DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

Domain Adaptation for Light Microscopy Image Segmentation

Qing Sun

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

Domain Adaptation for Light Microscopy Image Segmentation

Domänenanpassung für die Segmentierung von Lichtmikroskopiebildern

Author:	Qing Sun
Supervisor:	Dr. Felix Dietrich
Submission Date:	15.11.2022

Acknowledgments

My deepest and most sincere gratitude goes to Dr. Felix Dietrich, who has been and will continue to guide my academic journey. A big thank you to my parents for their unconditional love and support as I pursue my dream, even though they have no clue what I am doing. My final thanks go to everyone who has helped me in any way. Without you my life would never be the same.

Abstract

While deep learning techniques are nowadays commonly used in the field of computer vision and have achieved outstanding performance, collecting sufficient data for training a deep neural network remains a challenge. In this work, domain adaptation techniques are utilized in order to learn from a different yet related domain, where data are more readily available. In this thesis, a framework called Synergistic Image and Feature Adaptation (SIFA) is implemented using PyTorch, and in the project it is demonstrated how to perform semantic segmentation task on light microscopy images of fungal-colonized root sections. The identified classes are arbuscules, vesicles and root cortex. This helps the analysis of arbuscular mycorrhizal fungi (AMF). The training dataset is composed of images from two domains: unlabeled real-world light microscopy images and labeled synthetic images. The synthetic images and the labels are acquired from a 3D model of a root section. A baseline model adopting the U-Net architecture trained with the synthetic images suffers from severe performance degradation when applied to the light microscopy images, while the segmentation model developed in this work is able to create better segmentations. The result is overall promising, and future developments and possible improvements are suggested in the final section.

Contents

1 Introduction

Semantic segmentation has been a popular task in the field of computer vision. It is the process of classifying each pixel in an image into a certain class. In recent years, Deep Neural Networks (DNNs) are booming, and they are becoming ubiquitous in this field. If DNNs are provided with sufficient training data, they can learn very well. In semantic segmentation tasks, various DNNs have outperformed the state of art and become the new benchmark models. However, these segmentation models cannot generalize well to new settings, such as new datasets. For example, a semantic segmentation model is trained with a set of synthetic images of a colonized root section, the model can achieve good performance in predicting synthetic images, but its performance is severely degraded when predicting real-world light microscopy images of a root section. This problem is referred to as domain shift. Many *domain adaptation* techniques focus on solving this problem.

The study of domain adaptation is of great significance. One of its contributions is the transfer of knowledge from one domain to another. This allows the model to learn from one domain and apply the knowledge to another related domain. DNNs for semantic segmentation can also benefit from the domain adaptation techniques, as the lack of training data is one of the biggest obstacles in some cases. In fields such as medicine and biology, obtaining sufficient images for training a DNN is challenging enough, not to mention the amount of time and effort required to manually annotate the images. With the help of domain adaptation techniques, it is possible for a semantic segmentation model to learn from existing data in the source domain and then make predictions for a different target domain without obtaining additional training data.

In this work, light microscopy images of plant roots are to be predicted, and fungal objects in the image should be identified. However, these images have no ground truth labels, so training a DNN for segmentation in the traditional way with a large amount of training data is not an option. A 3D model of a colonized root section is available, which can provide synthetic images of the root as well as the labels. Therefore, domain adaptation techniques can be utilized to learn from these labeled images. Figure 1.1 is an illustration of the data pipeline in this work.

The thesis first provides some background knowledge of arbuscular mycorrhizal fungi (AMF), semantic segmentation and domain adaptation (Chapter 2). Then the main work is described in detail in Chapter 3. A framework called Synergistic Image Feature Adaptation (SIFA) is implemented, which utilizes domain adaptation techniques [3], while a U-Net trained without domain adaptation in [46] serves as a baseline model for comparison (Section 3.3). The experimental results are presented with images, figures and short discussions (Section 3.4). Finally, the work is summarized, and possible future improvements are discussed (Chapter 4).

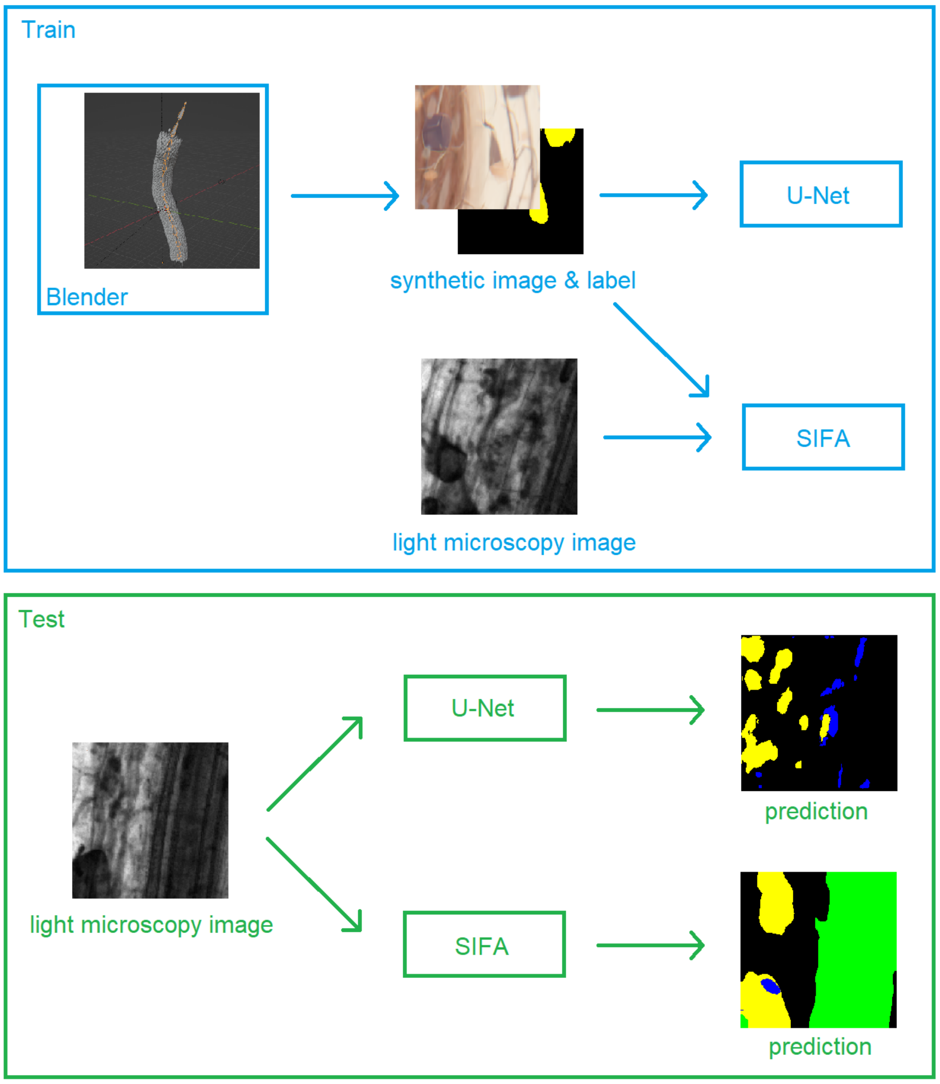

Figure 1.1: Data pipeline. Above is the training process. A 3D model created using the software Blender provides synthetic images and the corresponding labels. The baseline model (U-Net) is trained solely on the labeled synthetic images, while the model implemented in this work (SIFA) uses both the labeled synthetic images and unlabeled light microscopy images. Below is the test process. Both models perform semantic segmentation on light microscopy images. The model which utilizes domain adaptation techniques yields better results.

2 Related work

This chapter covers the basic knowledge applied in this work. A brief introduction of the biological background (Section 2.1), the computational methods (Section 2.2 and 2.3) and the preliminary work (Section 2.4) is given.

2.1 Arbuscular Mycorrhizal Fungi

Mycorrhizal symbiosis is a form of fungal symbiosis with plants. Based on the anatomy, it can be further divided into two categories: ectomycorrhizas (EMs) and endomycorrhizas, and EMs are further divided into three categories, including arbuscular mycorrhizas (AMs), which are the most common of all types [8, 2]. This association involving plants and arbuscular mycorrhizal fungi (AMF) is quite ancient and is found in most plant families [11, 1]. It promotes the absorption of nutrients from the soil, and has received increasing attention for its potential use in sustainable agriculture [33, 21, 5]. Figure 2.1 shows the root colonization structures. With the help of highly extended hyphal network of AMF, a larger area of soil is available for nutrient uptake. Through arbuscules, the plant and the fungus exchange nutrients such as minerals and carbon, and this is mutually beneficial.

The traditional method for analyzing AMF is to collect light microscopy images and manually mark the regions of interest, which is time consuming. First, AMF samples are prepared as described in [46]:

> Lotus japonicus ecotype Gifu wild-type (plant-name) seeds were scarified and surface sterilized. The imbibed seeds were germinated for 10-14 days and cultivated in quartz sand as substrate. For colonization with Rhizophagus irregularis (fungus-name) the roots were inoculated with 500 spores per plant and harvested after five weeks after inoculation.

Next, the samples are stained using the method described in [45], so that the plant structures are visible and can be imaged by a microscope, but there are also risks that the staining is not successful. After that, the plant biologists analyze the images and count the fungal structures.

2.2 Semantic segmentation

Image segmentation is the process of analyzing an image and identifying different regions and structures. Each and every pixel that constitutes the image should be labeled with a class. There are two main types of image segmentation. One is semantic segmentation. In a

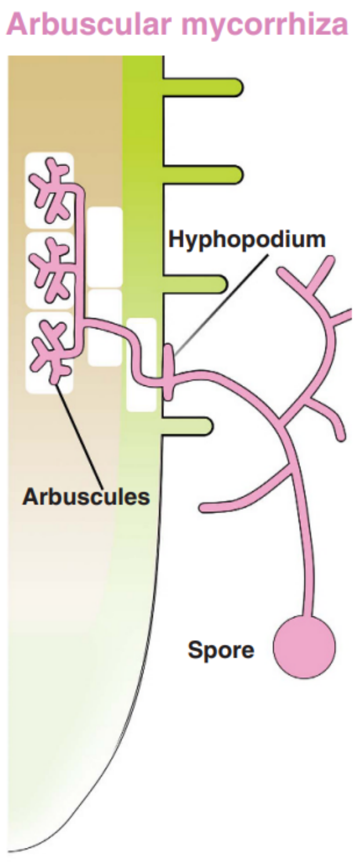

Figure 2.1: Illustration of arbuscular mycorrhizal (AM) colonization structure. In the AM colonization, the hyphal network of AMF developed from the spore in the soil spread inside the plant root, and arbuscules are formed inside the root cells. Taken from [2].

semantic segmentation task, objects of the same kind should be given the same label, and this requires recognizing and understanding the semantic meaning of pixel-level contents in the image. The other is instance segmentation. It is similar to semantic segmentation, but different instances should also be distinguished. In other words, when doing instance segmentation, a unique label is given to each instance, even if they are objects of the same class, while in semantic segmentation, they have the same label.

Semantic segmentation has long been a popular topic in the field of computer vision, because it has the potential to help in many areas such as autonomous driving and medical image diagnosis. There were attempts to solve this problem even before the booming of deep learning [17, 41] (see Figure 2.2). Modern computer vision algorithms are based on convolutional neural networks (CNNs). CNN is a multilayer neural network specifically designed to deal with images, and it outperformed other traditional techniques at that time [28, 29]. Later, deeper and more complex CNNs appeared, yielding increasingly better results. For example, in [32, 4, 37, 48, 39, 7], CNNs are trained for supervised semantic segmentation tasks (see Figure 2.3).

However, powerful deep learning comes at a cost. It typically requires huge amounts of

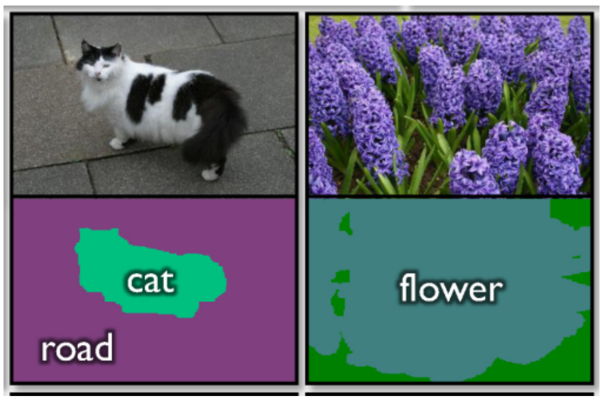

Figure 2.2: Example semantic segmentation results on the MSRC 21-class database with a non-neural network algorithm. Taken from [41].

training data. In the case of supervised semantic segmentation, pixel-wise manual annotation of images is inefficient and expensive. Therefore, people turn to unsupervised learning techniques, where ground truth labels are not involved in the training. Some works focus on unsupervised image segmentation by combining the idea of clustering with deep learning in various ways, so that image patches and pixels which share the same semantic meaning can be identified and grouped using only unlabeled data samples [22, 6, 14].

2.3 Domain adaptation

In the field of computer vision, supervised learning has made tremendous progress. Most methods require that the test data and the training data come from the same distribution. However, this is not always the case. In some applications, the testing scenarios may vary

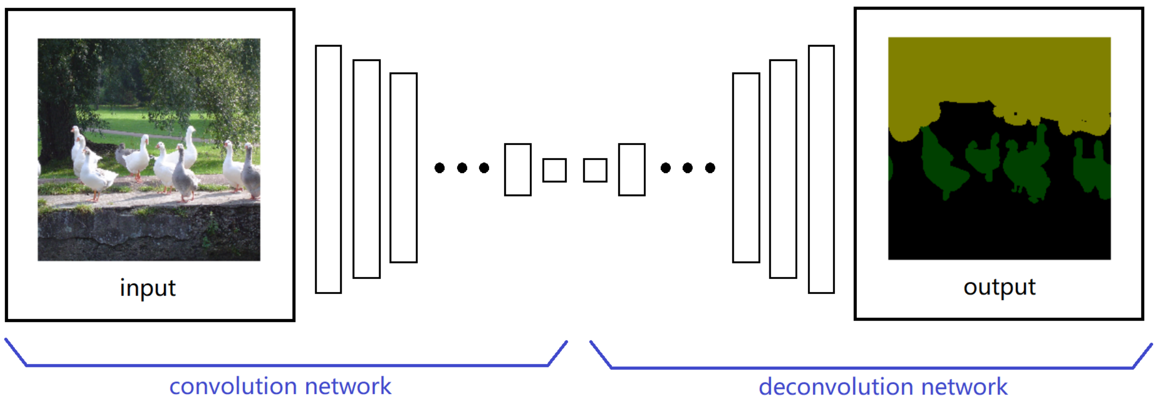

Figure 2.3: Architecture of a CNN designed for semantic segmentation. The convolution network (left) extracts features from input image. With convolution layers and max pooling layers, the size of the image shrinks and the number of filters increases. The deconvolution network (right) reconstructs the image to its original size and assigns a label to each pixel. The architecture of the deconvolution network is typically a mirrored version of the convolution network, but with transposed convolution layers and unpooling layers.

significantly. For example, a model trained for autonomous driving on a particular dataset may encounter different weather conditions and different geographic regions during test, and this can result in dramatically degraded performance. This problem is referred to as *domain shift* (see Figure 2.4). A naive idea to solve this problem is to add more training data that cover more scenes, but acquiring labeled data can be a big problem by itself. In some cases, labeled data for training are only available in one domain, called the *source domain*, while test is performed on data which come from another domain, called the *target domain*. Due to the scarcity of labeled data, various *domain adaptation* methods have been investigated to bridge the gap between training data and test data, so that knowledge gained from one domain can be transferred to another related domain where only a few or even no labeled data are available.

There are many interesting ideas on how to solve this problem for both image classification and segmentation tasks. Some early works try to learn a feature space transformation between two domains to compensate for the domain-induced changes [40, 27, 12]. Figure 2.5 is a simple illustration of this idea. Later more works focus on the idea of adversarial adaptation methods, for example, introducing a domain classifier, thus encouraging the CNN model to extract domain-invariant features during training in order to confuse the classifier [43, 9, 10, 19] (see Figure 2.6). The well-known Generative Adversarial Network (GAN) also belongs to the category of adversarial learning [13]. It has inspired the use of a GAN-based loss as the adversarial loss [44], as well as the use of generative models in domain adaptation [31, 36].

GAN is known for generating realistic fake images, e.g., human faces (see Figure 2.7). A vanilla GAN consists of a generator and a discriminator. The generator randomly generates

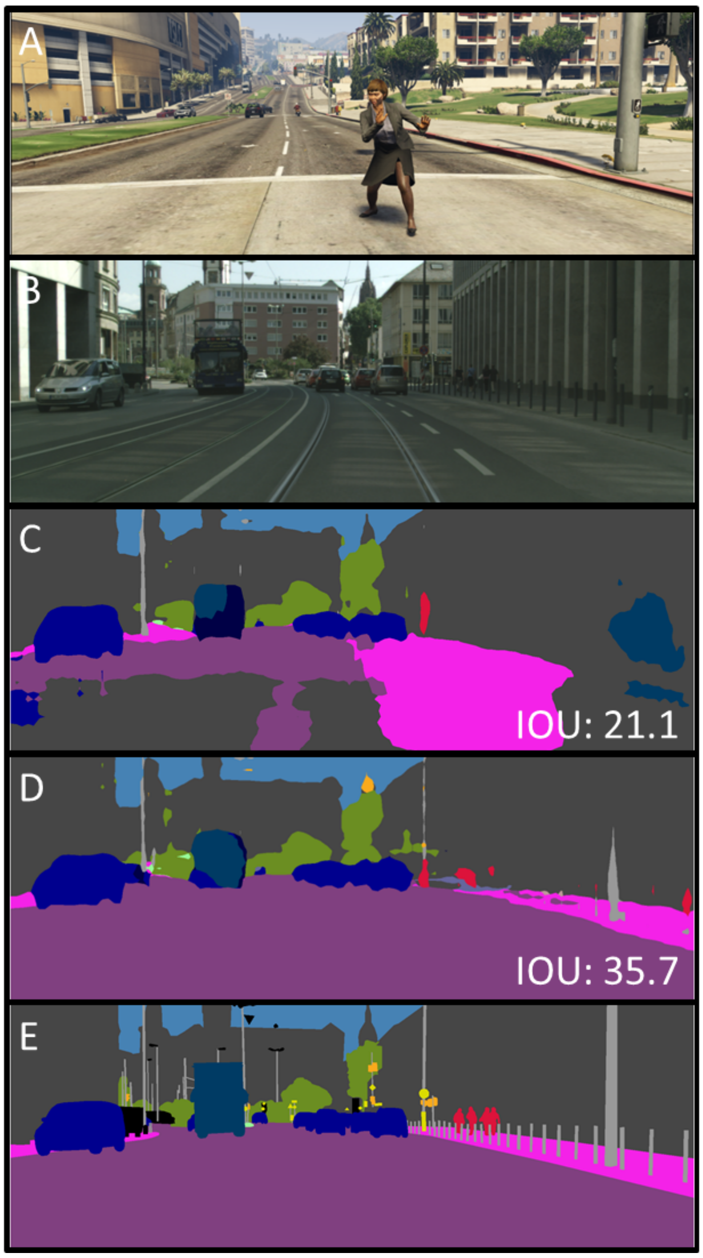

Figure 2.4: An example showing the impact of domain shift on semantic segmentation. If a model is trained on the synthetic GTA5 dataset **A** and is tested on the real Cityscapes dataset **B** without any domain adaptation, the segmentation result **C** is not very good. An unsupervised domain adaptation framework proposed in [36] yields better result **D**. Comparing the two results with the ground truth **E** reveals that large errors on roads, sidewalks and buildings are fixed. The mean Intersection Over Union (IOU) values are reported. Taken from [36].

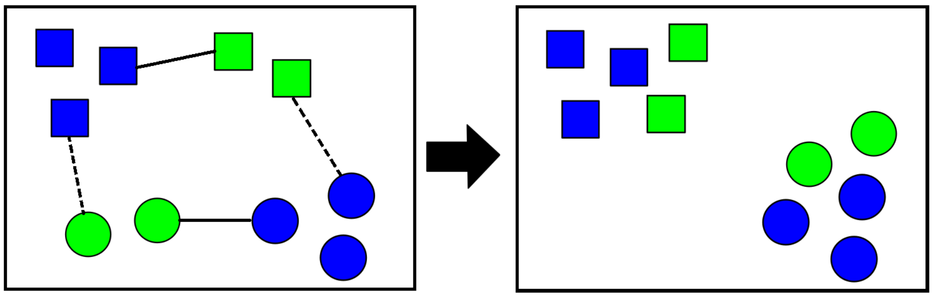

Figure 2.5: Illustration of Information-theoretic Metric Learning (ITML) method proposed in [40]. Data come from two different domains $\mathcal{A}$ and $\mathcal{B}$ (blue and green). A linear transformation W between $\mathcal{A}$ and $\mathcal{B}$ is learned by optimizing the (dis)similarity constraint $x^T W y$ (solid and dashed lines), where $x \in \mathcal{A}$ and $y \in \mathcal{B}$. This is applied in kernel space, so that non-linear transformations can be learned.

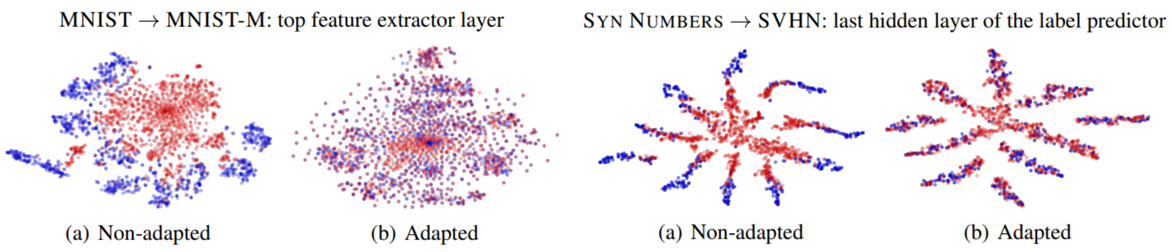

Figure 2.6: T-SNE embedding visualization of distributions of extracted features. Blue are source domain images. Red are target domain images. The adaptation method proposed in [9] makes the two distributions closer. Taken from [9].

images, and the discriminator distinguishes between real and fake images. They compete against each other, thus encouraging the generator to generate realistic-looking images. In more detail, the generator tries to generate images from the target distribution, while the discriminator tries to predict if the image is from the target distribution or not. They are jointly trained, and the train losses are exactly the opposite. In the end, the generator will learn to approximate the target distribution, and the discriminator cannot distinguish between them, so it has to predict randomly.

In many works, researchers have gone beyond obtaining completely random images and

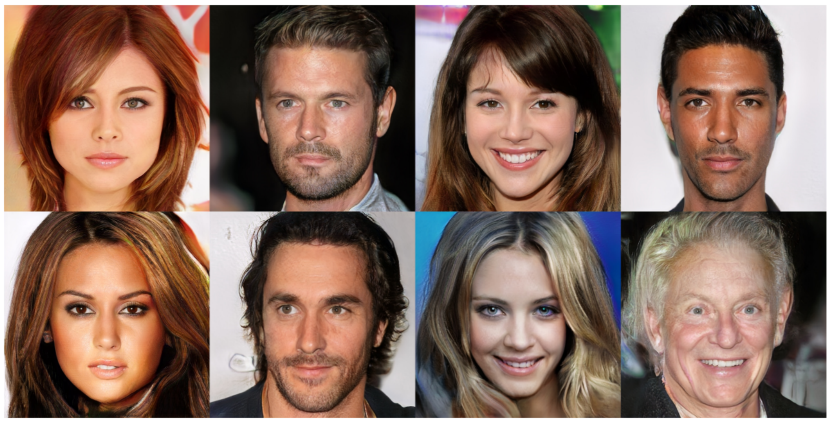

Figure 2.7: Fake human faces generated by Progressive Growing GAN (ProGAN) proposed in [24]. ProGAN is an extension of GAN which adopts a new training process and can produce large high-quality images. Taken from [24].

move further to modify the GAN model so that the generator can produce certain outputs based on given inputs. GANs applied in the conditional settings are called conditional GANs (cGANs). The conditions can be labels [35], text [38], and even images, for example, the pix2pix model converts images from one style to another [20] (see Figure 2.8). There are also other models that do similar things [23, 47] (see Figure 2.9). However, cGANs are not suitable for solving domain adaptation problems, because they require paired inputs. The core idea of cGAN is presenting the condition/target (ground truth images) pair and condition/output (fake images generated by the generator) pair to the discriminator. For comparison, the discriminator in a vanilla GAN takes in target and output (with no conditions) and tries to distinguish them. In the case of domain adaptation, the condition is an image from one domain, but there is no guarantee that there is a paired image from the other domain to be used as the target.

It is Cycle Generative Adversarial Network (CycleGAN) that makes the idea of unpaired image-to-image translation possible [49]. The approach is easy to understand. Images $\{x_i\}_{i=1}^{N}$ and $\{y_j\}_{j=1}^{M}$ are available for training. They come from two different domains X and Y, where $x \in X$ and $y \in Y$. For these images, a mapping function $G : X \to Y$ should be learned by the generator. The discriminator D_Y is responsible for distinguishing between images $\{y\}$ and generated images $\{G(x)\}$. In order to do the mapping in the other direction, a second set of

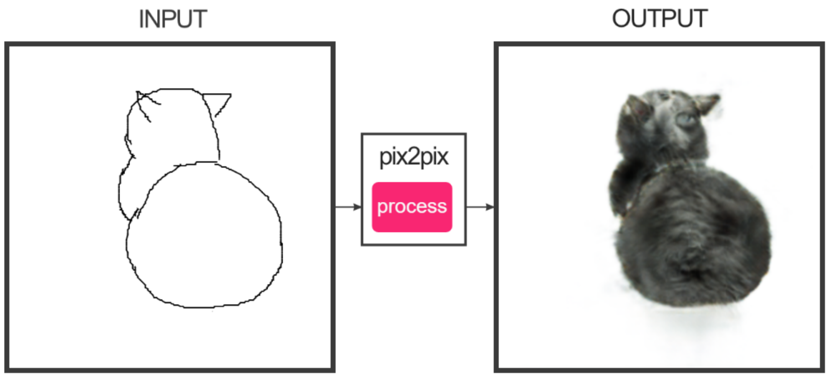

Figure 2.8: A cat image generated by the pix2pix model that corresponds to the edges. This model is trained on cat images and their edges. The pix2pix model can do a lot of other things, such as generating street scenes given the labels and generating colored images given the gray images. The setting is always the same. It maps pixels to pixels. The interactive demo made by Christopher Hesse is available under the link: `https://affinelayer.com/pixsrv/`.

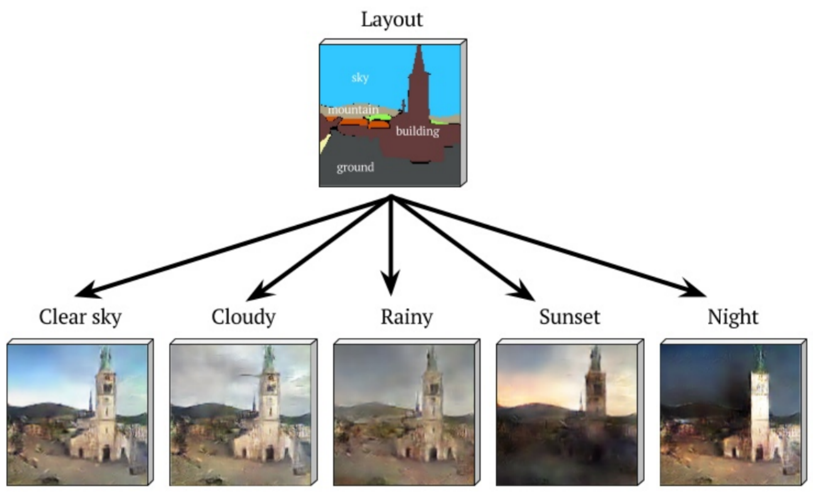

Figure 2.9: Images generated showing different weather conditions given the semantic layout. This model proposed in [23] is named AL-CGAN. Taken from [23].

generator and discriminator is trained in a similar way. The generator learns the mapping $F : Y \to X$, and the corresponding discriminator is denoted as D_X. However, this process alone cannot guarantee that the resulting mapping is ideal. As [49] points out:

> With large enough capacity, a network can map the same set of input images to any random permutation of images in the target domain, where any of the learned mappings can induce an output distribution that matches the target distribution.

Therefore, the constraint of cycle-consistency is introduced, which states that the transformation of an image from one domain to the other and then back again should result in the original image, i.e., $x \to G(x) \to F(G(x)) \approx x$. This process is called forward cycle consistency. Similarly, the backward cycle consistency $y \to F(y) \to G(F(y)) \approx y$ should also be satisfied. This concept is illustrated in Figure 2.10.

CycleGAN is one of the benchmarks for domain adaptation. There are frameworks

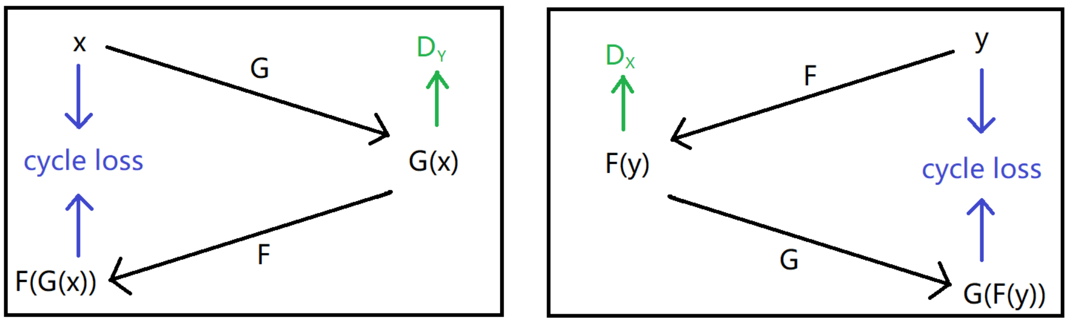

Figure 2.10: Illustration of the CycleGAN technique. CycleGAN model consists of two generators which learns the two mappings $G : X \to Y$ and $F : Y \to X$ and two corresponding discriminators. Furthermore, forward and backward cycle consistency are required, i.e., $x \to G(x) \to F(G(x)) \approx x$ and $y \to F(y) \to G(F(y)) \approx y$.

based on CycleGAN for image classification and segmentation, for example, Cycle-Consistent Adversarial Domain Adaptation (CyCADA), Synergistic Image and Feature Adaptation (SIFA) and Cycle Consistency Panoptic Domain Adaptive Mask R-CNN (CyC-PDAM) [18, 3, 30]. CyCADA is a direct extension of CycleGAN. In addition to pixel loss and cycle consistency, CyCADA also considers feature loss and semantic loss. SIFA, which is developed for semantic segmentation of medical images, adopts a similar idea to CyCADA, but has a more elegant architecture by fusing the CycleGAN with a semantic segmentation model. This framework will be explained in detail in Section 3.3.3. CyC-PDAM focuses on instance segmentation. It combines CyCADA and an instance segmentation framework called Mask R-CNN [15], and it also has a lot of other features, such as panoptic level domain adaptation inspired by [26] and [25].

These three frameworks are different from each other, but they share the same basic idea of using CycleGAN to disguise images from the source domain as images from the target domain, which makes sense. Usually, there is more information available for data from the source domain, such as the ground truth labels for doing semantic segmentation, while little information is given for the data from the target domain, which makes the training difficult or even infeasible. With the help of CycleGAN, source images are stylized as target images, which are used to train the model, and the training can benefit from the additional information on the source images. Of course, a naive implementation of this idea will face a lot of problems. The frameworks contain plenty of other details, which are not mentioned here.

2.4 Light microscopy image simulation using Blender

The tradition way to investigate AMF is explained in Section 2.1. However, if an image segmentation model is to be trained for this task, it is not ideal to obtain training data in this way. Light microscopy images are typically of very high resolution, and manual annotation of different plant structures per pixel is nearly infeasible. In addition, identifying plant structures is not easy, and manual labeling cannot guarantee 100% accuracy. If these flawed data are used for training, the performance of the model will degrade.

In order to solve this problem, a 3D model of a fungus-colonized root section is built in [46] using the 3D computer graphics software Blender, so that synthetic 2D images and labels used for training a semantic segmentation model can be generated automatically. The root anatomy is modeled, and deformation is allowed so that the root can be imaged in different positions and poses (see Figure 2.11). The rendering and output configurations are also set (see Figure 2.12), and images can be directly exported. With this 3D model, training data can be easily obtained.

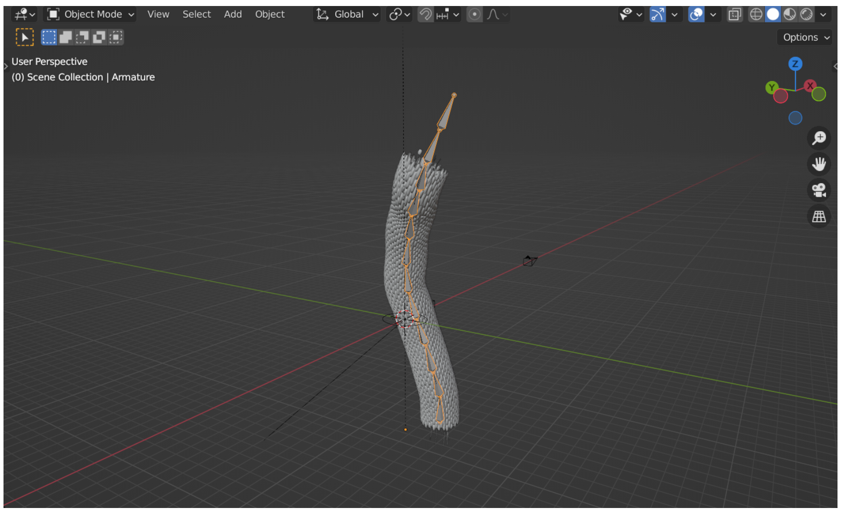

Figure 2.11: Screenshot of the software Blender showing the root model. Bones (structure with orange outlines in the middle) are added to the model and enable the deformation of the 3D mesh (gray structure surrounding the bones).

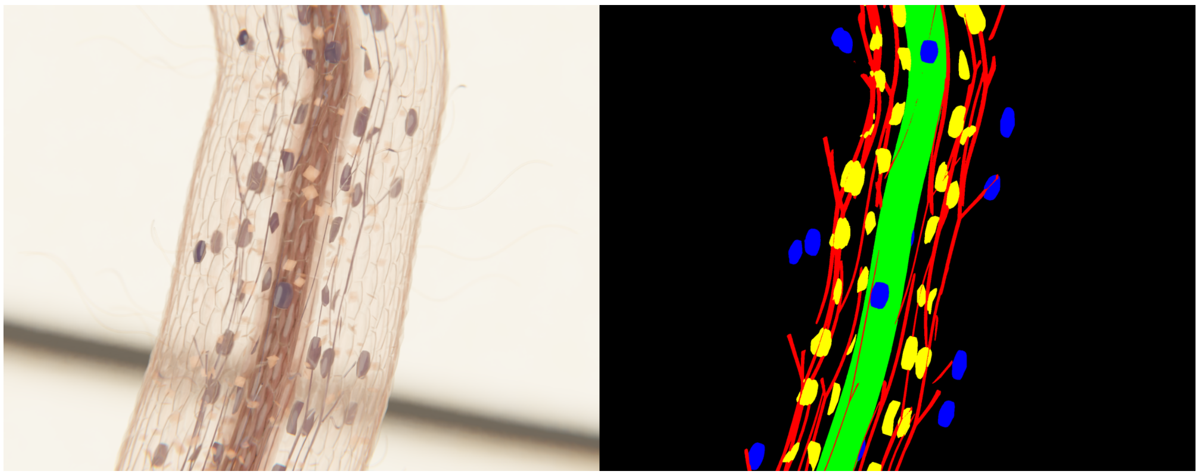

Figure 2.12: Images exported by the software Blender. Left is a realistic rendered image of the root. The black line in the background is the simulation of the edge of an air bubble, as cotton fibres and air bubbles can sometimes be seen in light microscopy images. Right are the corresponding labels. There are four classes in addition to the background: root cortex (green), vesicles (blue), arbuscules (yellow) and hyphae (red).

3 Light microscopy image segmentation

This chapter describes the development of the segmentation model for light microscopy images. First, the background and the goal of this work is given (Section 3.1). Next, the evaluation metric of the model is discussed (Section 3.2). Then the model and its implementation are described in detail (Section 3.3). After that, experimental results are presented (Section 3.4). Finally, possible improvements of the current model are suggested (Section 3.5).

3.1 Problem description

Light microscopy images of roots are provided by the Life Science department at TUM (see Figure 3.1). A segmentation model should be developed to identify the AMF colonization in these images. The main challenge is the lack of ground truth labels for the light microscopy images. In the conventional method of training a semantic segmentation model, each pixel of an image should be assigned a label, which is not possible in this case. Training without manually labeled images is the problem this project aims to overcome. In addition, the number of light microscopy images is limited, because the process of obtaining these images is tedious and requires a lot of effort. This also adds to the difficulty of this project.

There has been much effort devoted to this project. A simulation pipeline is developed

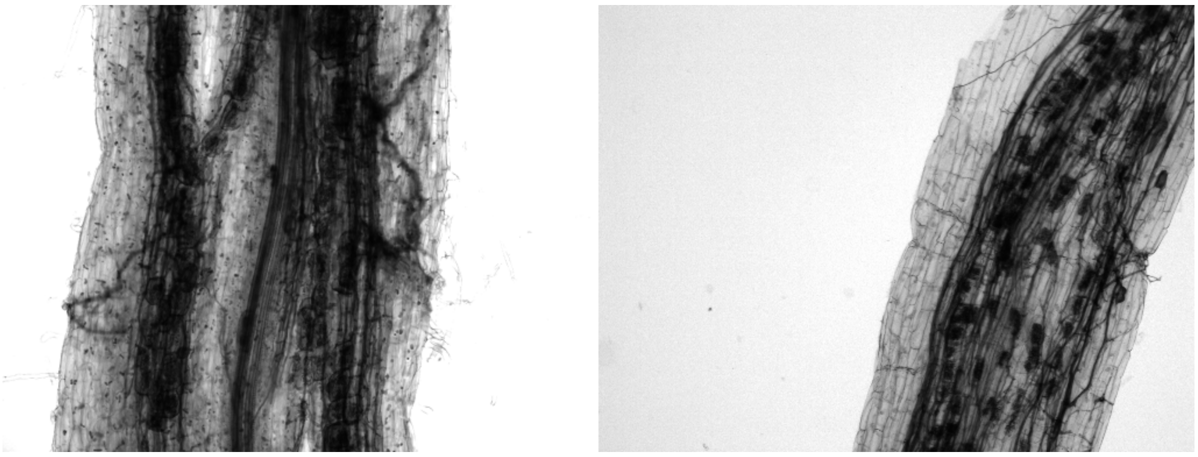

Figure 3.1: Light microscopy images of roots available through the Life Science department at TUM, lab of Prof. Caroline Gutjahr. These images are of very high resolution (2048 $\times$ 1536). Manual annotation of these images is very costly.

in the 3D modeling environment Blender to simulate the light microscopy images [46]. As shown in Figure 2.11 and Figure 2.12, the 3D model of the plant root can be adjusted into different poses and then rendered. Specific parts of the root are highlighted to generate labels for segmentation tasks. This ensures that there are adequate data to train a semantic segmentation model. However, the model will face problems when dealing with real-world data, as it has only been trained on synthetic data. Another work focuses on the problem of the limited number of light microscopy images and explores the field of deep generative modeling [42]. A model which generates realistic light microscopy image data is developed (see Figure 3.2). These data can potentially be used to train a segmentation model, but the corresponding labels are still not available. New approaches should be adopted in order to combine these two efforts and solve the above problems. This is the starting point for this work.

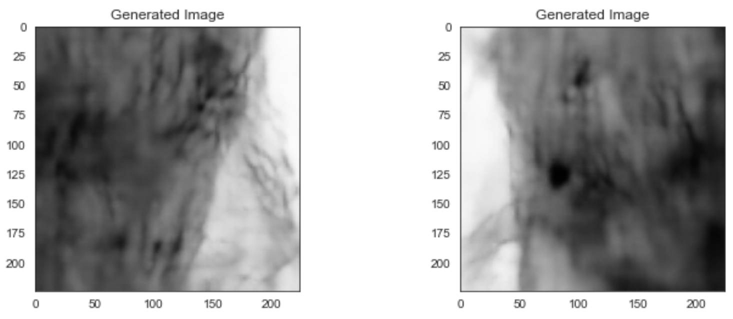

Figure 3.2: Sample image patches generated by a Variational Autoencoder (VAE) with a ResNet-18 architecture trained in [42]. The model is able to learn details in the light microscopy images. Taken from [42].

3.2 Evaluation metrics

When evaluating classification models in machine learning, predictions are classified into four categories: true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). Among them, TP and TN are correct predictions, while FP and FN are wrong predictions. Table 3.1 is a 2×2 confusion matrix that summarizes the relationship between these four outcomes. Figure 3.3 is an illustration of classification results.

In this work, a semantic segmentation model should be developed. Pixel accuracy, Intersection over Union (IoU) and Dice similarity coefficient (DSC) are evaluation metrics commonly used for semantic segmentation model evaluation, and they will be further explained in the following subsections.

Table 3.1: Confusion matrix.

	Ground truth	
Prediction	$y = 1$	$y = 0$
$\hat{y} = 1$	TP	FP
$\hat{y} = 0$	FN	TN

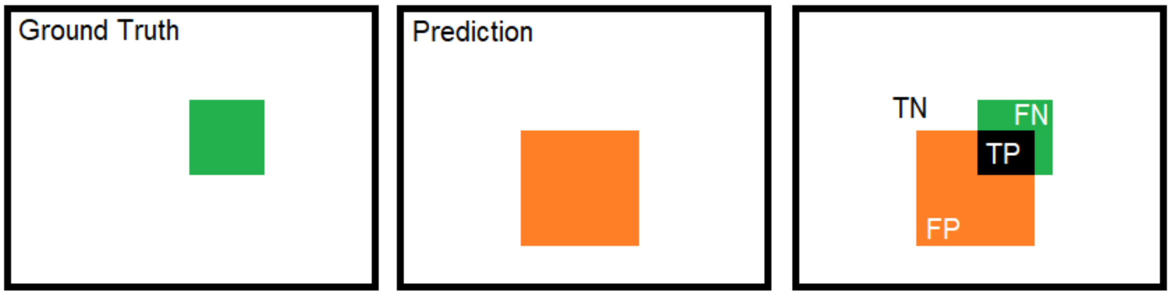

Figure 3.3: Classification result of a certain class. A pixel either belongs to the class (positive) or it does not (negative). Left and middle are the ground truth and the prediction repectively. Right is the classification result. TP and TN are pixels that are successfully predicted by the model to be positive (black) or negative (white). FP are pixels predicted to be positive, but actually they are not (orange). Similarly, FN are pixels falsely predicted to be negative (green).

3.2.1 Pixel accuracy

Pixel accuracy is one of the most basic evaluation metrics. It considers the percentage of pixels that are correctly classified. When evaluating a semantic segmentation model, there are two ways to calculate pixel accuracy. One is per-class pixel accuracy, which reports accuracy for each class. This is equivalent to a binary classification of the class in question while masking out all the other classes. The calculation of the per-class pixel accuracy is

$$acc = \frac{TP + TN}{TP + TN + FP + FN}. \tag{3.1}$$

The other is overall accuracy, which simply calculates the ratio of correctly classified pixels to total pixels. However, this evaluation metric can be problematic when working with unbalanced data. For example, if an image has a large background, a dummy model that predicts all pixels to be background will be evaluated as having high pixel accuracy, and this result is misleading.

3.2.2 Intersection over Union

IoU is an alternative metric to evaluate a semantic segmentation model. It is also known as Jaccard index. The definition of IoU is

$$IoU = \frac{|gt \cap prediction|}{|gt \cup prediction|}, \tag{3.2}$$

where gt is the ground truth. As can be seen in Figure 3.3, the intersection of the ground truth and the prediction is TP, and the union of the ground truth and the prediction is $TP + FP + FN$. Therefore, IoU can be calculated as

$$IoU = \frac{TP}{TP + FP + FN}. \tag{3.3}$$

This should be calculated for each class, and then the scores are averaged over all classes. The averaged IoU score is referred to as mean IoU.
This evaluation metric performs better when dealing with unbalanced data. As described in Section 3.2.1, if a dummy model ignores the object class and only predicts that all pixels belong to the background class, evaluation with pixel accuracy fails, because the model correctly and yet meaninglessly states that either there are many pixels that are background (TP) or there are many pixels that are not object (TN), so the pixel accuracy is always high. However, with IoU evaluation metric, the score for the object class is 0, because the model does not correctly predict any pixel to belong to the object class. This result is more relevant in this case.

3.2.3 Dice similarity coefficient

DSC, also known as the F1 score, is similar to the IoU metric. The definition is

$$DSC = \frac{2|gt \cap prediction|}{|gt| + |prediction|}. \tag{3.4}$$

In the binary class cases, this is equivalent to

$$DSC = \frac{2TP}{2TP + FP + FN}. \tag{3.5}$$

The ratio between IoU and DSC is

$$\frac{IoU}{DSC} = \frac{1}{2} + \frac{IoU}{2}, \tag{3.6}$$

where both DSC and IoU range from 0 to 1. It can then be deduced that $DSC \geq IoU \geq \frac{DSC}{2}$. The ratio is close to $\frac{1}{2}$ when both values are close to 0, which means that the IoU metric tends to punish bad classifications harder than the DSC metric. In general, the DSC metric has a similar performance to the IoU metric, and they are positively related.
In this work, the DSC evaluation metric is adopted. The training images contain a large amount of background, and it is important to identify minor classes which are of interest. This makes the DSC metric a proper choice.

3.3 Implementation

3.3.1 Dataset

This work utilizes data from two domains: synthetic images as the source domain and real-world light microscopy images as the target domain. Synthetic images of the plant root as well as the corresponding labels are generated by the 3D model developed in [46]. Light microscopy images are provided by Dr. Catarina Cardoso from the Gutjahr Lab at the TUM School of Life Sciences. The images are cropped and normalized.

The original images are of high resolution. They are cropped into small patches, and these patches constitute the dataset used for developing the model in this work. The cropping is done in a sliding window fashion, as illustrated in Figure 3.4. The resulting image patches have parts that overlap each other. The advantage of doing this is that the limited number of original images is fully utilized and a large amount of data is obtained. The original images have a lot of blank parts, and these blank patches are discarded. These cropped image patches are originally RGB images, which have three color channels. Later, they are converted to grayscale images, which have one color channel. Table 3.2 shows more details of the image cropping process.

Data augmentation is a strategy commonly used in deep learning to increase the amount of data available for training. Models are usually trained to handle real-world data, which are complex, so the training dataset should ideally cover various circumstances that the model may encounter during test. However, it is not always feasible to collect more data for training, so the existing data are augmented, which increases the diversity as well as the size of the dataset. For image data, artificial transformations are performed for data augmentation, such as rotating, flipping, blurring, shearing, adding noise and changing brightness. In this work, image patches are first flipped horizontally and then rotated by 90 degrees for data augmentation, as shown in Figure 3.5. This doubles the number of image patches.

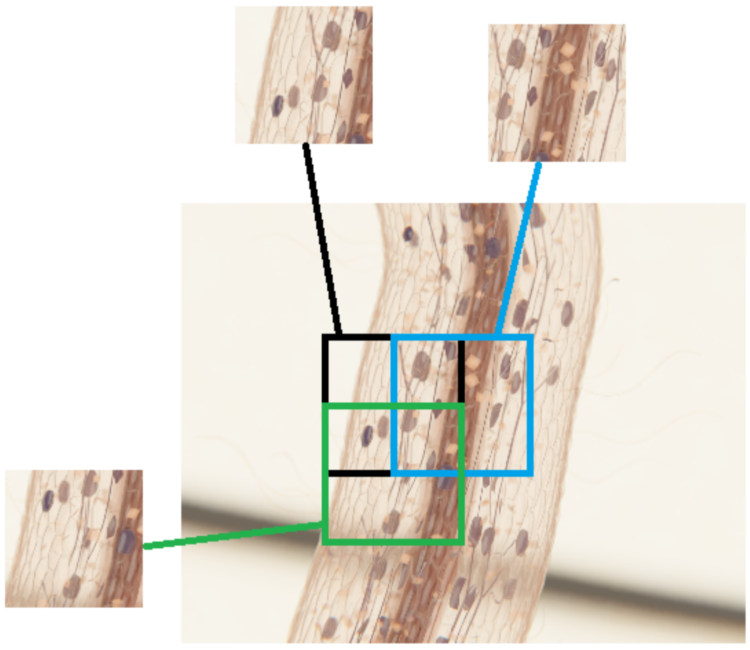

Figure 3.4: Example of image cropping using a synthetic image. The three bounding boxes indicate the three cropped patches. The patches can overlap each other. The corresponding label image is also cropped in the same way, which is not shown here.

Table 3.2: Image cropping. Images are cropped into small patches before being fed into the model. The cropping is done in a sliding window fashion, so that a single image can produce more image patches. Blank patches are not included in the dataset.

	Synthetic images + labels (source domain)	Light microscopy images (target domain)
Original image size	2048 × 1600	2048 × 1536
Cropped image patch size	256 × 256	256 × 256
Number of original images	15	16
Number of cropped patches	5011	4964

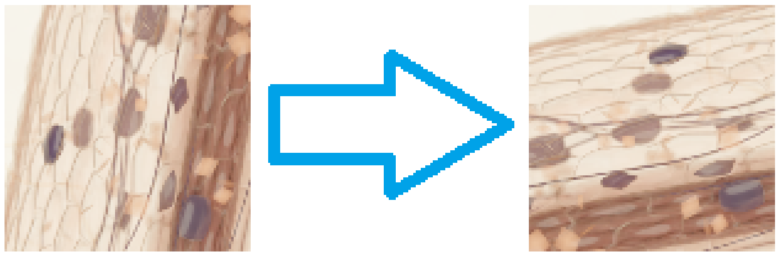

Figure 3.5: Data augmentation of an image patch. Flipping and rotating are commonly used data augmentation techniques. Image patches are first flipped horizontally and then rotated by 90 degrees. These augmented image patches are added to the original dataset, so the size of the dataset is doubled.

Normalization is often used in data pre-processing to obtain ordered and consistent data. For image data, the pixel values change according to the chosen normalization scheme. In this work, min-max normalization is performed on each image patch. The minimum pixel value of the image patch is transformed to 0, and the maximum pixel value is transformed to 1. All other pixel values are transformed to decimals between 0 and 1 depending on their original scale. The calculation formula is

$$x' = \frac{x - min(x)}{max(x) - min(x)}, \tag{3.7}$$

where x is the original image patch and x' is the normalized image patch.

The image patches are divided into two datasets: the training dataset and the validation dataset. The training data are used to train the model, and the validation data are used to assess the model during training and to assist in hyperparameter tuning. 80% data are training data, and 20% data are validation data. It is worth noting that the validation dataset also contains augmented images. Since variations are expected in real-world data, a mild augmentation of the validation data may help to better evaluate the model during training.

The test data come from two microscopy images. They show the same colonized root section, but they are obtained at different z-positions, which means the two images are sightly different from each other and they share the same label. Each image is cropped into 400 image patches. These image patches have ground truth and are used to test the performance of the model after training. The ground truth label is obtained by manually labeling the microscopy image. Figure 3.6 shows the light microscopy image annotated by plant biologists. The label is created based on the annotations. Due to current circumstances, the label is not guaranteed to be pixel-wise accurate, but it is sufficient to serve as an evaluation standard. Table 3.3 shows details of the three datasets.

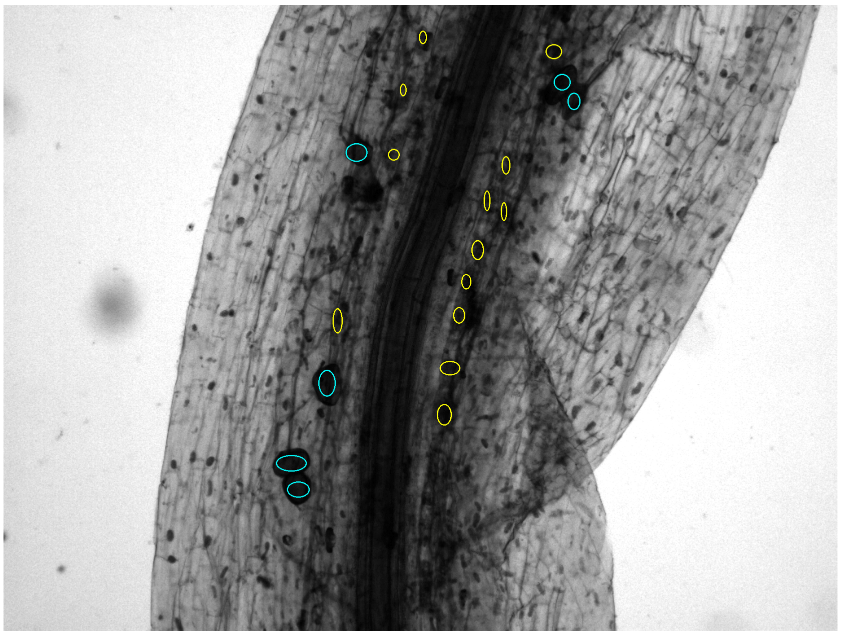

Figure 3.6: A light microscopy image annotated by plant biologists in the lab from Prof. Gutjahr. The yellow circles indicate the arbuscules, and the blue circles indicate the vesicles. The ground truth label of this image is created with the help of the annotations and is used as test data.

Table 3.3: Division of dastaset. Image patches are divided into a training dataset, a validation dataset and a test dataset. The test dataset does not contain synthetic images, as the test process only involves predicting light microscopy images.

	Synthetic images + labels (source domain)	Light microscopy images (target domain)
Number of patches (with data augmentation)	10022	9928
Training + validation	8018 + 2004	7943 +1985
Test	-	800 (with label)

3.3.2 Baseline model

In [46], a U-Net is developed for semantic segmentation of synthetic images with fungal objects. Figure 3.7 shows the architecture of the model. This model is able to make predictions with high accuracy. However, the model suffers from a serious performance degradation problem when dealing with real-world light microscopy images, because it is only trained with the synthetic images. In this work, the model serves as the baseline model for semantic segmentation without domain adaptation.

The U-Net architecture is first proposed in [39] for biomedical image segmentation. This

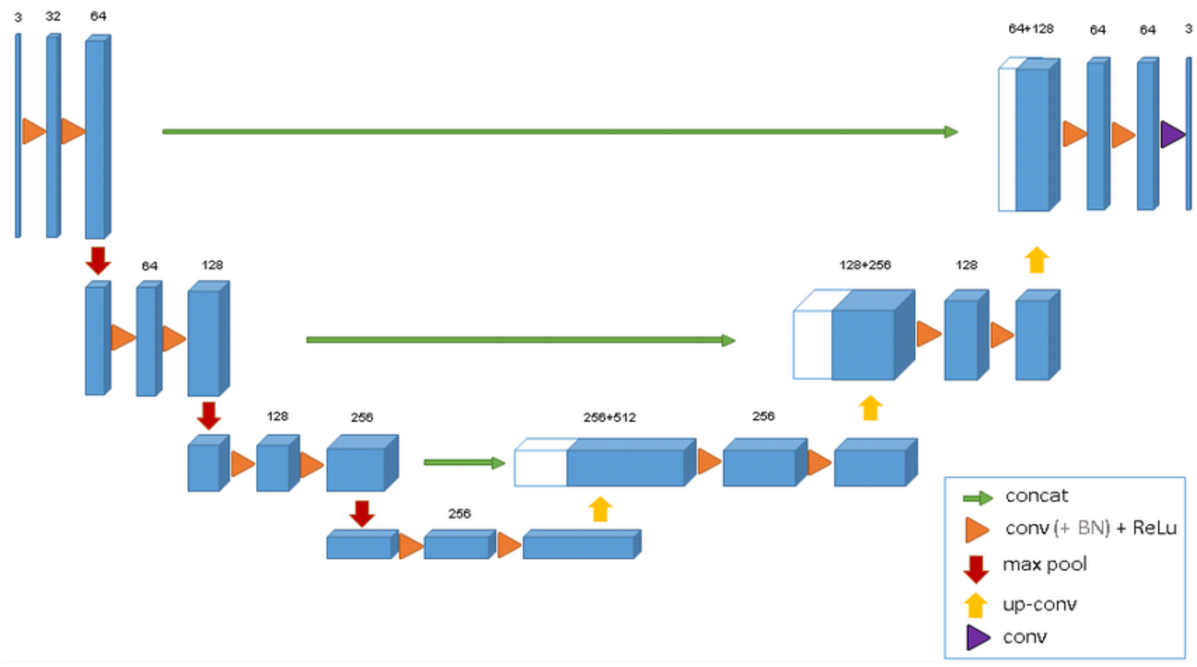

Figure 3.7: The U-Net architecture. The semantic segmentation model in [46] adopts this architecture. Taken from [46].

network only needs to be trained with a few images to achieve a good performance, which makes it a popular choice for biomedical tasks. As shown in Figure 3.7, the U-Net has a symmetric architecture. The contracting path on the left is a typical fully convolutional network that extracts features, and the expansive path on the right brings the image to its original size using transposed convolutions. A skip connection at each level brings additional information about the low-level features, and these connections also enable the fusion of features from different levels. The model in [46] is a 4-class model. For simplicity, the hyphae class is omitted in the following figures. The model makes good predictions for background, arbuscules and vesicles, but it fails in predicting the real-world images. Figure 3.8 shows the prediction results.

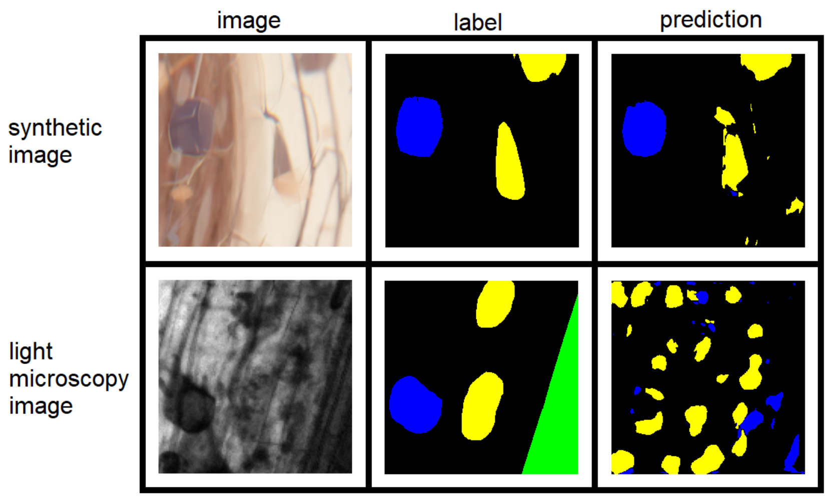

Figure 3.8: Prediction results of the baseline model. The baseline model is a U-Net trained on synthetic images. It makes good predictions for the synthetic images (top row), but it cannot predict the real-world images because of the domain gap (bottom row). The root cortex, vesicles and the arbuscules are represented by the green, blue and yellow colors. The baseline model does not include the root cortex class.

3.3.3 SIFA framework

The Synergistic Image and Feature Adaptation (SIFA) framework is proposed in [3] for unsupervised domain adaptation in medical image segmentation. The segmentation model is integrated with the domain adaptation network. It considers both pixel and feature adaptations and has achieved good performance on the task. This work implements the SIFA framework for light microscopy image segmentation using PyTorch. The implementation details are discussed in this subsection.

Architecture

The SIFA framework can be divided into 6 modules. For clarity, the source domain (synthetic image) is denoted by the letter A and the target domain (light microscopy image) is denoted by the letter B. The aim is to perform domain adaptation from A to B. The 6 modules are listed in Table 3.4. These modules are sequentially optimized during the training process, and the training is end-to-end. Figure 3.9 shows the architecture of the framework.

The network G_A receives images from domain A, denoted as images a, and converts them to images from domain B, denoted as images b. The network D_B receives both fake images b generated by G_A and real images b, and tries to distinguish which images are real. The G_B and D_A pair is similar and works in the opposite direction, but the architecture of G_B is different from G_A. G_B consists of an encoder E and a decoder U. The encoder E is shared in the segmentation network S_B. G_A, D_B. G_B and D_A together form a CycleGAN, but it is different from the vanilla CycleGAN and has some other features. These features will be

Table 3.4: Modules in the SIFA framework. The modules in the SIFA framework as well as the number of trainable parameters are given in the table. It should be noted that the encoder E is shared between the module S_B and the module G_B.

Module type	Module name	Number of trainable parameters
Generator	G_A	2844289
Discriminator	D_B	2762689
Segmentation model	S_B (E and C)	27527524 (27525472 + 2052)
Generator	G_B (E and U)	29426849 (27525472 +1901377)
Discriminator	D_A	2770882
Discriminator	D_P	2765761

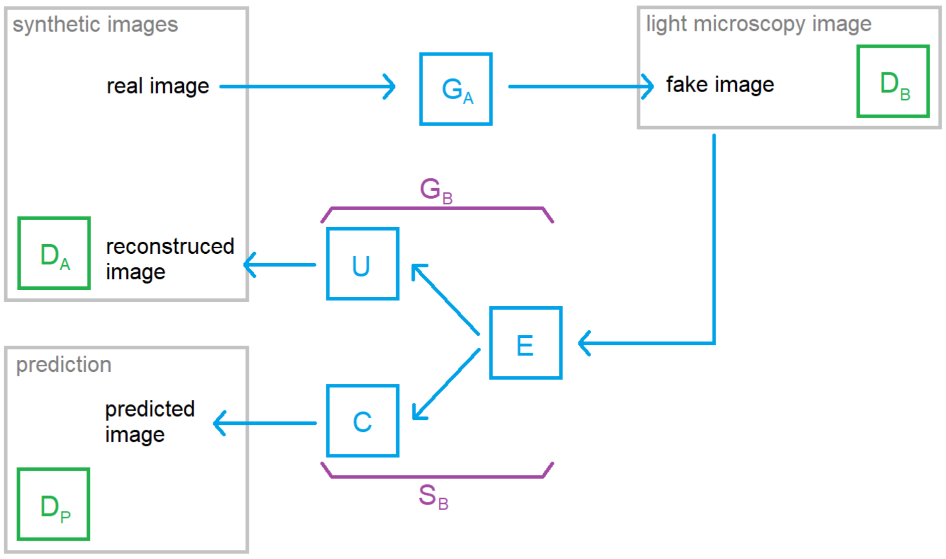

Figure 3.9: Architecture of the SIFA framework. It can be divided into 6 modules: G_A, D_B, S_B, G_B, D_A and D_P. E is an encoder, U is a decoder, and C is a classifier.

further explained afterwards. Figure 3.10 illustrates how these 4 modules work.

The segmentation model S_B consists of an encoder E and a classifier C. It receives images

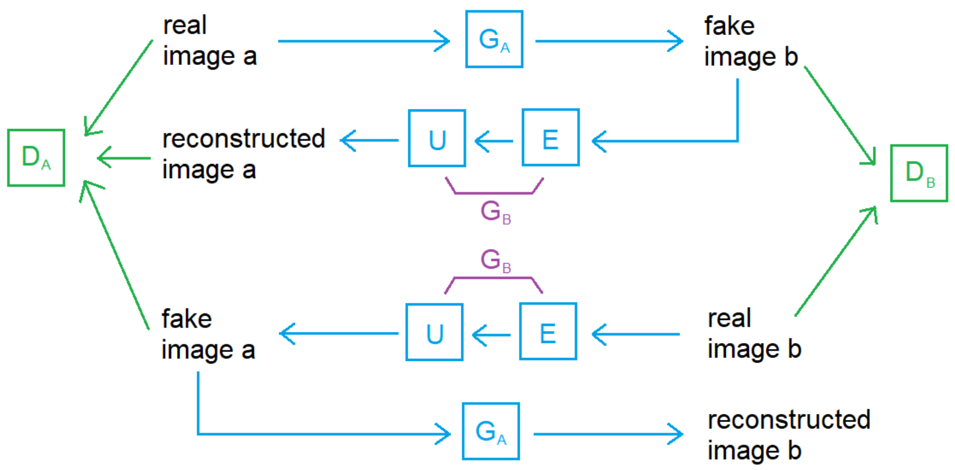

Figure 3.10: Illustration of G_A, D_B, G_B and D_A modules. G_A is a generator which generates images from domain B given images from domain A. The discriminator D_B should distinguish whether the received images are real images or generated by G_A. The other generative module G_B is composed of an encoder E and a decoder U. The decoder U is shared in another module S_B, which is not shown in this figure. G_B converts images b to images a, and a discriminator D_A distinguishes between real and fake images a. In addition, D_A is asked to differentiate between fake images a and reconstructed images a so that the encoder E is encouraged to extract domain-invariant features.

from domain B and assigns a label to each pixel of the images. The label belongs to one of the following 4 classes:

- Background
- Root cortex
- Vesicles
- Arbuscules

Both real images b and fake images b generated by G_A are presented to S_B for semantic segmentation. Fake images b have ground truth labels, since they are converted from real images a. These ground truth labels are used to optimize S_B by encouraging it to make predictions close to the ground truth, as in other semantic segmentation models without domain adaptation. Real images b do not have ground truth labels. The discriminator D_P receives predictions of fake images b and real images b, and distinguishes between them. Figure 3.11 is an illustration of these 2 modules.

During the test, only the module for semantic segmentation S_B is required, namely the

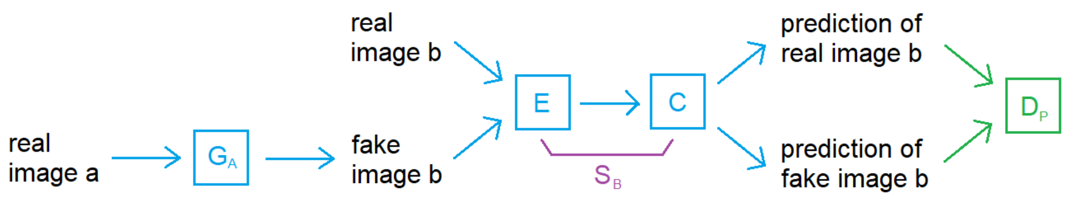

Figure 3.11: Illustration of S_B and D_P modules. S_B is a semantic segmentation model which consists of an encoder E and a classifier C. It receives images b and makes predictions. The discriminator D_P differentiates whether the images are predictions of real images b or fake images b.

encoder E and the classifier C. This trained segmentation model receives real-world light microscopy images and makes prediction for the images.

Losses

In order to optimize the modules, losses in different aspects are calculated and minimized:

- Image adaptation: $\mathcal{L}^b_{ls}$, $\mathcal{L}^a_{ls}$, $\mathcal{L}_{cyc}$ and $\mathcal{L}_{seg}$
- Feature adaptation: $\mathcal{L}^p_{ls}$ and $\mathcal{L}^{\tilde{a}}_{ls}$

Image adaptation aims to align the appearance of images. In some cases, a domain adaptation network can already achieve good results with only image adaptation. Feature-level adaptation provides additional improvement and leads to better performance when the domain shift is significant.

Common GANs use minimax loss, which is derived from the binary cross-entropy. This network uses least squares loss function instead, and the GAN is called Least Squares GAN (LSGAN) [34]. The generator G_A converts image a to image b. The process can be described as

$$G_A(x^a) = x^{a\to b}, \tag{3.8}$$

where x^a is real image a and $x^{a\to b}$ is fake image b. The objective functions for LSGAN are defined as

$$\begin{aligned} \mathcal{L}^b_{ls}(D_B) =& \frac{1}{2}\mathbb{E}_{x^b\sim X^b}[(D_B(x^b)-1)^2]+ \\ & \frac{1}{2}\mathbb{E}_{x^a\sim X^a}[(D_B(G_A(x^a)))^2], \\ \mathcal{L}^b_{ls}(G_A) =& \frac{1}{2}\mathbb{E}_{x^a\sim X^a}[(D_B(G_A(x^a))-1)^2]. \end{aligned} \tag{3.9}$$

Minimizing the objective function is equivalent to minimizing the Pearson χ^2 divergence, and this encourage the generator to generate images as real as possible.

The other pair G_B and D_A is trained in the same way with the loss $\mathcal{L}_{ls}^a$. The loss functions are

$$\begin{aligned} \mathcal{L}_{ls}^a(D_A) =& \frac{1}{2}\mathbb{E}_{x^a \sim X^a}[(D_A(x^a) - 1)^2] + \\ & \frac{1}{2}\mathbb{E}_{x^b \sim X^b}[(D_A(U(E(x^b))))^2], \\ \mathcal{L}_{ls}^a(E, U) =& \frac{1}{2}\mathbb{E}_{x^b \sim X^b}[(D_A(U(E(x^a))) - 1)^2]. \end{aligned} \tag{3.10}$$

G_B receives images b and converts them to image a, which is similar to Equation 3.8 but in an opposite direction. D_A distinguishes between real and fake images in domain A. Unlike G_A, G_B consists of an encoder E and a decoder U. The encoder E is shared in the semantic segmentation module.

The cycle-consistency constraint is imposed on the image conversion process. As shown in Figure 3.10, real image x^a is converted to fake image $x^{a \to b}$ by G_A and then converted back by G_B. The reconstructed image should be close to the original image, which can be written as

$$G_B(G_A(x^a)) = U(E(G_A(x^a))) \approx x^a. \tag{3.11}$$

Similarly, the original image b should also be recovered after the series of conversions, which can be written as

$$G_A(G_B(x^b)) = G_A(U(E(x^b))) \approx x^b. \tag{3.12}$$

The corresponding loss can be calculated by imposing an L1 penalty on the reconstruction error in Equation 3.11 and 3.12, and the loss function is

$$\begin{aligned} \mathcal{L}_{cyc}(G_A, E, U) =& \mathbb{E}_{x^a \sim X^a}[\|U(E(G_A(x^a))) - x^a\|_1] + \\ & \mathbb{E}_{x^b \sim X^b}[\|G_A(U(E(x^b))) - x^b\|_1], \end{aligned} \tag{3.13}$$

which is referred to as the cycle-consistency loss.

The S_B module is a semantic segmentation network for domain B. It consists of an encoder E and a classifier C and is trained by minimizing a hybrid segmentation loss $\mathcal{L}_{seg}$. As in Equation 3.8, image x^a is converted to image $x^{a \to b}$, and the image has ground truth label y^a. The encoder E receives image $x^{a \to b}$ and extracts features, then the classifier C makes prediction for the image. This process can be written as

$$\hat{y}^{a \to b} = C(E(x^{a \to b})). \tag{3.14}$$

The prediction $\hat{y}^{a \to b}$ is supposed to be the same as y^a, and this pair is used to calculate the loss. The cross-entropy loss is widely used to measure the performance of classification. However, the Dice loss is preferred in medical image segmentation, because it has a better performance when encountering unbalanced dataset. Therefore, a hybrid of these two losses is used, which is

$$\mathcal{L}_{seg}(E, C) = CE(y^a, \hat{y}^{a \to b}) + \alpha \cdot Dice(y^a, \hat{y}^{a \to b}), \tag{3.15}$$

where the first term is the cross-entropy loss and the second term is the Dice loss. α is a hyperparameter, which assigns different weights to the two losses.

In addition to pixel-level methods, feature-level adaptation is considered, which encourages the network to extract domain-invariant features. The common approach is to use adversarial learning directly in the feature space. This work takes a different approach, because the feature space is high-dimensional. It is easier to work with the lower-dimensional semantic prediction space. As shown in Figure 3.11, the module S_B receives both real image x^b and fake image $x^{a\to b}$, and makes predictions $\hat{y}^b$ and $\hat{y}^{a\to b}$ for them (see Equation 3.14). The module D_P discriminates between $\hat{y}^b$ and $\hat{y}^{a\to b}$. This leads to the losses

$$\begin{aligned}\mathcal{L}^p_{ls}(D_P) =&\frac{1}{2}\mathbb{E}_{x^{a\to b}\sim X^{a\to b}}[(D_P(C(E(x^{a\to b}))) - 1)^2]+\\&\frac{1}{2}\mathbb{E}_{x^b\sim X^b}[(D_P(C(E(x^b))))^2],\\\mathcal{L}^p_{ls}(E,C) =&\frac{1}{2}\mathbb{E}_{x^b\sim X^b}[(D_P(C(E(x^b))) - 1)^2].\end{aligned} \tag{3.16}$$

This encourages the module S_B to extract aligned features from x^b and $x^{a\to b}$ so that the discriminator D_P cannot distinguish between them.

The discriminator D_A differentiate not only between real image a and fake image a, but also between fake image a and reconstructed image a (see Figure 3.10). Fake image a and reconstructed image a are generated by the module G_B, which consists of an encoder E and decoder U. The encoder E is encouraged to extract domain-invariant features so that the discriminator D_A cannot tell the difference. The losses are

$$\begin{aligned}\mathcal{L}^{\tilde{a}}_{ls}(D_A) =&\frac{1}{2}\mathbb{E}_{x^{a\to b}\sim X^{a\to b}}[(D_A(U(E(x^{a\to b}))) - 1)^2]+\\&\frac{1}{2}\mathbb{E}_{x^b\sim X^b}[(D_A(U(E(x^b))))^2],\\\mathcal{L}^{\tilde{a}}_{ls}(E) =&\frac{1}{2}\mathbb{E}_{x^b\sim X^b}[(D_A(U(E(x^b))) - 1)^2].\end{aligned} \tag{3.17}$$

Implementation details

The generator G_A consists of 3 convolution layers, 9 residual blocks, 2 deconvolution layers and 1 output convolution layer. The convolution layers downsample the input. The spatial size of the input is reduced, while more features are extracted. The deconvolution is achieved by using transposed convolution layers. The input is upsampled and the original size of the image is recovered. The residual blocks are first introduced in [16] as the building blocks of the ResNet. They are widely used in deep neural networks. Figure 3.12 shows a single residual block. After two layers, the output is added together with the input. A traditional network learns directly the output, while the residual block learns the difference between the output and the input, namely the residue. The skip in the residual block can prevent the gradient vanishing problem, because even when the weights and the biases of the layer become zero, the same input value is kept and a non-linearity is added, so the residual block can only improve the performance of the network. Figure 3.13 shows the complete architecture of the generator module. The last convolution layer is followed by a tanh activation layer. An additional skip connection is introduced.

The discriminators D_B, D_A and D_P use the PatchGAN discriminator structure [20]. It is

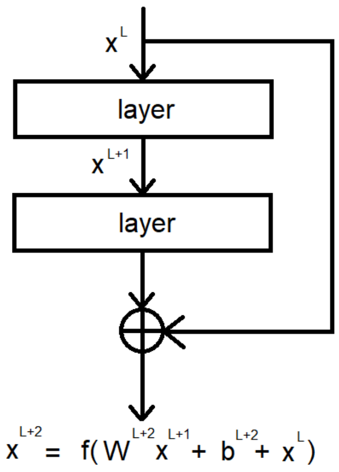

Figure 3.12: A residual block. W is the weight, b is the bias and f is the non-linear activation function. Even when the weights and the biases go to zero, the residual block has an output $x^{L+2} = f(x^L)$. The identity is easy to learn, so introducing residual blocks into a network will not hurt its performance.

basically a set of convolution layers and an average pooling layer as the final layer. Each pixel in the output of the convolution network has a receptive field of 70×70, and its value indicates whether the 70×70 patch in the 256×256 input image is real or fake. This is equivalent to cropping the input image into overlapping 70×70 patches and applying a normal discriminator to each patch. The output of the convolution layers is then averaged to produce the final result. The output of D_A has 2 channels instead of 1, because it not only discriminates between real images and fake images, but also between fakes images and reconstructed images.

An encoder E and a decoder U constitute the G_B module. The encoder is composed of a series of convolution layers, residual blocks and max-pooling layers. For a residual block, if the input and output dimensions do not align, the dimensions are converted with zero padding. A max-pooling layer reduces the spatial size of the input. While a convolution layer computes the features in a region, a max-pooling layer select the strongest feature in a region. The encoder also uses dilated convolutions. In a normal convolution, if the kernel size is $k \times k$, features in a $k \times k$ region is computed. In a dilated convolution, holes are inserted into the $k \times k$ kernel so that it can cover a region larger than $k \times k$ (see Figure 3.15). This enlarges the receptive field, which is the region of input that affects an output unit. The decoder has 1 convolution layer, 4 residual blocks, 3 deconvolution layers and 1 output convolution layer. It receives the latent information provided by the encoder and transforms it back to an image. The decoder has a skip connection, which requires the input image of the G_B module. Figure 3.16 shows the architecture of the encoder and the decoder.

The classifier C consists of a convolution layer and an upsampling layer. It receives latent

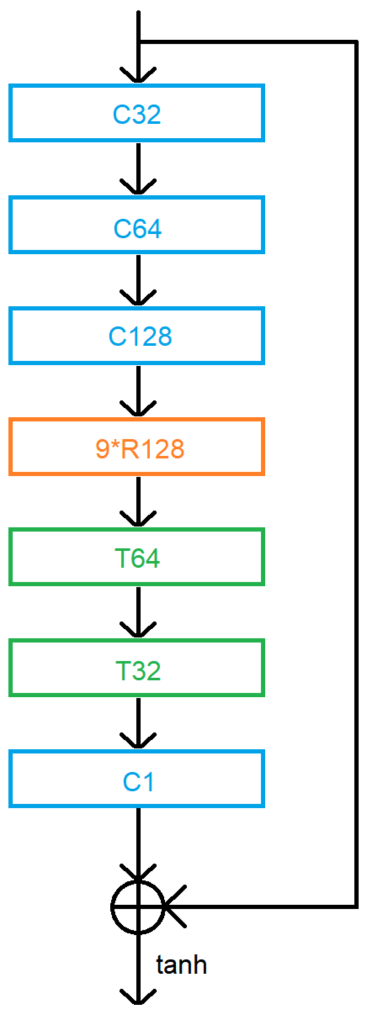

Figure 3.13: Architecture of the generator G_A. C denotes a convolution layer, R denotes a residual block and T denotes a transposed convolution layer. The number in the image indicates the number of output channels in each layer. For the first 3 convolution layers, the number of channels is doubled and the image size is halved in each layer. In the following 9 residual blocks, the dimensions remain unchanged. Then 2 transposed convolution layers reconstruct the image to its original size. In the last convolution layer, the output channel number is 1, because the grayscale image has 1 color channel. This module uses a skip connection. The input and output are summed before a tanh layer.

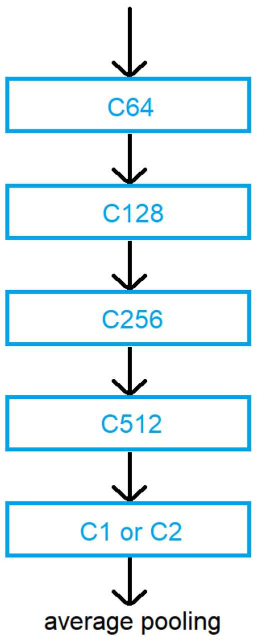

Figure 3.14: Architecture of the discriminators D_B, D_A and D_P. C denotes a convolution layer. The number in the image indicates the number of output channels in the layer. Each pixel in the output of the convolution layers indicates the result of the discrimination of a 70×70 patch in the original input image. The output is then averaged to produce the final result of the discriminator. The output of D_B and D_P has 1 channel, while the output of D_A has 2 channels.

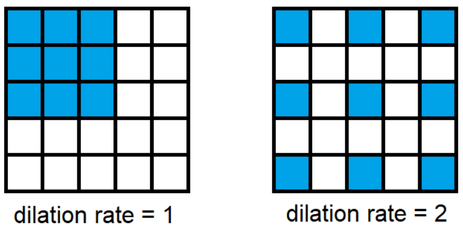

Figure 3.15: Illustration of dilated convolution. Assume that the kernel size is 3×3 (blue) and the stride of the convolution is 1. Left is a normal convolution. The receptive field for this layer is 3×3. Right is a dilated convolution with a dilation rate of 2. Holes are inserted into the kernel, which means that some pixels are skipped during the convolution, and this enlarges the input region that is covered. The receptive field is 5×5 in this example.

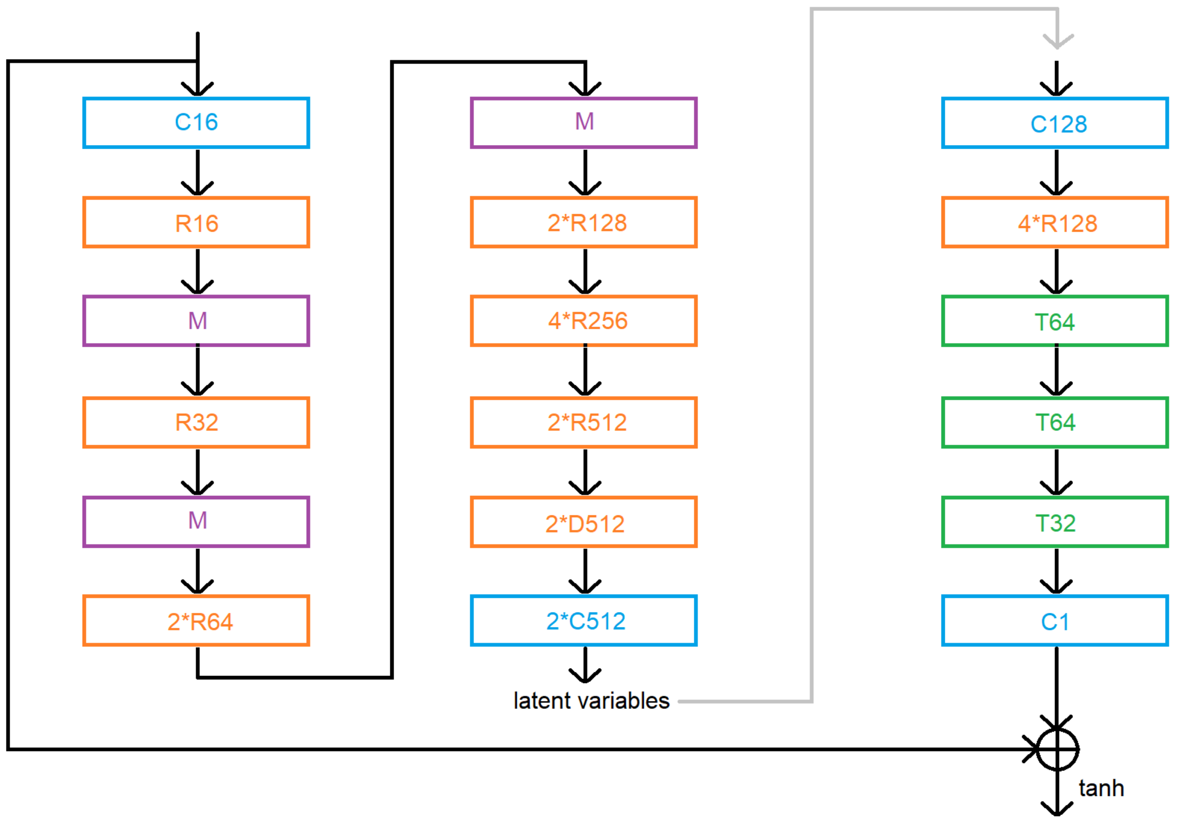

Figure 3.16: Architecture of the encoder E (2 columns on the left) and the decoder U (1 column on the right). C denotes a convolution layer, R denotes a residual block, M denotes a max-pooling layer, D denotes a residual block with dilated convolutions and T denotes a transposed convolution layer. The number indicates the number of output channels. The max-pooling layers in the encoder halve the spatial size, so the output of the encoder has a dimension of $32 \times 32 \times 512$. The transposed convolution layers in the decoder double the spatial size, so the size of the output image is converted back to the original $256 \times 256 \times 1$.

variables from the encoder E and makes predictions for all the four classes. It should be noted that the classifier does not include the softmax function as an output layer, so the output values cannot be interpreted as probabilities.

Table 3.5 is a summary of the module architecture. The notation of the layers in the table corresponds to the one in Figure 3.13, 3.14 and 3.16. More specifically, Ck denotes a convolution layer, Tk denotes a transposed convolution layer, Rk denotes a residual block, Dk denotes a residual block with dilated convolutions and M denotes a max-pooling layer. k is the number of output channels in the layer. The output layers of some modules are omitted. In this table, encoder E, decoder U and classifier C are listed as individual modules, whereas in previous introductions (Table 3.4), they are addressed as part of the segmentation model S_B and the generator G_B.

The hyperparameters of the model mainly follow the setting in [3]. Adam optimization

Table 3.5: Module architecture. The configurations of the modules in the SIFA framework are listed in this table. The type of layers and the number of output channels are given.

Module	Layers
G_A	$C32$, $C64$, $C128$, $9 \times R128$, $T64$, $T32$, $C1$
D_B, D_A, D_P	$C64$, $C128$, $C256$, $C512$, $C1$ or $C2$
E	$C16$, $R16$, M, $R32$, M, $2 \times R64$, M, $2 \times R128$, $4 \times R256$, $2 \times R512$, $2 \times D512$, $2 \times C512$
U	$C128$, $4 \times R128$, $T64$, $T64$, $T32$, $C1$
C	$C4$

is adopted for optimizing the modules. Except for the module S_B, all other modules have a base learning rate of 0.0002. Module S_B has a learning rate of 0.001. Learning rate scheduler is also applied during training, which gradually decreases the learning rate according to the chosen scheme. For module D_B and D_A, the fake images which should be distinguished are picked from a pool. The size of the pool is set as 50, and the pool is continually updated during training. The batch size is 4 due to the GPU memory limitation. The number of training epochs is set based on the performance of the model on the validation dataset during training. A direct evaluation of the performance is challenging, because the light microscopy images do not have ground truth labels, with which a semantic segmentation model can simply be evaluated using DSC or similar metrics. Therefore, the model performance is monitored by tracking the cross-entropy loss and the Dice loss in Equation 3.15 as well as visually comparing the quality of the generated light microscopy images. Although these losses are obtained with the fake light microscopy images generated by the generator and the domain gap still exists, they can verify whether the generator and the segmentation model are working well or not and thus help to monitor the model performance.

3.4 Experimental results

The model is trained for 26 epochs. Figure 3.17 shows the training loss and the validation loss curves. This loss is the combination of the cross-entropy loss and the Dice loss, which indicates how well the segmentation task is done with the fake light microscopy images generated by the generator. Figure 3.18 shows the result images from the validation dataset. It is observed that the predictions for the fake light microscopy images are better than the predictions for the real light microscopy images. This implies that the images generated by the generator are not close enough to the real images. Even though they look like light microscopy images and the domain gap is narrowed, the difference is significant enough to negatively affect the performance of the segmentation model.

The model is tested with the test dataset and the DSC values are in Table 3.6. Figure

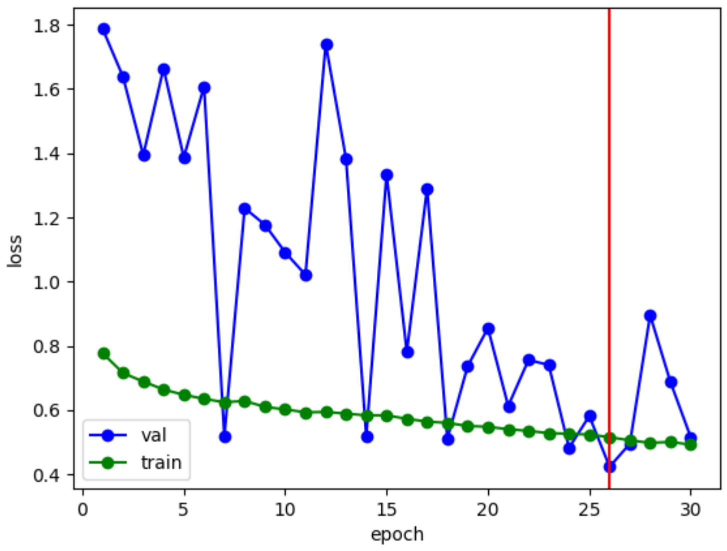

Figure 3.17: Loss curves. The green curve is the training loss and the blue curve is the validation loss. The horizontal axis shows the number of training epochs, and the vertical axis shows the weighted sum of the cross-entropy loss and the Dice loss. The red line indicates when the training is stopped.

3.19 shows some sample images that are representative and can reveal the inadequacy of the model. The model tends to make false positive predictions for the arbuscules, as observed in row 2 in the figure. One possible reason lies in the 3D model that provides the synthetic images used for training. The 3D model of a colonized root section has a lot of arbuscules, while the root sections in the light microscopy images do not necessarily contain many arbuscules. When the generator generates fake light microscopy images, it mimics the style of the real light microscopy image, which has a large background, so the arbuscules in the synthetic images are converted into background patterns. The semantic segmentation model

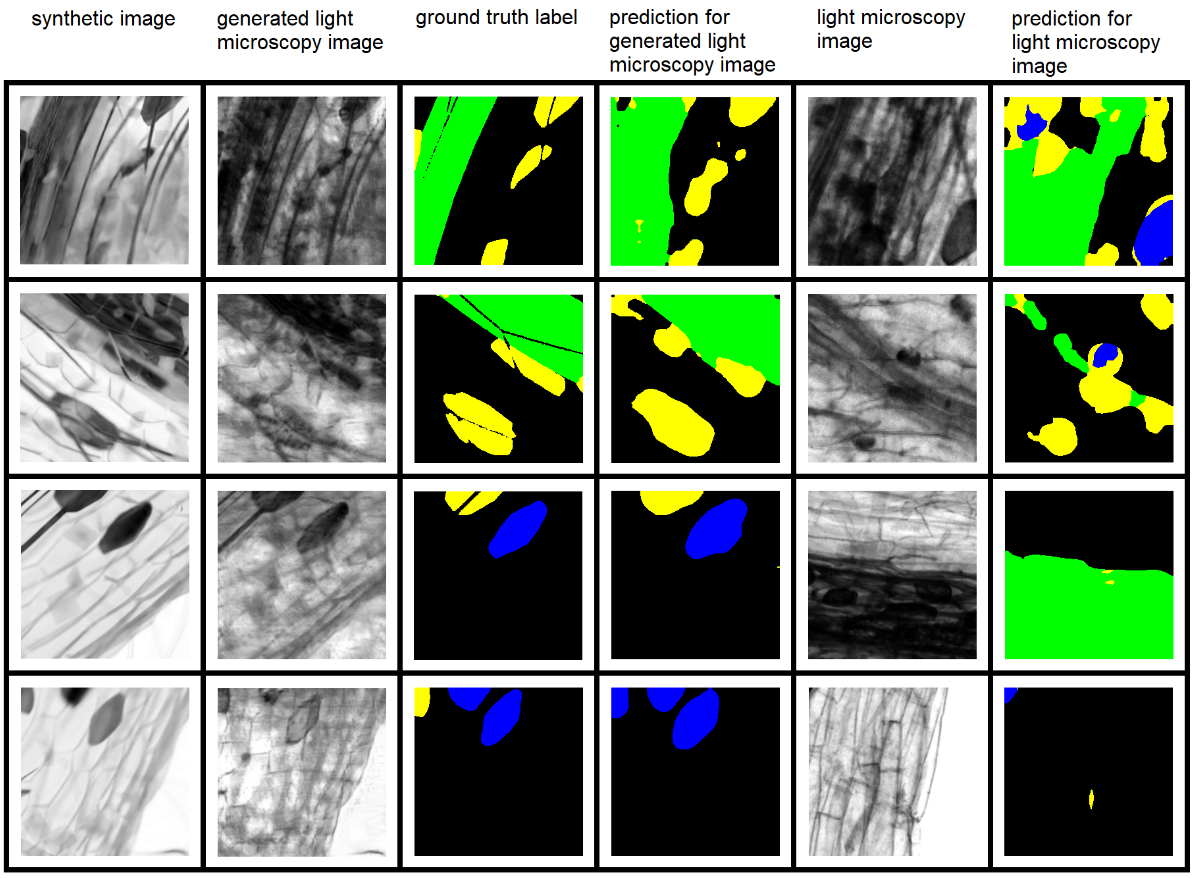
synthetic image
generated light microscopy image
ground truth label
prediction for generated light microscopy image
light microscopy image
prediction for light microscopy image

Figure 3.18: Result images from the validation dataset. Columns 1-4 are the segmentation task using the synthetic images disguised as the light microscopy images. Columns 5-6 are the segmentation task using the real light microscopy images. The real light microscopy images do not have corresponding synthetic images, and the ground truth label is not available. The root cortex, the vesicles and the arbuscules are indicated by the green, blue and yellow colors respectively. Each row corresponds to one example.

receives images with background patterns, but they are labeled as arbuscules, because in the original synthetic images, these regions are truly arbuscules. This results in false positive predictions for the arbuscules. Furthermore, the arbuscules are hard to identify, as they exist in different depths, and sometimes they blend into the background, which makes it difficult for the model to distinguish between the background patterns and the arbuscules. Since the model is encouraged to identify the minor classes as it is trained with weighted losses, it falsely recognizes the background patterns as the arbuscules. This results in a low DSC value for the arbuscules class, as shown in Table 3.6. The hyphae can be mistaken as root cortex, which can be seen in row 1 in this figure. The different z-slices of the light microscopy image result in different predictions. In row 3, the root cortex is not recognized by the model, while in row 4 the prediction is fairly decent. The predictions at the edge of the image patch is not ideal in some cases. In row 3, the model predicts that two arbuscules at the lower edge, which are actually shadows in the background.

Figure 3.20 shows the prediction results of two different models. Both the baseline

Table 3.6: Comparison of two models. Dice similarity coefficient (DSC) on the test data are reported. This table also includes the total number of trainable parameters and the number of images used for training and validation. These images refer to the original synthetic images or light microscopy images rather than the cropped images patches.

Methods	DSC on test data				Parameters	Images
	Root cortex	Vesicles	Arbuscules	Average		
U-Net	-	0.03	0.06	0.05	1362650	420
SIFA	0.31	0.25	0.07	0.21	40572522	31

model and the model developed in this work make predictions on the same image patch. In comparison with the baseline model, the improvement is noticeable. Although mistakes can still be seen, the prediction is overall reasonable. Local textures are more correctly identified by the model. Row 3 shows an interesting case where both models fail to predict the images patch. The prediction results from the two models are similar to each other. Irrelevant objects are falsely predicted as arbuscules. The low accuracy of prediction for the arbuscules remains a major problem of the model. In addition, Table 3.6 reports the test performance as well as the number of trainable parameters and images of both models. The model developed in this work is much larger than the baseline model. The baseline model uses more images for training and validation, but these synthetic images are similar to each other and they are very different from the light microscopy images, so they do not contribute much to improve the model performance.

To further investigate the model, the predictions for different z-slices are compared. Light microscopy images are taken with a z-stack, containing many z-slices. The z-slices are the images taken at different depths, and they show the same root section. The test data come from two different z-slices. In Figure 3.21, predictions are made for the two z-slices, and the results are different. In row 2, the root cortex is not identified. In row 3, some arbuscules are

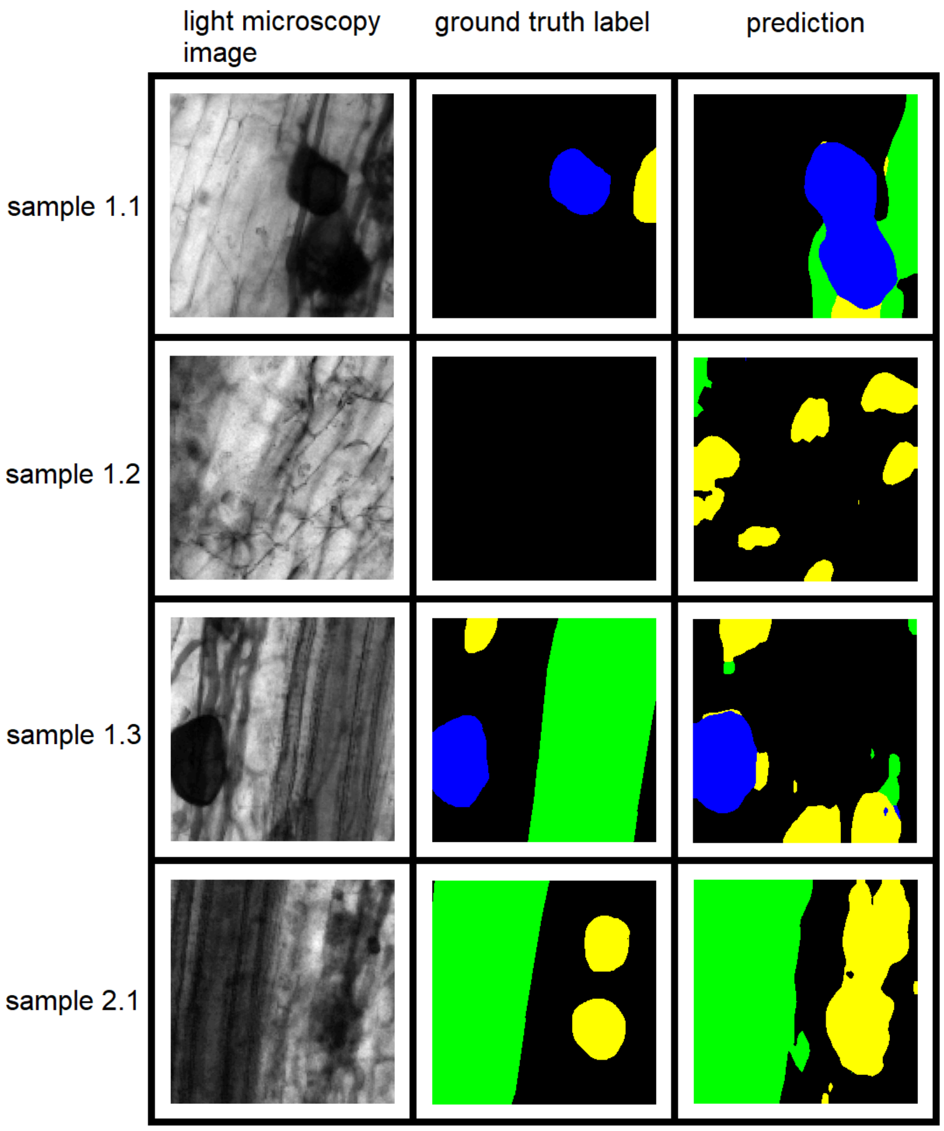

Figure 3.19: Result images from the test dataset. Each row corresponds to one example. The ground truth label is obtained by manual annotation. Rows 1-3 are image patches cropped from the same light microscopy image, and the image patch in row 4 comes from the other light microscopy image, which basically shows the same root section but with a different depth.

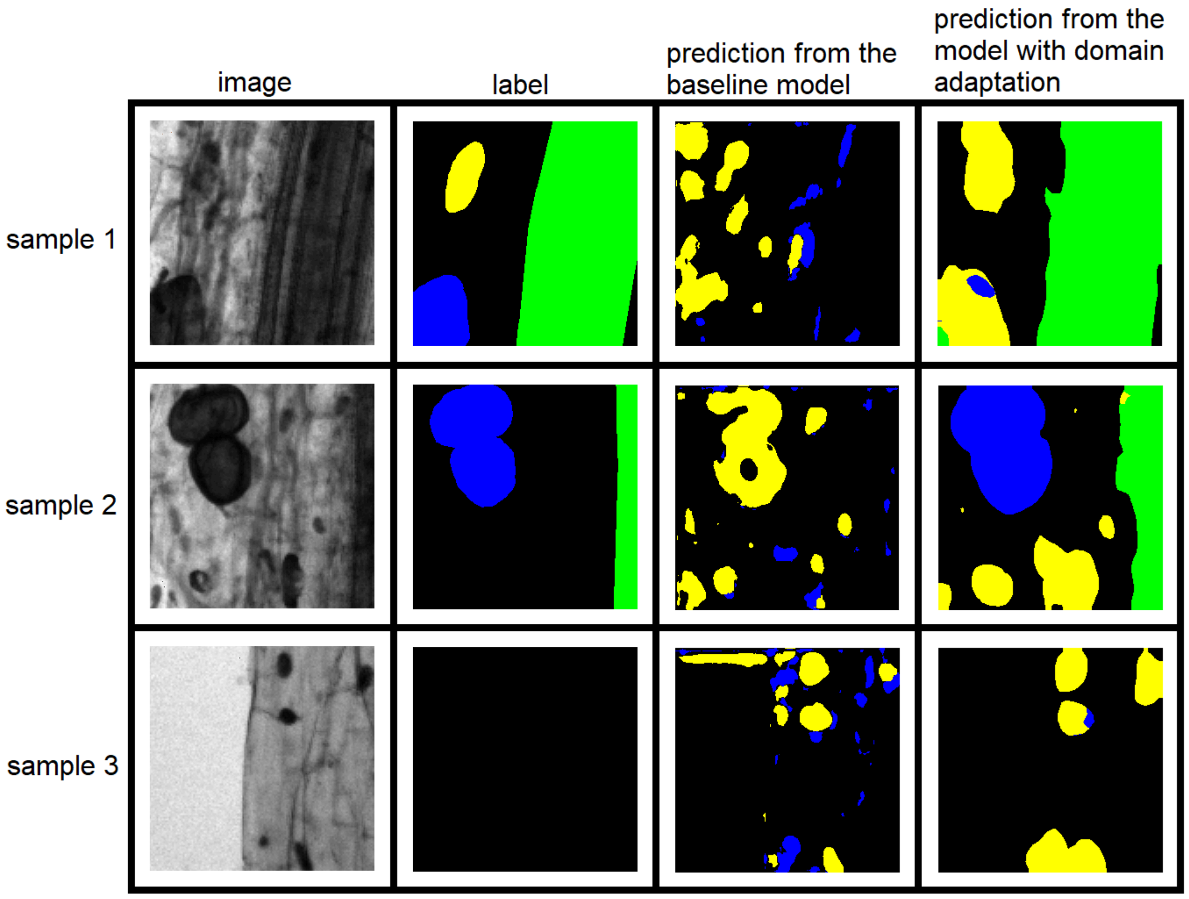
image
label
prediction from the baseline model
prediction from the model with domain adaptation
sample 1
sample 2
sample 3

Figure 3.20: Comparison of the two models. The three rows show the predictions of three light microscopy image patches. Column 3 is the prediction made by the baseline model. Column 4 is the predictions made by the model developed in this work, which is trained with domain adaptation technique. It should be noted that the baseline model does not include the root cortex class (green).

missing. The model treats each z-slice individually, so the prediction results vary depending on the z-slice. The set of predictions also reveals a problem with the model that hyphae, which should be counted as a background object, is sometimes recognized as root cortex or arbuscules.

When training the model, the normalization of the data is performed on each image patch, so the same should be done for the test data. This results in different predictions for the same object if it is captured in different image patches. Figure 3.22 show the predictions of the same arbuscule. It is located differently in three image patches. In row 1-2, the arbuscule is identified, while hyphae is misclassified as root cortex. In row 3, the whole area is identified as arbuscules, and it is not a successful prediction. If the normalization is performed on the complete light microscopy image, the prediction will be consistent. Although it is uncertain how this will affect the overall model performance, the experiment is worthwhile and can be done for future improvements.

Furthermore, a simple web application is designed to facilitate the use of the model. It allows uploading local light microscopy images for prediction. Any arbitrary region of the uploaded image can be selected, and a 256×256 image patch will be cropped based on the center point of the selected region. This is shown in Figure 3.23. The model makes prediction for the image patch. Both the image patch and its prediction are displayed on the page. In addition, a toggle function is added, which allows the displayed image to be switched between the image patch and the prediction, so it is easier to figure out what each pixel is classified as, since the prediction result and the input image patch are shown in the same location (see Figure 3.24).

3.5 Suggested improvements

Due to the time constraints in writing this thesis, the study of the factors affecting the training and the performance of the model is not conducted in depth. More experiments can be done in the future, and improvements in model performance are expected.

Hyperparameter tuning is an important step in training a model, as hyperparameter settings have a significant impact on the training results. Random search and grid search are two common methods for hyperparameter tuning. They can also be used in combination. For example, a course random search is first performed, followed by a finer grid search, and the grid search uses the range of values that yields the most promising results in the random search. In this work, trials can be performed with the learning rate of the modules being set differently in order to investigate how the learning rate affects the training. In addition, the learning rate scheduler adjusts the learning rate during training according to the selected method. Different learning rate schedulers may also be considered in future experiments. Loss functions for updating the modules are composed of many terms, and different weights are assigned to each term. These weights can be included in the set of hyperparameters which are tuned during the training process.

Modifications can also be made to the dataset and its preprocessing. A different data normalization other than the min-max normalization applied in this work can be considered,

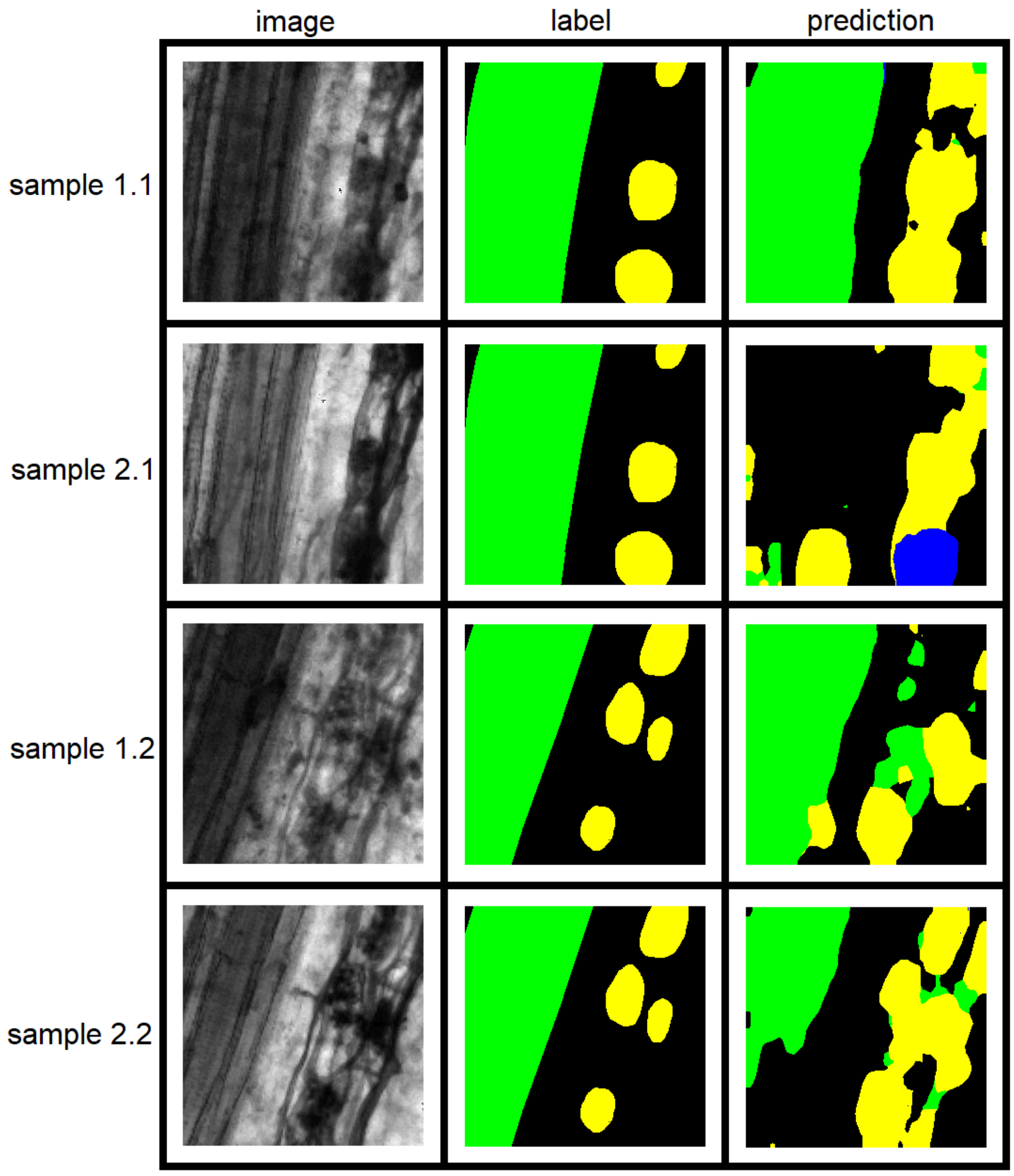

Figure 3.21: Comparison of the prediction results for different z-slices. Row 1-2 and row 3-4 are two examples showing the predictions for different z-slices at the same location. The image patches in row 1 and 3 are taken from one z-slice, and the image patches in row 2 and 4 are taken from the another z-slice.

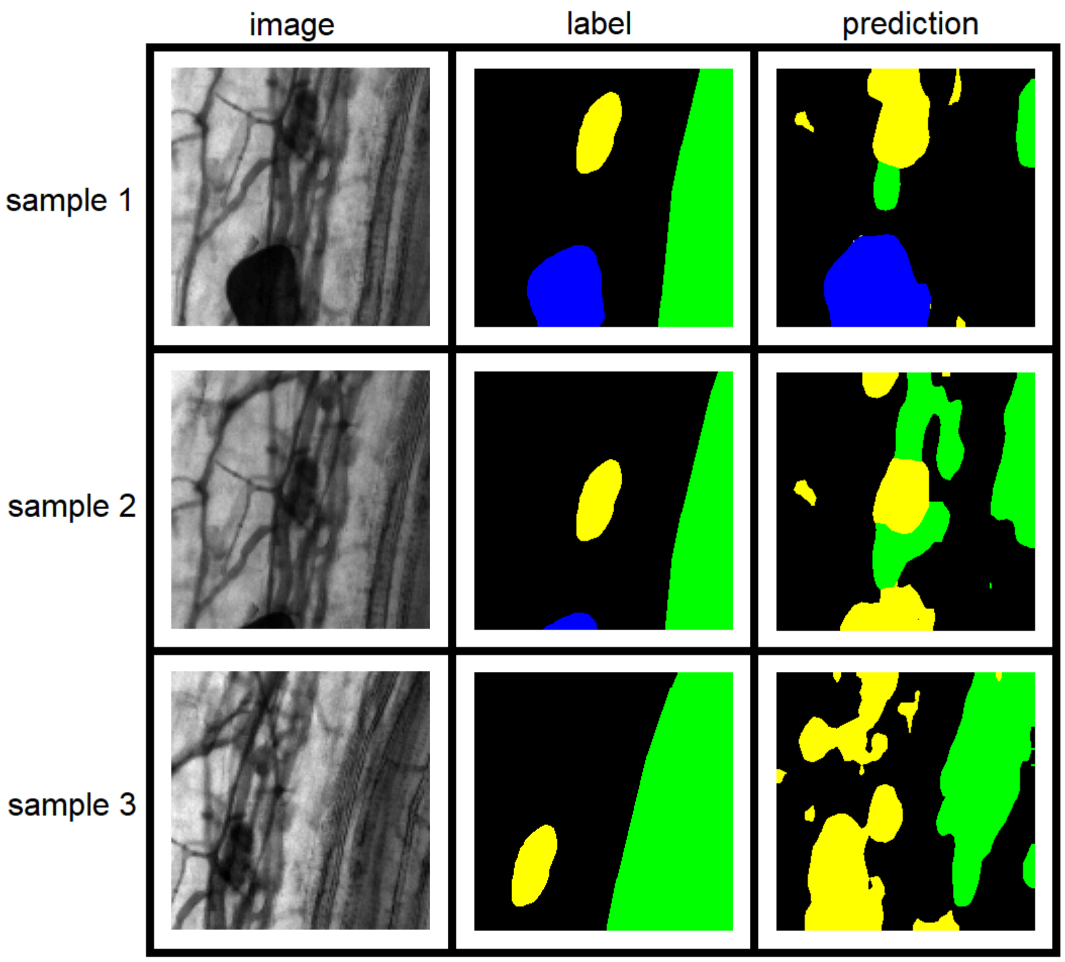

Figure 3.22: Comparison of the prediction results for different locations. The three image patches come from one light microscopy image. The arbuscule (yellow) in the patches are the same fungal object.

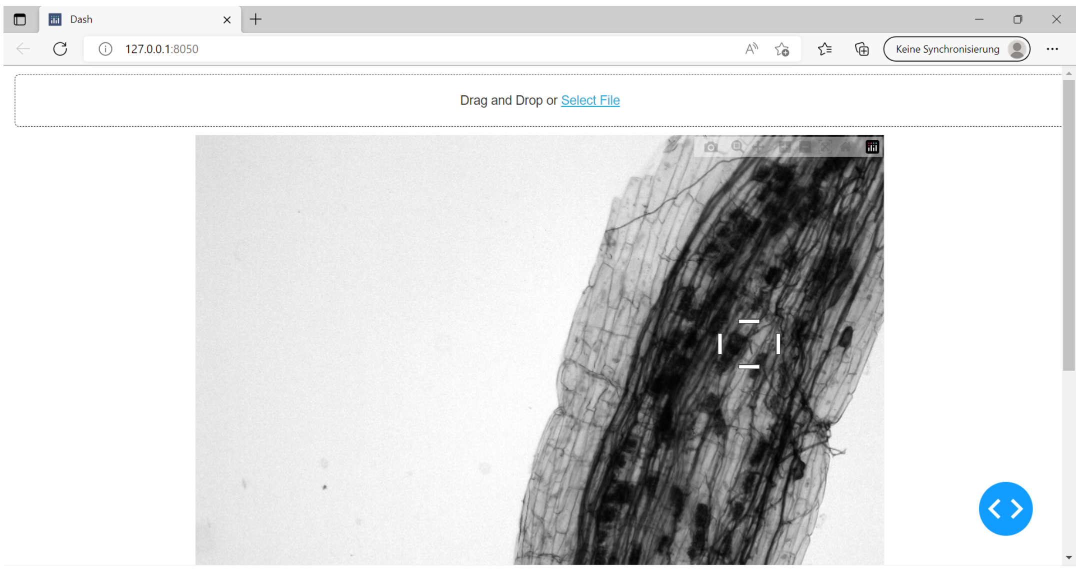

Figure 3.23: A preview of the web application. Local light microscopy images can be uploaded using the upload button. Regions of interest can be selected from this image.

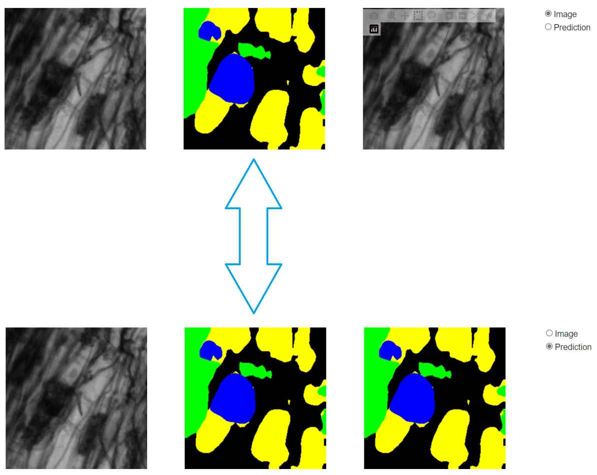

Figure 3.24: The prediction result displayed by the web application. Altogether three images are displayed. The first image is the selected image patch, the second image is the prediction made by the model, and the third image can be switched between the image patch and its prediction, depending on the selected option listed on the right.

such as the z-score normalization. The data normalization is now applied to image patches, but it is also possible to apply the normalization to the whole image before cropping. The synthetic images used for training comes from the same 3D model, which shows a root section colonized by a large number of arbuscules and vesicles. The lack of variety is a disadvantage because the dataset is biased. Therefore, the 3D model can be altered in order to obtain more different training data.

4 Conclusion

This work presents the approach for light microscopy image segmentation using a domain adaptation framework SIFA. A 3D model of a colonized root section developed with the 3D computer graphics software Blender and the light microscopy images provided by the Life Science department at TUM are the source of the data for training (Section 3.1). Labeled synthetic images are generated by the 3D model, while the real-world microscopy images are unlabeled. The data preprocessing steps include cropping the images into small patches, flipping and rotating the image patches as data augmentation and preforming a min-max normalization on the image patches (Section 3.3.1). The SIFA framework combines the domain adaptation and the semantic segmentation. In the end-to-end training, the synthetic images are converted to fake light microscopy images, which are then fed into the segmentation model. This enables the segmentation model to make predictions for the light microscopy images (Section 3.3.3). The implementation of the SIFA framework in this work is written in Python using the library PyTorch. Predictions and evaluation results are presented in Section 3.4. Improvements have been achieved in comparison to the baseline model, which is a U-Net trained with only the labeled synthetic images (Section 3.3.2). The predictions are generally reasonable, and they can assist the plant biologists in locating the fungal structures of interest. It can be concluded that the overall goal of this work, namely the development of a semantic segmentation model capable of predicting light microscopy images utilizing domain adaptation techniques, has been accomplished.

Compared with the baseline model U-Net, the semantic segmentation model in this work can produce better results. The U-Net is trained with synthetic images, and it has good performance if applied to synthetic images. However, the domain gap between the light microscopy images and the synthetic images is so large that the U-Net can no longer extract meaningful features. This results in the inability of the U-Net to make predictions for the real-world light microscopy images. The SIFA framework in this work attempts to address this problem from two aspects, namely appearance alignment and feature adaptation. The synthetic images are converted to light microscopy images before being fed into the segmentation model for training, and this is achieved by using GANs, where a generator generates fake light microscopy images based on the given synthetic images. Moreover, the encoder in the segmentation model is encouraged to extract domain-invariant features in order to further reduce the domain gap. Additional discrimination tasks are introduced, which force the encoder to extract aligned features from both domains so as to successfully elude the discriminators. While improvements in light microscopy images segmentation are indeed achieved in this work, flaws in the predictions are yet to be fixed. This is a difficult task in general, as the reliability, robustness and explainability of deep learning models is still an ongoing research, which is fundamental to how the network can be improved and generalized. Nevertheless, instinctive and qualitative explanations are possible, which can lead to methods for developing models with better performance.

There are a number of directions in which the work can be further improved and developed. Training data of higher quality can be acquired by extending the 3D model to a wider variety. In real-world light microscopy images, the root section is not necessarily colonized by AMF, while the current 3D model contains a large number of arbuscules and vesicles. Biased training data can lead to suboptimal predictions of the segmentation model, which makes it sensible to add synthetic images showing different forms of root to the dataset. Loss functions can be changed, for example, by assigning weights to different classes in order to balance the dataset. The fungal objects are the classes of interest, but they constitute only a small fraction of the entire dataset. A carefully balanced training dataset should improve the performance of the model. The light microscopy images are taken with a z-stack, which is a series of images taken at different focal-planes. These z-slices are highly correlated with each other, and may serve as an advanced data augmentation techniques. In this work, no distinction is made between wild-type and mutant-type arbuscules, and the 3D model only renders one type of arbuscules. This can be a focus for future work, as it is relevant for AMF-research. The segmentation model accepts an image patch and makes predictions on it. It would be helpful if an entire light microscopy image could be predicted, rather than merely one patch. Therefore, a pipeline can be set up to automatically crop the light microscopy image for semantic segmentation and resemble the predictions into a complete picture. In addition, the pipeline may also include the processing of a TIFF file, which is a file containing multiple images. The z-stack is stored in this form, which means that each z-slice should first be separated from the file, because the segmentation model only accepts one light microscopy image at a time. It would be convenient if the pipeline could produce predictions for all z-slices or a chosen z-slice.

Bibliography

[1] M. A. Ahanger, A. Hashem, E. F. Abd-Allah, and P. Ahmad. "Chapter 3 - Arbuscular Mycorrhiza in Crop Improvement under Environmental Stress". In: *Emerging Technologies and Management of Crop Stress Tolerance*. Ed. by P. Ahmad and S. Rasool. San Diego: Academic Press, 2014, pp. 69–95. ISBN: 978-0-12-800875-1. DOI: https://doi.org/10.1016/B978-0-12-800875-1.00003-X. URL: https://www.sciencedirect.com/science/article/pii/B978012800875100003X.

[2] P. Bonfante and A. Genre. "Mechanisms underlying beneficial plant-fungus interactions in mycorrhizal symbiosis". In: *Nature communications* 1 (July 2010), p. 48. DOI: 10.1038/ncomms1046.

[3] C. Chen, Q. Dou, H. Chen, J. Qin, and P. Heng. "Synergistic Image and Feature Adaptation: Towards Cross-Modality Domain Adaptation for Medical Image Segmentation". In: *CoRR* abs/1901.08211 (2019). arXiv: 1901.08211. URL: http://arxiv.org/abs/1901.08211.

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. *Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs*. 2014. DOI: 10.48550/ARXIV.1412.7062. URL: https://arxiv.org/abs/1412.7062.

[5] M. Chen, M. Arato, L. Borghi, E. Nouri, and D. Reinhardt. "Beneficial Services of Arbuscular Mycorrhizal Fungi – From Ecology to Application". In: *Frontiers in Plant Science* 9 (2018). ISSN: 1664-462X. DOI: 10.3389/fpls.2018.01270. URL: https://www.frontiersin.org/article/10.3389/fpls.2018.01270.

[6] J. H. Cho, U. Mall, K. Bala, and B. Hariharan. "PiCIE: Unsupervised Semantic Segmentation using Invariance and Equivariance in Clustering". In: *CoRR* abs/2103.17070 (2021). arXiv: 2103.17070. URL: https://arxiv.org/abs/2103.17070.

[7] J. Dai, K. He, and J. Sun. "Instance-Aware Semantic Segmentation via Multi-task Network Cascades". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 3150–3158. DOI: 10.1109/CVPR.2016.343.

[8] R. D. Finlay. "Ecological aspects of mycorrhizal symbiosis: with special emphasis on the functional diversity of interactions involving the extraradical mycelium". In: *Journal of Experimental Botany* 59.5 (Mar. 2008), pp. 1115–1126. ISSN: 0022-0957. DOI: 10.1093/jxb/ern059. eprint: https://academic.oup.com/jxb/article-pdf/59/5/1115/1440502/ern059.pdf. URL: https://doi.org/10.1093/jxb/ern059.

[9] Y. Ganin and V. Lempitsky. *Unsupervised Domain Adaptation by Backpropagation*. 2014. DOI: 10.48550/ARXIV.1409.7495. URL: https://arxiv.org/abs/1409.7495.

[10] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. "Domain-Adversarial Training of Neural Networks". In: (2015). DOI: `10.48550/ARXIV.1505.07818`. URL: `https://arxiv.org/abs/1505.07818`.

[11] M. Giovannetti. "Structure, Extent and Functional Significance of Belowground Arbuscular Mycorrhizal Networks". In: *Mycorrhiza: State of the Art, Genetics and Molecular Biology, Eco-Function, Biotechnology, Eco-Physiology, Structure and Systematics*. Ed. by A. Varma. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 59–72. ISBN: 978-3-540-78826-3. DOI: `10.1007/978-3-540-78826-3_3`. URL: `https://doi.org/10.1007/978-3-540-78826-3_3`.

[12] B. Gong, Y. Shi, F. Sha, and K. Grauman. "Geodesic flow kernel for unsupervised domain adaptation". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), pp. 2066–2073.

[13] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. *Generative Adversarial Networks*. 2014. DOI: `10.48550/ARXIV.1406.2661`. URL: `https://arxiv.org/abs/1406.2661`.

[14] M. Hamilton, Z. Zhang, B. Hariharan, N. Snavely, and W. T. Freeman. *Unsupervised Semantic Segmentation by Distilling Feature Correspondences*. 2022. DOI: `10.48550/ARXIV.2203.08414`. URL: `https://arxiv.org/abs/2203.08414`.

[15] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. "Mask R-CNN". In: *CoRR* abs/1703.06870 (2017). arXiv: `1703.06870`. URL: `http://arxiv.org/abs/1703.06870`.

[16] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: `1512.03385`. URL: `http://arxiv.org/abs/1512.03385`.

[17] X. He, R. S. Zemel, and D. Ray. "Learning and Incorporating Top-Down Cues in Image Segmentation". In: *Computer Vision – ECCV 2006*. Ed. by A. Leonardis, H. Bischof, and A. Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 338–351. ISBN: 978-3-540-33833-8.

[18] J. Hoffman, E. Tzeng, T. Park, J. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell. "CyCADA: Cycle-Consistent Adversarial Domain Adaptation". In: *CoRR* abs/1711.03213 (2017). arXiv: `1711.03213`. URL: `http://arxiv.org/abs/1711.03213`.

[19] J. Hoffman, D. Wang, F. Yu, and T. Darrell. "FCNs in the Wild: Pixel-level Adversarial and Constraint-based Adaptation". In: *CoRR* abs/1612.02649 (2016). arXiv: `1612.02649`. URL: `http://arxiv.org/abs/1612.02649`.

[20] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. "Image-to-Image Translation with Conditional Adversarial Networks". In: *CoRR* abs/1611.07004 (2016). arXiv: `1611.07004`. URL: `http://arxiv.org/abs/1611.07004`.

[21] C. N. Jacott, J. D. Murray, and C. J. Ridout. "Trade-Offs in Arbuscular Mycorrhizal Symbiosis: Disease Resistance, Growth Responses and Perspectives for Crop Breeding". In: *Agronomy* 7.4 (2017). ISSN: 2073-4395. DOI: `10.3390/agronomy7040075`. URL: `https://www.mdpi.com/2073-4395/7/4/75`.

[22] X. Ji, J. F. Henriques, and A. Vedaldi. "Invariant Information Distillation for Unsupervised Image Segmentation and Clustering". In: *CoRR* abs/1807.06653 (2018). arXiv: 1807.06653. URL: http://arxiv.org/abs/1807.06653.

[23] L. Karacan, Z. Akata, A. Erdem, and E. Erdem. "Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts". In: *CoRR* abs/1612.00215 (2016). arXiv: 1612.00215. URL: http://arxiv.org/abs/1612.00215.

[24] T. Karras, T. Aila, S. Laine, and J. Lehtinen. "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *CoRR* abs/1710.10196 (2017). arXiv: 1710.10196. URL: http://arxiv.org/abs/1710.10196.

[25] A. Kirillov, R. B. Girshick, K. He, and P. Dollár. "Panoptic Feature Pyramid Networks". In: *CoRR* abs/1901.02446 (2019). arXiv: 1901.02446. URL: http://arxiv.org/abs/1901.02446.

[26] A. Kirillov, K. He, R. B. Girshick, C. Rother, and P. Dollár. "Panoptic Segmentation". In: *CoRR* abs/1801.00868 (2018). arXiv: 1801.00868. URL: http://arxiv.org/abs/1801.00868.

[27] B. Kulis, K. Saenko, and T. Darrell. "What you saw is not what you get: Domain adaptation using asymmetric kernel transforms". In: *CVPR 2011* (2011), pp. 1785–1792.

[28] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". English (US). In: *Proceedings of the Institute of Radio Engineers* 86.11 (1998), pp. 2278–2323. ISSN: 0018-9219. DOI: 10.1109/5.726791.

[30] D. Liu, D. Zhang, Y. Song, F. Zhang, L. O'Donnell, H. Huang, M. Chen, and W. Cai. "Unsupervised Instance Segmentation in Microscopy Images via Panoptic Domain Adaptation and Task Re-weighting". In: *CoRR* abs/2005.02066 (2020). arXiv: 2005.02066. URL: https://arxiv.org/abs/2005.02066.

[31] M. Liu and O. Tuzel. "Coupled Generative Adversarial Networks". In: *CoRR* abs/1606.07536 (2016). arXiv: 1606.07536. URL: http://arxiv.org/abs/1606.07536.

[32] J. Long, E. Shelhamer, and T. Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *CoRR* abs/1411.4038 (2014). arXiv: 1411.4038. URL: http://arxiv.org/abs/1411.4038.

[33] L. H. Luginbuehl and G. E. Oldroyd. "Understanding the Arbuscule at the Heart of Endomycorrhizal Symbioses in Plants". In: *Current Biology* 27.17 (2017), R952–R963. ISSN: 0960-9822. DOI: https://doi.org/10.1016/j.cub.2017.06.042. URL: https://www.sciencedirect.com/science/article/pii/S0960982217307790.

[34] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, and Z. Wang. "Multi-class Generative Adversarial Networks with the L2 Loss Function". In: *CoRR* abs/1611.04076 (2016). arXiv: 1611.04076. URL: http://arxiv.org/abs/1611.04076.

[35] M. Mirza and S. Osindero. "Conditional Generative Adversarial Nets". In: *ArXiv* abs/1411.1784 (2014).

[36] Z. Murez, S. Kolouri, D. J. Kriegman, R. Ramamoorthi, and K. Kim. "Image to Image Translation for Domain Adaptation". In: *CoRR* abs/1712.00479 (2017). arXiv: 1712.00479. URL: http://arxiv.org/abs/1712.00479.

[37] H. Noh, S. Hong, and B. Han. "Learning Deconvolution Network for Semantic Segmentation". In: *CoRR* abs/1505.04366 (2015). arXiv: 1505.04366. URL: http://arxiv.org/abs/1505.04366.

[38] S. E. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. "Generative Adversarial Text to Image Synthesis". In: *CoRR* abs/1605.05396 (2016). arXiv: 1605.05396. URL: http://arxiv.org/abs/1605.05396.

[39] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[40] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. "Adapting Visual Category Models to New Domains". In: *Computer Vision – ECCV 2010*. Ed. by K. Daniilidis, P. Maragos, and N. Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 213–226. ISBN: 978-3-642-15561-1.

[41] J. Shotton, J. Winn, C. Rother, and A. Criminisi. *TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context*. 2007.

[42] A. Stamatouli. "Deep Generative Modelling of Microscopy Image Data". Masterarbeit. Technical University of Munich, Mar. 2022.

[43] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. "Simultaneous Deep Transfer Across Domains and Tasks". In: *CoRR* abs/1510.02192 (2015). arXiv: 1510.02192. URL: http://arxiv.org/abs/1510.02192.

[44] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. "Adversarial Discriminative Domain Adaptation". In: *CoRR* abs/1702.05464 (2017). arXiv: 1702.05464. URL: http://arxiv.org/abs/1702.05464.

[45] H. Vierheilig, A. P. Coughlan, U. P. Wyss, and Y. Piché. "Ink and Vinegar, a Simple Staining Technique for Arbuscular-Mycorrhizal Fungi". In: *Applied and Environmental Microbiology* 64 (1998), pp. 5004–5007.

[46] J. Watter. "Light Microscopy Image Analysis using Neural Networks". Masterarbeit. Technical University of Munich, Apr. 2021.

[47] W. Xian, P. Sangkloy, V. Agrawal, A. Raj, J. Lu, C. Fang, F. Yu, and J. Hays. "TextureGAN: Controlling Deep Image Synthesis with Texture Patches". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8456–8465. DOI: 10.1109/CVPR.2018.00882.

[48] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. "Conditional Random Fields as Recurrent Neural Networks". In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 1529–1537.

[49] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. 2017.