6. Supplementary of Semi-Supervised Semantic Segmentation of Vessel Images using Leaking Perturbations

6.1. Algorithm summary of the proposed model

Our model is summarized in Algo. 1. Furthermore, LeakGAN can be used for the cross-domain scenarios, which the unlabelled target $\mathbf{X}^t_{\text{ul}}$ also is fed into the model, as shown in Algo. 2.

Algorithm 1: Training of the LeakGAN

Input: Source: $\mathbf{X}^s_l, y^s_l, \mathbf{X}^s_{\text{ul}}$
Result: Segmentation classifier (discriminator)
$\boldsymbol{\theta}_D, \boldsymbol{\theta}_G, \boldsymbol{\theta}_{\text{MT}} \leftarrow$ initialization
for *iterations of traning* **do**
  $\mathbf{X}^s_l, y^s_l, \mathbf{X}^s_{\text{ul}} \leftarrow$ sample mini-batch
  $\boldsymbol{z} \leftarrow$ sample from $\mathcal{N}(0, I)$
  Generate $\widetilde{\mathbf{X}}_f$ by feeding $\boldsymbol{z}$ through generator G
  $\mathcal{L}_{\text{sup}}, \mathcal{L}_{\text{unsup}}, \mathcal{L}_{\text{cons-FL}}, \mathcal{L}_{\text{adv-}} \leftarrow$ calculated by Eq. (2), Eq. (3), Eq. (13) and Eq. (15)
  for *iterations of inference model updating* **do**
    $\boldsymbol{\theta}_D \leftarrow - \Delta_{\boldsymbol{\theta}_D} \mathcal{L}^*$
  end
  for *iterations of discriminator updating* **do**
    $\boldsymbol{\theta}_G \leftarrow -\Delta_{\theta_G} \mathcal{L}_{\text{adv-}}$
  end
end

Algorithm 2: Training of the cross-domain LeakGAN

Input: Source: $\mathbf{X}^s_l, y^s_l, \mathbf{X}^s_{\text{ul}}$, Target:$\mathbf{X}^t_{\text{ul}}$
Result: Segmentation classifier (discriminator)
$\boldsymbol{\theta}_D, \boldsymbol{\theta}_G, \boldsymbol{\theta}_{\text{MT}} \leftarrow$ initialization
for *iterations of traning* **do**
  $\mathbf{X}^s_l, y^s_l, \mathbf{X}^s_{\text{ul}}, \mathbf{X}^t_{\text{ul}} \leftarrow$ sample mini-batch
  $\boldsymbol{z} \leftarrow$ sample from $\mathcal{N}(0, I)$
  Generate $\widetilde{\mathbf{X}}_f$ by feeding $\boldsymbol{z}$ through generator G
  $\mathcal{L}_{\text{sup}}, \mathcal{L}_{\text{unsup}}, \mathcal{L}_{\text{cons-FL}}, \mathcal{L}_{\text{adv-}} \leftarrow$ calculated by Eq. (2), Eq. (3), Eq. (13) and Eq. (15)
  for *iterations of inference model updating* **do**
    $\boldsymbol{\theta}_D \leftarrow - \Delta_{\boldsymbol{\theta}_D} \mathcal{L}^*$
  end
  for *iterations of discriminator updating* **do**
    $\boldsymbol{\theta}_G \leftarrow -\Delta_{\theta_G} \mathcal{L}_{\text{adv-}}$
  end
end

6.2. Implementation

Our experiments are based on implementations using TensorFlow [1]. Our model is trained by the patches randomly cropped from the training images. The patch size is 64×64. The structure of the generator is similar to that of DCGAN [31], which performs well on image generation tasks. A 1-D random normal noise with length 100 is fed to the generator. It passes through a fully connected layer, and the activation is reshaped to 8×8. There are overall six convolution layers following the fully-connected layer. LeakyReLU is used as the activation function except for the last layer, where tanh() is used instead. Transposed convolution layers with stride 2 are utilized for the upsampling. Batch normalization is added between the convolutional layers. The intermediate outputs of the generator, which have the same frame size as the discriminator's, are used as the leaked information. The discriminator is of U-Net style and has a structure similar to the model proposed in [28]. The kernel of the consecutive convolution layers is 3×3, and max-pooling with step 2 is utilized to downsample the image patches. After four stacked blocks of convolution and downsampling layers, the patches are encoded to the size of $4\times4\times512$. The decoder of the discriminator has a structure symmetric to the encoder. The encoder's output, also the intermediate outputs of the generator, are concatenated to the decoder for the further upsampling. During the experiments, the output of the first convolutional layer of the generator (features with size 8×8) is utilized for the leaking module. It spends around 1.69s, 1.72s, and 4.70s to generate each image prediction in patch-level for DRIVE, STARE, and CHASE_DB1, respectively, when our model is run on a single NVIDIA GTX1080 GPU.

6.3. Qualitative results for STARE and CHASE_DB1

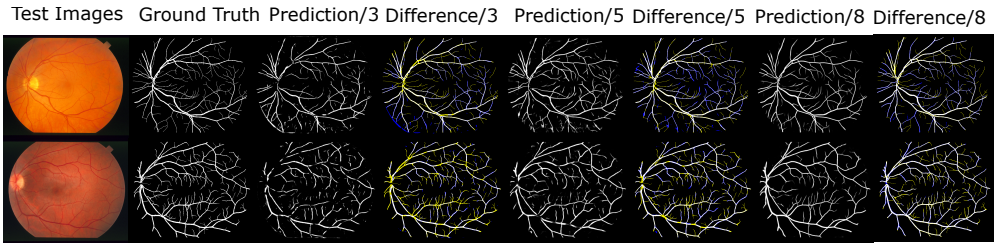
In this section, more visualization of the experiments on STARE and CHASE_DB1 are demonstrated.

The whole-image prediction of the STARE and CHASE_DB1 are shown in Fig. 6.

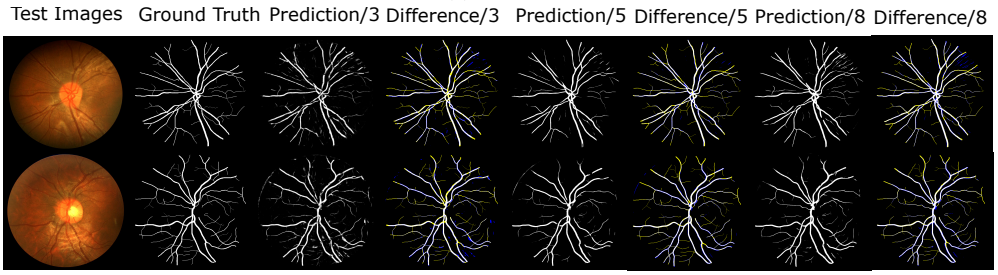
Also, in the sixth column of Fig. 7a, the vessels of STARE are detected well though it has some blue pixels around them. However, some vessels' structures disappear (as false negative) in the predictions, e.g. as shown in the fourth column of Fig. 7b.

6.4. More scenarios of cross-domain evaluations

We extend the cross-domain segmentation evaluation onto another scenario: training the LeakGAN framework on the healthy images and testing it on the pathological images. For the DRIVE dataset, there are seven images from diabetic patients. We also evaluate the performance for the health retina images as training images, the diabetic patient's retina images for the test. The results are given in Table 7, and

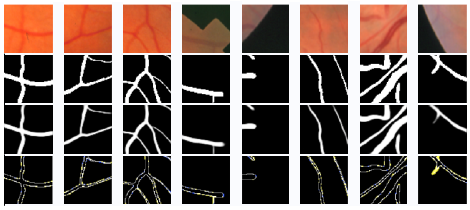


(a) STARE

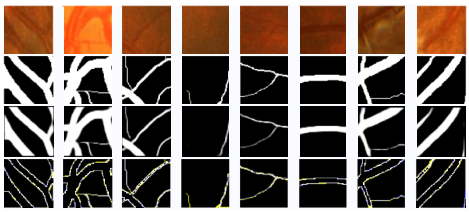


(b) CHASE_DB1

Figure 6: Visualization of the semantic segmentation on testing images. The first two columns are test images and their ground truth. Then the results are shown when 3, 5 and 8 labelled training images are used. Its differences to ground truth are shown next to the prediction columns, with blue pixels being false positive, and yellow being false negative.



(a) Patch segmentation of STARE



(b) Patch segmentation of CHASE_DB1

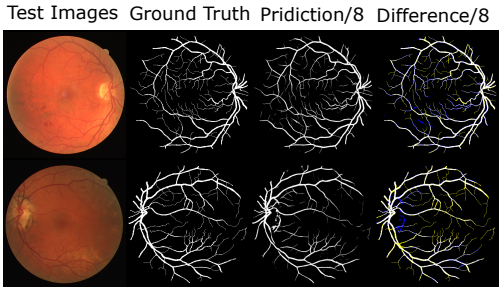
Figure 7: Patched segmentation results for the scenario of 8 labelled training image using different datasets. The testing patches and their ground truth are in first and second row respectively. The third row is for the predictions. The difference between the prediction and ground truth is shown in the last row, blue pixels being false positive, and yellow being false negative.

segmentation examples are shown in Figure 8a. Also the visualization of the scenario from STARE to DRIVE is shown in Fig. 8b. From these results, we can see that the model performs reasonably in this scenario.

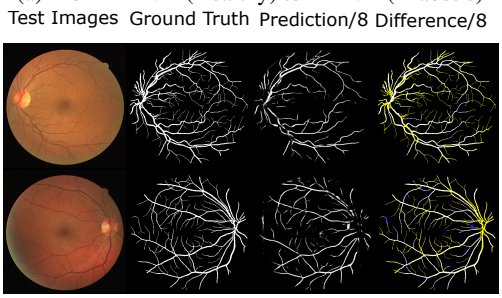
Furthermore, we evaluate our model's generalization when it is applied to unseen datasets. During the experiments, the model is trained following Algorithm 1 on one dataset, while the testing is done on retina images of another dataset. For example, these results are obtained: Acc(%)=95.06, Sp(%)=92.15, and Se(%)=83.37, when the transfer from DRIVE (training) to STARE (testing) is considered. This shows that our model also generalizes well to unseen data resources.

Table 7: Performance for training on DRIVE(H), testing on DRIVE(D). Numbers for the training setup are for labelled and unlabelled images respectively.

Train Dataset Test Dataset	DRIVE(H)(5+5) DRIVE(D)	DRIVE(H)(10+10) DRIVE(D)	DRIVE(H)(10+23) DRIVE(D)
Acc(%)	92.15	92.23	93.61
Sp(%)	85.39	85.32	80.52
Se(%)	97.53	95.16	98.30



(a) From DRIVE (Healthy) to DRIVE (Diabetic)



(b) From STARE to DRIVE

Figure 8: Visualization of the cross-domain prediction results. The test images and their ground truth are shown in the first two columns. The predictions are listed on the third. The difference between the prediction and ground-truth is shown in the last column overlaying on ground-truth, with blue pixels indicating false positive, and yellow indicating false negative.