$, instead of the original $\log(1 - D_\beta(G_\theta(\boldsymbol{y}, \boldsymbol{z})))$. To summarize, the generator G is learned by minimizing

$$L_{\mathcal{G}}(G_\theta) = -\sum_i \log D_\beta(G_\theta(\boldsymbol{y}_i, \boldsymbol{z}_i), \boldsymbol{y}_i) + \lambda\|\boldsymbol{x}_i - G_\theta(\boldsymbol{y}_i, \boldsymbol{z}_i)\|_1. \quad (3)$$

On the other hand, discriminator D attempts to properly separate the real images from the synthetic ones by maximizing Eq. (1), which can be simplified by maximizing

$$L_{\mathcal{D}}(D_\beta) = \sum_i \log D_\beta(\boldsymbol{x}_i, \boldsymbol{y}_i) + \log\left(1 - D_\beta(G_\theta(\boldsymbol{y}_i, \boldsymbol{z}_i), \boldsymbol{y}_i)\right). \quad (4)$$

Note that the sum is used to approximate the expectation. The learning process is carried out by alternating between these two optimization operations, which are similarly adopted by GANs and variants (Goodfellow et al., 2014; Radford et al., 2015; Isola et al., 2016). Unfortunately this optimization procedure is not guaranteed to reach Nash equilibrium. In practice, we have observed convergence with reasonable image synthesis outputs, which is also demonstrated in the above GANs related work. In the following paragraphs, we describe in detail the specific neural net architectures of our functions G and D.

3.1.1. *Generator* G *and discriminator* D

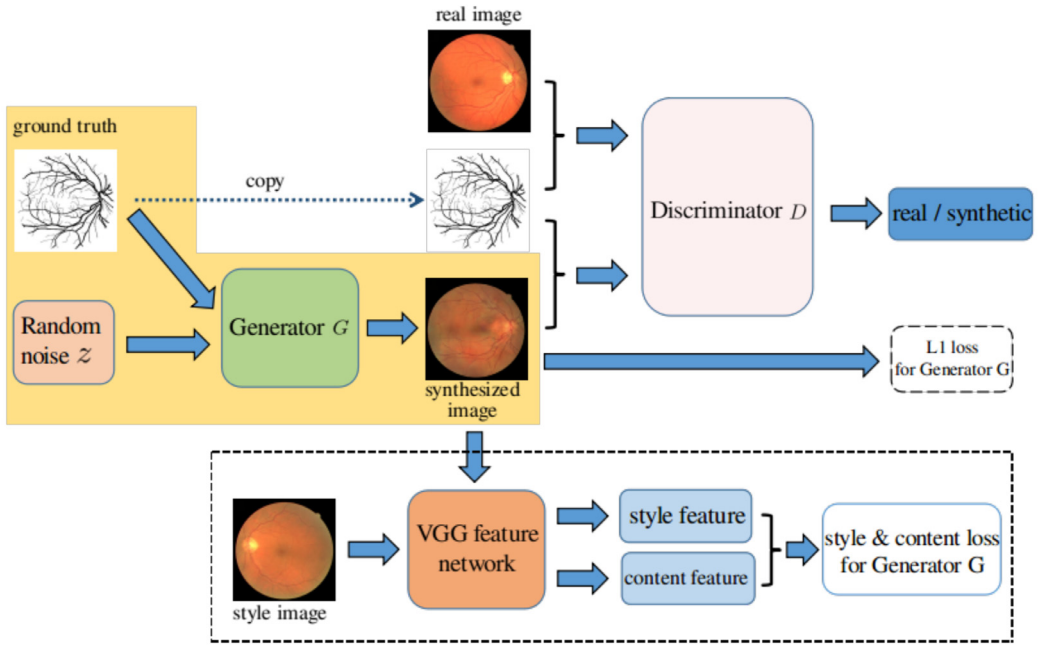
For the generator, the commonly used encoder-decoder strategy is adopted here (Wang and Gupta, 2016; Mao et al., 2016; Pathak et al., 2016; Isola et al., 2016), which allows the introduction of noise code in a natural manner. The encoder part acts as a feature extractor where the multiple layered structure captures both local and more global data representations. The 400-dimensional noise code $\boldsymbol{z}$ 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 retain the input tubular structured morphology during image generating process. For this, the skip connections of U-Net (Ronneberger et al., 2015) are also considered here. That is, the previous layer is duplicated by being appended onto the current layer in a mirrored fashion that skips odd-number layers with the center coding layer as its origin. It is worth noting that when the 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 challenging. 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 (He et al., 2016), 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 practically helps to generate much better results.

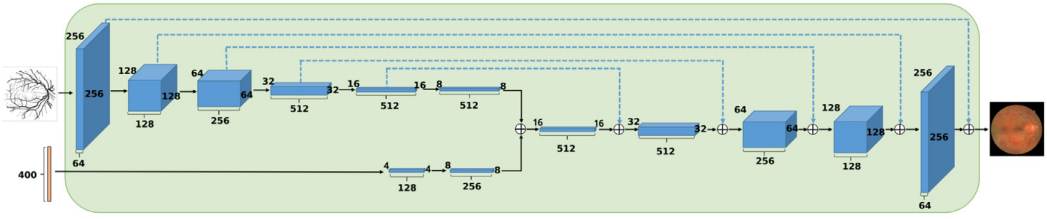
We follow the basic network architecture proposed in Radford et al. (2015), to build the layers of the generator with multiple Convolution-BatchNorm-LeakyRelu components as shown in Fig. 1(b). The activation function of the output layer is *tanh* to keep the value between -1 and 1. Note that *tanh* is used here instead of *sigmoid*, which increases the stability of training GANs.

On par with our generator, the same Convolution-BatchNorm-LeakyRelu building blocks are used in building our discriminator as shown in Fig. 1(c). Here the activation function of the output layer is *sigmoid*. The feature map sizes are halved after each convolutional layer (*e.g.* the input size is 512×512, and the size becomes 256×256 after one convolutional layer) while the number of filters (*i.e.* the number of feature maps) are doubled from 32 all the way to 512.

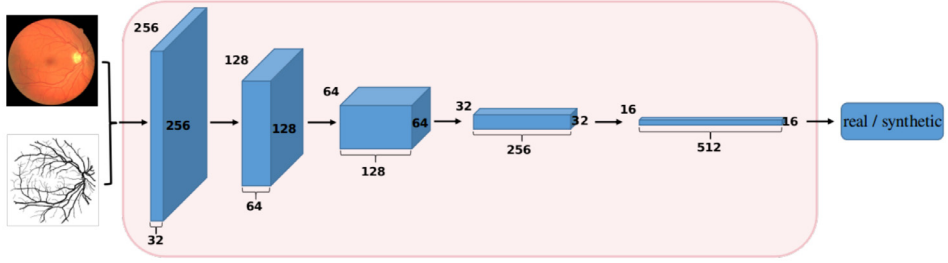
Up to now, our approach considers to learn a generic representation from a (usually limited) set of training examples, which is then employed in generating tubular structured phantoms, which is termed *Tub-GAN*. Next, we consider a variant inspired by the recent style transfer techniques.



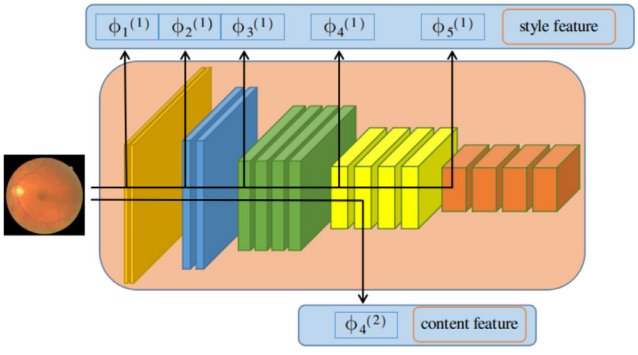
(a) Flowchart of our model



(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 yellow shaded area highlights the generator module that will be engaged in testing. The VGG feature networks are described in (d), where the top row indicates the specific layers (*e.g.* $\phi_1^{(1)}$, $\phi_2^{(1)}$, ...) extracted as style features, while the bottom row are the layers (*e.g.* $\phi_4^{(2)}$) used as content features. The $\oplus$ sign denotes a concatenation operation. See text for details.

3.2. Tub-sGAN: the style transfer framework

Inspired by the recent advances in image style transfer, 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 $\boldsymbol{y}$ that delineates its tubular structured content, the generated phantom $\hat{\boldsymbol{x}}$ is expected to possess the similar style (*i.e.* texture) of a target style input $\boldsymbol{x}_\mathrm{s}$, while still keeping the content of $\boldsymbol{y}$ that has been presented during training stage. This variant is termed *Tub-sGAN*, to highlight the fact that the synthetic image is now based on a *particular* style representation provided by $\boldsymbol{x}_\mathrm{s}$, instead of the generic representation we have aimed for in our basic approach *Tub-GAN*. This is made possible by introducing a style image as an additional training input, *i.e.* a new tubular structured image $\boldsymbol{x}_\mathrm{s}$ with a different style or texture. Note that in general $\boldsymbol{x}_\mathrm{s}$ has its own tubular structure, which is usually different from the structure in the input argument $\boldsymbol{y}$. It is also worth noting that this design makes practical sense in a 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.

Without loss of generality, we follow the style transfer idea of Ulyanov et al. (2016); Johnson et al. (2016) to utilize the convolutional neural network (CNN) of VGG-19 (Simonyan and Zisserman, 2014) to extract features from its multiple layers. According 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 denote 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 $\lambda \in \Lambda$. For a tubular structured image $\boldsymbol{x}$ (being either the real $\boldsymbol{x}$ or phantom $\hat{\boldsymbol{x}}$), $\phi_\gamma^{(\lambda)}(\boldsymbol{X})$ denotes the λth layer of CNN block γ. The VGG-19 net is obtained by training on the ImageNet image classification task and its details are illustrated in Fig. 1(d). In terms of the optimization problem, it is realized in our approach by explicitly incorporating two perceptual loss components of Gatys et al. (2016), namely the style loss and the content loss, as well as a total variation loss, which we describe next.

3.2.1. Style loss

The style loss is used to minimize the textural deviation between the target style $\boldsymbol{x}_\mathrm{s}$ and the phantom $\hat{\boldsymbol{x}}$. The textural feature is characterized by the *Gram matrix*, where each element $\mathbb{G}_{\gamma_\mathrm{s},ij}^{(\lambda_\mathrm{s})}$ represents an inner product between the ith and jth feature maps in λ_sth layer of γ_sth block:

$$\mathbb{G}_{\gamma_\mathrm{s},ij}^{(\lambda_\mathrm{s})} = \sum_k \phi_{\gamma_\mathrm{s},ik}^{(\lambda_\mathrm{s})} \phi_{\gamma_\mathrm{s},jk}^{(\lambda_\mathrm{s})}. \tag{5}$$

In Eq. (5), consider Γ_s as the set of CNN blocks, and for each block index $\gamma_\mathrm{s} \in \Gamma_\mathrm{s}$, its set of layers is represented by Λ_s. $\phi_{\gamma_\mathrm{s}}^{(\lambda_\mathrm{s})}(\boldsymbol{X})$ is defined as the λ_sth layer of block γ_s, where $\boldsymbol{X} = \boldsymbol{x}_\mathrm{s}$ or $\boldsymbol{X} = \hat{\boldsymbol{x}}$. With a slight abuse of notation, denote the number of feature maps in current layer λ_s as $|\lambda_\mathrm{s}|$. Let i (or j) denote the feature map of interest, and k denotes an element of the current feature map i (or j).

The style loss is calculated by the difference between the Gram matrices of $\boldsymbol{x}_\mathrm{s}$ and $\hat{\boldsymbol{x}}$ during training

$$l_\mathrm{sty}(G_\theta) = \sum_{\gamma_\mathrm{s}\in\Gamma_\mathrm{s},\lambda_\mathrm{s}\in\Lambda_\mathrm{s}} \frac{\varpi_{\gamma_\mathrm{s}}}{W_{\gamma_\mathrm{s}} H_{\gamma_\mathrm{s}}} \left\| \mathbb{G}_{\gamma_\mathrm{s}}^{(\lambda_\mathrm{s})}(\boldsymbol{x}_\mathrm{s}) - \mathbb{G}_{\gamma_\mathrm{s}}^{(\lambda_\mathrm{s})}(\hat{\boldsymbol{x}}) \right\|_\mathrm{F}^2, \tag{6}$$

where $\|\cdot\|_\mathrm{F}$ is the matrix Frobenius norm, $\varpi_{\gamma_\mathrm{s}}$ denotes the weight of γ_sth block *Gram matrix* and it is set to 0.2 in practice. Note $\hat{\boldsymbol{x}} = G_\theta(\boldsymbol{y}, \boldsymbol{z})$ by definition.

3.2.2. Content loss

For content loss, consider the following notation: let Γ_c refer to the set of CNN blocks, and for each block index $\gamma_\mathrm{c} \in \Gamma_\mathrm{c}$, its set of layers is denoted by Λ_c. As already stated, the synthetic phantom $\hat{\boldsymbol{x}}$ is expected to abide by the tubular structure as prescribed in the real raw image $\boldsymbol{x}$ of the segmentation input, which is carried out by minimizing the following Frobenius norm of the difference between input and generated CNN features:

$$l_\mathrm{cont}(G_\theta) = \sum_{\gamma_\mathrm{c}\in\Gamma_\mathrm{c},\lambda_\mathrm{c}\in\Lambda_\mathrm{c}} \frac{1}{W_{\gamma_\mathrm{c}} H_{\gamma_\mathrm{c}}} \left\| \phi_{\gamma_\mathrm{c}}^{(\lambda_\mathrm{c})}(\boldsymbol{x}) - \phi_{\gamma_\mathrm{c}}^{(\lambda_\mathrm{c})}(\hat{\boldsymbol{x}}) \right\|_\mathrm{F}^2. \tag{7}$$

3.2.3. Total variation loss

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

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

where $w,\ h \in W,\ H$, and $\hat{x}_{w,h}$ denotes pixel value of a given location in phantom image $\hat{\boldsymbol{x}}$.

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

$$L_\mathcal{G}(G_\theta) = -\sum_i \log D_\gamma(G_\theta(\boldsymbol{y}_i, \boldsymbol{z})) + L_\mathrm{ST}(G_\theta), \tag{9}$$

The training and testing processes of this variant remain the same as *Tub-sGAN*: the training is carried out in batch mode over the n annotated training examples. The aforementioned generator and the discriminator functions still carry on as before. The differences lie in the objective function of generator G, where Eq. (3) is now replaced by Eq. (9). On the other hand, 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 $\boldsymbol{x}_\mathrm{s}$ is obtained by back-propagation optimization of the above objective function, while the remaining aspects of our approach are kept the same.

4. Experimental set-up

4.1. Datasets and preparation

Our approach is tested on four standard benchmarks that cover a broad spectrum of tubular structured images including both retinal blood vessels and neurons. They are DRIVE (Staal et al., 2004), STARE (Hoover et al., 2000), high-res fundus or HRF (Köhler et al., 2013), as well as 2D Neurons or NeuB1 (De et al., 2016). The image sizes and the amount of training examples are different across these datasets: DRIVE contains 20 training examples and 20 testing images, with each of size 584×565. STARE and HRF are two fundus image datasets with image sizes of 700×605 and 3304×2336 respectively, and the splits of training/testing images are 10/10 and 22/23 respectively. The NeuB1 dataset is microscopic neuronal images, which contains 112 images of size 512×512. We also follow the standard train/test split of 37/75 as in De et al. (2016).

To summarize, the image sizes of DRIVE, STARE, and NeuB1 are roughly similar, while HRF contains high resolution (and subsequently much larger size) images. In preprocessing, raw images of these first three datasets are all resized to a standard size of 512×512. 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, and

all foreground pixels are still kept in the cropped images. They are further resized to 512×512 by bicubic interpolation; For STARE, as margins outside of the foreground mask are rather small, images are directly resized to the size of 512×512 by bicubic interpolation; For HRF, as its raw images are of much larger size of 3304×2336, these images are all resized to 2048×2048 instead of the 512×512 template size used previously, in order to retain sufficient information of the original images. For each of the benchmarks, the pixel intensity values of all the images are scaled to the same range of $[-1, 1]$. As a result, the learned generator in our approach will produce a phantom image of size 512×512 for DRIVE, STARE, and NeuB1, and size 2048×2048 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 pixels inside the mask are retained as foreground.

4.2. Model architecture and parameters

Fig. 1(b)–(d) illustrates the architecture of our approach. The image width W and height H are 2048 for HRF and 512 for other datasets. In the generator and discriminator modules, each 3D box denotes a CNN layer consisting of its feature maps. A directed edge usually represents a convolutional (or transposed convolutional) 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) and (c) specify the intrinsic parameter values of our networks G and D, such as the size of each layer, and the length of the noise code $Z = 400$. In particular, in generator G, the $\bigoplus$ sign together with the two directed edges pointing to it denote a concatenation operation. For example, the first $\bigoplus$ sign shows a concatenation of an $8 \times 8 \times 512$ tensor with an $8 \times 8 \times 256$ tensor that produces an $8 \times 8 \times 768$ tensor; This is followed by a transposed convolutional (also referred to as deconvolutional) operation of filter size $4 \times 4 \times 512$ that produces the 3D box of size $16 \times 16 \times 512$.

Throughout experiments in our approach, the internal parameters are empirically considered: The TensorFlow 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 β, θ), a $[-0.04, 0.04]$ truncated normal distribution of zero-mean and standard deviation of 0.02 is used. The weights θ of G are updated with mini-batch of size 1 using the Adam optimizer (Kingma and Ba, 2014). The vanilla stochastic gradient descent is employed for D to update β. The learning rate is set to 0.0002 for the generator and 0.0001 for the discriminator during back-propagation training. λ in our *Tub-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 a zero-mean Gaussian distribution with standard deviation 0.001. At testing stage, it is sampled in the same manner but with a different standard deviation of 1. This is found useful in maintaining a proper level of diversity for our small sample-size situation. We follow the practice of Radford et al. (2015) where batch normalization is not performed to the output layer of G nor the input layer of D. Instead, batch normalization (Ioffe and Szegedy, 2015) is applied right after each convolutional layer. Better model training behaviors have been observed for both G and D nets by such settings.

For our style transfer variant, *Tub-sGAN*, the VGG-19 nets are employed to produce the feature descriptors. Some of their layers are used for extracting style/content features, which are illustrated in Fig. 1(d). 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_{cont}, w_{sty}, w_{tv}, are 1, 10, and 100, respectively.

4.3. Sensitivity of parameters

Here we focus on the noise code, and proper layers for loss functions. More details are provided in the supplementary file.

4.3.1. Noise code z

Experiments have been carried out to examine the impact of engaging noise code $\boldsymbol{z}$ of varying lengths such as 1, 10, 40, 400, 4000. It is observed that when the vector length is relatively small (*e.g.* 1, 10, 40), differences in the generated images using distinct noise codes are not noticeable; On the other hand, when the length is relatively large (*e.g.* 400, 4000), our approach produces visually diversified outputs. There is also not much difference in terms of diversifying capacity, when the length is 400 or 4000. In our applications, 400 is used as the length of $\boldsymbol{z}$.

As presented in Fig. 1(b), the noise code $\boldsymbol{z}$ passes through two additional network layers prior to concatenation. This network structure has the advantage of facilitating a compact representation of length 400 instead of the full-sized length $16{,}384 = 8 \times 8 \times 256$. Moreover, it is also observed that by either adopting the full-sized noise code or not, there is not much difference in terms of the generated images. We attribute it to the fact that in a very high dimensional space, *any* two vectors are far away from each other, and as a result, they do not help much with diversifying the outputs compared to the more compact 400 dimensional space as in our case.

We have observed little difference for a noise code to be sampled from a Gaussian vs. from a uniform distribution. In practice, our noise code is sampled from a Gaussian distribution.

4.3.2. Proper layers for loss functions in our Tub-sGAN variant

Fig. 1(a) presents both style and content loss functions utilized in our *Tub-sGAN*: content loss relates to the tubular morphological structures, while style loss can be regarded as enforcing the detailed textural information attached to the underlying tubular structures. It has been observed that *style* features from lower layers such as $\phi_1^{(1)}$, $\phi_2^{(1)}$ preserve detailed pixel information, while the higher layer features tend to maintain high-level semantic textures. By including multiple layers, the model is able to capture the multi-scale textural information on the image. Therefore, we consider a style loss function based on the multi-layer feature representation of $\phi_1^{(1)}, \phi_2^{(1)}, \phi_3^{(1)}, \phi_4^{(1)}, \phi_5^{(1)}$, as well as a content loss function based on the single layer feature representation of $\phi_4^{(2)}$, as illustrated in Fig. 1(d). This setting can produce visually appealing results.

4.4. 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 *Tub-GAN* and *Tub-sGAN* is in Python. Training time of *Tub-GAN* and *Tub-sGAN* on DRIVE or other datasets with the same image size takes around 108 min and 184 min, respectively. For the high resolution dataset, HRF, the training time increases to around 438 min. The average run-time speed in synthesizing a DRIVE-size image is 0.45 s, which becomes 1.35 s for a HRF-sized image.

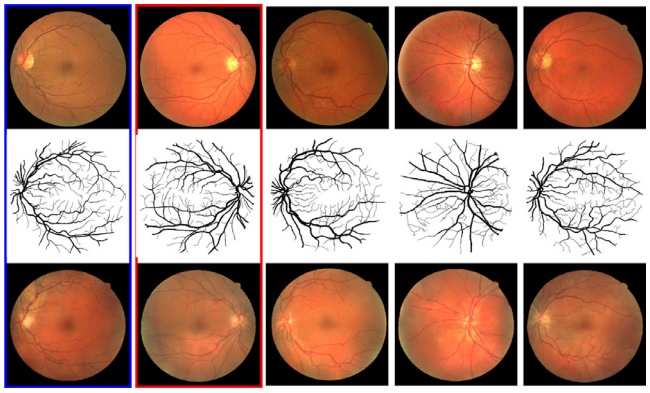


Fig. 2. Exemplar DRIVE phantoms generated by *Tub-GAN*. For each column, the 1st row presents a real image, the 2nd row is the corresponding segmentation annotation, and the 3rd displays one generated phantom.

5. Experimental results

5.1. Visual results

Fig. 2 displays exemplar synthetic results of applying *Tub-GAN* on DRIVE, where the first row presents the real images, the second row shows the corresponding segmentation annotations, 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 the figure), while being able to present different yet realistic-looking texture. It is interesting to note that the very bright colored areas are usually properly situated around the optic disc locations of the segmentation images, which suggests that our phantom generation model could capture this intrinsic feature without explicit human interventions for introducing such prior knowledge. A zoomed-in view of the synthetic image is presented in Fig. 3 to show that detailed information is well generated by our approach. Similar results can also be obtained in datasets such as STARE as shown in Fig. 4, as well as for high-resolution dataset HRF in Fig. 5, 2D neuronal dataset NeuB1 in Fig. 6, and 3D BigNeuron dataset in Fig. 7.

To demonstrate the strength of *Tub-GAN* in generating multiple distinct syntheses from the same segmentation annotation, Fig. 8 presents more synthetic results of the first and second annotation inputs of Fig. 2, which are obtained by i.i.d. random noise codes $\boldsymbol{z}$. Clearly, for each of these segmentation inputs, the results synthesized by *Tub-GAN* are visually different. It can be observed that *Tub-GAN* is relatively more powerful in emulating textural diversity and less in capturing illumination changes.

In addition, this *Tub-GAN* model trained on the DRIVE training set is applied to images from other datasets. Here we take the STARE dataset (Hoover et al., 2000) as an example and the results are presented in Fig. 9: two random samples are selected, where from left to right are real image, segmentation annotation, and phantom, respectively. Not surprisingly, the two synthetic STARE phantoms bear DRIVE-like textures. Finally, to demonstrate that *Tub-GAN* works well even when the segmentation annotations are not available, 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 can be used here. As displayed in Fig. 10, the generated phantoms are also visually plausible. In terms of 3D images, Fig. 7 provides exemplar visual results of synthesizing the 3D neuronal image stacks of the Gold166 dataset (Peng et al., 2015a), and more details and examples can be found in the supplementary file and video.

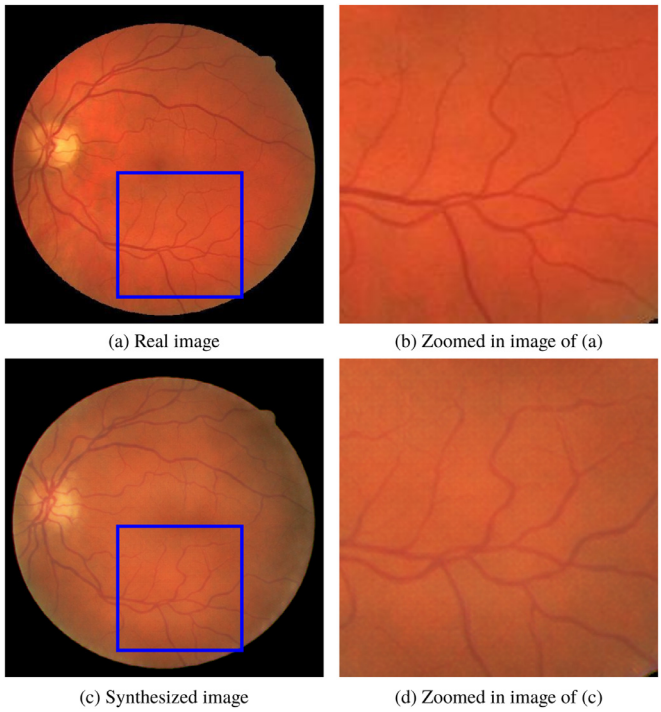


Fig. 3. A zoomed-in view of the synthesized image. Top row presents a real DRIVE image in (a), and for a region highlighted in blue, (b) displays the zoomed-in view. In the same way, (c) shows a corresponding synthetic image based on the same segmentation annotation of (a), and (d) displays the zoomed-in view of the same region. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

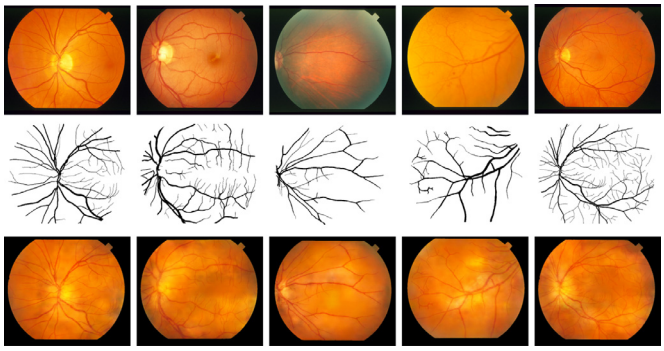


Fig. 4. Exemplar STARE phantoms generated by *Tub-GAN*. For each column, the 1st row presents a real retinal image, the 2nd row is the corresponding segmentation annotation, while the 3rd row displays one generated phantom.

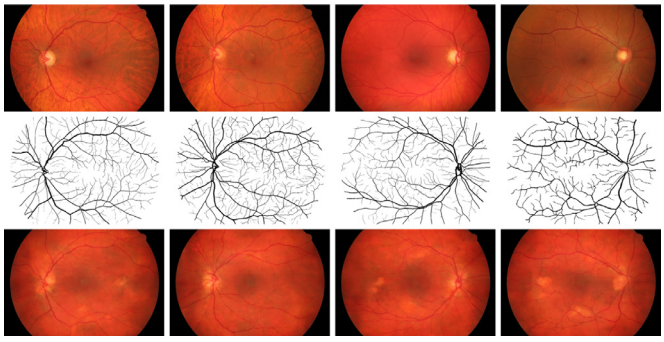


Fig. 5. Exemplar HRF phantoms generated by our *Tub-GAN* model. For each column, the 1st row presents a real retinal image, the 2nd row is the corresponding segmentation annotation, while the 3rd row displays one generated phantom. A zoomed-in figure is presented in the supplementary file.

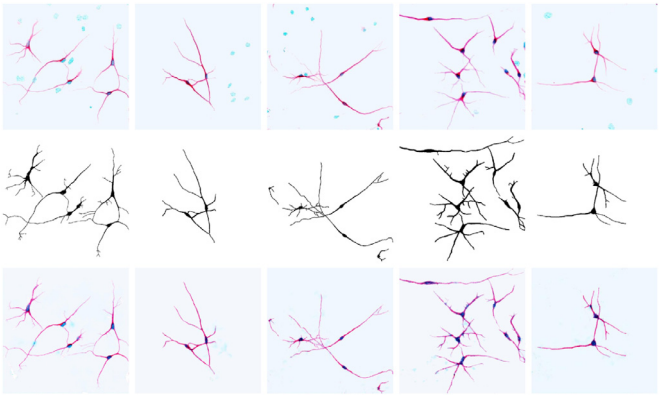


Fig. 6. Exemplar NeuB1 phantoms generated by our *Tub-GAN*. For each column, the 1st row presents a real neuronal image, the 2nd is its corresponding segmentation annotation, while the 3rd displays one generated phantom.

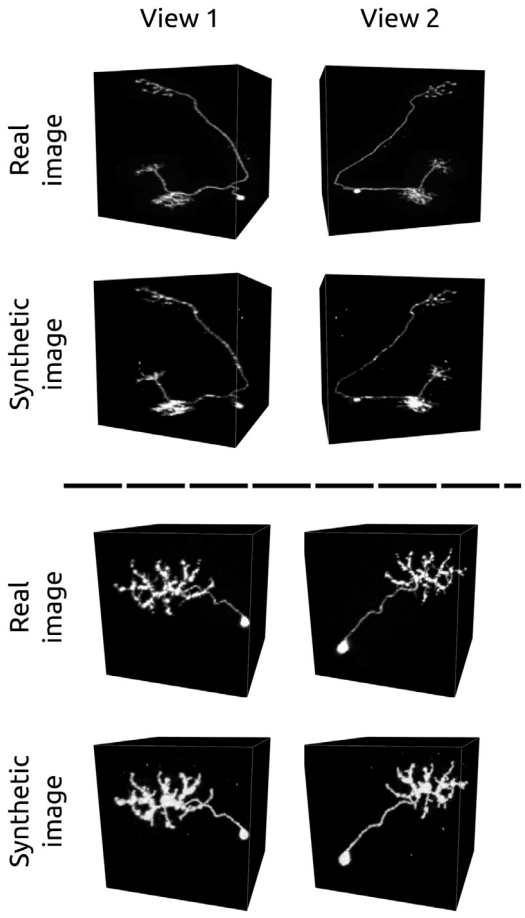


Fig. 7. Visual demonstration of synthesizing 3D neuronal image stacks. The first row displays the real images, while the second row presents synthetic images. Two such examples are provided here.

Now we turn to examine our style transfer variant, *Tub-sGAN*. Fig. 11 presents a gallery of collective results to the same set of DRIVE retinal images (shown in the 2nd to the 6th columns) when trained with different style images (shown in the first column). The generated images with same style are displayed in the same row. Here the hindsight real images are presented in the first row for reference purpose, followed by their corresponding segmentation annotations in the second row. From the 3rd row onwards, the generated phantoms with different styles are presented. DRIVE,

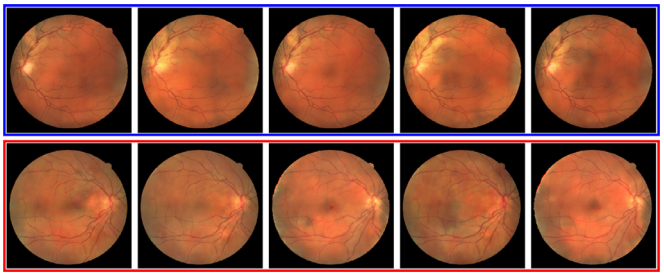


Fig. 8. The two rows in this figure display multiple synthesized results derived from the first two columns of Fig. 2 respectively, to showcase the ability of *Tub-GAN* in generating diverse outputs.

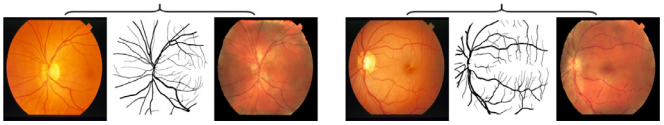


Fig. 9. Application of the *Tub-GAN* model trained on the DRIVE training set (the same model used in Fig. 2) to STARE tubular annotations. Two random samples are selected, from left to right are real STARE image, segmentation annotation, and generated phantom, respectively.

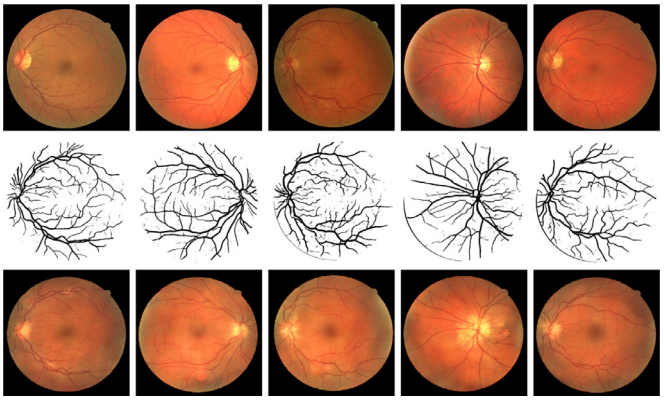


Fig. 10. Exemplar phantoms generated from segmentation predictions (instead of segmentation annotations) of a typical segmentation method trained on DRIVE. In each column, the 1st row presents a real DRIVE image, the 2nd row is the corresponding segmentation hypotheses obtained by a state-of-the-art segmentation method, the 3rd row displays the generated phantoms.

Kaggle,[2] and STARE denote the three distinct style sources. It is observed that the texture style of each phantom synthesized by our *Tub-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 of the images. At the same time, the respective tubular structures are well preserved. Moreover, visually diverse results are again generated by varying the noise input $\boldsymbol{z}$, as presented in the supplementary file. Visually, *Tub-sGAN* is shown to be capable of producing realistic looking phantoms of very different styles.

An image quality assessment metric, structural similarity (SSIM) (Wang et al., 2004) is applied here to provide a quality evaluation of the synthesized images. Taken as input a pair of real and synthetic images, a SSIM score is provided as a combined evaluation of luminance, contrast and structure comparisons. Here we focus on the DRIVE test set images. The average SSIM scores of *Tub-GAN* and *Tub-sGAN* are 0.8924 and 0.8980, respectively, which

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

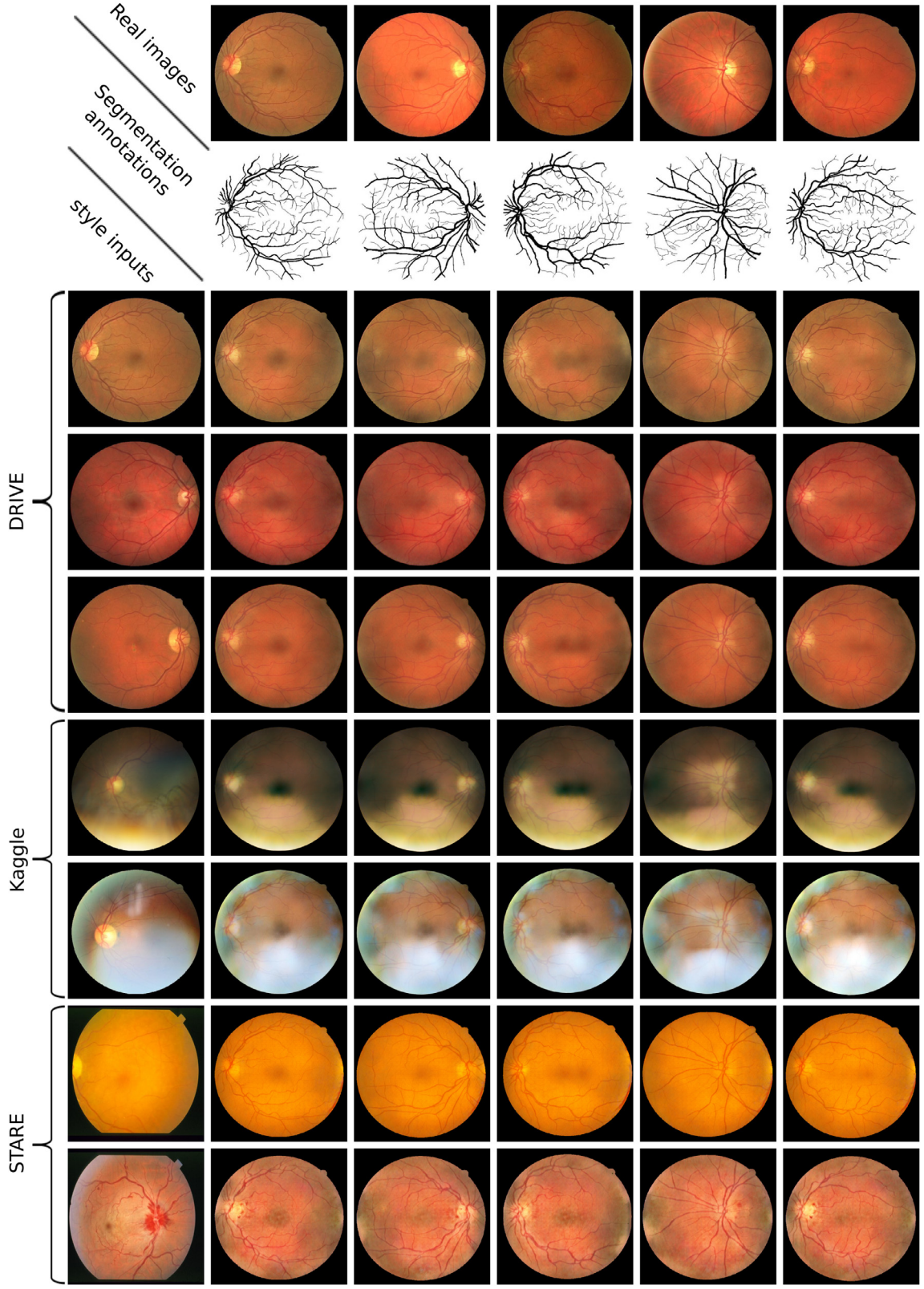


Fig. 11. Phantoms generated by *Tub-sGAN*. The first and second rows display the real DRIVE images, and the corresponding segmentation annotations, respectively. From the third row onwards, each row presents the phantoms synthesized from a specific style image shown in the first column.

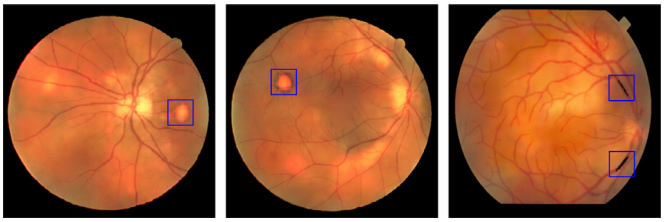


Fig. 12. Example cases of failure.

suggest the performance of the *Tub-sGAN* variant being better or at least on par in terms of image quality evaluation.

5.1.1. Failure cases

It is observed that 90% of the generated images are realistic looking. For example, our learned generator G_θ of the *Tub-GAN* is applied on the DRIVE test set 20 times, and each time with a randomly sampled noise vector. From the 400 synthesized images, only about 40 images (*i.e.* 10%) are non-realistic looking ones that may contain strange artifacts. Several example cases of failure are presented in Fig. 12, with the outlier regions being marked out in blue. We also note that these failure cases are noticeably reduced in our *Tub-sGAN* variant.

5.2. Quantitative results

It is often difficult to quantitatively evaluate the quality of synthetic results, as is also mentioned in Isola et al. (2016); Salimans et al. (2016). In this paper, we consider a relatively straightforward evaluation scheme, which is to examine the usefulness of these newly generated images in boosting the performance of the segmentation method. As part of the quantitative evaluation process, a segmentation baseline is required. Broadly speaking, any reasonable supervised segmentation method could be applicable. Practically two very different CNN baselines are considered: (1) a patch-based CNN method, and (2) our re-implementation of the DRIU method (Maninis et al., 2016). Our patch-based CNN is a pixel-wise residual-block based classification method, where each input instance is a 27×27 image patch centered on the current pixel of interest. More details are provided in the supplementary file. Another evaluation scheme is through investigating the differences in segmentation results, when applying the same trained segmentation model on both synthetic and real images, which can be found in the supplementary file. In what follows, only the first scheme is considered.

5.2.1. Patch-based CNN baseline

As our patch-based CNN baseline segmentation method takes image patches as input, 400K image patches are obtained from real images in the training set, fully exhausting the number of real examples (*i.e.* real image patches) that can be acquired from the DRIVE training set. Additionally we have another 400 K image patches from the synthetic training images. We consider three scenarios, which are: *synthetic images* where the segmentation baseline is trained only on the 400 K synthetic patches; *real images* where it is trained on the set of 400 K real image patches; *real + synthetic images* for training on the set of 400 K real patches plus the 400 K synthetic patches.

As summarized in Table 1, the segmentation model performs worst when training only on synthetic images, which is to be expected. Meanwhile, taking the DRIVE benchmark for example, the introduction of additional synthetic images, *i.e. real + synthetic images*, is demonstrated to improve the segmentation performance to 80.33%, compared with 79.15% F1-score when training on 400 K real patches. It suggests that adding synthetic images generated by *Tub-GAN* helps to improve segmentation performance. The same trend is consistently observed when working with all different benchmarks.

Table 1
Segmentation quantitative evaluation. Here a patch-based CNN baseline segmentation method is engaged on the DRIVE, STARE, HRF, NeuB1 test sets while training on different sets of images (as depicted in the left-most column). Synthetic images are generated by our *Tub-GAN* variant. Results are evaluated with average F1-score (%), Specificity (%), and Sensitivity(%). Note in each column, the same number of *synthetic images* and *real images* are used, and the set of synthetic images remains the same in the column. See text for details.

		DRIVE	STARE	HRF	NeuB1
Synthetic images	F1-score	71.44	75.34	68.28	77.69
	Specificity	97.79	98.35	97.74	99.19
	Sensitivity	68.57	72.79	67.29	82.46
Real images	F1-score	79.15	78.11	78.68	83.91
	Specificity	97.85	98.26	98.09	99.49
	Sensitivity	80.33	77.87	79.73	86.70
Real + synthetic images	F1-score	80.33	79.02	79.50	85.06
	Specificity	98.15	98.41	98.23	99.54
	Sensitivity	80.38	78.96	80.01	87.26

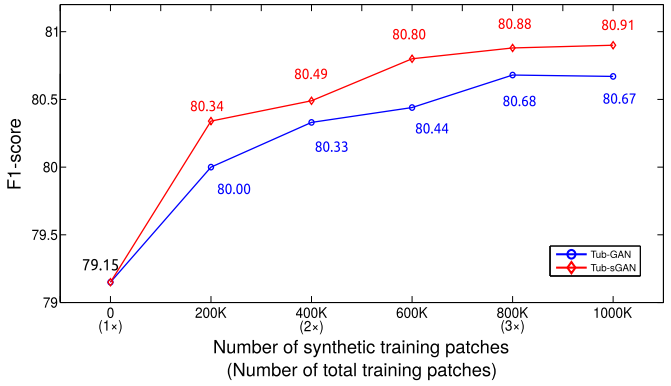


Fig. 13. Performance of our patch-based CNN baseline segmentation method on the DRIVE benchmark as a function of the number of synthetic training patches. The blue and the red curves are the segmentation performances when our *Tub-GAN* and *Tub-sGAN* variants are used to synthesize images, respectively. See text for details. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The Wilcoxon signed-rank test with p-value threshold .05 is used to compare between the two results: *real images* and *real + synthetic images*, in order to demonstrate that the incorporation of synthetic training images improves the segmentation performance with statistical significance. Similar experimental observations can be obtained for our *Tub-sGAN* variant. Taking the DRIVE benchmark for example, similar patterns as presented in Table 1 for *Tub-GAN* can also be achieved for *Tub-sGAN*, where the segmentation results of *real + synthetic images* lead to a boost of performance to 80.49%. More details are shown in the supplementary file.

Fig. 13 displays the performance on the DRIVE dataset when our patch-based CNN baseline method is employed. Two curves are plotted for *Tub-GAN* and *Tub-sGAN*, respectively. Clearly, the performance of our *Tub-sGAN* variant is consistently better than the *Tub-GAN* variant. The first point of both curves starts at 79.15%, where the aforementioned baseline is trained with 400 K real patches only. As more synthetic patches are incorporated into the training set, the performances are improved for both of our synthesizing approaches. In particular, at $2\times$ (*i.e.* 400 K synthetic patches are incorporated into the training set), the respective performances of *Tub-GAN* and *Tub-sGAN* are 80.33% and 80.49%. At $3\times$ (*i.e.* 800 K synthetic patches are incorporated into the training set), the respective performances are increased to 80.68% and 80.88%; Both

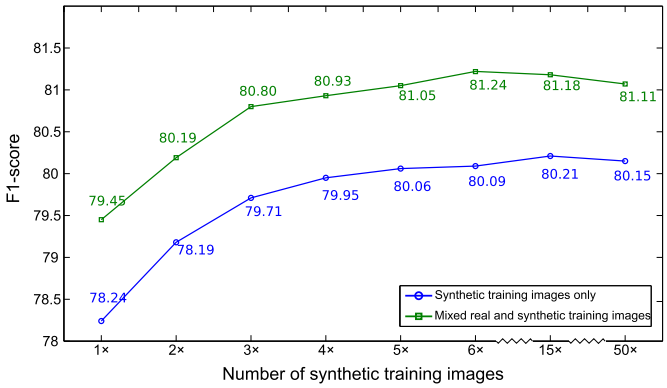


Fig. 14. Performance of our DRIU (Maninis et al., 2016) baseline segmentation model on the DRIVE benchmark as a function of the number of synthetic training images. The blue curve presents results from purely synthetic training images, that is, all images in the training set including the first 20 images are synthesized; The green curve displays results by augmenting the original training set with synthetic images, where the first 20 images are those original training images. See text for details. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

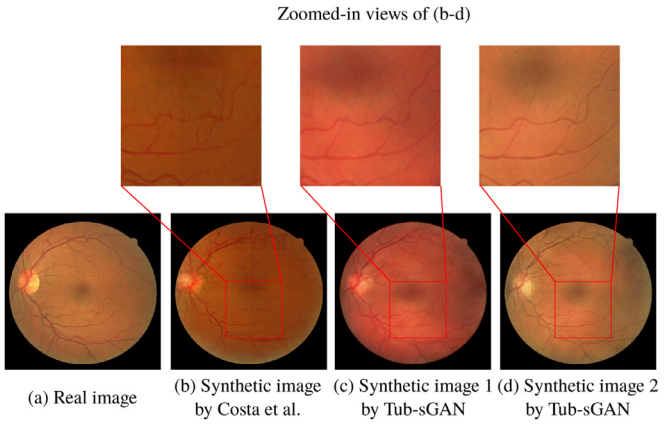


Fig. 15. Visual comparison of images generated by Costa *et al.* and by our approach. The same tubular structured annotation from the DRIVE test set is used as input for all comparison methods. (a) presents the corresponding real fundus image; (b) is the image generated by Costa et al. (2017); The remaining two images on the right-hand side in (c) and (d) are those generated by our approach. The zoomed-in views of (b)–(d) show the finer-detailed textures of the synthesized images.

curves reach a plateau at 1000 K synthetic training patches with their F1 scores of 80.67% and 80.91%, respectively.

5.2.2. DRIU baseline (Maninis et al., 2016)

By working with our DRIU segmentation baseline on the DRIVE benchmark, as shown in Fig. 14, the performance improves as more synthetic images are included as additional training examples. Here two training scenarios are considered, namely synthetic mixed images, and pure synthetic images, which are represented by the green and blue curves in the figure, respectively. $1\times$ refers to the entire 20 original training images, $2\times$ stands for the inclusion of additional 20 training images, etc, because DRIU takes an entire image as an input instead of working with patch inputs as in Fig. 13. In particular, both green and blue curves reach a plateau at around the point of $15\times$. Note that in the mixed scenario of the green curve, only 20 real images are involved which are the original training images, the rest are all synthetic images.

It is also worth noting that training our DRIU model with 1920 real images obtained by rotation and flipping data augmentation of the 20 original training images gives a state-of-the-art performance 81.66%, if we further augment these 1920 real images with 1920 synthetic training images, the performance is boosted to 81.72%. Two conclusions can be drawn from these experimental results. For segmenting tubular structured images, our approach could be useful as a data augmentation tool when the training set is *small*. The improvement however has its limit and it will diminish when excessive images are synthesized from a small pool of real training images. On the other hand, the synthetic images generated from our approach could still be beneficial even when the training set is large.

6. Discussions

6.1. Comparison with the method of Costa et al. (2017)

Experiments are conducted to specifically compare with the work of Costa et al. (2017). Fig. 15 presents the results of Costa et al. (2017) obtained with their original implementation versus those of ours. As is evidenced from this example, the images generated by Costa et al. (2017) contain artifacts such as chess board patterns and broken tubular structures. In general the phantom results are not as realistic as those of ours which usually excel in preserving proper connectivity of the vessel trees. We attribute this to the adoption of manual tubular structured annotation when training our models. The most important distinction is that our models are capable of producing different outputs from the same tubular structured annotations. This is in fact evidenced not only in *Tub-GAN* by varying the noise code $\boldsymbol{z}$ as *e.g.* shown in Fig. 8, but also in *Tub-sGAN* by engaging different style inputs as displayed in Fig. 15 as well as in Fig. 11.

To further validate quantitatively the performance of our results, we also use the SSIM score (Wang et al., 2004) for image quality assessment. More specifically, manual vessel annotations from the DRIVE test set are input into both the method of Costa et al. (2017) and that of our *Tub-sGAN* to generate phantom outputs. Then the SSIM score is computed between a pair of a real and a synthetic images. The average SSIM score of Costa et al. (2017) is 0.8716, which is elevated to 0.8980 by our approach. In order to compare the vessel tree structures generated by the two competing methods, the structure comparison module in SSIM is employed alone. The SSIM structure comparison scores of Costa et al. (2017) and ours are 0.9680 and 0.9812 respectively. These SSIM scores also indicate that our approach is capable of generating better results than that of Costa et al. (2017).

6.2. Pathological cases

Now we examine on how the proposed approach performs on clinical pathological cases both visually and quantitatively. As displayed in Fig. 16, several pathological images are used as style inputs to generate pathological looking retinal images, where tubular structured annotations from DRIVE are used as the corresponding content input. With different types of pathological cases such as cataract and diabetic retinopathy, our *Tub-sGAN* model is able to synthesize textural appearances following the style inputs.

In addition to the visual inspection above, quantitative evaluations are also conducted on pathological cases. A further analysis on the STARE dataset is presented in Table 2. Everything else is carried out as in Table 1, except for that the STARE testing images are split into two subsets: the sets of healthy and pathological images, respectively. It is observed that the segmentation performance of *e.g.* patch-based CNN on pathological cases is worse than that on healthy ones. However, when focusing on the pathological subset,the segmentation model trained with *real + synthetic images* still outperforms the one trained with *real images* only. This again reinforces what has been previously seen from Table 1.

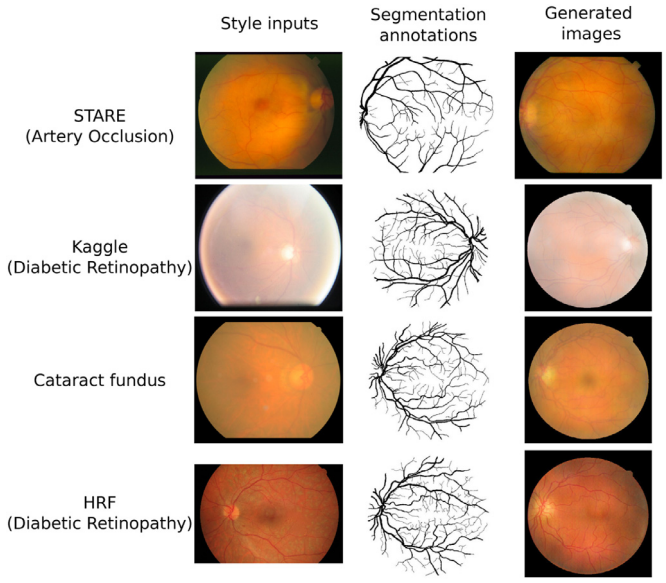


Fig. 16. Visual results of various pathological cases generated by our *Tub-sGAN* model, where the pathological styles are shown in the first column, the generated images are displayed in the last column. The tubular structured annotations in the middle column are all from DRIVE.

Table 2
Segmentation performance of patch-based CNN on the STARE testing set that is split into healthy and pathological cases. Here we directly employ the *Tub-GAN* model that is trained on the entire STARE train set and has been evaluated in Table 1. Results are provided using average F1-score (%). See text for details.

	Healthy	Pathological
Synthetic images	79.12	66.51
Real images	80.92	71.53
Real + synthetic images	82.20	72.54

Table 3
Segmentation performance of patch-based CNN on the STARE testset. Here, the healthy training images are used as the training set for retraining with the *Tub-GAN* model, while the pathological testing images are used as the testing set. Results are evaluated in terms of average F1-score (%), Specificity (%), and Sensitivity (%).

	Real images	*Real + synthetic images*
F1-score	74.08	75.18
Specificity	97.79	97.81
Sensitivity	74.76	76.69

We continue with another experiment on the STARE dataset. Now, the healthy training images are used alone as the training set. The trained model is then examined on pathological images. The same training and testing procedure with patch-based CNN is also adopted here, with the result shown in Table 3. Once again, it can be observed that the segmentation performance of training with *real + synthetic images* outperforms that of training with *real images* only, which follows the same trend we have observed previously.

6.3. Model limitation

Overall, our approach is capable of generating images that are realistic looking. The synthesized images could maintain the same tubular structures with different textural appearances. However, there are limitations that can be further addressed. For instance, some anatomical details are less than perfect. The position of the optic disc can be correctly generated in general, but the boundaries are often not as clear as those of the real images. Take for example the real retinal images in Fig. 17(a), where there usually exists a bright and clear boundary that separates the optic disc region away from the rest of the eye fundus. In comparison, the proposed model tends to synthesize a relatively more blurred boundary. Moreover, the macular region is sometimes also not entirely accurate. Fig. 17(b) presents the zoomed-in view of the macular regions. In most cases, such as the first synthesized image of Fig. 17(b), the generated macular regions are capable of visually resembling the real ones that possess a nearly round shape. Sporadically, the synthesized macular regions may deviate notably from the real ones, as for instance displayed in the second synthesized image of Fig. 17(b). As to neuronal images, much less issues have been observed, which we attribute to the comparably less sophisticated textural appearances in a typical neuronal image. Additionally, the style loss we have utilized considers a global difference for our *Tub-sGAN*. It works well in transferring the global textural style, but it is relatively weak in synthesizing local details, such as exudate regions. These issues remain to be investigated in the future.

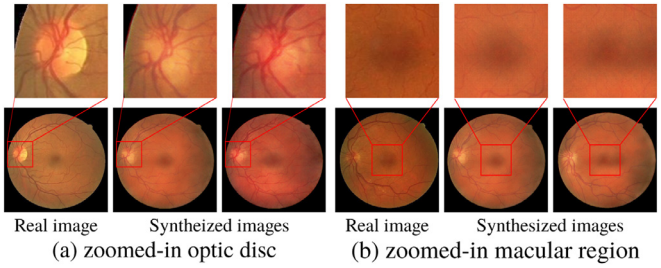


Fig. 17. Zoomed-in views of (a) optic disc and (b) macular regions.

7. Conclusion and future work

A data-driven approach is proposed to synthesize retinal fundus and neuronal images. Based on the same tubular structured annotation, multiple realistic-looking phantoms could be generated. Moreover, the model is capable of learning from small training sets of as few as 10–20 examples. Experimental results suggest that it helps to improve image segmentation performance when our synthesized images are used as additional training images. For future work, we consider the exploration of possible performance gains in other downstream tasks such as image level recognition or grading of diabetic retinopathy. We also plan to investigate the combination of different styles in one training procedure instead of training dedicated models for individual styles. Finally, its applications in related medical images such as magnetic resonance angiography and x-ray angiography are also of interest.

Acknowledgments

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, Malyatha Shridharan, and Nastaran Okati for their help with this project. We also thank the editors and the reviewers for their generous help in improving the presentation of this paper.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.media.2018.07.001.

References

Abramoff, M., Garvin, M., Sonka, M., 2010. Retinal imaging and image analysis. IEEE Trans. Med. Imaging 3, 169–208.

Annunziata, R., Kheirkhah, A., Aggarwal, S., Hamrah, P., Trucco, E., 2016. A fully automated tortuosity quantification system with application to corneal nerve fibres in confocal microscopy images. Med. Image Anal. 32, 216–232.
Arjovsky, M., Chintala, S., Bottou, L., 2017. Wasserstein GAN. arXiv.
Ascoli, G., Krichmar, J., 2000. L-Neuron: a modeling tool for the efficient generation and parsimonious description of dendritic morphology. Neurocomputing 32–33, 1003–1011.
Bower, J., Cornelis, H., Beeman, D., 2014. GENESIS, the general neural simulation system. In: Jaeger, D., Jung, R. (Eds.), Encyclopedia of Computational Neuroscience. Springer, pp. 1–8.
Carnevale, N., Hines, M., 2006. The NEURON Book. Cambridge University Press.
Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P., 2016. InfoGAN: interpretable representation learning by information maximizing generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2172–2180.
Cheng, L., Vishwanathan, S., Zhang, X., 2008. Consistent image analogies using semi-supervised learning. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8.
Ciregan, D., Meier, U., Schmidhuber, J., 2012. Multi-column deep neural networks for image classification. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 3642–3649.
Costa, P., Galdran, A., Meyer, M., Abramoff, M., Mendonca, A., Campilho, A., 2017. Adversarial synthesis of retinal images from vessel trees. In: ICIAR, pp. 516–523.
Costa, P., Galdran, A., Meyer, M., Niemeijer, M., Abràmoff, M., Mendonça, A., Campilho, A., 2018. End-to-end adversarial retinal image synthesis. IEEE Trans. Med. Imaging 37 (3), 781–791.
De, J., Cheng, L., Zhang, X., Lin, F., Li, H., Ong, K., Yu, W., Yu, Y., Ahmed, S., 2016. A graph-theoretical approach for tracing filamentary structures in neuronal and retinal images. IEEE Trans. Med. Imaging 35 (1), 257–272.
Denton, E., Chintala, S., Fergus, R., 2015. Deep generative image models using a Laplacian pyramid of adversarial networks. In: Advances in Neural Information Processing Systems, pp. 1486–1494.
Fiorini, S., Biasi, M., Ballerini, L., Trucco, E., Ruggeri, A., 2014. Automatic generation of synthetic retinal fundus images. In: Eurographics Italian Chapter Conference, pp. 41–44.
Fraz, M., Remagnino, P., Hoppe, A., Uyyanonvara, B., Rudnicka, A., Owen, C., Barman, S., 2012. Blood vessel segmentation methodologies in retinal images - a survey. Comput. Methods Programs Biomed. 108 (1), 407–433.
Fritzsche, K., Can, A., Shen, H., Tsai, C., Turner, J., Tanenbuam, H., Stewart, C., Roysam, B., 2003. Automated model based segmentation, tracing and analysis of retinal vasculature from digital fundus images. In: Suri, J., Laxminarayan, S. (Eds.), State-of-The-Art Angiography, Applications and Plaque Imaging Using MR, CT, Ultrasound and X-rays. Academic Press, pp. 225–298.
Gagalowicz, A., Philips, W., 2009. Computer Vision/Computer Graphics Collaboration Techniques. Springer.
Gatys, L., Ecker, A., Bethge, M., 2016. Image style transfer using convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2414–2423.
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680.
Grenander, U., 1976. Lectures in Pattern Theory I, II and III: Pattern Analysis, Pattern Synthesis and Regular Structures. Springer-Verlag.
Gu, L., Zhang, X., Zhao, H., Li, H., Cheng, L., 2017. Segment 2d and 3d filaments by learning structured and contextual features. IEEE Trans. Med. Imaging 36 (2), 596–606.
He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778.
Hertzmann, A., Jacobs, C., Oliver, N., Curless, B., Salesin, D., 2001. Image analogies. In: Proceedings of the 28th annual conference on Computer Graphics and Interactive Techniques, pp. 327–340.
Hoover, A., Kouznetsova, V., Goldbaum, M., 2000. Locating blood vessels in retinal images by piece-wise threshold probing of a matched filter response. IEEE Trans. Med. Imaging 19 (3), 203–210.
Ioffe, S., Szegedy, C., 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456.
Isola, P., Zhu, J., Zhou, T., Efros, A., 2016. Image-to-image translation with conditional adversarial networks. arXiv.
Johnson, J., Alahi, A., Fei-Fei, L., 2016. Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision, pp. 694–711.
Kingma, D., Ba, J., 2014. Adam: a method for stochastic optimization. arXiv.
Kingma, D., Welling, M., 2014. Auto-encoding variational Bayes. In: International Conference on Learning Representations.
Kirbas, C., Quek, F., 2000. A review of vessel extraction techniques and algorithms. ACM Comput. Surv. 36 (2), 81–121.
Köhler, T., Budai, A., Kraus, M., Odstrčilik, J., Michelson, G., Hornegger, J., 2013. Automatic no-reference quality assessment for retinal fundus images using vessel segmentation. In: IEEE Int. Sym. on Computer-Based Medical Systems, pp. 95–100.
Krizhevsky, A., Sutskever, I., Hinton, G., 2012. ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105.
Lesage, D., Angelini, E., Bloch, I., Funka-Lea, G., 2009. A review of 3D vessel lumen segmentation techniques: models, features and extraction schemes. Med. Image Anal. 13 (6), 819–845.
Maninis, K., Pont-Tuset, J., Arbelaez, P., Gool, L.V., 2016. Deep retinal image understanding. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 140–148.
Mao, X., Shen, C., Yang, Y., 2016. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In: Advances in Neural Information Processing Systems, pp. 2802–2810.
Martinez-Perez, M., Hughes, A., Stanton, A., Thom, S., Chapman, N., Bharath, A., Parker, K., 2000. Geometrical and topological analysis of vascular branches from fundus retinal images. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 756–765.
Menti, E., Bonald, L., Ballerini, L., Rugger, F., Trucco, E., 2016. Automatic generation of synthetic retinal fundus images: vascular network. In: Int. Workshop on Simulation and Synthesis in Medical Imaging, pp. 167–176.
Mirza, M., Osindero, S., 2014. Conditional generative adversarial nets. arXiv.
Nie, D., Trullo, R., anad S. Ruan, C.P., Shen, D., 2017. Medical image synthesis with context-aware generative adversarial networks. arXiv.
Oord, A.V., Kalchbrenner, N., Kavukcuoglu, K., 2016. Pixel recurrent neural networks. In: International Conference on Machine Learning, pp. 1747–1756.
Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A., 2016. Context encoders: feature learning by inpainting. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2536–2544.
Peng, H., Hawrylycz, M., Roskams, J., Hill, S., Spruston, N., Meijering, E., Ascoli, G., 2015. BigNeuron: large-scale 3d neuron reconstruction from optical microscopy images. Neuron 87 (2), 252–256.
Peng, H., Meijering, E., Ascoli, G., 2015. From DIADEM to BigNeuron. Neuroinformatics 13 (3), 259–260.
Radford, A., Metz, L., Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv.
Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 234–241.
Sagar, M., Bullivant, D., Mallinson, G., Hunter, P., 1994. A virtual environment and model of the eye for surgical simulation. In: Proceedings of the 21st annual Conference on Computer Graphics and Interactive Techniques, pp. 205–212.
Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., 2016. Improved techniques for training GANs. In: Advances in Neural Information Processing Systems, pp. 2234–2242.
Scarpa, F., Zheng, X., Ohashi, Y., Ruggeri, A., 2011. Automatic evaluation of corneal nerve tortuosity in images from in vivo confocal microscopy. Investig. Ophthalmol. Visual Sci. 52 (9), 6404–6408.
Shotton, J., Girshick, R., Fitzgibbon, A., Sharp, T., Cook, M., Finocchio, M., Moore, R., Kohli, P., Criminisi, A., Kipman, A., Blake, A., 2013. Efficient human pose estimation from single depth images. IEEE Trans. Pattern Anal. Mach. Intell. 35 (12), 2821–2840.
Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv.
Staal, J., Abràmoff, M., Niemeijer, M., Viergever, M., Ginneken, B.V., 2004. Ridge-based vessel segmentation in color images of the retina. IEEE Trans. Med. Imaging 23 (4), 501–509.
Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9.
Tenenbaum, J., Freeman, W., 2000. Separating style and content with bilinear models. Neural Comput. 12 (6), 1247–1283.
Trucco, E., Ruggeri, A., Karnowski, T., Giancardo, L., Chaum, E., Hubschman, J., Al-Diri, B., Cheung, C., Wong, D., Abramoff, M., Lim, G., Kumar, D., Burlina, P., Bressler, N., Jelinek, H., Meriaudeau, M., Quellec, G., MacGillivray, T., Dhillon, B., 2013. Validating retinal fundus image analysis algorithms: issues and a proposalvalidating retinal fundus image analysis algorithms. Investig. Ophthalmol. Visual Sci. 54 (5), 3546–3559.
Ulyanov, D., Lebedev, V., Vedaldi, A., Lempitsky, V., 2016. Texture networks: Feed–forward synthesis of textures and stylized images. In: International Conference on Machine Learning, pp. 1349–1357.
Wang, X., Gupta, A., 2016. Generative image modeling using style and structure adversarial networks. In: European Conference on Computer Vision, pp. 318–335.
Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E., 2004. Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. 13 (4), 600–612.
Wolterink, J., Leiner, T., Viergever, M., Isgum, I., 2017. Generative adversarial networks for noise reduction in low-dose CT. IEEE Trans. Med. Imaging 36, 2536–2545.
Xu, C., Cheng, L., 2013. Efficient hand pose estimation from a single depth image. In: IEEE International Conference on Computer Vision, pp. 3456–3462.