TECHNICAL UNIVERSITY OF MUNICH

TUM Data Innovation Lab

Domain Adaptation for Annotation-Efficient Image Segmentation

Authors	Hanzhi Chen, Yun-Fei Hsu, Zeju Qiu
Mentor(s)	Dr. Ghazal Ghazaei, Dr. Stratis Tzoumas (Carl Zeiss AG)
Co-Mentor	M.Sc. Laure Vuaille (Department of Mathematics)
Project Lead	Dr. Ricardo Acevedo Cabra (Department of Mathematics)
Supervisor	Prof. Dr. Massimo Fornasier (Department of Mathematics)

Jul 2021

Abstract

Optical coherence tomography (OCT) is an imaging technique that captures micrometer resolution of 3D images from within biological tissue. Tracking the OCT volumes of three fluid types including Intraretinal Fluid (IRF), Subretinal Fluid (SRF), and Pigment Epithelial Detachment (PED) has proved to be helpful in medical treatment decision-making. Recently, learning-based method for semantic segmentation has indicated superior performance for medical image understanding and analysis, which benefit OCT volumes component tracking as well. However, such a method requires a large amount of manual annotations to supervise the training procedure. Domain adaptation technique is suggested to be one possible technique to tackle such issue. The idea behind it is to leverage a small amount of annotated dataset for training, and further utilize such model for the unseen domain to generate plausible predictions with less requirement for annotations.
In this project, we first build a baseline UNet model to perform semantic segmentation tasks and then investigate two methods for domain adaptation, namely the CycleGAN and the FCAN method. We further evaluate all variants of our method on the RETOUCH dataset with respect to both style consistency and semantic segmentation performance in the target domain. The experimental results show that our methods can successfully achieve adaptation on unseen domains and further effectively benefit semantic segmentation tasks requiring less effort for manual annotations.

Contents

1 Introduction

Optical coherence tomography (OCT) is an imaging technique that capture micrometer resolution of 3D images from within biological tissue. It can provide high-quality images of the retinal structure and visualize the accumulated fluid within the intracellular space of the retina to monitor diseases such as age-related macular degeneration (AMD). As AMD is the major cause of blindness in developed countries, the developed treatment regimens are deeply depend on the retinal bleeding or fluid accumulation. Therefore, tools for quantitative assessment of retinal health are urgently needed.

1.1 Medical Image Semantic Segmentation

Tracking the volume of three fluid types including Intraretinal Fluid (IRF), Subretinal Fluid (SRF), and Pigment Epithelial Detachment (PED) has proved to be helpful in treatment decision-making[22][21][13]. Although 3D OCT images can visualize the region of the fluids, it acquires large amounts of information that is difficult to manually interpret the changes of fluids. Thus, the quantitative measurements of the region size require automated computational algorithms. A fully convolutional neural network (FCN) is typically used for such semantic segmentation tasks[12], in particular, the U-net[16] structure is superior to other methods in segmenting medical images. In the RETOUCH challenge launched in 2017, 8 teams are competing to create a representative benchmark to evaluate algorithms for detecting and segmenting all three types of fluids[2]. Five of the eight teams, including the first prize, built their models based on U-net. Therefore, the network we used for segmentation is based on the U-net structure as well, which will be illustrated in Section 2.3.

1.2 Domain Adaptation

Although the fully convolutional neural network has led to great advances in medical image segmentation, the performance highly depends on the amounts of labeled samples and the variation in the training and testing domain. In practice, marking medical images requires expertise and the labeled data are often expansive or infeasible to obtain. In addition, the domain on which the segmentation network is trained might have different appearance distribution with the collected images. The variation often happens when the different scanners are used, different imaging sequences, or the bias in patient cohorts[20]. One way to close the performance gap is to apply domain adaptation, which is usually used to solve the domain disparity problem and can be categorized into three types: supervised, semi-supervised and unsupervised. Our work mainly focuses on unsupervised adaptation which adapts domains without using labeled data from the target domain. We tackle the adaptation task from two different directions: pixel-level adaptation and feature-level adaptation. The CycleGAN model[27] provides an image to image translation, while the FCAN[26] combines both pixel adaptation and latent space calibration. The detail of the two methods will be illustrated in Section 3.

1.3 Project Objectives

The objective of this work is to conduct a label-efficient unsupervised domain adaptation for semantic segmentation. We realize domain mapping approaches that can generalize to new domains with a minimal amount of annotations. In the article, we show the reliable segmentation performance on two domain adaptation techniques. We also provide a comprehensive comparison of each method on the appearance and the segmentation results in both quantitative and qualitative ways.

2 Medical Image Segmentation

In this chapter, we will mainly focus on how we design and implement the segmentation model that will be later used for evaluation on domain adaptation performance in Chapter 3. We first introduce the OCT dataset we experimented with, then formulate our problem, introduce the implementation details and evaluation metrics. Finally, we carried out several ablative experiments to study the effect of different loss settings and illustrate the final segmentation performance of the trained models on Cirrus dataset and Spectralis dataset.

2.1 RETOUCH dataset

OCT scans from the RETOUCH dataset are utilized in our experiments. The dataset consists of 70 volumes in total collected from three different devices, with 24 volumes from Cirrus (Zeiss Meditec), 24 volumes from Spectralis (Heidelberg Engineering) and the rest 22 volumes from T-1000 and T-2000 (Topcon). There are 128 B-scans for each volume from Cirrus and Topcon, and 49 B-scans from Spectralis. In our experiments, we mainly utilize volumes acquired from Cirrus and Spectralis, which give out 3072 2D B-scans and 1176 2D B-scans in total, respectively.
Three different classes of fluid manually labeled with ground truth in pixel-level for each OCT B-scan: Intraretinal Fluid (IRF), Subretinal Fluid (SRF) and Pigment Epithelial Detachment (PED).
In order to establish baseline evaluation for domain adaptation performance, we train two independent segmentation baseline network for B-scans from Cirrus and Spectralis, respectively. For these two different baseline models, We always select 18 volumes for training, 2 volumes for validation, and 2 volumes for testing.

2.2 Problem Formulation

Before diving into the implementation details of the segmentation model, we would like to formulate the semantic segmentation problem at first.
Given an input image $I \in \mathbb{R}^{h \times w}$ (B-scan in our scenario), a segmentation model f_θ parameterized with θ is utilized to predict a label $\hat{Y}^{(i)} \in \{1, ..., K\}$ for each queried pixel x with k being the number of classes and $h \times w$ being the resolution of the image. In our scenario, K equals 4 since there are three different fluids and a background class in total, and we resize the input B-scan to a resolution of 512×512 for all of the experiments.

Generally, f_θ is modeled as a convolutional neural networks (CNN). In that case, the parameters θ are learned in a fully supervised manner from a training set $S = \{I^{(i)}, Y^{(i)}\}, 1 \leq i \leq N$, with N training pairs of images I^i together with the corresponding annotated pixel-level labels $Y^{(i)}$. To supervise such model, a pixel-wise loss function $\mathcal{L}(f_\theta(I^{(i)}), Y^{(i)})$ that imposes penalization between discrepancy between predicted labels and annotated ground-truth labels will be minimized to guide the model to learn a meaningful mapping from color intensity to class label. We will introduce the pixel-wise loss in details in the coming sections.
In practice, one-hot encoding is utilized to map the label $Y^{(i)}$ from $\mathbb{R}$ to $\mathbb{R}^K$ to acquire $\mathbf{y}^{(i)}$ for training. As a result, f_θ is also trained to provide a prediction on probabilistic distribution of the class labels. To be more specific, given each pixel intensity of an input image $I^{(i)}_{m,n} \in \mathbb{R}$ with (m, n) being the coordinate of the pixel, the model is formulated as a mapping $f_\theta : \mathbb{R} \to \mathbb{R}^K$ to obtain scores on each class $\hat{\mathbf{y}}^{(i)}_{m,n} \in \mathbb{R}^K$. During training phase, we obtain the probability of the label k, $1 \leq k \leq K$, through Softmax normalization operator σ defined as following:

$$\sigma : \mathbb{R}^K \to \mathbb{R}^K, \sigma(\hat{\mathbf{y}}^{(i)}_{m,n})_k = \frac{\exp(\hat{\mathbf{y}}^{(i)}_{m,n})_k}{\sum_{k'=1}^{K} \exp(\hat{\mathbf{y}}^{(i)}_{m,n})_{k'}} \tag{1}$$

where $\mathbf{z}$ is the output scores of each class, and we normalize $\mathbf{z}$ to obtain class probability distribution that summed to be 1.
While during inference phase, we simply take the class with highest score without Softmax normalization as the final prediction $\hat{Y}^{(i)}_{m,n} \in \mathbb{R}$:

$$\hat{Y}^{(i)}_{m,n} = \underset{k' \in \{1,\ldots,K\}}{\arg\max}\ f_\theta(I^{(i)}_{m,n})_{k'} \tag{2}$$

2.3 Implementation Details

2.3.1 Fully Convolutional Neural Network

A shown in Fig. 1, The network architecture shares a similar structure as the standard U-Net architecture [16], which is widely used in medical image segmentation problems.
We mainly made three modifications on top of the baseline architecture. First, we replace every transpose convolution layer with bilinear upsampling layer. The second change we made is to add Batch Normalization layer [10]after every convolutional block to stabilize the learning process. Finally, a Dropout layer [18] was inserted before the final 1×1 convolutional layer for class prediction to avoid overfitting. During the training phase, only a portion of the units were randomly sampled to be fed into the next convolutional block, while in the testing stage, all the units were kept to generate the prediction. This operation can reduce overfitting by preventing co-adpation on the training data.

2.3.2 Loss Functions

As described in the last sections, a pixel-wise loss function will be optimized to train the model. During the training phase, we mainly experimented two different loss functions, namely Cross-Entropy Loss and Dice Loss. We will give a more detailed definition in this section.

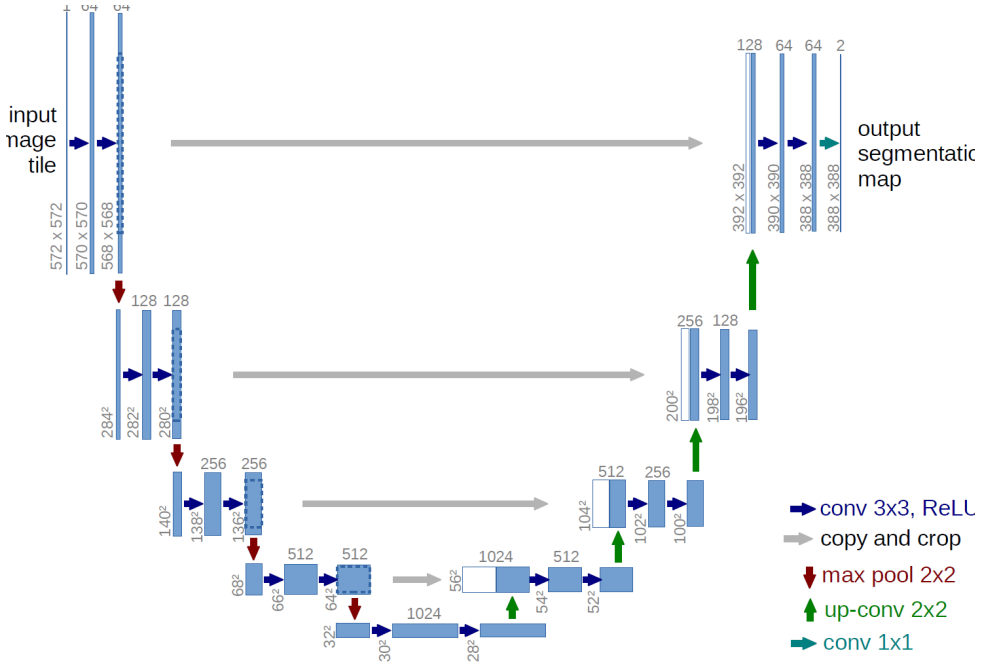

Figure 1: UNet architecture

Cross-Entropy Loss. The Cross-Entropy Loss is defined as the following:

$$\mathcal{L}_{CE} = -\frac{1}{h \times w \times N} \sum_{i=1}^{N} \sum_{n=1}^{h} \sum_{m=1}^{w} \sum_{k=1}^{K} \left([(\mathbf{y}_{m,n}^{(i)})_k = 1] \log(\sigma(\hat{\mathbf{y}}_{m,n}^{(i)})_k) \right) \tag{3}$$

where $[\cdot]$ is the Iverson bracket, $\sigma(\cdot)$ is the Softmax operator for normalization defined above , $\mathbf{y}_{m,n}^{(i)}$ is the one-hot encoded ground truth label, and $\hat{\mathbf{y}}_{m,n}^{(i)}$ is the predicted scores on each class.

Dice Loss. We also experiment with Dice Loss, which aims to deal with the scenarios where the positive and negative samples are strongly imbalanced in semantic segmentation. It is formulated as the following:

$$\mathcal{L}_{Dice} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \left(1 - \frac{\sum_{n=1}^{h} \sum_{m=1}^{w} [(\hat{\mathbf{y}}_{m,n}^{(i)})_k \cdot (\mathbf{y}_{m,n}^{(i)})_k]}{\sum_{n=1}^{h} \sum_{m=1}^{w} [(\hat{\mathbf{y}}_{m,n}^{(i)})_k + (\mathbf{y}_{m,n}^{(i)})_k] + \epsilon} \right) \tag{4}$$

where $\mathbf{y}_{m,n}^{(i)}$ is the one-hot encoded ground truth label, and $\hat{\mathbf{y}}_{m,n}^{(i)}$ is the predicted scores on each class, which shares consistent definition as above. ϵ is used here to ensure numerical stability, and it is to 1×10^{-5} in all experiments.

Finial Training Loss Finally, we combine Cross-Entropy Loss $\mathcal{L}_{CE}$ and Dice Loss$\mathcal{L}_{Dice}$ together, with equally weighted effect as final loss $\mathcal{L}$ to supervise final segmentation models trained on Cirrus and Spectralis dataset:

$$\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_{Dice} \tag{5}$$

2.3.3 Training Strategies

Our models are implemented in PyTorch [14], using Adam optimizer [11] with a batch size of 6 and an input/output resolution of 512×512 unless otherwise specified. We use a learning rate of 0.001 at the beginning and half it every 15 epochs, and train for a total of 80 epochs on an NVIDIA Quadro P5000. The Dropout layer before the last convolutional layer is initialized with a probability $p = 0.3$.
Due to the limited number of training samples of each device especially for Spectralis, we also perform data augmentation to increase the robustness and invariance properties of the network. We maily apply random horizontal flipping with a probability of 0.4, and random rotation within $-20°$ and $+20°$ around image center with a probability of 0.5 to each image during training phase. It is worth noting that during inference phase no data augmentation is applied on input images.
We evaluate the performance of the segmentation models by computing the Dice score accuracy on the validation set after every 5 epochs to select the best segmentation model for later domain adaptation performance evaluation.

2.4 Evaluation Metrics

F1-score metric (Dice score). We use Dice Score to evaluate the performance of the segmentation model and the improvement of the performance in segmentation tasks before and after the image adaptation. Regardless of the background, we only consider the dice score of the three fluid types. For each class, given two sets X and Y, the formula is defined as following:

$$Dice(X, Y) = \frac{2 * |X \cap Y|}{|X| + |Y|} \tag{6}$$

where X denote the set of predictions and Y denote the set of one-hot encoded ground truth label of the class, $|X|$ and $|Y|$ are the cardinalities of the two sets. We filter out images within a batch that doesn't include any pixels belonging to the class since we are more interested in the non-empty label set $Y \notin \emptyset$.

2.5 Results & Ablative Study

In this section, we first present the ablative study results for different effect of the Cross-Entropy Loss $\mathcal{L}_{CE}$ and Dice Loss$\mathcal{L}_{Dice}$. Notably, every model variant for ablative study is trained on Spectralis dataset. After deciding the best setting of the loss, we trained two segmentation models with the same training strategy on Cirrus dataset and Spectralis dataset, respectively. These two models will be further utilized to evaluate the performance of the domain adaptation in the next chapter.

Table 1: Performance on different model variants

Loss Setting	Trained Dataset	Test Dataset	Mean-Dice	IRF	SRF	PED
$\mathcal{L}_{CE}$	Spectralis	Spectralis	0.735	0.687	0.875	0.643
$\mathcal{L}_{Dice}$	Spectralis	Spectralis	0.742	0.698	0.883	0.646
$\mathcal{L}_{CE} + \mathcal{L}_{Dice}$	Spectralis	Spectralis	0.751	0.722	0.876	0.656
$\mathcal{L}_{CE} + \mathcal{L}_{Dice}$	Cirrus	Cirrus	0.505	0.466	0.651	0.400
$\mathcal{L}_{CE} + \mathcal{L}_{Dice}$	Spectralis	Cirrus	0.002	0.006	0.000	0.000
$\mathcal{L}_{CE} + \mathcal{L}_{Dice}$	Cirrus	Spectralis	0.489	0.299	0.620	0.548

Loss effect As illustrated in Table 1, both Cross-Entropy Loss and Dice Loss can contribute to the performance of the model. Since Cross-Entropy Loss can be treated as a standard baseline choice for segmentation task, we mainly investigate how much the Dice Loss can boost the accuracy in this section. It can be seen that the combination of both losses yields the best performance of the model, so we opt to this loss setting for the segmentation model training.

Segmentation Performance We trained two segmentation models on Cirrus dataset and Spectralis dataset with the best training strategy exploited. From Table 1, we can see that our segmentation models can generate reliable segmentation predictions and successfully generalize well to unseen data sampled from the trained domain, which are scans collected from Cirrus or Spectralis.

Generalization on Unseen Domain We also carried out cross-domain validation experiments to verify the generalization ability of our model on unseen domain. It can be seen that the model trained with Cirrus dataset reveals decay of performance when testing on Spectralis dataset, while for the model trained on Spectralis dataset, it shows poor generalization on unseen domain. We attribute it to the unequal amount of dataset since the total amount of Spectralis dataset is approximately 30% of that of Cirrus dataset.
It is worth noting that in this project we don't intend to design a model that is perfectly robust across domains. Instead, the models that is able to generate accurate predictions from trained domain and performs poorly on unseen domain are sufficient for us to assess the domain adaptation performance, which also reflects the importance of cross-domain adaptation for more robust predictions.

3 Domain Adaptation

3.1 GAN-based Methods

3.1.1 Cycle-GAN Framework

Cycle-GAN [27]is a framework trained with unpaired images drawn from two domains that for allows bidirectional domain-to-domain translation. The two domains in our scenario

would be Cirrus images and Spectralis images.
The goal of this method is to learn functions between two domains X and y with the training samples drawn from their corresponding domain in a *unsupervised* manner. Here we denote the training set from domain X and domain Y as $\{\mathbf{x}^{(i)}\}, 1 \leq i \leq N$, and $\{\mathbf{y}^{(j)}\}, 1 \leq j \leq M$.
The adaptation models consist of two mapping functions $G_Y : X \rightarrow Y$ and $G_X : Y \rightarrow X$, which are generators parameterized using convolutional neural networks. Apart from the generators above, two discriminators D_X and D_Y are established as well to train the corresponding generators G_X and G_Y in an adversarial manner.
To illustrate, the generator G_X receives input image $\mathbf{y}$ drawn from counterpart domain Y and outputs a synthetic translated result $\hat{\mathbf{x}} = G_X(\mathbf{y})$ which appeared to be drawn from domain X. Meanwhile, the discriminator D_X receives the synthetic results $\hat{\mathbf{x}}$ and an *unpaired* sample $\mathbf{x}$ randomly drawn from the domain X.
Therefore, G_X and D_X together are essentially under a generative adversarial model (GAN) setting, where G_X and D_X are pitted against each other in competition. On the one hand, D_X aims to distinguish between the translated samples and the real samples drawn from domain X; on the other hand, G_X is optimized to generate the synthetic images with high quality that are able to deceive D_X. As for generator G_Y and discriminator D_Y, they work in an analogous manner to G_X and D_X.
This training procedure is formulated as a min-max optimization guided by an adversarial loss $\mathcal{L}_{adv}$, which will be further discussed in the next section.

3.1.2 Loss Functions

As described in the last section, an adversarial loss will be optimized to train the generators and discriminators. In this section, we would first introduce the adversarial loss that plays a key role. Besides, Cycle Consistency Loss, Style Transfer Loss and Content Preserving loss utilized to boost the adaptation performance will also be discussed.

Adversarial Loss. Binary-Cross Entropy loss is applied to both pairs of generators and discriminators as adversarial loss [5]. We can at first define The adversarial loss for one pair of generator and discriminator, e.g, G_X and D_X, which is expressed as below:

$$\mathcal{L}_{adv}(G_X, D_X) = \mathbb{E}_{x \sim p_{data}(x)}[\log(D_X(x))] + \mathbb{E}_{y \sim p_{data}(y)}[\log(1 - D_X(G_X(y))] \tag{7}$$

where G_X aims to minimize this objective against D_X that tries to maximize it. Analogously, the same adversarial objective is also designed for the other pair of generator and discriminator, namely G_Y and D_Y: $\mathcal{L}_{adv}(G_Y, D_Y)$.

Cycle Consistency Loss. As discussed in [27], even adversarial training could theoretically guide the generators G_X and G_Y to learn the distribution of the target domain, i.e. X and Y. However, adversarial loss only is still not sufficient since it could possibly lead the generators to map the same input images from source domain to the target domain with random permutation. Therefore, a cycle consistency loss is proposed as a regularization to constraint the space of mapping in the target domain. This is achieved by enforcing

the two generators, namely G_X and G_Y, to be cycle-consistent with each other. More specifically, for each sample x drawn from domain X, the cycle-translated results (first X to Y and then Y to X) shall be identical to its original input, i.e., $x \approx G_X(G_Y(x))$. This behavior is formulated with pixel-wise L1 norm objective as the following:

$$\mathcal{L}_{cyc}(G_X, G_Y) = \mathbb{E}_{x\sim p_{data}(x)}[\|G_X(G_Y(x)) - x\|_1] + \mathbb{E}_{y\sim p_{data}(y)}[\|G_Y(G_X(y)) - y\|_1] \quad (8)$$

The success of Cycle-GAN mainly grounded on introducing cycle consistency loss $\mathcal{L}_{cyc}$ that could reduce the space of possible mapping functions and further avoid mismatches between input images and synthetic images that could occur due to unpaired training. Moreover, $\mathcal{L}_{cyc}$ can also be well understood as a regularizer for contents preserving to avoid random permutation as domain translation.
However, as discussed in previous study that pixel-wise losses for such regularization usually fail to capture the perceptual aspect on human judgements [15]. Therefore, we propose to appraise the style similarity and perceptual similarity for input images and synthetic translated images in the feature space with extractor F, which further brings the style transfer loss and content preserving loss. On the one hand, style transfer loss is utilized to ensure the synthetic image shares a similar style and texture details as samples drawn from the target domain by imposing a constraint on Gram matrices; on the other hand, content preserving loss is applied to better ensure the semantics within the input image is well preserved regardless of domain translation. These two losses can be treated as a regularizer for each other as well.

Style Transfer Loss. Given input images x and its synthetic counterpart $G_Y(x)$, we feed them together to a feature extractor F to acquire multi-level feature maps and compute the internal correlations at each level. These internal correlations are represented by the Gram matrices, which is defined as:

$$Gr(x)^l_{m,n} = \text{vec}[F(x)^l_m]^\top \text{vec}[F(x)^l_n] \quad (9)$$

where vec$[\cdot]$ is the vectorization operation, $F(x)^l_m$ refers to the $m-$th channel of the feature maps extracted in the $l-$th level from feature extractor F for input x.
The style transfer loss is further computed as follows:

$$\mathcal{L}_{style}(G_X, G_Y) = \mathbb{E}_{(x,y)\sim p_{data}(x,y)} \sum_{l=1}^{2} \Big[\|Gr(G_Y(x)) - Gr(y)\|_F + \|Gr(G_X(y)) - Gr(x)\|_F \Big] \quad (10)$$

where $\|\cdot\|_F$ represents the Frobenius norm, and we compare the style of features maps from the first two layers of feature extractor F between the translated image and a random sample randomly drawn from the target domain.

Content Preserving Loss. Content preserving loss is applied to ensure better semantics preserving after domain adaptation, which is realized by comparing features maps extracted. We formulate the content preserving loss as follows:

$$\mathcal{L}_{content}(G_X, G_Y) = \mathbb{E}_{(x,y)\sim p_{data}(x,y)} \sum_{l=3}^{4} \Big[\|F(G_Y(x))^l - F(x)^l\|_1 + \|F(G_X(y))^l - F(y)^l\|_1 \Big] \quad (11)$$

where $\|\cdot\|_1$ represents the L1 norm, and we only compare features maps from the last two layers of feature extractor F between the original input and its translated counterpart.

Full Objective. Combining the above losses, the full optimization task of our proposed model is given by:

$$\min_{G_X,G_Y}\max_{D_X,D_Y}\mathcal{L}(G_X,G_Y,D_X,D_Y)=\mathcal{L}_{adv}(G_X,D_X)+\mathcal{L}_{adv}(G_Y,D_Y)+\lambda_{cyc}\mathcal{L}_{cyc}(G_X,G_Y) \\ +\lambda_{style}\mathcal{L}_{style}(G_X,G_Y)+\lambda_{content}\mathcal{L}_{cotent}(G_X,G_Y) \tag{12}$$

where λ_{cyc}, λ_{style} and λ_{cotent} are weight factors to balance the effect of cycle consistency loss, style transfer loss, and content preserving loss. We aim to solve:

$$G_X^\star,G_Y^\star=\arg\min_{G_X,G_Y}\max_{D_X,D_Y}\mathcal{L}(G_X,G_Y,D_X,D_Y) \tag{13}$$

3.1.3 Feature Extractors

As discussed in Section 3.1.2, a feature extractor is desired to measure the difference regarding style and content after domain translation to boost the performance of adaptation. In this section we will introduce 2 variants of feature extractors we experimented.

Discriminator-based Extractors For discriminator-based extractors, the discriminators are not only utilized to distinguish fake or real images, but required to extract high-level features for the input images, which can be interpreted that the feature extractor is trained with generators and discriminators as well. We then take every level of features from discriminators to compute style transfer loss and content preserving loss. More specifically, the feature extractor F is defined as below:

$$F(z)=\begin{cases} D_X(z) & z\in X \\ D_Y(z) & z\in Y \end{cases} \tag{14}$$

where z could be real images or synthetic translated images. For instance, given x drawn from domain X, it will be fed into D_X for feature extraction, while the corresponding feature extractor for its synthetic counterpart $G_Y(x)$ would be D_Y since it essentially belongs to domain Y after translation regardless of equality.

Shared Feature Extractors We also experiment with pretrained shared feature extractors similar to other works [1]. This feature extractor is trained together with a decoder M using image reconstruction loss in an unsupervised fashion across domains. The image reconstruction loss $\mathcal{L}_{rec}$ is defined as:

$$\mathcal{L}_{rec}(F,M)=\mathbb{E}_{(x,y)\sim p_{data}(x,y)}\Big[\|M(F(x))-x\|_1+\|M(F(y))-y\|_1\Big] \tag{15}$$

The shared feature extractor F will then be utilized to compute the style transfer loss and content preserving loss during the training of generators and discriminators with fixed weights.

3.1.4 Implementation Details

Network Architecture The generators are an encoder/decoder UNet-based architecture with skip connections, enabling us to represent both deep abstract features as well as local information. In encoder, we apply 6 consecutive downsampling blocks to the input images, each block contains Leaky ReLU activation, convolutional layer and Batch Normalization [23] [10]. As for the decoder, it shares a symmetrical structure with the decoder, the only difference is that we replace the convolutional layer to transpose convolutional layer for upsampling.
For the discriminators, it uses 4 convolutional blocks, and each block contains Leaky ReLU activation, a convolutional layer, and Instance Normalization [19], which aim to classify whether overlapping image patches are real or fake.
As for the pretrained shared feature extractor, it shares a similar architecture to that of the generator. The only modification we made is to replace the encoder with ResNet-18 [8].

Training Strategy Our models are implemented in PyTorch [14] using Adam optimizer [11] with a batch size of 6 and an input/output resolution of 512×512 unless otherwise specified. We use a learning rate of 0.0002 at the first 20 epochs and linearly decay the rate to zero for the last 5 epochs. We apply random horizontal flipping and vertical flipping with a probability of 0.5, respectively. For the factors λ_{cyc}, λ_{style} and λ_{cotent} that control the relative importance of each loss term, we set it to 30, 0.1 and 0.1, respectively. We also follow the strategy as previous work by applying a history pool of generated images for discriminators and replacing the negative log-likelihood objective with a least-squares loss to stabilize the model training procedure.
As for the shared feature extractor, we train it for 20 epochs with a batch size of 4 and an input/output resolution of 512×512 as well. We use Adam optimizer starting with a learning rate of 0.0001 for the first 15 epochs, which is then multiplied with 0.1 for the remainder.

3.2 FCAN Methods

Another possibility to tackle our problem is to use a method similar to Style Transfer with some extensions. Style transfer [3] typically refers to an optimization technique, which takes an image and extracts its style, and merges with the content of another image. A state-of-the-art network named Fully Convolutional Adaptation Network builds upon this idea and extends it with adversarial learning to directly perform semantic segmentation in the target domain. We use the Fully Convolutional Adaptation Networks (FCAN) as our baseline, which we intend to improve on. The FCAN network contains an Appearance Adaptation Network (AAN) and a Representation Adaptation Network (RAN), where the AAN focuses on the domain adaptation task and the RAN focuses on the semantic segmentation task. The core idea of FCAN is to overcome the domain shift coming with the domain adaptation under the appearance- and representation viewpoints [26]. The major challenge of using FCAN as our baseline is the implementation: since there is no code base and the description of the architecture is vague, we are not able to exactly reproduce this project. We had to implement the network ourselves and searching for

other possibilities to make the network working. Our proposed network is illustrated in the Fig. 2.

3.2.1 Appearance Adaptation Networks (AAN)

The Appearance Adaptation Network (AAN) intends to make the images from the source and target domain appear visually similar. The major idea is to use a white noise image and iteratively optimize the image to have a similar appearance with the target image and similar content with the source image. We replace the white noise image with a random image from the target domain and interpret the AAN as an image-to-image translation. The reason for using the source image is after experimenting we found it has less noise than starting from the white noise image and it trivially has the same content (and style) as the source domain, so our AAN could focus on changing the style. We employ the two losses from the original FCAN to measure the similarity between the target and the source image with the input image: the content loss and the style loss. We use a pre-trained ResNet50 [8] as our backbone for feature extraction and register the response maps at certain convolutional layer l as $M^l \in R^{N_l \times H_l \times W_l}$. The content loss is defined as the weighted per-pixel mean squared loss between input and source feature maps:

$$w_s^l Dist(M_i^l, M_s^l) \tag{16}$$

The style loss is defined as the weighted per-pixel mean squared loss between the gram matrix of the input and the target image feature maps. During the translation, we randomly choose a subset of images from the target domain $X_t = \{x_t^i | i = 1, ..., n\}$ and calculate the average of their feature maps to calculate the style loss:

$$w_t^l Dist(G_i^l, G_t^l) \tag{17}$$

We weigh the style loss with a constant α to make our AAN focus more on adapting the style. Our AAN translates all the images from the source domain $X_s = \{x_s^i | i = 1, ..., m\}$ one-by-one to the target domain. The overall objective L_{AAN} defined as

$$L_{AAN}(x_i) = \sum_{l \in L} w_s^l Dist(M_i^l, M_s^l) + \alpha \sum_{l \in L} w_t^l Dist(G_i^l, G_t^l) \tag{18}$$

with i denote the input image, t an image from the target domain and s am image from the source domain.

3.2.2 Representation Adaptation Networks (RAN)

As mentioned before, while CycleGAN only focuses on domain adaptation, FCAN combines the tasks of domain adaptation and semantic segmentation together. The Representation Adaptation Networks (RAN) contains two branches: one adversarial branch and one segmentation branch. The adversarial branch contains a discriminator and a feature extractor. The discriminator consists of four Atrous Spatial Pyramid Pooling layers and attempts to differentiate between source and target feature representations. The feature encoder is based on a pre-trained ResNet101 architecture and the network utilizes this network to extract feature representations from the source and target domain and subsequently performs feature-level image segmentation.

We abandon this proposed feature extractor and replaced it with a U-Net encoder. The original FCAN uses a ResNet101 network to generate image representation and then uses bilinear interpolation to upsample the feature maps to the original size to perform pixel-level classification. We believe by incorporating our pre-trained, more sophisticated U-Net we can produce better segmentation results. Our idea is to use the down-sampling part of our U-Net as a feature extractor, this U-Net encoder with shared weights is trained in both the segmentation branch and the adversarial branch. We train our discriminator and our U-Net encoder to make our original U-Net learn to extract domain-invariant representations, while using the segmentation branch as supervision to fine-tune our entire U-Net (including the U-Net encoder) to keep our segmentation loss at a low level. We use the same objective function for RAN from the original FCAN to our modified RAN:

$$\max_{F} \min_{D} \{L_{adv}(X_s, X_t) - \lambda L_{seg}(X_s)\} \tag{19}$$

The RAN calculates the adversarial loss L_{adv} as the average classification loss over all spatial units and the segmentation loss L_{adv} is the same as our previous medical image segmentation section.

$$L_{adv}(X_s, X_t) = -E_{x_t \sim X_t}[\frac{1}{Z}\sum_{i=1}^{Z} log(D_i(F(x_t))] - E_{x_s \sim X_s}[\frac{1}{Z}\sum_{i=1}^{Z} log(1 - D_i(F(x_s))] \tag{20}$$

We noticed that this could be seen as a standard Generative Adversarial Network (GAN) [5], with the target domain feature as the true data and the source domain feature as generated data.
However, after testing the RAN using the proposed standard GAN model we faced the vanishing gradient problem: the model stops to improve after 15k iterations. This problem occurs when the trained discriminator becomes optimal and thus does not provide any information for the generator to further improve. This is not the only problem that occurred during training: we also noticed that by using the standard GAN the segmentation performance does not improve compared to the initial model. When evaluating the segmentation results during training, we noticed a strong oscillation in the performance: the model does occasionally produce better segmentation results, but we conclude that there is not enough performance gain that we can conclude the RAN is improving our model's segmentation capability in the target domain.
Instead, we use the Wasserstein GAN [7] to counter this problem. The WGAN-GP uses the Wasserstein loss with gradient penalty to achieve Lipschitz continuity and performs better than the original WGAN with instead simple weight clipping. The overall objective function is illustrated as follow:

$$L_{WGAN\_GP} = E_{\tilde{x} \sim P_g}[D(\tilde{x})] - E_{x \sim P_r}[D(x)] + \lambda E_{\hat{x} \sim P_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2] \tag{21}$$

3.2.3 Implementation Details

The detailed illustration of our extended Fully Convolutional Adaptation Network is illustrated in Fig. 2. Our network is implemented with the Pytorch framework. We use an Nvidia RTX 2070s GPU for training. We use the same parameters for pre-training our U-Net on the source domain and the AAN-transformed source domain as mentioned in

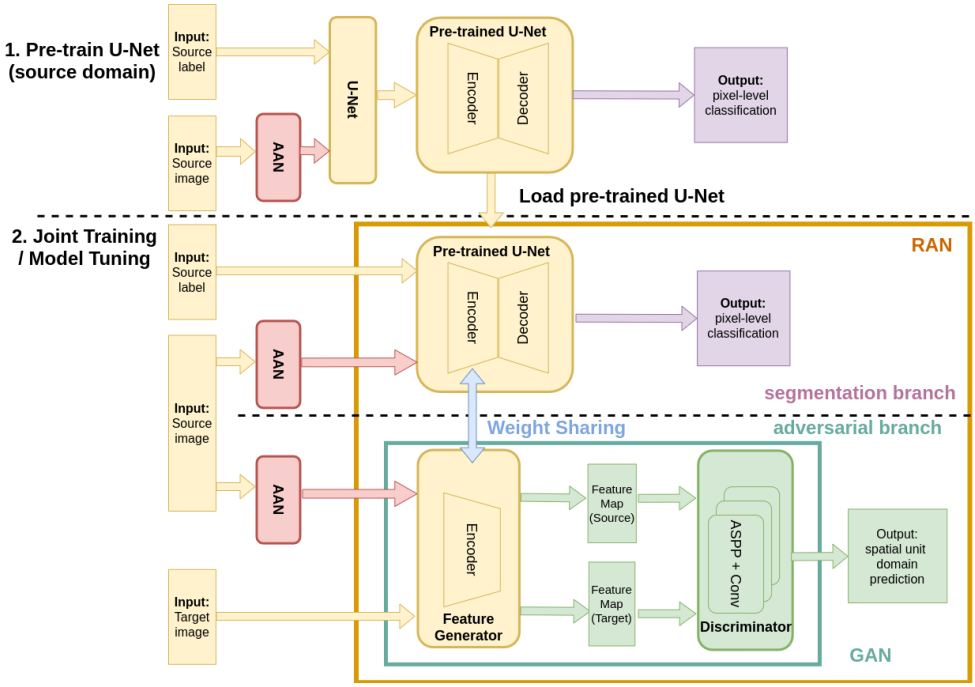

Figure 2: Our proposed model based on the original FCAN. In the first stage, we pre-train our U-Net on the source domain images. In the second stage, we use our per-trained U-Net and jointly train our segmentation branch and adversarial branch.

the Medical Image Segmentation. We use the pre-trained ResNet-50 as our backbone for feature extraction. For the AAN we use the activation maps of the five convolutional layers $L = \{conv1, res2c, res3d, res4f, res5c\}$ as mentioned in the original paper. However, the proposed hyperparameters from the original paper did not work well for our data sets, instead, we found the following set of parameters by hyperparameter-tuning: the weight α which balances the content loss and the style loss, the initial learning rate β, the maximum number of iteration $I = 500$, the 10 layer weights w_s^l and w_t^l. Our weights using ResNet50 are: $\alpha = 12358755.73$, $w_s^l(conv1) = 1$, $w_s^l(res2c) = 3.7$, $w_s^l(res3d) = 13.97$, $w_s^l(res4f) = 43.15$, $w_s^l(res5c) = 91.18$, $w_s^l(conv1) = 1$, $w_s^l(res2c) = 2.85$, $w_s^l(res3d) = 8.16$, $w_s^l(res4f) = 46.44$, $w_s^l(res5c) = 74.68$. Our weights using VGG19 are: $\alpha = 30000000$, $w_s^l(conv1) = 0$, $w_s^l(res2c) = 0$, $w_s^l(res3d) = 0$, $w_s^l(res4f) = 1$, $w_s^l(res5c) = 1$, $w_s^l(conv1) = 1$, $w_s^l(res2c) = 1$, $w_s^l(res3d) = 1$, $w_s^l(res4f) = 1$, $w_s^l(res5c) = 1$.
For the RAN we use our pre-trained U-Net and jointly fine-tune the RAN with segmentation loss and adversarial loss. We use the training set of CIRRUS and SPECTRALIS for training RAN and use a batch size of 2 because of the limited memory of our local machine. The maximum number of iteration is set to be 20k, because we only want to fine-tune our U-Net with our RAN pipeline and we also notice the vanishing gradient problem after 20k.

3.3 Original FCAN vs. proposed FCAN

The original Fully Convolutional Adaptation Network (FCAN) examined the usage of the following architecture: ABN(adaptive batch normalization), ADA(adversarial domain adaptation), Conv + ASPP (discriminator with an extended image region for classification). We have successfully implemented all the mentioned architectures to achieve the best performance. We extended the ASPP with Xavier weight initialization and examined the usage of WGAN+GP to enhance the performance of adversarial learning.
For evaluating the impact of AAN on the FCAN, we only examined the two cases: we use original samples from the source and the target domain images and secondly, we use AAN-processed source domain samples and original samples. The reason for that is in the original FCAN paper the author claims the first case has the worst and the second case has the best performance.

3.4 Evaluation Metrics

We evaluate the quality of domain adaptation from two directions: appearance similarity and segmentation performance.

3.4.1 Style Metric

The Effectiveness(E) statistics[25] is a quantitative evaluation metric that measures the extent to which a given style has been transferred to the target. The higher the E statistics, the adapted image has a closer appearance to the style image. The key idea of the E statistics is to calculate the divergence between convolutional feature layer distributions of the style image and the transferred image. The E statistics can be formulated as follow:

$$E_i = -\log(D_{KL_i}(\mathcal{N}_0||\mathcal{N}_1)) \tag{22}$$

where $D_{KL_i}(\mathcal{N}_0||\mathcal{N}_1)$ is the KL divergence of i'th layers between the t-dimensional Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ of the transferred image I_0 and the style image I_1. The KL distance can be expressed as follow:

$$D_{KL}(\mathcal{N}_0||\mathcal{N}_1) = \frac{1}{2}(tr(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^T\Sigma_1^{-1}(\mu_1 - \mu_0) - t + \ln(\frac{\det\Sigma_1}{\det\Sigma_0})) \tag{23}$$

To calculate the parameters μ and Σ, we project the feature map of both style images and transferred the image's summary statistics to a t-dimensional representation. The statistics of the style images are the average summary statistics over a randomly selected volume from the Spectralis dataset. We randomly select one volume from the Cirrus dataset to calculate the fixed projection basis. We use the pretrained VGG model to obtain the convolutional feature map of layer Relu1_1, Relu2_1, Relu3_1, Relu4_1, and Relu5_1. For each layer, we use dimensions 32, 48, 128, 256, 256 respectively. We investigated the E statistics through five convolutional layers and the average of 5 layers and the first 3 layers. Since the lower layer of the convolutional features captures more information of the image style[4], we use the average E statistic of the first three layers to be the main quantitative metric to evaluate the style transfer.

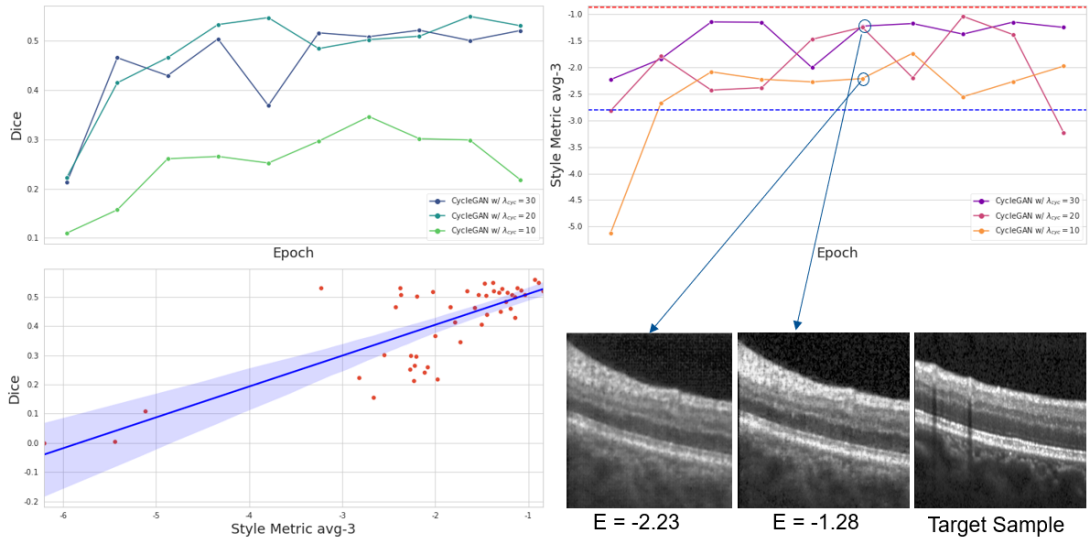

Figure 3: Correlation results between $avg-3$ from style metric and Dice

3.4.2 Dice Metric

We use Dice Score to evaluate the segmentation performance on three fluid types before and after the image adaptation. The idea of evaluating domain adaptation through the segmentation tasks is that while the domain adaptation model successfully adapts the appearance of an image to the style image, the segmentation model trained on style images can better segment the adapted image and performed a higher score on the Dice metric.

3.4.3 Correlation between Dice & Style Metric

We also utilize the CycleGAN model to further study the correlation between Dice and style metric. As shown in upper 2 curves of Fig. 3, the performance of segmentation and dice all indicate a rising trend. We further carried out the curve fitting between $avg-3$ from style metric and Dice using tested results from different model variants of each epoch to study the correlation between two metrics. The result illustrate a positive correlation between them, which reveals the fact that the better the adaptation, the higher the both performance. Therefore, we opt to utilize this two metrics to study the adaptation performance of different methods. It is worth noting that style metric could reflect human precepts on image quality as well. As shown in Figure 3, example with higher style metric also indicates better appearance similarity to the target domain, which is in line with our intuitive visual judgement.

3.5 Domain Adaptation Peformance

3.5.1 Style Metric Performance

We first compare all model variants for domain adaptation with respect to style consistency. As can be shown from Fig. 4. All variants are able to provide synthetic results that share high visual consistency with the target domain. To further investigate the style-

Table 2: Performance on Style Consistency

Model	E1	E2	E3	E4	E5	avg-5	avg-3
Upper Bound	-0.134	-0.539	-1.921	-3.793	-4.710	-2.219	-0.865
Cycle-GAN	**-0.315**	**-0.494**	**-1.993**	**-3.919**	**-4.798**	**-2.304**	**-0.934**
AAN	-1.809	-2.200	-3.128	-4.511	-5.028	-3.335	-2.379
Lower Bound	-2.416	-2.296	-3.981	-4.965	-5.187	-3.769	-2.897

consistency beyond human precepts, we evaluate the results of style metric proposed in Section 4.1 as well.
As can be seen from Table 2, CycleGAN baseline obtain a higher value in style metric compared to AAN baseline which is consistent with the qualitative results shown in Fig.4. The lower bound is evaluate on the same test set of the Cirrus samples before adapted, and the upper bound is evaluated on the test samples of the Spectralis. Compare to the lower bound, both CycleGAN and AAN baselines can successfully transfer the source images into images with target style appearance.

3.5.2 Semantic Segmentation Performance

Table 3: Performance on Semantic Segmentation Task

Model	Mean-Dice	IRF	SRF	PED
Cycle-GAN	**0.561**	**0.578**	**0.720**	0.386
AAN	0.527	0.512	0.719	0.349
FCAN	0.521	0.476	0.688	**0.399**

To further verify the effectiveness of our proposed adaptation models, we carried out semantic segmentation experiments for the adapted results using the pretrained segmentation from Chapter 2. There are several sources of the B-scans tested with corresponding pretrained models: (1) Synthetic images from Cirrus to Spectralis using CycleGAN methods tested with pretrained models on Spectralis dataset; (2) Synthetic images from Cirrus to Spectralis using AAN methods tested with U-net model pretrained on Spectralis dataset; (3) FCAN with the adversarial branch trained directly on Cirrus samples.
As can be seen from Table 3, CycleGAN baseline outperforms the other two methods on the segmentation tasks. Fig. 5 shows the qualitative results of the segmentation tasks. The segmentation results of CycleGAN baseline acquires the majority of ground truth labels. It is worth noting that AAN requires only a small amount of style images and the adaptation process is relatively easy compared to CycleGAN. Besides, the FCAN method needs only a few hours of computation. Take the amount of data and the computational effort into account, the results of AAN and FCAN are also acceptable.

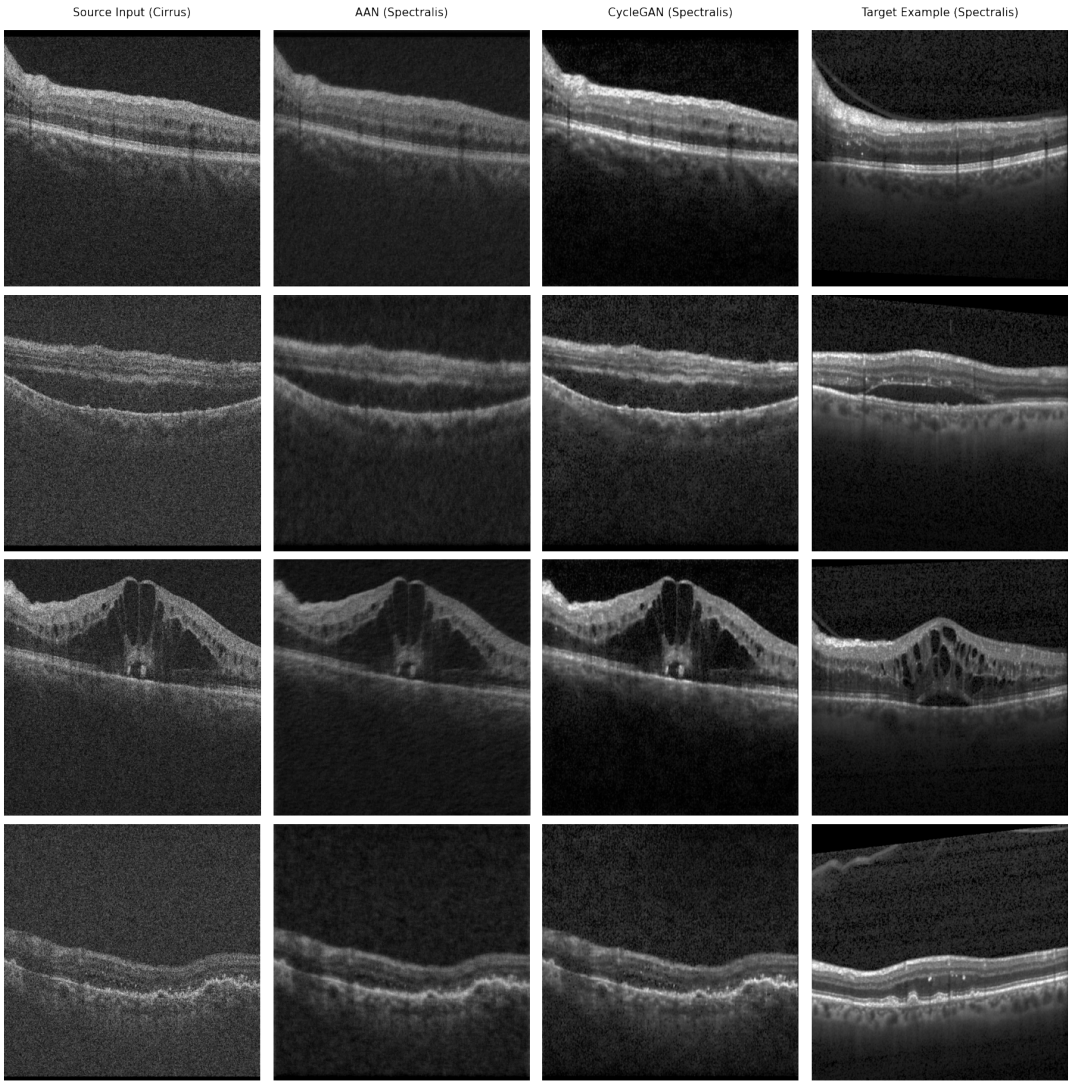

Figure 4: Qualitative results on style adaptation performance

3.6 Ablative Study

3.6.1 Ablative Study on CycleGAN

We further carried out several experiments to study the effect of proposed loss and different feature extractors.

Effect of Cycle-Consistency Weight. We first carried out several experiments to empirically find the best selection for the value of λ_{cyc}. As can be shown from 6, it can be seen that when setting $\lambda_{cyc} = 10$, the model suffers a significant drop in both style performance and segmentation performance. While for the other two model variants with λ_{cyc} equalling 20 or 30, it shows a very marginal improvements with respect to both performance. As a result, we empirically set $\lambda_{cyc} = 30$ and run all the other experiment under such hyperparameter setting.

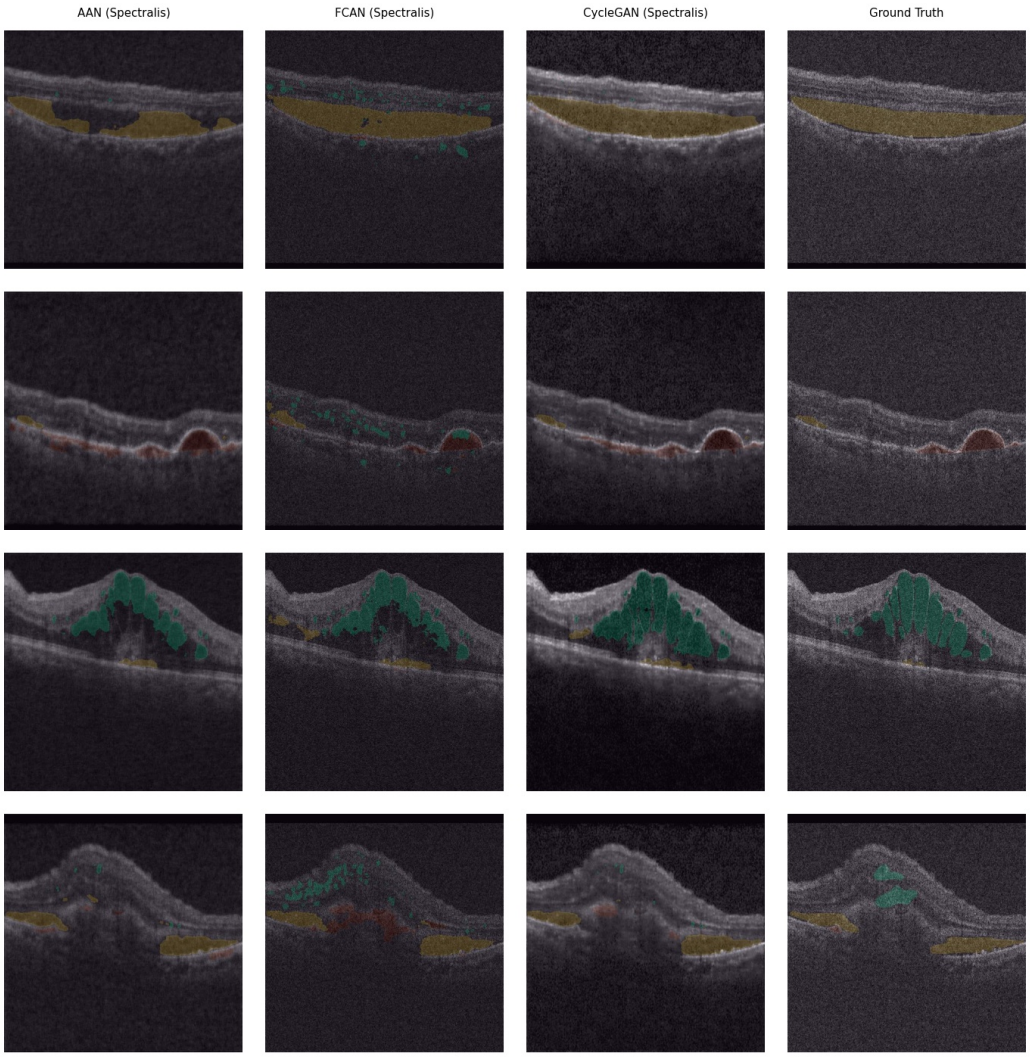

Figure 5: Qualitative results on semantic segmentation performance

Effect of Loss Formulation. We also studied the individual effect of style transfer loss and content preserving loss. As reflected from the results, content preserving loss itself has a very marginal contribution to the performance and style preserving loss even degrades the performance. We explain it is because style transfer loss could help to generate higher quality results with realistic appearance to the target domain, while leads to loss of semantic information, which causes a drop in performance. As a result, content preserving loss acts as a regularizer for style transfer loss to further ensure the contents shall be consistent with original input after translation. Therefore, we can observe a significant improvement on performance of the full model with style transfer loss and content preserving loss together compared with the baseline.

Effect of Feature Extractor. We then carried out experiment on different variants of feature extractors. We can at first conclude that introducing style transfer loss and content preserving loss regardless choice of feature extractors boost the performance to a

Table 4: Performance on different model variants of Cycle-GAN

Loss Setting	λ_{cyc}	Feature Extractors	Dice-Mean	Style-avg-3
CycleGAN	10	-	0.312	-1.978
CycleGAN	20	-	0.476	-1.386
CycleGAN	30	-	0.487	-1.349
CycleGAN + $\mathcal{L}_{style}$	30	Discriminators	0.458	**-0.846**
CycleGAN + $\mathcal{L}_{cotent}$	30	Discriminators	0.505	-1.372
CycleGAN + $\mathcal{L}_{cotent}$ + $\mathcal{L}_{style}$	30	Discriminators	**0.561**	-0.934
CycleGAN + $\mathcal{L}_{cotent}$ + $\mathcal{L}_{style}$	30	Shared	0.511	-1.322

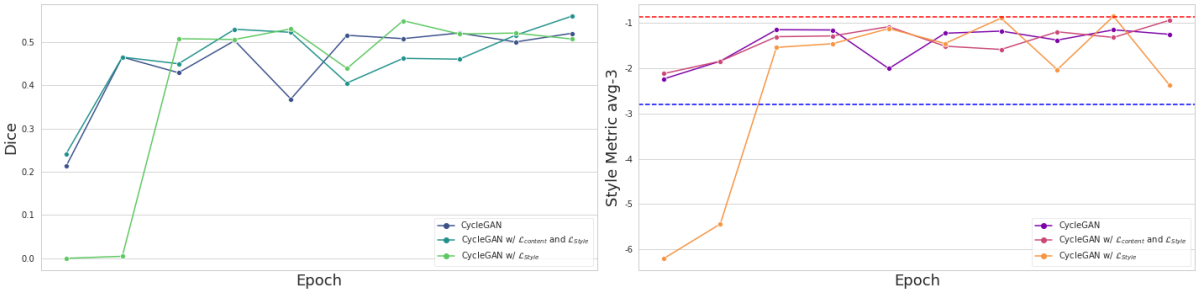

Figure 6: Performance of selected CycleGAN model variants w.r.t each epoch

certain extent. Moreover, we found that the discriminator-based feature extractor yields better performance compared with shared feature extractor. We assume the reason lies in that the discriminator-based feature extractors are essentially trained simultaneously with generators and discriminators compared with pretrained shared feature extractor, which potentially guides the discriminator-based feature extractors to learn better feature representation for style and content measurement. It's also worth noting that training with discriminator-based feature extractors is much father than that with shared feature extractor since it requires no pretraining procedure and less computational effort because the extracted features will be simultaneously stored during the forward pass of discriminators.

3.6.2 Ablative Study on AAN

The major advantage of using an Appearance Adaptation Network (AAN) for domain adaptation is because of its light-weightiness: one can easily use the same network and apply it to another data set without any per-training. Because of its relative simple network architecture, we are aware of its lack of performance compared to other networks like CycleGAN. We therefore think, it is of great importance to focus on the time factor of our AAN: the AAN should generate good results in a short period.
We compare the impact of using a different feature extraction backbone (ResNet50 vs VGG19) and between using a different batch size of the target images. The reason we compared it with VGG19 is because the original Neural Style Transfer paper uses VGG19 as backbone [3] and utilizes the first five convolutional layers to compute the style and

content loss. Features extracted from deeper layers tend to have more style and semantic information. ResNet has the ResNet blocks and skip connections architecture designs, which allows us to train deeper network. We want compare the performance difference using these two different architectural designs. The top 2 segmentation results of each backbone are shown in Table 5.

Table 5: Dice score comparison with different batch size of target images and feature extraction backbone (IRF/SRF/PED)

Epoch	ResNet50 - 3	ResNet50 - 5	VGG19 - 1	VGG19 - 5
100	0.289	0.317	0.318	0.373
	0.310/0.272/0.284	0.262/0.375/0.313	0.336/0.341/0.276	0.363/0.365/0.393
200	0.454	0.463	0.348	0.409
	0.377/0.609/0.377	0.338/0.629/0.421	0.371/0.386/0.287	0.378/0.429/0.418
300	0.499	0.490	0.358	**0.417**
	0.428/0.710/0.358	0.372/0.684/0.413	0.388/0.399/0.286	0.383/0.443/0.426
400	0.518	0.500	0.362	0.411
	0.479/0.722/0.353	0.406/0.687/0.406	0.399/0.404/0.285	0.378/0.431/0.423
500	**0.527**	**0.502**	**0.363**	0.408
	0.512/0.719/0.349	0.428/0.670/0.407	0.403/0.406/0.281	0.375/0.423/0.425

To summarize, when comparing the best results (independent of the number of iterations) of each architectural design in Table 6 w.r.t time of each epoch, we found out that overall the ResNet backbone outperforms the VGG backbone in the segmentation tasks. It is worth noting that for the ResNet backbone, images adapted from the model with batch size = 1 have the most similar style with the target image, however, the model with batch size = 3 perform the best in semantic segmentation, even better than the model with batch size = 5. Fig. 7 shows that for the dice metric, the ResNet backbone has great progress on segmentation when epochs increase, while the curve of the VGG backbone remains flat.

Table 6: Performance comparison with different model parameters and feature extraction backbone

Parameters	Backbone	Style-avg-3	Dice-Mean	IRF	SRF	PED	Time(s)
500 (5)	ResNet50	-2.760	0.502	0.428	0.670	0.407	0.25
500 (3)	ResNet50	-2.379	**0.527**	**0.512**	**0.719**	0.349	0.2
500 (1)	ResNet50	**-1.471**	0.425	0.463	0.589	0.222	0.1
500 (5)	VGG	-1.627	0.408	0.375	0.423	**0.425**	0.22
500 (1)	VGG	-2.137	0.363	0.403	0.406	0.281	0.09

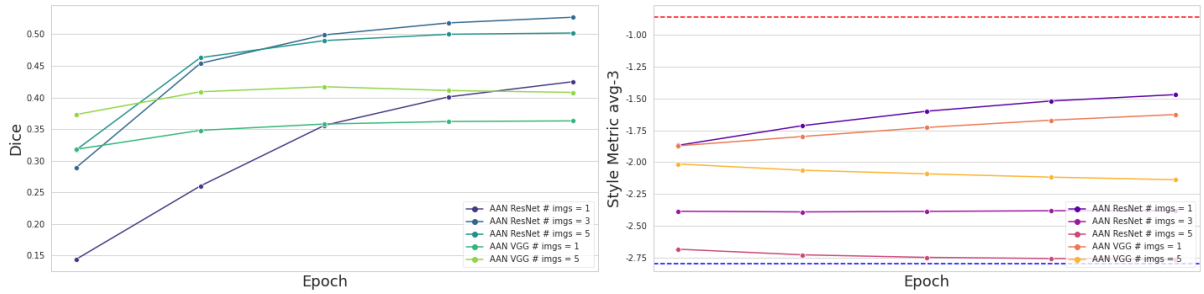

Figure 7: Performance of AAN with with different model parameters and feature extraction backbone w.r.t epoch

3.6.3 Ablative Study on FCAN

We now examine how the different design influences the overall performance of FCAN. For the baseline we directly use the U-Net pre-trained on the Spectralis data set and test it directly with samples from the Cirrus data set. We differentiate between two settings: when training our adversarial branch we could either use feature representations from Cirrus or AAN-processed Cirrus samples (Cirrus samples which appear like Spectralis). By incorporating AAN transferred samples, we enforce our U-Net to also learn to segment samples with the target style.

Table 7: Imoprovment achieved using FCAN

Design Choise	RAN Data	Mean	IRF	SRF	PED
baseline	Spectralis - Cirrus	0.002	0.006	0.000	0.000
FCAN (GAN)	Spectralis - Cirrus	0.521	0.476	0.688	0.399
FCAN (WGAN)	Spectralis - Cirrus	0.448	0.428	0.647	0.271
FCAN (GAN)	Spectralis (AAN) - Cirrus	0.500	0.457	0.654	0.389
FCAN (WGAN)	Spectralis (AAN) - Cirrus	0.422	0.589	0.544	0.133

We tested four different scenarios to evaluate the different design approaches of FCAN: using the original images from Spectralis or using AAN translated Spectralis images. We also evaluate the effect of using normal GAN and Wasserstein GAN for adversarial training. We can see that against our previous assumption, the usage of simple GAN for adversarial training and the usage of original images produce the best results. Since we are trying to evaluate the results on the plain Cirrus data set, it might be beneficial to let our network to learn to extract Cirrus-Spectralis invariant features. The usage of WGAN with gradient clipping did not achieve the expected results: in the initial testing it did demonstrate superior performance than normal GAN, we have to further analyze the reason for the performance drop.

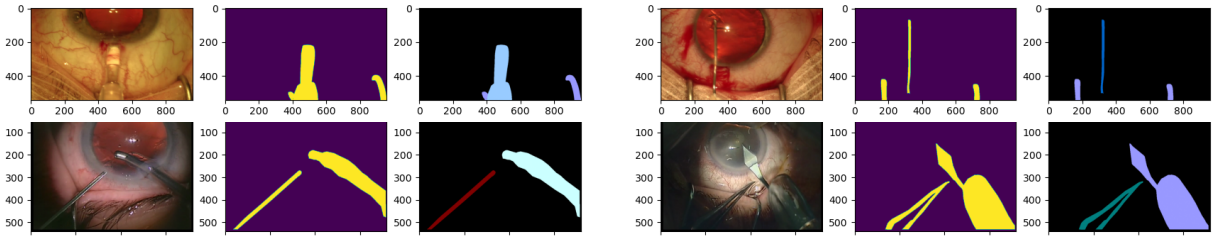

Figure 8: Upper row: two samples from CaDIS with the original image, overall segmentation mask, per-category segmentation mask. Buttom row: two samples from Cataract101 with the original image, overall segmentation mask, per-category segmentation mask. Each color defines a unique category of surgical tool.

4 Extras

4.1 Cadis - Cataract Data Set

The initial objective of the project is to evaluate the domain adaptation methods on two medical imaging use cases: the Retouch data set (CIRRUS/SPECTRALIS) and the Cataract data set (Cataract101/CaDIS) [17][6]. The segmentation task is to perform pixel-wise segmentation of different categories of surgical tools in both domains. In the Fig. 8, we provide some visualizations of the Cataract101 and CaDIS data set using our custom dataloader, to give the reader a feeling about the task setting.
We managed to prove the efficacy of our models on the Retouch data set, but because one of our team member dropped the project, we did not have enough time to also evaluate our models on the other data set. Nevertheless, because of the light-weighted nature of our defined Appearance Adaptation Network, we still managed to provide some qualitative results of domain adaptation. In the following we depict three batches of images: images from the the original CaDIS data set, images from the original Cataract101 data set and AAN-processed images from the CaDIS data set. The reason we choose the the CaDIS as our source domain is because it contains in total 4740 (3584/540/616) samples, while the Cataract101 data set only contains 385 (237/61/87) samples. Finding a method to translate images from the CaDIS to the Cataract101 data set would provide the better gain. In the Fig. 9 we provide some qualitative results of our AAN-processed images.
We can clearly observe a change of style: the AAN-processed images appear to be more dark and change from warm tone to cold tone. The AAN-processed images still maintain details like blood vessels and one can still spot the medical apparatus. It is worth mentioning, we did perform any hyperparameter-tuning, instead we just use the same set of hyper-parameters for the Retouch data set. It is justified to believe with some systematic hyper-parameter tuning the CaDis images will have a stronger resemblance with the Cataract101 images.

4.2 Region Proposal Network

During the implementation of our segmentation baseline we encountered some difficulties to get good segmentation: our segmentation results (IoU and DICE score) did not reach a satisfying level. So we explored different methods to further boost the performance of

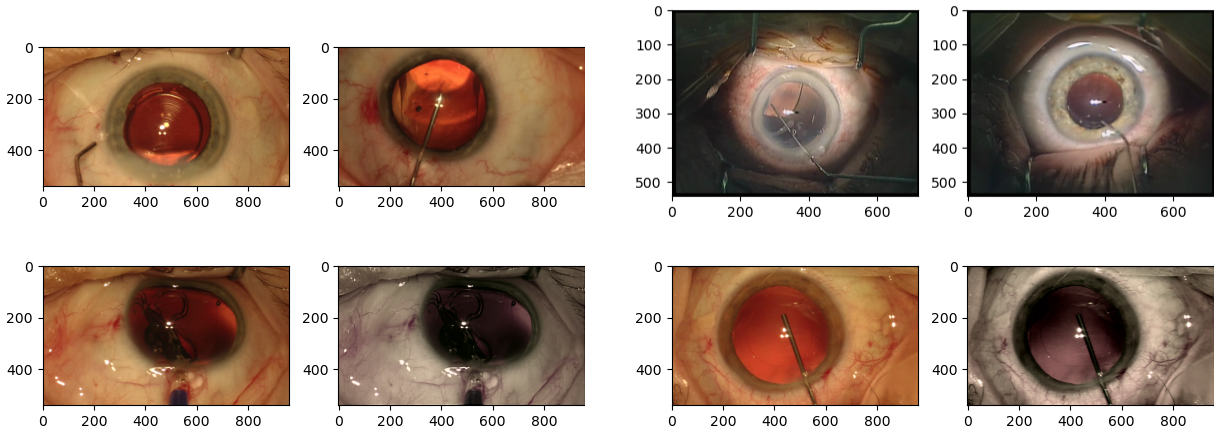

Figure 9: CaDIS and Cataract101 samples and qualitative results when performing domain adaptation using AAN. Upper row: original samples form the CaDIS (left) and Cataract101 (right) data set. Buttom row: two samples of image translation from CaDIS to Cataract101.

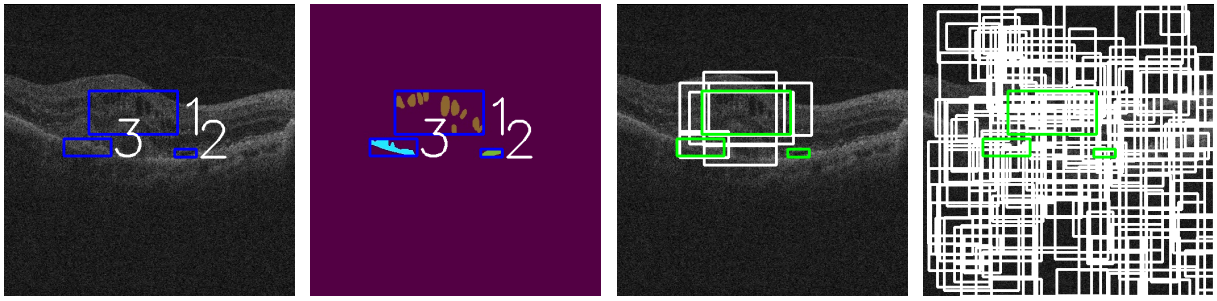

Figure 10: Left 1: one original image from Cirrus with the bounding boxes. Left 2: the segmentation masks with the bounding boxes per fluid category. Right 3-4: positive and negative examples of the generated bounding box proposals (white). Positive bounding boxes have large IoU values with the ground truth bounding boxes (green).

our model. One of the most common state-of-the-art network for instance segmentation is Mask R-CNN [9]. One of the most important reason for its good performance is the usage of a Region Proposal Network (RPN). Through literature research, we also found some other projects incorporate a Region Proposal Network to perform instance segmentation in medical domain. Xu et al. performs whole heart segmentation of cardiac CT volumes by combining a 3D U-Net with Faster R-CNN (predecessor of Mask R-CNN) [24].
The main idea of RPN is to guide the network to focus on some important regions with a higher probability of containing useful information for instance segmentation. We successfully implemented an own RPN network, the major process can be summarized with the following steps: we first use the VGG19 as backbone to extract image features. Then we uniformly sample a fixed number of anchor points on the extracted, down-sized feature map. We use these anchor points as center points and put 9 varying-sized bounding boxes around it. Finally, by employing two shallow networks the network the network jointly outputs box regression and box classification [9]. In Fig. 10 we visualize some intermediate results.

We did not further follow this path, because after fixing the U-Net baseline, we got acceptable results and our major focus was on the domain adaptation task. But because of the simplicity and generalizability of our implemented Region Proposal Network, we could consider incorporate this network into our future work to further improve the segmentation results.

5 Conclusions

We have presented CycleGAN and FCAN methods that explore domain adaptation for semantic segmentation on medical images. Particularly, we study the problem from the viewpoint of both visual appearance-level and segmentation-level evaluation. In summary, CycleGAN performs the best on both style consistency and semantic segmentation compare to AAN and FCAN. Additionally, adding both content and style loss can further improve the adaptation results of CycleGAN. However, the AAN model and the FCAN model have their own advantages. The AAN requires a minimum amount of target domain data for style transfer, which is quite beneficial for difficult to obtain style images. The FCAN requires only a few hours for training, which can save computational effort compare to CycleGAN. Both AAN and FCAN can obtain a reliable segmentation performance. One possible direction of our future work is to conduct end-to-end training that simultaneously trains the domain adaptation network and the segmentation network. In such a case, it will be easier to evaluate the adaptation performance, and it is possible to obtain better results on semantic segmentation.

References

[1] Karim Armanious et al. "Unsupervised Medical Image Translation Using Cycle-MedGAN". In: *27th European Signal Processing Conference, EUSIPCO 2019, A Coruña, Spain, September 2-6, 2019*. IEEE, 2019, pp. 1–5. DOI: 10.23919/EUSIPCO.2019.8902799. URL: https://doi.org/10.23919/EUSIPCO.2019.8902799.

[2] Hrvoje BogunoviÄ‡ et al. "RETOUCH: The Retinal OCT Fluid Detection and Segmentation Benchmark and Challenge". In: *IEEE Transactions on Medical Imaging* 38.8 (2019), pp. 1858–1874. DOI: 10.1109/TMI.2019.2901398.

[3] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. "A neural algorithm of artistic style". In: *arXiv preprint arXiv:1508.06576* (2015).

[4] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. "Image Style Transfer Using Convolutional Neural Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2414–2423. DOI: 10.1109/CVPR.2016.265.

[5] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[6] Maria Grammatikopoulou et al. "CaDIS: Cataract dataset for image segmentation". In: *arXiv preprint arXiv:1906.11586* (2019).

[7] Ishaan Gulrajani et al. "Improved training of wasserstein gans". In: *arXiv preprint arXiv:1704.00028* (2017).

[8] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[9] Kaiming He et al. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[10] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. Ed. by Francis R. Bach and David M. Blei. Vol. 37. JMLR Workshop and Conference Proceedings. JMLR.org, 2015, pp. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html.

[11] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[12] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[13] Maureen G. Maguire et al. "Five-Year Outcomes with Antiâ€"Vascular Endothelial Growth Factor Treatment of Neovascular Age-Related Macular Degeneration: The Comparison of Age-Related Macular Degeneration Treatments Trials". In: *Ophthalmology* 123.8 (2016), pp. 1751–1761. ISSN: 0161-6420. DOI: https://doi.org/10.1016/j.ophtha.2016.03.045. URL: https://www.sciencedirect.com/science/article/pii/S0161642016300926.

[14] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 8024–8035. URL: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[15] Deepak Pathak et al. "Context Encoders: Feature Learning by Inpainting". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 2536–2544. DOI: 10.1109/CVPR.2016.278. URL: https://doi.org/10.1109/CVPR.2016.278.

[16] Olaf Ronneberger. "Invited Talk: U-Net Convolutional Networks for Biomedical Image Segmentation". In: *Bildverarbeitung für die Medizin 2017 - Algorithmen - Systeme - Anwendungen. Proceedings des Workshops vom 12. bis 14. März 2017 in Heidelberg*. Ed. by Klaus Hermann Maier-Hein et al. Informatik Aktuell. Springer, 2017, p. 3. DOI: 10.1007/978-3-662-54345-0_3. URL: https://doi.org/10.1007/978-3-662-54345-0%5C_3.

[17] Klaus Schoeffmann et al. "Cataract-101: video dataset of 101 cataract surgeries". In: *Proceedings of the 9th ACM multimedia systems conference*. 2018, pp. 421–425.

[18] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1929–1958. URL: http://dl.acm.org/citation.cfm?id=2670313.

[19] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Instance normalization: The missing ingredient for fast stylization". In: *arXiv preprint arXiv:1607.08022* (2016).

[20] Vanya V. Valindria et al. "Domain Adaptation for MRI Organ Segmentation using Reverse Classification Accuracy". In: *CoRR* abs/1806.00363 (2018). URL: http://arxiv.org/abs/1806.00363.

[21] Sebastian M. Waldstein et al. "Correlation of 3-Dimensionally Quantified Intraretinal and Subretinal Fluid With Visual Acuity in Neovascular Age-Related Macular Degeneration". In: *JAMA Ophthalmology* 134.2 (Feb. 2016), pp. 182–190. ISSN: 2168-6165. DOI: 10.1001/jamaophthalmol.2015.4948. URL: https://doi.org/10.1001/jamaophthalmol.2015.4948.

[22] Sebastian M. Waldstein et al. "Morphology and Visual Acuity in Aflibercept and Ranibizumab Therapy for Neovascular Age-Related Macular Degeneration in the VIEW Trials". In: *Ophthalmology* 123.7 (2016), pp. 1521–1529. ISSN: 0161-6420. DOI: https://doi.org/10.1016/j.ophtha.2016.03.037. URL: https://www.sciencedirect.com/science/article/pii/S0161642016300835.

[23] Bing Xu et al. “Empirical Evaluation of Rectified Activations in Convolutional Network”. In: *CoRR* abs/1505.00853 (2015). arXiv: 1505.00853. URL: http://arxiv.org/abs/1505.00853.

[24] Zhanwei Xu, Ziyi Wu, and Jianjiang Feng. “CFUN: Combining faster R-CNN and U-net network for efficient whole heart segmentation”. In: *arXiv preprint arXiv:1812.04914* (2018).

[25] Mao Chuang Yeh et al. “Improving Style Transfer with Calibrated Metrics”. In: *CoRR* abs/1910.09447 (2019). URL: http://arxiv.org/abs/1910.09447.

[26] Yiheng Zhang et al. “Fully convolutional adaptation networks for semantic segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6810–6818.

[27] Jun-Yan Zhu et al. “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks”. In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 2242–2251. DOI: 10.1109/ICCV.2017.244. URL: https://doi.org/10.1109/ICCV.2017.244.