Domain adaptation of microscope cell images

Amanda Aarnio, Antti Huttunen, Juhani Kolehmainen, Anni-Mari Niskanen, Lauri Pohjola, Tuomas Poutanen

School of Science

Espoo 27.5.2022

Supervisor

Prof. Tomi Laurila

Advisor

Ossi Arasalo, Arttu Lehtonen

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract

Author Amanda Aarnio, Antti Huttunen, Juhani Kolehmainen, Anni-Mari Niskanen, Lauri Pohjola, Tuomas Poutanen

Title Domain adaptation of microscope cell images

Degree programme

Major **Code of major**

Teacher in charge Prof. Tomi Laurila

Advisor Ossi Arasalo, Arttu Lehtonen

Date 27.5.2022 **Number of pages** 30+6 **Language** English

Abstract

This report is the outcome of a project where the task was to automatically label cells in unlabelled cell images. This task was tackled with two machine learning (ML) based solutions, semantic segmentation and domain adaptation (DA). Semantic segmentation was utilised for the labeling, i.e. pixel-wise classification, of the images itself. Due to the lack of labelled training data, domain adaptation was utilised as well: with its help, labelled cell images from a different but related domain could be utilised to train the model.

The U-Net model was chosen for the semantic segmentation task and the backpropagation and CycleGAN models for domain adaptation. In addition, U-Net was trained without any domain adaptation model, but simply with slightly augmented data aiming to resemble the unlabelled target data. The models were trained with various datasets and their accuracies evaluated with dice score. The results showed that the plain U-Net trained with augmented data performed the best, backpropagation performing slightly worse and CycleGAN performing the worst of all three. In addition, the scores of all three approaches showed room for improvement.

Keywords Unsupervised domain adaptation, transfer learning, semantic segmentation, computer vision

Contents

1 Introduction

Automatic identification of cells in biomedical images without a trained human eye has many applications in the modern world. When successful, it might not only save time of medical professionals, but potentially also exceed them in terms of accuracy. Nonetheless, this task and possible solutions to it have not been widely studied. The objective of this report is to introduce, explore and compare various domain adaptation methods in this task of identifying cells in microscopic cell images by machine learning, and specifically semantic segmentation. If successful, the results of this report could be utilised in a variety of applications where cell detection is required or beneficial. Additionally, this report aims to add to the knowledge of how the nature of biological data affects the performance and choice of the domain adaptation models applied to it.

Semantic segmentation is a fundamental but challenging problem in the field of computer vision. It means the task of pixel-wise classification. That is, the aim of a semantic segmentation model is to assign a label to each pixel of an image. Semantic segmentation algorithms have a wide variety of applications, such as self-driving vehicles, detecting medical instruments in operations and computer-aided diagnosis [1, 2]. While a large number of semantic segmentation methods, such as partial differential equation and random forest based methods, were proposed before the deep learning era, the rapid emergence and development of deep learning has greatly advanced research on semantic segmentation [1]. Nowadays deep learning methods are widely considered the best performing methods for semantic segmentation tasks.

Semantic segmentation models require a huge amount of labelled data in order to perform accurately [3]. However, labelling training data in order to train a new semantic segmentation model for a dataset is extremely expensive and time-consuming. Furthermore, a sufficient amount of training data may not be available in many scenarios. Domain adaptation (DA), a case of transfer learning, offers a solution to this issue. It utilises the fact that a large amount of labelled data often exists in another domain somehow related to the unlabelled domain of interest. In domain adaptation, a model is trained in the label-rich domain, called the source domain, and then applied in another related domain, called the target domain. This is accomplished by diminishing the difference between the two domains by various strategies and learning domain invariant features. In this report, our target domain data, the data where we want to identify cells, is unlabelled, and we utilise labelled cell data available on the Internet as the source domain.

Semantic segmentation and domain adaptation are widely useful in biomedical tasks. A common task is automated cell segmentation because large-scale annotated cell image datasets are scarce and expensive to create. Various implementations of cell segmentation with domain adaptation exist, for example the CellSegUDA model [4] and the DASGAN model [5]. The nature of biological data affects the

performance and creates new constraints for domain adaptation models applied on biological data, however, these constraints are not yet completely understood.

2 Background

In this section, we present the outline of the problem. The task falls into the category of pixel-wise classification of images, namely semantic segmentation. The main goal is to transfer the models performance from one dataset to another, a problem known as domain adaptation. Various strategies have been recognized and attempted for domain adaptation, and a short overview of these strategies is presented in this chapter.

2.1 Semantic segmentation

Computer vision is the task of computationally understanding something from visual data such as images and videos. Numerous studies have shown that modern Deep Learning methods perform well in computer vision tasks. That being the case, computer vision allows the automation of the tasks that are usually done by human vision.

Figure 1 represents some of the most common computer vision tasks. One of these is semantic segmentation that is used to pixel-wise label images. From Figure 1 it can be seen that unlike object detection that can detect a dog and a cat in an image, semantic segmentation shows a label grass, cat, tree or sky for each pixel. Therefore, semantic segmentation does not differentiate objects and only observes pixels [1].

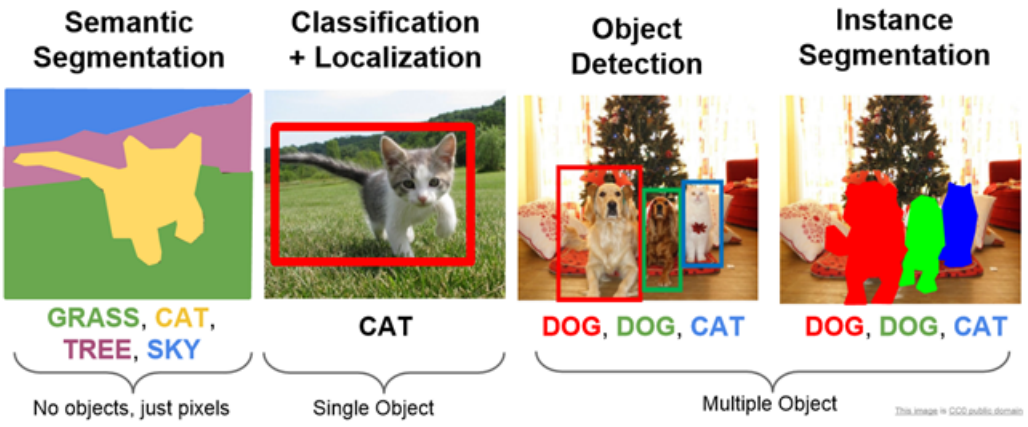


Figure 1: Common computer vision tasks

Cell segmenting is a difficult task due to the significant variation in cytoplasm shape and size. In addition, cells that are in contact with each other can have unclear boundaries. This complicates the problem even further. Segmentation algorithms can also be distracted by colour variations in the background of the cell culture because these colour variations can be misinterpreted as cells [6].

Some semantic segmentation methods are used to identify cells of different classes using multiple output channels. For example, one channel can mark the cell nuclei and one the cytoplasm. However, when only the whole cells are segmented, one channel grey-scale models are used [6]. A common problem in medical image segmentation is the lack of data. Acquiring data can be time consuming and/or expensive and so affects the model selection.

Next, two commonly used semantic segmentation methods, Fully Convolutional and Encoder-Decoder Networks are described. Fully Convolutional Networks and Encoder-Decoder models are Deep Learning (DL) approaches. DL models differ from simpler machine learning models by using several functions to find approximation which describes the data. DL functions are often called layers, and the layers are connected too each other. In a Feed-Forward Network (FNN), the layers are hierarchically connected and always feed information to the next layer. Additionally, there may be loops and skips in the architecture that make the structure nonlinear. Deep neural networks contain multiple layers. In addition, DL models use non-linear activation functions (e.g., Tanh, ReLU etc.) that allows the model find non-linear dependencies [7].

Fully Convolutional Networks (FCNs) are often used in classification of whole images but can be modified to be used in semantic segmentation. Normally, FCNs return a classification output but for semantic segmentation, the last layers must be modified to return an image with pixel-wise predictions (Figure 2). This can be done by changing fully connected layers to convolutional layers and modifying the last layer so that it will match the input image shape. FCNs are used because they can efficiently learn to make dense predictions in pixel-wise tasks like semantic segmentation [8].

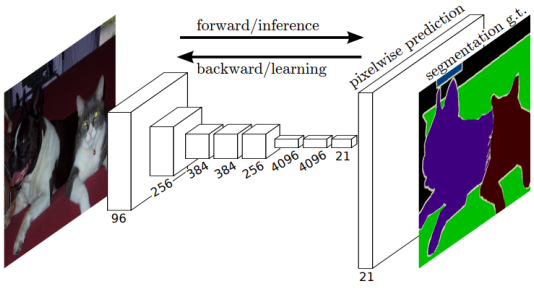


Figure 2: Fully Convolutional Network for semantic segmentation [8]

One of the most commonly used semantic segmentation approaches is the encoder-decoder architecture (Figure 3). Encoder-decoder networks are commonly used because the encoder captures semantic information (labels) and the decoder recovers spatial information (pixel locations) [9].

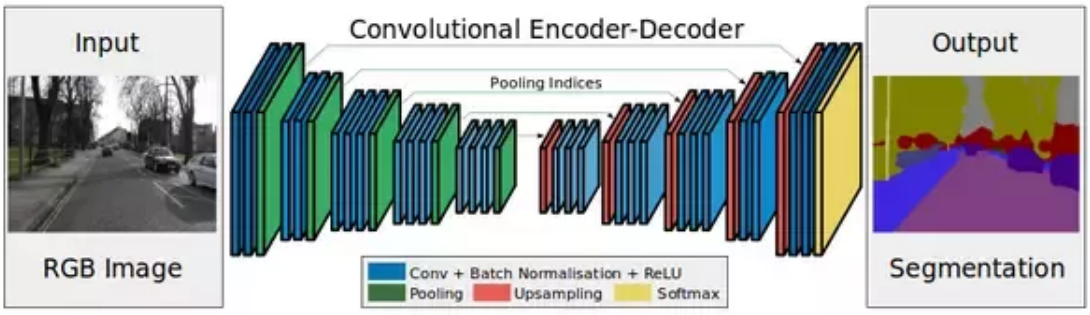


Figure 3: The left side of the figure represents the encoder that produces a lower-dimensional feature representation of the original image to capture the image's context. Meanwhile, the right side represents the decoder, which interprets the feature representation created by the encoder and makes a prediction. Finally, the decoder returns the final segmentation mask as an output [9]

2.2 Domain adaptation

This section presents an overview of domain adaptation and related techniques. The chapter will address Domain adaptation (DA) in context of deep neural networks for semantic segmentation. Domain adaptation and its related terms are presented at the beginning of the chapter. The chapter will then focus mostly on a version of domain adaptation called unsupervised domain adaptation (UDA) where little or no labelled target data is available during model training. Three different levels of domain adaptation, input, output and feature levels, will be covered after UDA. Finally, different DA strategies will be introduced.

2.2.1 Domain adaptation theory

Most machine learning models assume that training and target datasets share a same underlying distribution from which the data is obtained [10]. However, in many cases trained models are used with data from a different domain that has its own distribution compared to the training data. The difference between these distributions, i.e. source and target domain, is called domain shift [3]. Domain shift degrades most of the traditional machine learning models, including neural networks, and prevents the generalization of the models between different domains [3, 11].

The described problem can be solved by using domain adaptation. DA is essentially a process that aims to overcome the domain shift. In domain adaption, a model is trained with one or more source domains that is then used in different but related target domain [12]. The domains have different distributions even though the high-level image content is shared. For example, in FMRI image analysis distributions of brain images are highly dependent of a scanner, which can be seen in Figure 4. Even though the high level information is invariant of the scanner type, the differences in image distributions, i.e. domains, affects the model performance. The overall effectiveness of domain adaptation is highly influenced by how similar the features of source and target domains are [3]. DA is not possible for distributions that do not share any common features.

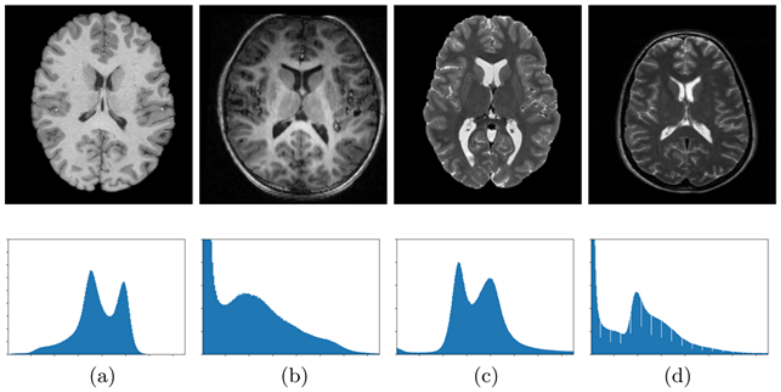


Figure 4: Images of a brain and corresponding histograms of normalized T1w (a,b) and T2w (c,d) MRIs taken using different scanners. Distributions (domains) differ significantly even though the high level information remains the same. [13]

Domain adaptation can be seen as a part of transfer learning [10]. According to Farahani et al. [11], transfer learning is related to machine learning problems where tasks and/or domains can change between source and target settings. In domain adaptation, only domains change and the task remains the same. Domain adaptation can be divided into different categories based on category gap, differences in label spaces, and domain gap, differences in marginal distributions. Category cap includes four main categories that are open set, closed set, partial, and universal domain adaptation. Domain gap on the other hand includes three main categories that are covariate shift, prior shift, and concept shift [11].

Domain adaptation includes several approaches that can be broadly divided into methods that use either shallow or deep architectures [3]. This report focuses on deep domain adaptation techniques on semantic segmentation which use deep architectures, i.e. deep neural networks. Most commonly deep domain adaptation techniques include convolutional, adversarial based, or autoencoder networks that

are used to reduce the domain shift [11]. Next section will focus on a particular version of domain adaptation called unsupervised domain adaptation.

2.2.2 Unsupervised domain adaptation

Unsupervised domain adaptation is a version of domain adaptation where no target labelled target data, or only a small set, is available [11]. The simplest form of UDA is to train a model using only labelled data from source domain. However, this leads to poor predictive performance in practice [3]. Because of this most UDA methods use unlabelled target data in the training process together with labelled source data. This means that there is no direct supervision on the target domain data, unlike in supervised machine learning. Various different unsupervised learning methods can be used but the main idea is to achieve good accuracy on both source and target domains.

In many real world problems, labelled target data is rarely available. Obtaining labelled data is a laborious and costly task, especially for semantic segmentation where each pixel has to be labelled [3]. UDA can be used with target raw data that is much simpler and easier to get, and one can still achieve good performance for the trained model. Obtaining source data is not usually a problem since we can use already annotated datasets e.g. from public databases.

2.2.3 Domain adaptation levels

Domain adaptation can take place on three different levels of the process: input-level, feature-level and output-level. The levels are illustrated in Figure 5.

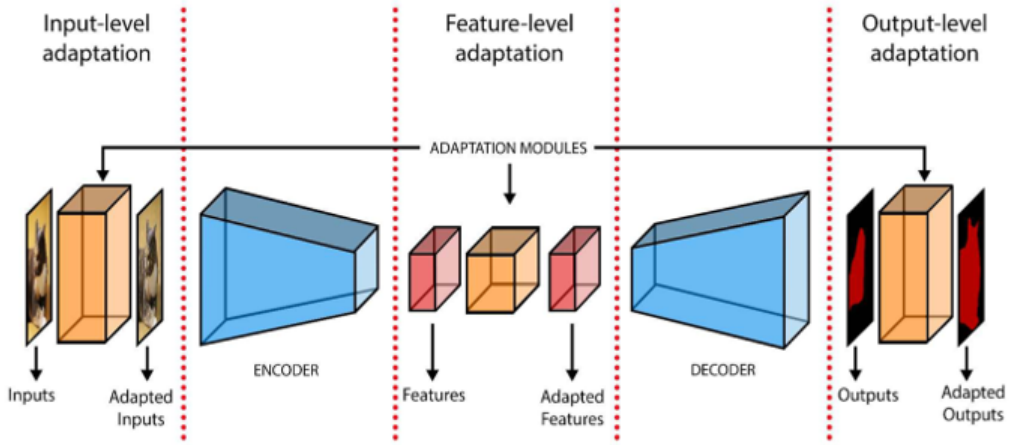


Figure 5: Domain adaptation levels in semantic segmentation model with encoder and decoder [3]

Input-level domain adaptation is the process of manipulating the training data. Instead of altering the model to handle the domain gap, the training dataset is modified to be more similar to the target dataset. Input-level domain adaptation includes simple methods adding noise to the training images and re-weighting. Re-weighting is a technique where training images with most resemblance to target domain gain more weight in training than other images. More advanced methods include approaches such as finding a function that maps source and / or target images to another, domain invariant space. In input-level domain adaptation this would be done before passing data to the model [3].

Feature-level domain adaptation tries to bridge over the domain gap by forcing the underlying feature representation to be as domain-invariant as possible. It is performed to the features created by the encoder, or more generally to the feature extractor part of the semantic segmentation model. In the context of deep domain adaptation this often means use of domain discriminators. The domain discriminator in the model tries to detect from which domain the given input is. Then by maximizing the loss of the domain discriminator as part of training, the encoder should find domain-invariant feature representations. One such solution is presented by Ganin et al. [14].

Output-level domain adaptation techniques try to map the semantic mask generated by the decoder across the domains. However, output-level domain adaptation is not widely used. It performs best in combination with feature-level adaptation or as part of full-level approaches that try to reduce domain gap at all levels of the semantic segmentation model. One possible method is to use the generative adversarial approach to the semantic masks as segmentation output from different domains often include a significant amount of similarities. This model impacts the feature-level representation through the backpropagation process during training [15].

2.2.4 Domain Adaptation strategies

According to Toldo et al., unsupervised domain adaptation strategies can be divided into seven main categories that are self-training, multi-task learning, curriculum learning, classifier discrepancy based, generative based, and domain adversarial based approaches. In this chapter, the descriptions of each strategy are based on the work of Toldo et al. [3].

Self-training approaches generate their own pseudo-labels during the training process. These pseudo-labels from unlabelled data are used to train the classifier in subsequent iterations. The main problem of self-training approaches is that the lack of external supervision can lead to over-confident predictions due to incorrect early labelling. Furthermore, self-learning can enforce prediction mistakes in each iteration through propagation mechanism that will progressively deteriorate

accuracy of the predictor. Several different variations of self-training have been tested, including offline techniques, self-training with weighted pseudo-labels, self-training with adversarial discriminative module, and self-training using prediction ensembling. Zou et al. created the first UDA solution for self-training [16]. By following what they call an "easy-to-hard" scheme, their algorithm transfers labels from source domain to target domain according to class balance at a hyper-parameterized pace. Their process is visualized in Figure 6.

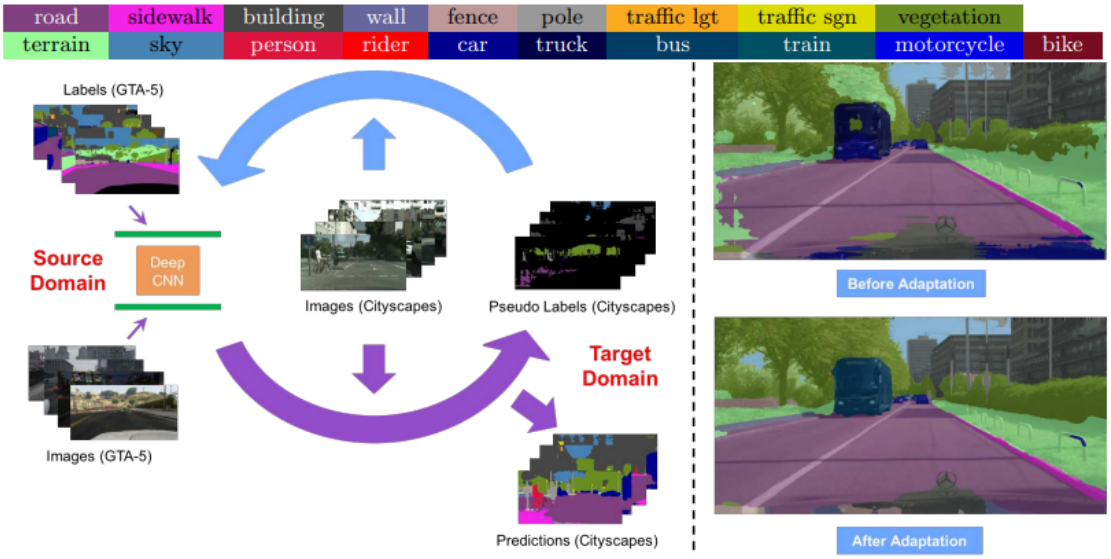


Figure 6: The self-training process of Zou et al. [16]

Multi-Task learning includes all models that exploit additional data from the source domain to improve the models performance in the target domain. In practice, the model solves many problems simultaneously and uses some logic to fuse results to boost performance. The additional data is called privileged information (PI), see Figure 7. For example, Watanabe et al. use the boundary information and depth encodings of source images to improve the performance of a semantic segmentation model [17].

Curriculum learning is the process of learning more complex concepts based on the simpler ones. It shares a lot with self-training. Instead of inferring the pseudo-labels through the usual output process, the preliminary predictions are done based on alternative information on the target domain. In the domain adaptation approach, a model works first with tasks that are less sensitive to domain, such as the label distribution over the global image or landmark superpixels. After that, the model is trained to find the target domain by following the properties found in the first step. Curriculum learning should result in faster training of the model and guide to better parameter space [18].

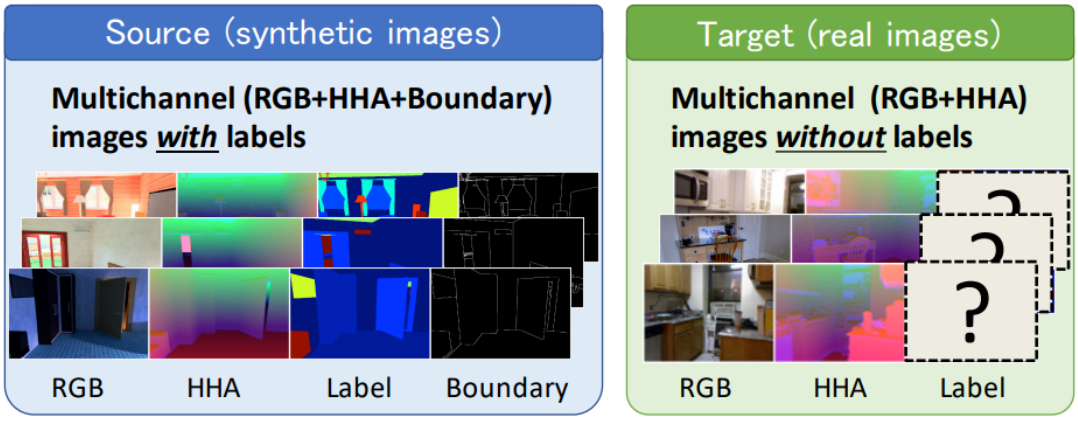


Figure 7: An example of privileged information available in source domain [17]

Classifier discrepancy utilizes the disagreement between the predictions of two classifiers. In this approach, two distinct classifiers are connected to a generator. The generator feeds it's features into both classifiers and the classifiers are trained to maximize their predictive discrepancy. The model is trained in three steps. First, maximize the discrepancy of the classifiers with source data. Then, minimize the discrepancy between the classifiers using target data. Finally, train a generator to generate features that maximize discrepancy on target data. The process is displayed in Figure 8. [19].

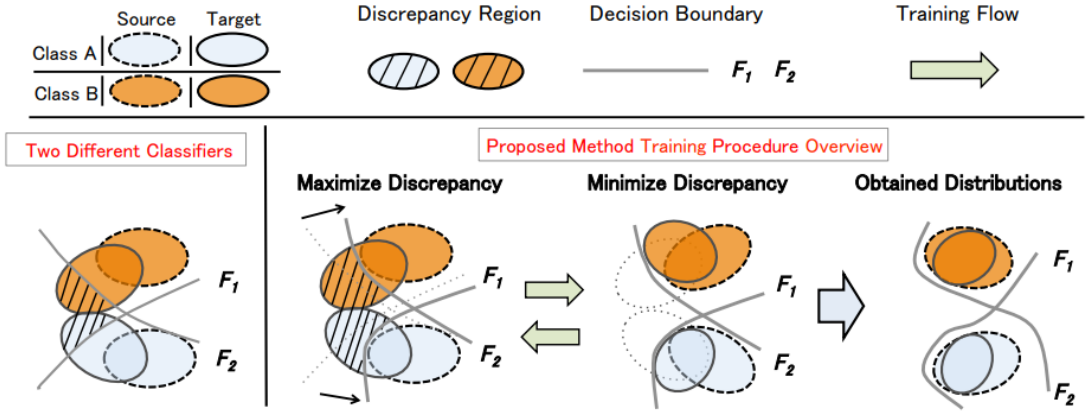


Figure 8: Illustration of the classifier discrepancy process [19]

Generative-based approaches aim to transfer visual attributes from one domain to another. This is done by learning a function for mapping images across domains. The idea is to generate new images using the images from either domain as

samples. The generated images should then be more similar to the images from the other domain. This achieves domain invariance in terms of visual appearance. A popular implementation is the CycleGAN framework. In CycleGAN, concurrent image translation is done both in source-to-target and target-to-source directions. The translation takes place at the input level, encouraging the projections to be inversions of each other. CycleGAN is presented in more detail in chapter 3.7. [20]

Domain adversarial discriminative (DAD) techniques exploit the adversarial property of Generative Adversarial Networks (GAN) by rewiring the real-fake discriminator into a source-target domain discriminator. The new source-target domain discriminator is then incorporated in the learning process, and the trained segmentation network aims to minimize segmentation loss simultaneously with the domain discriminator loss. DAD techniques can be roughly split into feature-discriminative and output-discriminative. Feature-discriminative solutions are more complex and aim to find the features that are most alike in source and target domains. Output-discriminative solutions on the other hand incorporate a discriminator with prediction maps from both domains. Hoffman et al. showed that presenting two new training objectives for training is enough to achieve domain adaptation in semantic segmentation. They propose minimizing the global distribution distances between target and source domain as well as transfering category specific statistics from source to target domain labels. [21]

Entropy minimization aims to answer the problem that models often produce over-confident segmentation masks for source data but are less confident with target data. Directly minimizing entropy on target domain should achieve similar confidence in both domains. Entropy minimization excels when the target domain is more complex than the source domain. Vu et al. showed that entropy minimization can be achieved both using an unsupervised entropy loss as well as adversarial training [22].

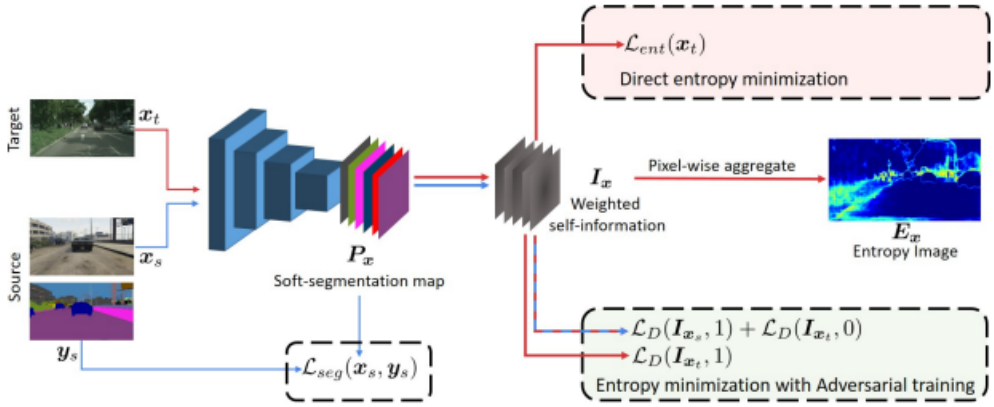


Figure 9: Overview of the entropy minimization approaches [22]

3 Data and methods

In this chapter, the available datasets and methods of Domain Adaptation and Semantic Segmentation are presented in detail. Two annotated sets, the LIVECell dataset and a synthetic Unity dataset are used to train the model to perform semantic segmentation on the unmasked target dataset. For semantic segmentation, we use the U-Net model. For domain adaptation, we use data augmentation and the CycleGAN architecture for input-level adaptation as well as one feature-level architecture abusing a domain classifier with backpropagation and a gradient-reversal layer.

3.1 Annotated training data

For training purposes, we use data available online as well as synthetic data. The LIVECell dataset is a versatile collection of 5239 2D cell culture images with segmentation masks [23]. Its main attraction is its size, as large annotated datasets are scarce. Additionally, the dataset contains various types of cells whereas the target dataset is limited to fibroblasts. This presents an unnecessary complication to the training process. Synthetic data was created with the Unity software and aimed to imitate target domain data while still looking simplistic. Synthetic images and masks for them can be generated in arbitrary amounts but the domain shift tends to remain quite large from synthetic to target data.

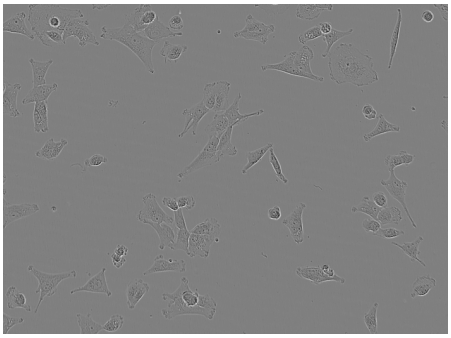
3.2 Target data

The aim of our model is to segment images from a 3D cell culture. The LIVECell images are from a 2D culture, which presents a new source of domain shift. There are small round magnetic particles in the 3D culture that are used for measuring the mechanical properties of the culture. Two different sizes of particles are used. The particles tend to get mislabeled as cells. Additionally, the cells are likely to behave slightly differently in a different environment as their shape and morphology changes, which may hinder the adaptation.

The dataset is a batch of 40 roughly 10 second videos from a 3D fibroblast culture. From the videos, five snapshots per second were captured, resulting in altogether 2224 images, see examples in Figure **??**. One of the videos was left out of training and used for evaluation and testing.

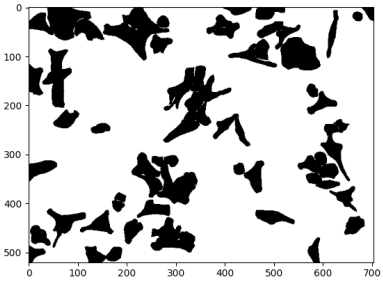
3.3 Augmenting data

Data augmentation is an input-level method of Domain Adaptation. To solve the issue of mislabeling magnetic particles as cells, the LIVECell dataset was augmented

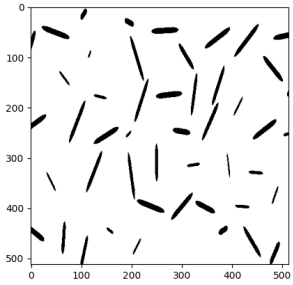


(a) LIVECell data

(b) Synthetic data using Unity software



(c) LIVECell mask

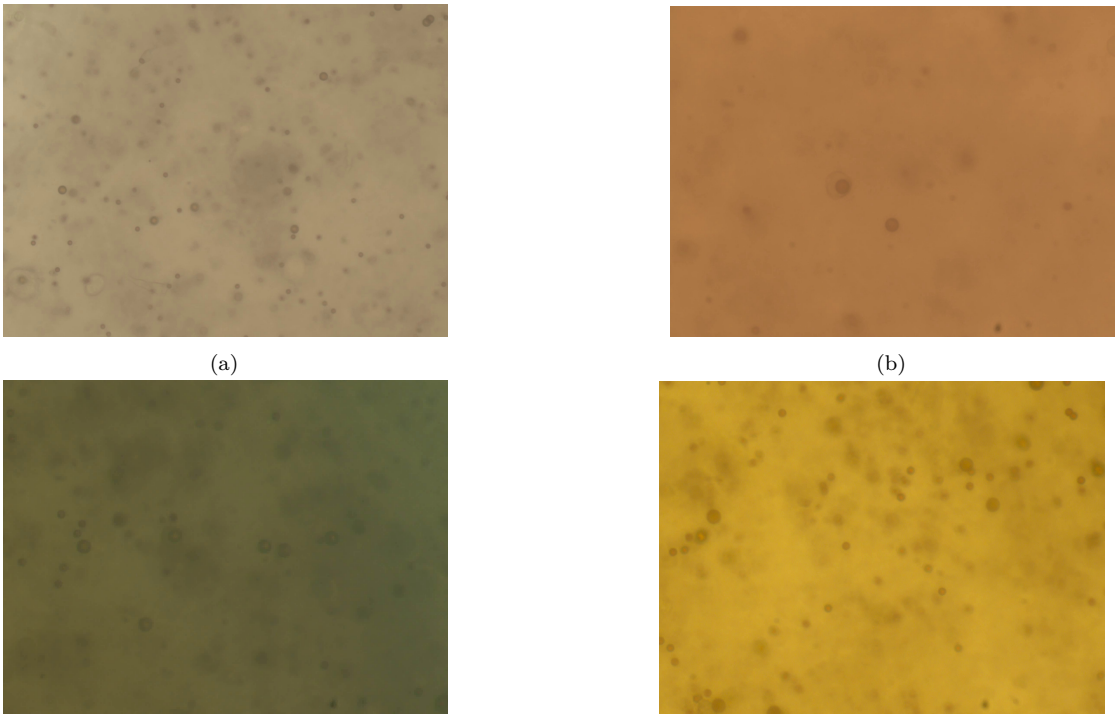


(d) Synthetic data mask

by adding small round black items. The goal of this approach is to reduce domain gap and teach the model to go around the magnetic particles while still learning to latch onto cells. Additionally, various image transformation methods from the Torchvision package such as adding Gaussian noise to the images were applied during training. The augmented images are shown in Figure 12.

3.4 Source data

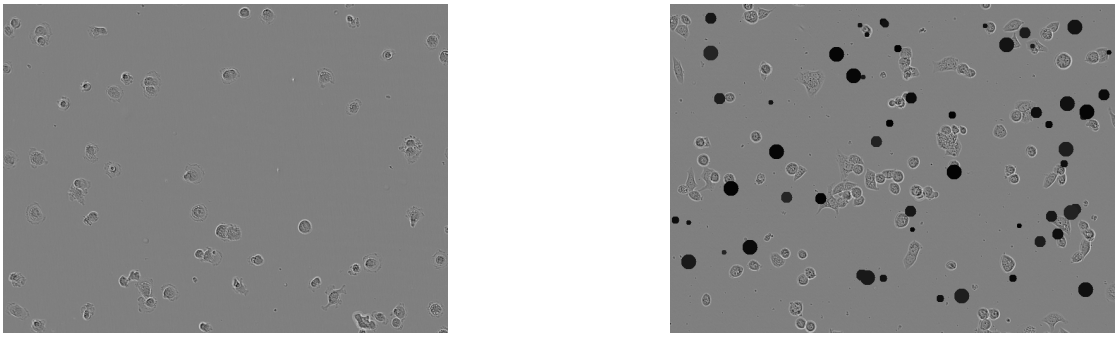
Three different source data sets were used to train the domain adaptation models and their performances were compared. The first data set consisted of pure LIVECell (Figure 12a) data and the second of augmented LIVECell data (Figure 12). The last data set was a combination of synthetic Unity (Figure 12b) and augmented LIVECell data.



(a) (b)

(c) (d)

Figure 11: Different target images. As it can be seen the colour of the images changes quite much from image (a) to image (d)



(a) (b)

Figure 12: An example image from the LIVECell dataset (a) augmented with magnetic particle like blobs (b)

3.5 Segmentation network

A common semantic segmentation architecture for medical imaging purposes is the U-Net model first proposed by Ronneberger et al. [24]. U-Net is a special kind of encoder-decoder CNN. The network feeds intermediate representations

from the encoder path to the decoder path to provide additional context to the feature representation. Ronneberger et al. recommend heavy data augmentation and transformations during training to amplify the networks performance when training data is few. The original architecture is presented in Figure 13.

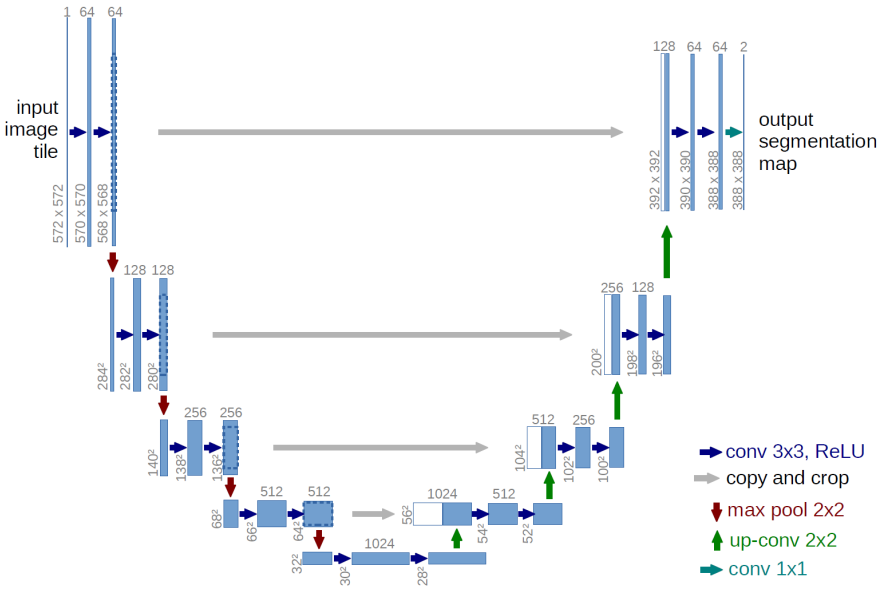


Figure 13: The U-Net architecture by Ronneberger et al. [24]

The architecture was developed to solve a semantic segmentation task with electron microscopy micrographs, but was already in the original paper shown to perform well in cell segmentation tasks, see Figure 14.

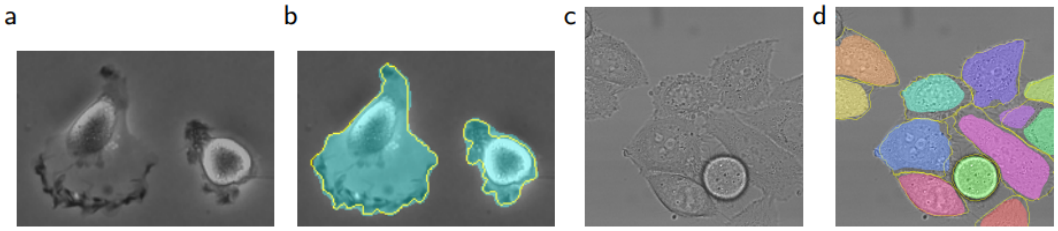


Figure 14: Semantic segmentation performance of the U-Net model [24]

3.6 Backpropagation

Backpropagation is an elementary algorithm in training neural networks [25]. Ganin et al. showed that it is possible to solve the domain adaptation task using ordinary backpropagation. This approach takes place on the feature level. The solution

involves adding a domain classifier G_d in between the deep feature extractor G_f and the deep label predictor G_y. The training objective is then to maximize domain classifier loss L_d simultaneously with minimizing feature extractor loss L_y and label predictor loss $\frac{\partial L_y}{\partial \theta_y}$. When the domain classifier loss is maximized, the feature extractor learns to extract domain invariant features. Combining these losses we have the functional

$$\begin{aligned} E(\theta_f, \theta_y, \theta_d) &= \sum_{i=1..N, d_i=0} L_y(G_y(G_f(x_i; \theta_f); \theta_y), y_i) \\ &\quad - \lambda \sum_i L_d(G_d(G_f(x_i; \theta_f); \theta_d), y_i) \\ &= \sum_{i=1..N, d_i=0} L_y^i(\theta_f, \theta_y) - \lambda \sum_{i=1..N} L_d^i(\theta_f, \theta_d), \end{aligned}$$

where λ is the control parameter, d_i is the domain label and θ are the collections of network parameters corresponding to each network, namely for the feature encoder θ_f, for label predictor θ_y and domain classifier θ_d. To find the point that maximizes loss of the domain discriminator and minimizes the rest, the optimization problem can be stated as finding the saddle point

$$(\theta_f, \theta_y) = arg \min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \theta_d) \tag{1}$$

$$\theta_d = arg \max_{\theta_d} E(\theta_f, \theta_y, \theta_d) \tag{2}$$

Simultaneously maximizing and minimizing losses is not possible with Stochastic Gradient Descent (SDG). However, the issue can be solved by introducing the Gradient Reversal Layer (GRL). The GRL acts as an identity transform in the forward pass, but multiplies the output by a control parameter $-\lambda$ in the backward pass. After augmenting the model with a GRL, also the domain discriminator loss can be minimized and SDG can solve the optimization problem. The proposed architecture can be seen in Figure 15 [14].

Adapting the approach of Ganin et al. to the U-Net model introduces a new challenge. In the U-Net architecture, the boundary between extracting features and predicting labels is blurred, as the expansive path is augmented with data from the contracting path on each level, see figure 13. The placement of the Domain Discriminator impacts where in terms of the network parameters, the domain invariance is maximized. As no trivial solution for the best placement was known to literature, all placements of the GRL and Domain Discriminator were tested starting from the bottom of U-Net and ending to the step prior to producing outputs. The different locations are illustrated in Figure 16.

The domain classifier proposed for classification task by Ganin et al. [14] was a multilayer perceptron (MLP) with two fully-connected hidden layers and output

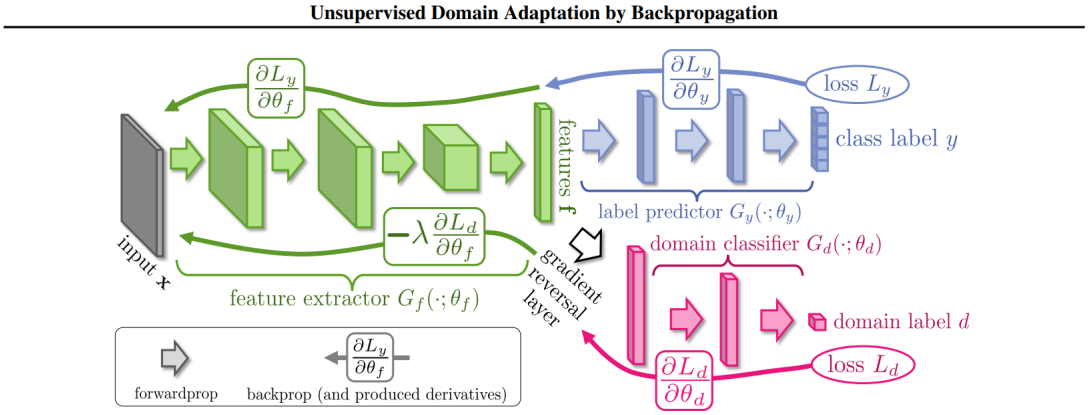


Figure 15: Neural network architecture for solving Domain Adaptation task using an adversarial Domain Discriminator Network and a Gradient Reversal Layer [14]

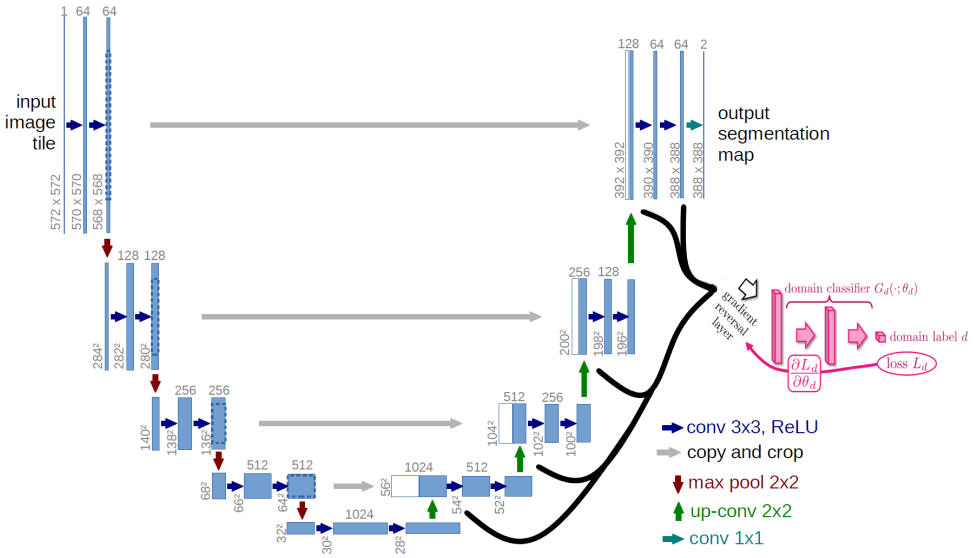


Figure 16: U-Net with domain discriminator

layer of a single node giving the domain label. In our semantic segmentation task that architecture need to be modified as U-Net model has different amount of features at each level. Therefore, an approach from Eliott et al. [26] was used, where domain classifier consists of a MLP with only one hidden layer and amount of down sampling double convolution layers of U-Net model depending on which placement was used for domain classifier. That way always the lowest resolution representation of the input image can be given to the MLP part of the domain classifier to ensure stable domain prediction.

3.7 CycleGAN

CycleGAN, originally proposed by Zhu et al. [20], is an image-to-image translation method. The main idea of the method is to train two mapping functions G and F which can translate images from some domain X to another domain Y and vice versa. If the two domains X and Y are considered to be source and target domains, CycleGAN can be viewed as an input-level domain adaptation method and therefore utilised in our task as well.

The CycleGAN [20] network consists of multiple deep neural networks which have different tasks. There are the two mappings, or generators, G and F which aim to translate images from one domain to another. Specifically, mapping $G : X \rightarrow Y$ should produce outputs $\hat{y} = G(x), x \in X$ indistinguishable from images $y \in Y$, and vice versa for mapping $F : Y \rightarrow X$. In order to evaluate how well the generators succeed in their task, two adversarial discriminators D_X and D_Y are introduced. D_X aims to distinguish between real images from domain X and images translated from domain Y to domain X. The task of D_Y is similar but for real images from domain Y and images translated from domain X to domain Y. The mappings and discriminators are illustrated in figure 17 (a).

Adversarial losses are applied to both mapping functions [20]. Denoting the data distributions with $x \sim p_{data}(x)$ and $y \sim p_{data}(y)$, the adversarial loss for mapping function G and its discriminator D_Y is:

$$\begin{aligned} \mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] \\ + \mathbb{E}_{x \sim p_{data}(x)}[\log 1 - D_Y(G(x))], \end{aligned} \tag{3}$$

where G aims to generate images $G(x)$ similar to images from domain Y, while D_Y tries to distinguish between real and translated samples $y \in Y$ and $G(x), x \in X$. Ideally, $D_Y(y)$ will output a value close to 1 when $y \in Y$, and otherwise will output a value close to 0. As can be seen from (3), the loss is minimised when the mapping G accurately generates images similar to images from domain Y, and the discriminator D_Y cannot tell the difference between real and translated samples. Therefore the mapping function successfully fools the discriminator, as is desired, and the single loss (3) can be utilised to train both the parameters of the mapping function and its discriminator. A similar adversarial loss is introduced for mapping function F and its discriminator D_X as well: $\mathcal{L}_{GAN}(F, D_X, Y, X)$.

It is theoretically possible to learn mappings G and F utilising only the adversarial losses introduced above [20]. However, without any additional constraints in the model, adversarial losses alone do not guarantee that the learned mappings will map inputs x_i to the desired outputs y_i. Because of this, another constraint called cycle consistency is introduced to the model. Cycle consistency ensures that images first translated from domain X to domain Y and then back to domain X are close to the original images, that is, $x \approx F(G(x))$, and similarly for images

from domain Y, $y \approx G(F(y))$. Cycle consistency is implemented to the model with a cycle consistency loss defined below. It is also illustrated in figure 17 (b-c).

$$\begin{aligned}\mathcal{L}_{cyc}(G, F) = &\mathbb{E}_{x\sim p_{data}(x)}[\|F(G(x)) - x\|_1] \\ &+ \mathbb{E}_{y\sim p_{data}(y)}[\|G(F(y)) - y\|_1]\end{aligned} \tag{4}$$

Therefore the full loss function becomes:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = &\mathcal{L}_{GAN}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{GAN}(F, D_X, Y, X) \\ &+ \lambda\mathcal{L}_{cyc}(G, F),\end{aligned} \tag{5}$$

where λ controls the relative importance between the adversarial and cycle consistency losses.

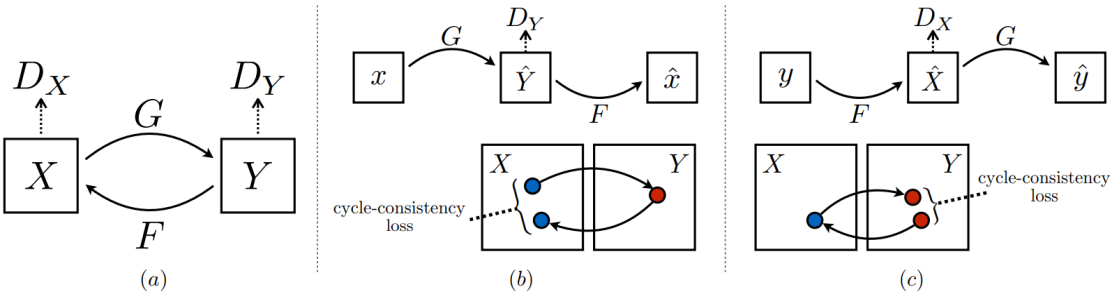


Figure 17: (a) The CycleGAN model [20] consists of two mapping functions $G : X \to Y$ and $F : Y \to X$ and two discriminators D_Y and D_X. D_Y encourages G to translate images from domain X to outputs indistinguishable from images of domain Y, and vice versa for D_X and F. In order to ensure that the translations are as desired and meaningful, cycle consistency losses were added to the model. They ensure that (b) $x \approx F(G(x))$, and (c) $y \approx G(F(y))$.

Our two CycleGAN generators G and F had the following architecture, consisting of an encoder, transformer, and decoder. The encoder consisted of three 2D convolutional layers with 64, 128, and 256 features and kernels of sizes 7×7, 3×3, and 3×3, respectively. The transformer consisted of three convolutional residual blocks, each of which had two 2D convolutional layers with 256 features and kernel size 3×3. The decoder had three 2D transposed convolutional layers with 512, 256, and 1 features and kernels of sizes 3×3, 3×3, and 7×7, respectively. In between these layers, pixel shuffle layers were utilised, and the Tanh non-linearity was applied at the very end. In addition to convolutional layers, the encoder, transformer, and decoder utilised instance normalisation, 2D reflection padding, and the ReLU non-linearity in various places.

Our CycleGAN model had two discriminators, D_X and D_Y, which both consisted of five 2D convolutional layers with kernels of size 4×4 and 64, 128, 256, 512, and 1 features, respectively. The leaky ReLU non-linearity and instance normalisations were also applied in multiple places.

4 Results

In this chapter we will present the overall performance of our models. In addition, the chapter goes through training parameters and possible adjustments to three training datasets shown in chapter 3.4. Dice score was chosen for measuring image similarity and evaluating predictions. Dice score which for two sets X and Y is defined as follows:

$$Dice = \frac{2|X \cap Y|}{|X| + |Y|} \tag{6}$$

Therefore, Dice score is a number between zero and one where higher value represents more similarity.

The evaluation scores are obtained by evaluating 15 manually annotated target images that are taken from one video. Then the target images of resolution 2064 x 1544 are divided into twelve 512 x 512 smaller images that cover nearly the full image. As network models take input of a 256x256 image, cropped images are resized and Dice score calculated for each mask. The performance of each model is reported as the means of these Dice scores.

The performance of each model was tested using different data. Firstly, the LIVECell dataset is the baseline labeled model. Secondly, the tables contain the AugLIVECell row, in which the LIVECell dataset was augmented with magnetic-particle-like blobs. Thirdly, we used a combination of augmented LIVECell and synthetic data.

The evaluation scores of all trained models are presented in Table 1. The more in-depth results of each model are discussed in their own sections that are U-Net, Backpropagation, and CycleGAN. The U-Net model trained with only LIVECell data is chosen to be baseline of the trained models since it shows how well the model performs without any domain adaptation attempts. The results of other models are compared to this baseline model. Overall, the scores indicate that neither Backpropagation nor CycleGan models managed to improve predictions compared to the baseline model.

4.1 U-Net

Three U-Net models were trained with each of the three source datasets. The training was done using Adam optimizer with learning rate of 0.001, 20 epochs and batch size of five. The training loss converged typically after 17 epochs. Predictions made with U-Net model are showcased in figures 18 and in Appendix A.

Table 1: The evaluation Dice score results of all models. The baseline model with data augmentations got the best score. Synthetic Unity data had positive impact most of the times.

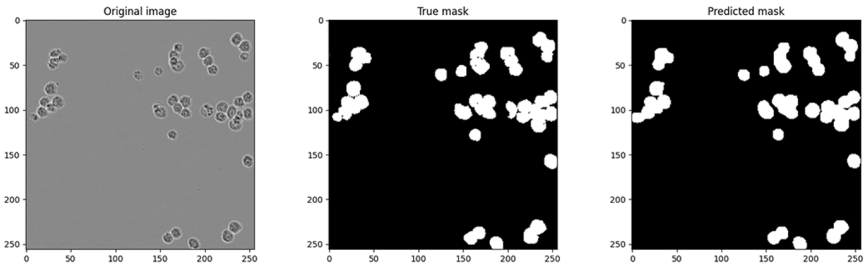
Utilised data	**Source images**	**Training epochs**	**Evaluation Dice score**
U-Net			
LIVECell	5239	20	0.622
AugLIVECell	5239	20	0.497
AugLIVECell+Unity	10239	20	**0.629**
Backpropagation			
LIVECell	7409	20	0.510
AugLIVECell	7409	20	0.337
AugLIVECell+Unity	12409	20	0.357
Re. LIVECell	4082	20	0.377
Re. AugLIVECell	4082	20	0.500
Re. AugLIVECell+Unity	9082	20	**0.574**
CycleGAN			
LIVECell	5239	50	**0.379**
LIVECell+empty	10239	50	0.346
AugLIVECell	5239	50	0.339
AugLIVECell+empty	10239	50	0.334
AugLIVECell+Unity	10239	50	0.367
AugLIVECell+empty+noise	10239	100	0.371

4.2 Backpropagation

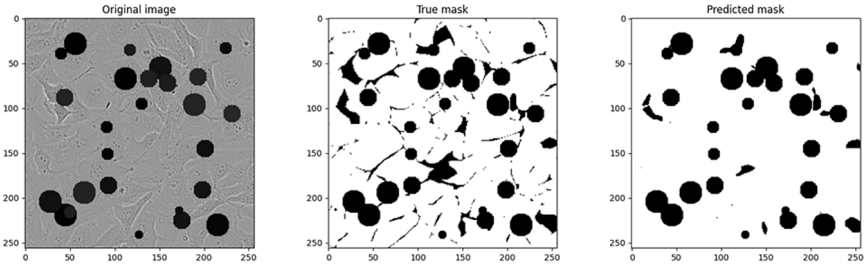
Six different models were trained for Backpropagation architecture. The implementation details are presented in chapter 3.6. Three of the models were trained using the same datasets as the basic U-Net, namely the LIVECell dataset, augmented LIVECell, and augmented LIVECell with synthetic images. The other models were trained with the same datasets but images starting with BV2, HUH7 and SkBr3 were only selected for training. These images contain smaller cells and thus are closer to our target domain cells. The aim was to reduce domain shift between domains. Table 2 shows exact number of images used in training for each model.

Models were trained using an Adam optimizer with a constant learning rate 0.001 and 20 epochs. A combination of dice loss and binary cross entropy loss (BCE) was used in the learning process. After 10 epochs, the dice loss steadily decreased with every epoch. However, the models were not trained with more than 20 epochs because in our testing evaluation dice did not significantly improve with further training. BCE loss on the other hand did change only little for both

(a) LIVECell



(b) Augmented LIVECell



(c) Combined Augmented LIVECell and synthetic Unity data

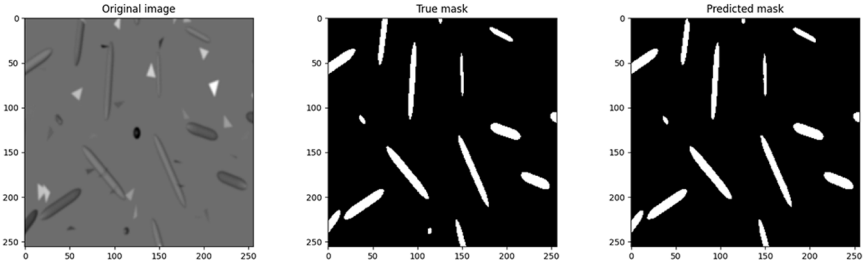


Figure 18: Examples of the training data predictions with the three U-Net models. As can be seen all models learned the training data well but large cells caused the most problems for the network. The synthetic data was the easiest task for the model even though it had extra shapes and noise added.

training and evaluation data during training.

Some transformations were applied to target data during training. The images were first randomly cropped to 512 x 512, then resized to 256 x 256. This was done to achieve better scaling when compared to source data. The source images were directly randomly cropped to size 256 x 256. Finally, the LIVECell dataset was augmented.

Most of the models perform well with training data. However, all of the models failed to surpass the basic U-Net models in evaluation scores despite high training scores. The highest evaluation dice score was obtained with a dataset that contained only LIVECell images of BV2, HUH7 and SkBr3 and synthetic data.

Some examples of target domain predictions can be seen in Appendix B. Overall, the predictions are more or less random and cells are not predicted reliably. Data augmentation and synthetic data seem to lower the threshold for detection of cells, which results in larger prediction areas.

Table 2: Number of training images for each Backpropagation model.

Utilised data	**Target**	**LIVECell**	**Unity**
LIVECell	2170	5239	0
AugLIVECell	2170	5239	0
AugLIVECell+Unity	2170	5239	5000
Re. LIVECell	2170	1912	0
Re. AugLIVECell	2170	1912	0
Re. AugLIVECell+Unity	2170	1912	5000

4.3 CycleGAN

The CycleGAN model was trained with six different datasets. The result of applying the trained CycleGAN and U-Net models to one target image is illustrated in Appendix C.

The original LIVECell dataset, as introduced in chapter 3.1, was utilised as the basis for all training datasets. Images imitating LIVECell data with no cells present were added to some training datasets and referred to as empty images. Augmented LIVECell images described in 3.3 and synthetic data created with the Unity engine introduced in 3.1 were utilised as well. To improve the results, some noise was added into augmented LIVECell images and augmented empty images. The used noise was a combination of random Gaussian blur and random changes in the brightness, the contrast, the saturation, and the hue of the image. In training, the Adam optimiser was utilised with the learning rate of 0.0002 and the betas (0.5, 0.999). The used batch size was 16 images.

5 Discussion and Conclusions

The results show that the adaptation task was quite hard. In this chapter we will present some thoughts on what made the task hard and how our work could be improved. We also touch on alternate strategies that could work well on this problem.

After trying various means of domain adaptation, it is remarkable how well the plain U-Net performs across domains. U-Net performance was improved by augmenting training data and using synthetic data. Without any data augmentation, U-Net seems to only find small cells in the target domain. Augmenting LIVECell data with magnetic-particle-like blobs achieved best results when used in combination with Unity data. The lack of magnetic particles in LIVECell images was an easy and distinguishable feature causing domain shift, and augmentation was used to reduce domain shift. The augmentation method used was quite simple, just adding two sizes of round particles into the data. It would be worth investigating how a more random and irregular augmentation strategies would impact the results.

Adding the Unity data to the training samples seemed to lower the threshold of detecting cells, making the model latch onto noise very easily. This might be related to how noise is added to the synthetic data. If adding noise to the source image blurs a synthetic cell beyond recognizable characteristics, but the mask still argues them to be cells, this would dispose the model to interpreting noise as cells. We also trained the model using only one set of synthetic data. Generating multiple data sets using different strategies to model noise and cells could reduce the model's fit to a single dataset and improve it's performance across domains. Making the synthetic sets sparser in cells could also reduce domain shift, since the target domain is sparse.

Other attempts at domain adaptation seemed to only degrade the performance of U-Net. We had limited time to optimize the model parameters, and the final models are likely not the optimal configurations. Too large nets can overfit to the data, or simply remember the training data. A simplified, less deep version of the U-net is worth investigating for this problem. Additionally, there is still room for additional data augmentation and transformations in the training procedure.

With the Backpropagation approach, the domain classifier was intended to catch domain invariant features of the cells. The results show, however, it does not perform very well in this task, but instead seems to obscure some important details that drive the model towards interpreting noise as cells. Removing significantly larger cells from the training data seemed to improve performance. This is most likely a result of smaller a domain gap, since the target domain consists of smaller cells. Saito et al. criticized domain classifier based solutions in their paper by stating that the domain classifier is agnostic to the task at hand [19]. As a result, the domain classifier may teach the feature encoder to find features that have nothing

to do with cells. Reducing the relative importance of the domain classifier loss could improve the results. Ensuring the U-Net is trained to perform well in one domain before attempting domain adaptation should ensure the features are relevant for the task. Another issue relates to the nonlinear encoder and decoder paths of the U-Net. Our attempts fed only a single layer of the U-Net to the domain discriminator. The idea was that the backpropagation algorithm should then propagate these effects deeper into other layers of the model. As U-Net contains multiple skips, the domain classifier may have simply recognized the domain only considering the skip path data. This would essentially leave the domain invariant feature encoding out altogether. Future work into the topic should study if there's a more robust way to combine this sort of nonlinear models with the backpropagation domain adaptation task, or attempt using a different model for semantic segmentation.

In the CycleGAN approach, our source and target domains could simply have been too far apart from each other. For example, the target data had a misty, blurry nature to it, while our source data did not. While Gaussian blur was added to our synthetic data, it was not added to the LIVECell data. In addition, both LIVECell and our synthetic data contained a large amount of cells in contrast to the target data. Therefore, there might have been too large of a domain shift between the source and target domain.

The most obvious difference in the source domains to the target domain was that our target domain is a 3D culture, whereas LIVECell images are from 2D cultures. The magnetic particles used to study the mechanic properties of the 3D cultures were prone to get misinterpreted as cells, and augmenting LIVECell data with magnetic-particle-like blobs showed an improvement in results, if only a small one. Another challenge was the sparsity of target domain cells. In both LIVECell and Unity datasets, all images contain multiple cells, whereas our target domain images have few to no cells. The CycleGAN results show that augmenting the data with simple, LIVECell-like images with no cells improved the model performance. This could be due to the difference in marginal distributions across the domains.

For evaluating model performance, we used the dice score, which is a metric of absolute pixel similarity. Other possible loss functions should be looked into as well. Especially in our case, where there's a label imbalance between target and source domains, dice loss scores can be deceptively small even when the model doesn't find any correct cells. For example, calculating the portion of pixels that are correctly predicted as cells would be another fitting metric and used in combination with dice loss.

Some sort of classifier discrepancy solution adapted from the work of Saito et al. could improve results on this behalf [19]. They report that their approach performs well in circumstances where there is a difference in marginal distributions. Another possibility would be to manipulate the source images to be more sparse before

training by, for example, augmenting the LIVECell images with some sort of white noise.

Some resulting images show that the predicted masks interpret random background noise as cells. This could be the result of simply too large a domain gap as the target domain is sparser as source domains, but they could also be the result of overconfident predictions. As these overconfident solutions become a problem, Entropy minimization approaches could be the solution. Entropy minimization directly minimizes entropy on target domain, forcing the model to be less confident also with training images. Some solution following the work of Vu et al. could improve the results [22].

References

[1] Hao, S., Zhou, Y., Guo, Y. A brief survey on semantic segmentation with deep learning. Neurocomputing, 2020. Vol. 406, P. 302–321.

[2] Thoma, M. A Survey of Semantic Segmentation. 2016. arXiv: 1602.06541 [cs.CV].

[3] Toldo, M. et al. Unsupervised Domain Adaptation in Semantic Segmentation: A Review. Technologies, 2020. Vol. 8:2. ISSN: 2227-7080. DOI: 10.3390/technologies8020035. URL: https://www.mdpi.com/2227-7080/8/2/35.

[4] Haq, M. M., Huang, J. Adversarial Domain Adaptation for Cell Segmentation. 2020. Vol. 121. Ed. by T. Arbel et al., P. 277–287. URL: https://proceedings.mlr.press/v121/haq20a.html.

[5] Kapil, A. et al. Domain Adaptation-Based Deep Learning for Automated Tumor Cell (TC) Scoring and Survival Analysis on PD-L1 Stained Tissue Images. IEEE Transactions on Medical Imaging, 2021. Vol. 40:9. P. 2513–2523. DOI: 10.1109/TMI.2021.3081396.

[6] Al-Kofahi, Y. et al. A deep learning-based algorithm for 2-D cell segmentation in microscopy images. BMC bioinformatics, 2018. Vol. 19:1. P. 1–11.

[7] Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[8] Long, J., Shelhamer, E., Darrell, T. Fully convolutional networks for semantic segmentation. 2015, P. 3431–3440.

[9] Minaee, S. et al. Image segmentation using deep learning: A survey. IEEE transactions on pattern analysis and machine intelligence, 2021.

[10] Pan, S. J., Yang, Q. A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 2009. Vol. 22:10. P. 1345–1359.

[11] Farahani, A. et al. A brief review of domain adaptation. Advances in Data Science and Information Engineering, 2021, P. 877–894.

[12] Sun, S., Shi, H., Wu, Y. A survey of multi-source domain adaptation. Information Fusion, 2015. Vol. 24, P. 84–92.

[13] Karani, N. et al. A lifelong learning approach to brain MR segmentation across scanners and protocols. 2018, P. 476–484.

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

[15] Tsai, Y. et al. Learning to Adapt Structured Output Space for Semantic Segmentation. CoRR, 2018. Vol. abs/1802.10349. arXiv: 1802.10349. URL: http://arxiv.org/abs/1802.10349.

[16] Zou, Y. et al. Unsupervised Domain Adaptation for Semantic Segmentation via Class-Balanced Self-Training. 2018.

[17] Watanabe, K. et al. Multichannel Semantic Segmentation with Unsupervised Domain Adaptation. CoRR, 2018. Vol. abs/1812.04351. arXiv: 1812.04351. URL: http://arxiv.org/abs/1812.04351.

[18] Bengio, Y. et al. Curriculum Learning. 2009, P. 41–48. DOI: 10.1145/1553374.1553380. URL: https://doi.org/10.1145/1553374.1553380.

[19] Saito, K. et al. Maximum Classifier Discrepancy for Unsupervised Domain Adaptation. CoRR, 2017. Vol. abs/1712.02560. arXiv: 1712.02560. URL: http://arxiv.org/abs/1712.02560.

[20] Zhu, J.-Y. et al. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. 2017. DOI: 10.48550/ARXIV.1703.10593. URL: https://arxiv.org/abs/1703.10593.

[21] Hoffman, J. et al. FCNs in the Wild: Pixel-level Adversarial and Constraint-based Adaptation. 2016. DOI: 10.48550/ARXIV.1612.02649. URL: https://arxiv.org/abs/1612.02649.

[22] Vu, T. et al. ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation. CoRR, 2018. Vol. abs/1811.12833. arXiv: 1811.12833. URL: http://arxiv.org/abs/1811.12833.

[23] Edlund, C. et al. LIVECell—A large-scale dataset for label-free live cell segmentation. Nature Methods, 2021. Vol. 18. DOI: 10.1038/s41592-021-01249-6.

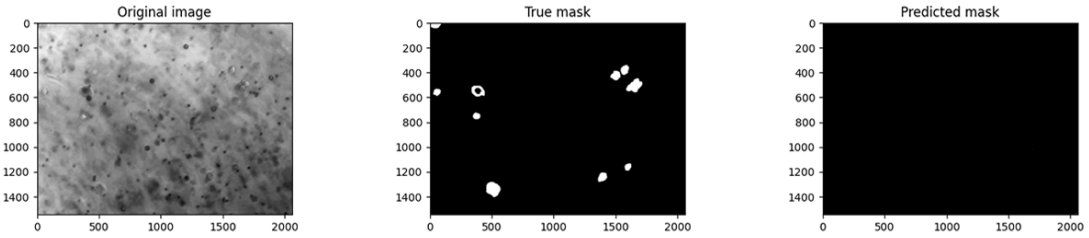
[24] Ronneberger, O., Fischer, P., Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015. DOI: 10.48550/ARXIV.1505.04597. URL: https://arxiv.org/abs/1505.04597.

[25] Rumelhart, D. E., Hinton, G. E., Williams, R. J. Learning Representations by Back-propagating Errors. Nature, 1986. Vol. 323:6088. P. 533–536. DOI: 10.1038/323533a0. URL: http://www.nature.com/articles/323533a0.

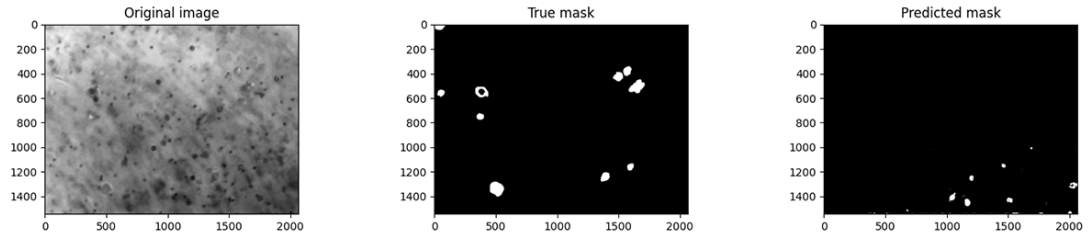
[26] Brion, E. et al. Domain adversarial networks and intensity-based data augmentation for male pelvic organ segmentation in cone beam CT. Computers in Biology and Medicine, 2021, P. 104269.

A The Evaluation Images of the U-Net

(a) LIVECell



(b) Augmented LIVECell



(c) The Combination of augmented LIVECell and synthetic unity data

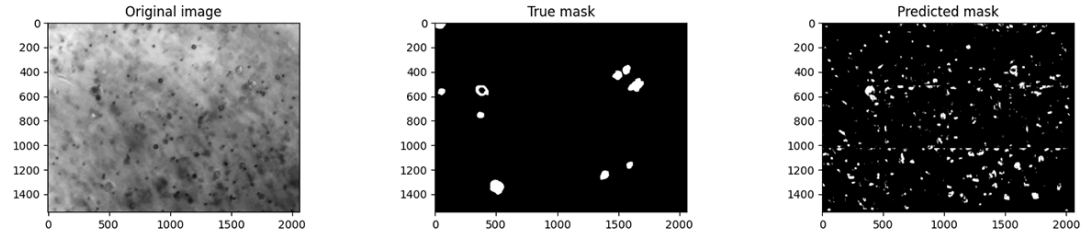
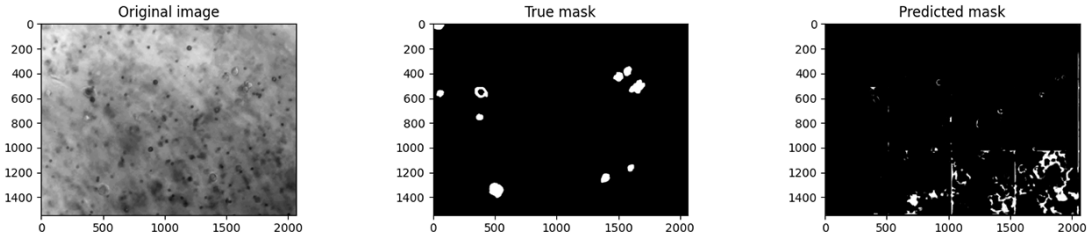


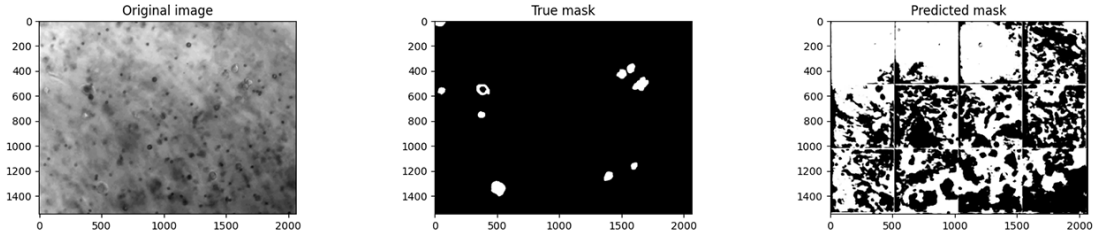
Figure A1: The results of the three U-Net models with target data. The models 1) and 2) predicted only a little pixels as cells and even then mostly incorrectly. Surprisingly data augmentation caused model to predict more magnetic particles as cells and therefore it had the lowest Dice score. The best model was achieved with the combined data (model 3) which predicted significantly more pixels as cells including often at least the outlines of actual cells.

B Result Images of the Evaluation of the Back-propagation

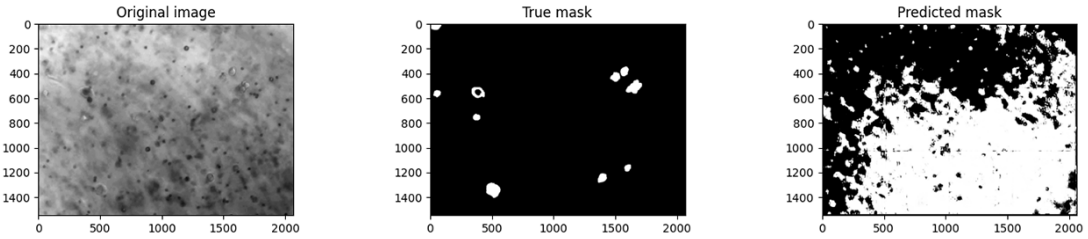
(a) LIVECell



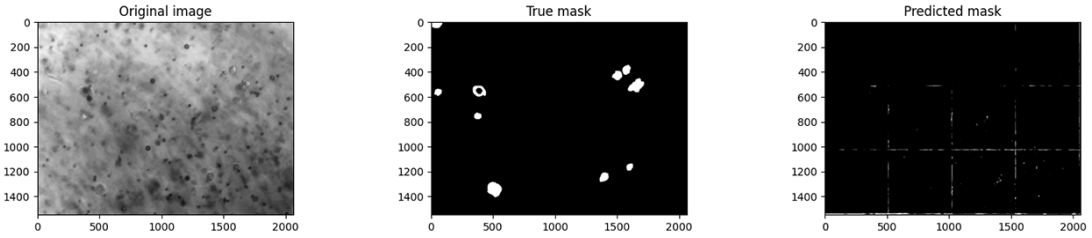
(b) Augmented LIVECell



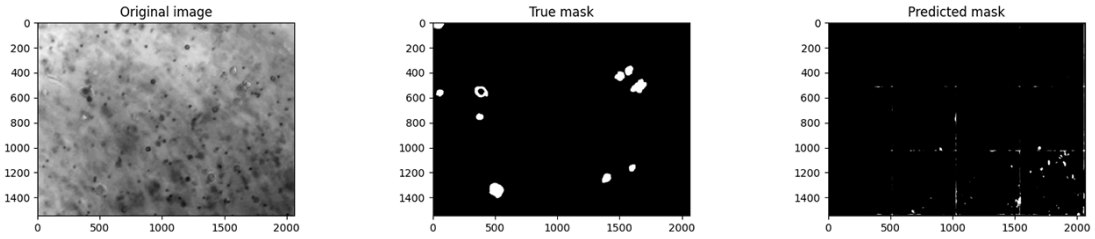
(c) The Combination of augmented LIVECell and synthetic unity data



(d) Reduced LIVECell



(e) Reduced Augmented LIVECell



(f) The Combination of reduced augmented LIVECell and synthetic unity data

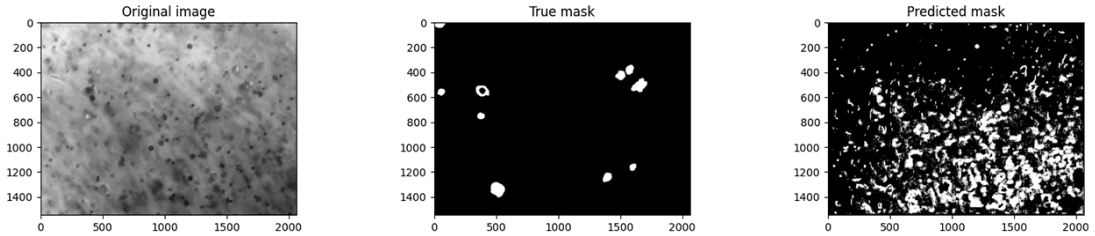
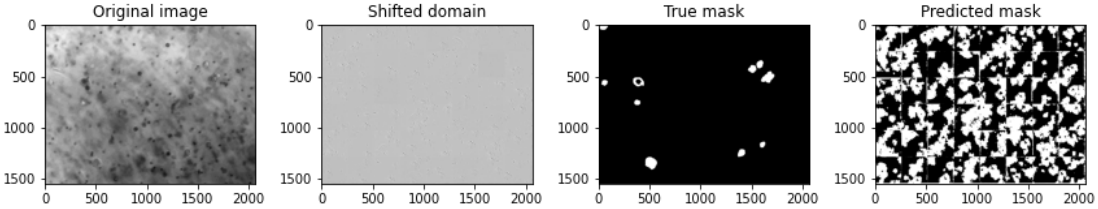


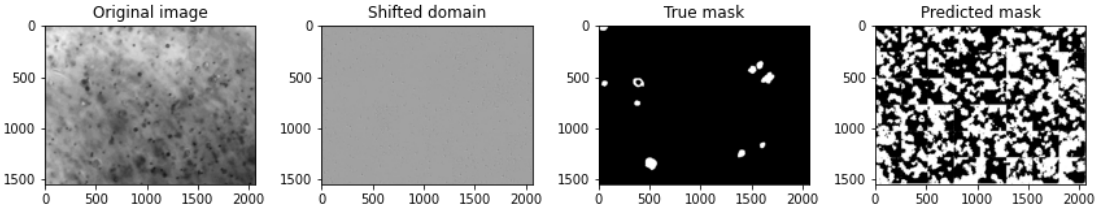
Figure B1: Target domain predictions of every Backpropagation model. Original image that is used as an input is first image of evaluation dataset and it is same for all of the models.

C Result Images of the Evaluation of the CycleGAN

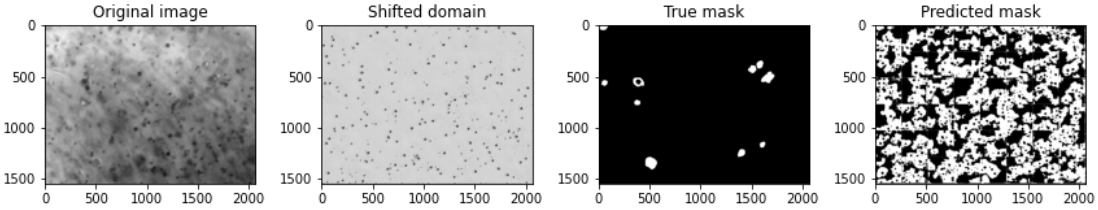
(a) LIVECell



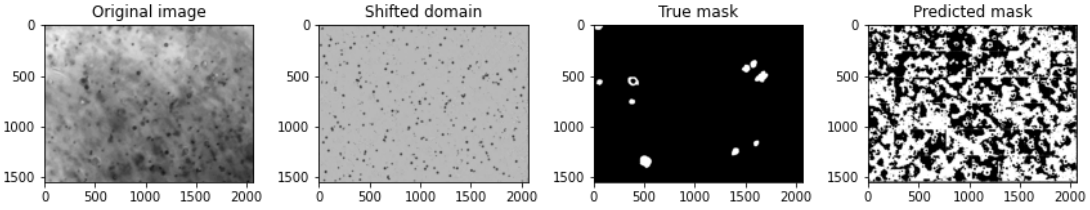
(b) LIVECell with empty images



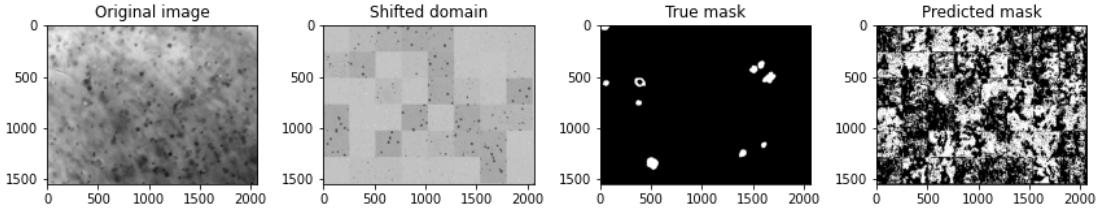
(c) augmented LIVECell



(d) augmented LIVECell with empty images



(e) Combined augmented LIVECell and synthetic data



(f) Augmented LIVECell with noise

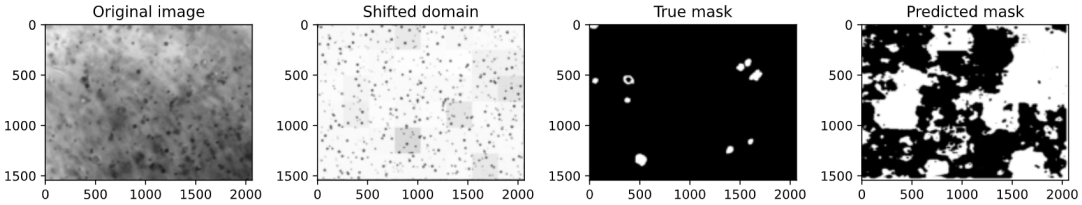


Figure C1: The results of CycleGAN models trained with different datasets