Generation of Synthetic Data for Large-Scale Image Analysis with Generative Adversarial Networks

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau des

Karlsruher Instituts für Technologie (KIT)

angenommene

DISSERTATION

von

M.Sc. Moritz Böhland

Tag der mündlichen Prüfung:	13. November 2024
Hauptreferent:	apl. Prof. Dr. Markus Reischl
Korreferenten:	Prof. Dr. Veit Hagenmeyer
	Prof. Dr. Sven Perner

Zusammenfassung

Bildverarbeitung wird in vielen Bereichen wie dem Ingenieurwesen oder den Biowissenschaften benötigt. Eine wichtige Komponente ist die Segmentierung, die primär mit überwachten neuronalen Netzwerken durchgeführt wird. Überwachte neuronale Netzwerke erfordern annotierte Daten. Die manuelle Annotation ist jedoch zeitaufwändig, teuer, und fehleranfällig. Seit kurzem werden Generative Adversarial Networks (GANs), die eine Bild-zu-Bild-Übersetzung zwischen zwei ungepaarten Bildmengen erlernen können verwendet, um die manuelle Annotation zu ersetzen. Eine der Bildmengen enthält zu segmentierende Bilder und die andere Bildmenge enthält synthetisch erzeugte ungepaarte annotierte Bilder, so genannte synthetische Labelbilder. Drei zentrale Komponenten dieser Arbeitsabläufe werden in dieser Arbeit betrachtet, optimiert, und die bestehenden Probleme werden gelöst. Der Einfluss der Eigenschaften der synthetischen Labelbilder auf die nachfolgenden Segmentierungsergebnisse wird untersucht. Die Ergebnisse können verwendet werden, um den Aufwand zur Synthese der Labelbilder zu reduzieren und bessere Segmentierungsergebnisse zu erzielen. Eine Methode zur Bewertung der Nützlichkeit von Validierungsmetriken während des GAN-Trainings wird vorgeschlagen. Mehrere Metriken werden auf zwei Benchmark-Datensätzen getestet. Daraus wird eine allgemeine Empfehlung zur Verwendung abgeleitet, um nach dem Training die beste GAN-Epoche auszuwählen. Eine neue Methode zur praktisch fehlerfreien Ausschnitts-basierten Inferenz von großformatigen Bildern unter Verwendung von GANs wird vorgestellt. Zusätzlich wird ein vollständig synthetischer 2D-Benchmark-Datensatz entwickelt, um die Fehler zu quantifizieren, die bei der Ausschnitts-basierten Inferenz auftreten. Dieser Benchmark-Datensatz wird verwendet, um den neuen Ansatz mit dem Stand der Technik zu vergleichen, wobei die neue Methode signifikant besser ist. Es wird gezeigt, dass sich die neue Methode auch auf 3D-Bilder übertragen lässt. Abschließend gibt diese Arbeit einen Ausblick auf zukünftige Arbeitsfelder zur Verbesserung der Ergebnisqualität, wenn GANs verwendet werden, um auf manuelle Annotation bei Segmentierungsaufgaben zu verzichten.

Abstract

Image processing is required in many areas such as engineering or life science. An important component is segmentation, which is performed primarily with supervised neural networks. Supervised neural networks require annotated data. However, manual annotation is time-consuming, expensive, and error-prone. Recently, Generative Adversarial Networks (GANs) capable of learning image-to-image translation between two unpaired sets of images have been used to replace manual annotation. One set contains the images to be segmented and the other set contains synthetically generated unpaired annotation images, so-called synthetic labels. Therefore, this thesis addresses the use of GANs in segmentation pipelines to replace the manual creation of labels. In this thesis, three central components of this pipeline are examined, evaluated and the existing problems are solved. The influence of the properties of the synthetic label images on the downstream segmentation performance is examined, and the results can be used to reduce the amount of work required to synthesize the label images and achieve better segmentation results. A method is proposed to evaluate the utility of validation metrics during GAN training. Several metrics are tested on two benchmark datasets and a general recommendation on how to use them to select the best GAN epoch after training is derived. A new method to allow virtually error-free patch-based inference of large-scale images using GANs is presented. Additionally, a fully synthetic 2D benchmark dataset is developed to quantify the errors that arise during patch-based inference. This benchmark dataset is used to compare state-of-the-art methods with the new method and the new method performs significantly better. It is shown that the new method can be adapted to 3D images. Finally, this thesis provides an outlook on future areas of application and fields of work to improve the quality of the results when using GANs to omit manual labeling for segmentation tasks.

Acknowledgments

First and foremost, I would like to express my gratitude to my direct supervisor, apl. Prof. Dr. Markus Reischl, whose outstanding guidance, inspiring conversations, and dedication were invaluable throughout my dissertation. He consistently supported and encouraged me to follow my own path. I really appreciate the open and personal conversations we shared during that time. I also thank the head of the Institute for Automation and Applied Informatics, Prof. Dr.-Ing. Veit Hagenmeyer. I would like to thank Dr. Maik Lorch for his tremendous support in familiarizing me with the organizational tasks and assisting with the institute's teaching responsibilities. I also express my gratitude to my project partners, Prof. Dr. Sven Perner and Dr. Lars Tharun. I greatly enjoyed discussing histopathology and the possibilities of data analysis with you. My sincere thanks go to apl. Prof. Dr. Ralf Mikut for his ideas, great conversations, and thoughtful attention during my scientific work.

This work would not have been possible without my colleagues. I thank Benni for his encouragement in the early months of my scientific journey and Andy for his willingness to help with technical and scientific questions. My special thanks go to Tim, with whom I shared an office and without whom many of my projects would not have been possible. I will never forget how you dedicated yourself to our joint contribution to the CoNIC Challenge when the deadline was extended. I never thought it would be possible to work with so many wonderful and down-to-earth people. I truly enjoyed the time I spent with each of you. Therefore, my special thanks go to Baifan, Hawo, Ines, Kaleb, Katharina, Lisa, Lorenz, Marian, Mark, Roman, Simon, Vojtech, Yanke, and Zehua.

I will always remember the Deep Learning Summer School in Gran Canaria with Luca, André, Oli, and Marcel, where I learned a lot but laughed even more. I also thank Dennis, Firaz, Hannes, and Ina for the exchange about deep learning. I really enjoyed the discussions about ideas, applications, and results. I would like to thank Friedrich and Luca, who presented our group's research together with me at the BMT and with whom I had a great time. I would also like to thank Andreas, Bernadette and Claudia, whose organizational skills and team spirit made our institute the best place to work that I could have wished for during my doctoral studies.

I would also like to thank my family, who always supported me in everything I did and encouraged me to pursue my path. Finally, I would like to express my special thanks to Celine, who listened to me when I excitedly told her about new ideas, who built me up when experiments failed, and who motivated me to maintain my determination while writing this thesis. Without you, this thesis would not have been possible.

Contents

Introduction 1

1.1 Motivation

Image analysis is used to extract insights from observations. The first applications in computer science date back to the 1960s [1]. Today, image analysis is used for a variety of tasks ranging from engineering to life science or remote sensing [2]–[9]. Entire images can be classified by systematically organizing them into categories based on shared characteristics. However, modern image processing also enables a much more granular evaluation in which individual objects in images are recognized and information is extracted. The detection of individual objects and the extraction of their shape is required for many tasks such as autonomous driving, quality control, biomedical image analysis, or microscopy [10]–[12].

Images are often generated automatically by mechatronic systems in large quantities [13], [14]. The large amount of data renders manual analysis impractical for many applications. Furthermore, manual analysis is not feasible for tasks that require real-time evaluation. Supervised deep learning models have replaced traditional methods in recent years [15]–[18]. Instead of using hand-crafted processing pipelines and feature extraction methods, supervised deep learning models learn to perform the desired task from annotated data. Although there are many annotated datasets, they are often not usable for new datasets because the imaging modalities or the objects in the images differ. Therefore, manual annotation is needed to train supervised deep learning models. However, manual annotation is expensive, time-consuming, and error-prone [19], [20].

Recently, Generative Adversarial Networks (GANs) capable of performing unpaired, also called unsupervised, image-to-image translation, are used to omit manual annotation [21], [22]. If GANs are used, annotations, which are also called label images, that are not paired with the existing images must be synthesized. Because these label images do not have to be paired with the existing data, synthesis can be automated, saving time and money. For the time being, the use of GANs still has some disadvantages. While the synthesis of the unpaired label images can be automated, the unpaired label images need to be individually adapted to the images that are to be analyzed. It is still unclear how to adapt the unpaired label images and which adaptions should be used. Furthermore, GANs are computationally expensive and training is unstable. Unlike supervised models, they are trained for a fixed number of epochs. This has disadvantages, such as an increase in computational expense required or a reduction in quality, since the last epoch is not necessarily the best. Finally, GANs require a large amount of computational resources. They are trained on graphics processing units (GPUs) and a large amount of video random access memory (VRAM) is essential. If VRAM is limited, large-scale images are synthesized patch-based during application. Patch-based synthesis reduces the quality of the final image.

The research presented in this thesis aims to further improve image processing pipelines that use GANs to replace the manual annotation of images for instance segmentation tasks. A generally applicable pipeline for the integration of GANs in workflows for image segmentation is used to examine the influence of synthetic label image properties on the downstream segmentation quality. Two benchmark datasets are developed to carry out the experiments. The insights gained enable faster creation of synthetic label images with less expert knowledge and higher downstream segmentation quality. Furthermore, a method to select the best epoch for inference after training a GAN is developed and tested on a biological and a biomedical benchmark dataset. Finally, the patch-based inference of GANs is investigated. Tests on a new custom benchmark dataset show that existing methods generate errors at the edges of the patches. Therefore, a new method for patch-based image-to-image translation is developed to increase the quality of the final GAN output.

1.2 Image Analysis

1.2.1 Image Analysis Pipeline

Image analysis is an old research field in computer science with the first applications in the 1960s [1]. Image analysis aims to extract meaningful information from images and the analysis pipeline depicted in Fig. 1.1 ranges from *Image acquisition* over *Information extraction* to *Information analysis*.

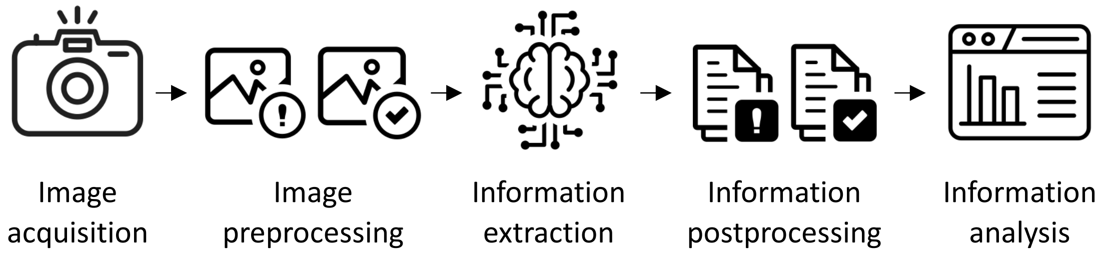


Figure 1.1 | Image analysis pipeline. Images are captured in the *Image acquisition* step. If necessary, *Image preprocessing* is performed. Subsequently, the desired information is extracted in the *Information extraction* step. *Information postprocessing* can be applied. Post processing is needed if the extracted information does not match the desired data representation. Finally, the results are assessed in the *Information analysis step.*

Image Acquisition The *Image acquisition* is the first step of the image analysis pipeline. The basic components of imaging devices are lenses to focus light, a light-sensitive surface, and a shutter to control light entry. The light-sensitive surface consists of a matrix of detectors for digital image acquisition. In each detector, the incoming photons of the light result in an electrical charge, and during readout, the electrical charge is converted to a brightness value of the corresponding point (pixel) in the detector matrix. The brightness values of grayscale images are typically integers in the range of 0 to 255 (8-bit) or in the range of 0 to 65535 (16-bit). The most prominent detectors are charge-coupled devices (CCDs) and complementary metal oxide semiconductor (CMOS) chips. CCDs are known for their good signal-to-noise ratio, and their reliable track record for scientific imaging but have high power consumption, and are more expensive than CMOS. CMOS chips are commonly used in mobile phones and have a low power requirement, but the signal-to-noise ratio in the past was poor [23]. As an image analyst, one often has no influence on the selection of the components for image acquisition and must work with the raw data provided. Low-quality raw data leads to increased effort in *image preprocessing*, *information extraction*, and *image postprocessing*.

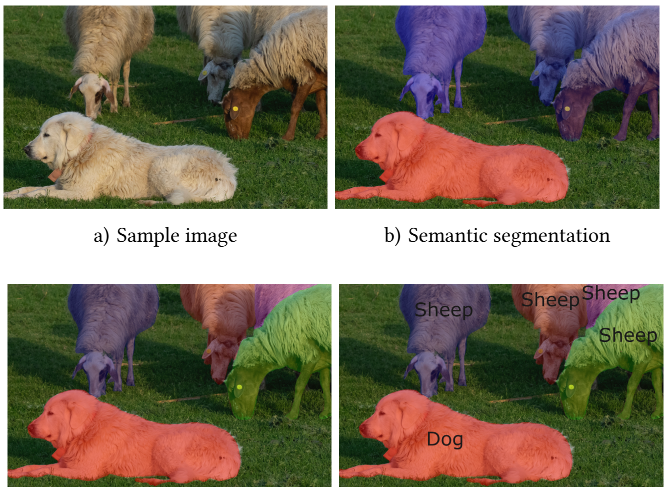
There is a wide variety of different acquisition modalities. Images can be acquired by remote sensing. Mostly, the reflected sunlight is captured, but wavelengths are often outside the range of human vision. Furthermore, the radiation from Earth itself or the light emitted by a laser or radar can be captured [8]. Mobile phones or digital single-lens reflex (DSLR) cameras are usually used to capture human visible light, e.g. in street scenes, for quality control during manufacturing, or in biomedical experiments with living animals [24], [25]. Computed tomography (CT) can be used to create cross-sectional images, e.g. of bones and dense structures in the human body. Meanwhile, Magnetic Resonance Imaging (MRI) provides good soft tissue contrast of organs such as the brain [26], [27].

Images with a very high spatial resolution need to be captured for biomedical use cases. Pathologists use bright-field microscopy to record human tissue and extract information from tumors. Staining, for example with Hematoxylin and Eosin (H&E), can be applied to enhance cell visualization [28]. In fluorescence microscopy, the sample is irradiated with light. The light is absorbed by the sample itself or by fluorescent agents, and light with less energy is emitted and directed to the camera sensor. Fluorescence microscopy is also used to capture images of cells [29].

Image Preprocessing Often times images need to be preprocessed before the information can be extracted. According to [30], "image preprocessing is a method to transform raw image data into clean image data, as most of the raw image data contain noise and contain some missing values or incomplete values, inconsistent values, and false values". Therefore, *image preprocessing* describes methods to manipulate images in such a way that the subsequent steps can be carried out in a standardized manner. A simple form of preprocessing is the removal of blurry images. This can be done for images of street scenes if there is a significant amount of motion. More advanced methods do not remove the images but try to enhance the quality of the images. Noise can be suppressed by filters, or inhomogeneous lighting can be corrected [30]. Fluorescence imaging often suffers from decreased brightness at the edges of images. This can be corrected with tools such as BaSiC, based on low-rank and sparse decomposition of the image [31]. Histopathological images suffer from variation in color and algorithms can be used to adapt the color to a reference image [32].

Information Extraction The desired information is extracted in the *Information extraction* step. Exemplarily, classification and segmentation are discussed in this introduction, but other information extraction tasks such as detection, regression, pose estimation, or tracking exist [33]–[39].

Classification can be performed with a classifier and traditional image processing methods or deep learning. Traditional image processing methods extract features from the images. Subsequently, classification is performed using the extracted features. Extracting meaningful features is a complex task and features often need to be invariant to scale, shift, and rotation. Reviews on image feature extraction are given in [40]–[43]. A variety of machine learning methods like Logistic Regression, Nearest-Neighbors, Naive Bayes, Support Vector Machines (SVMs), or Gradient Boosting can be applied to the extracted features to classify the underlying images [44]–[50]. The resulting pipeline is highly in-



a) Sample image b) Semantic segmentation

c) Instance segmentation d) Panoptic segmentation

Figure 1.2 | Different segmentation types with semantic segmentation, instance segmentation, and panoptic segmentation. Semantic segmentation can differentiate between different object classes, while instance segmentation cannot. Instance segmentation is used to delineate objects from each other. With panoptic segmentation, both are possible.

dividual and must be adapted to the underlying image distribution. If deep learning is used, the manual feature extraction step is omitted and features are learned by the neural network (NN). Classification is usually performed directly in the NN, but features can also be used to train a classifier, such as an SVM. However, there is a wide variety of network architectures and each architecture has multiple hyperparameters [51]–[53].

Segmentation tasks can be categorized into three different methods shown in Fig. 1.2. Semantic segmentation classifies each pixel in an image into specific object categories, enabling a pixel-level understanding of the scene. Instance segmentation goes a step further, not only classifying pixels but also distinguishing between individual instances of objects. Panoptic segmentation unifies semantic segmentation and instance segmentation, offering an understanding of the entire visual scene. The objects are not only separated, but also classified. Panoptic segmentation is used, for example, to assess the cellular composition of histopathological images [54], [55]. The segmentation step can be performed with traditional image processing or deep learning. Traditional segmentation can be performed in many ways. If the pixel intensities of the foreground and the background differ, global or local intensity thresholding can be used. The edges of objects can be used to separate foreground and background with an edge detector like the Canny filter. Region-based methods group pixels with similar properties. A detailed description is given in the following work [56]–[61]. Recently, deep learning methods have outperformed classical methods for many segmentation tasks [34], [54], [61]. This is especially notable for image segmentation challenges, where the results of different methods can be quantitatively compared [34], [54]. An aspect of superior performance is the learned discovery of the underlying patterns in the image, which results in better features than using hand-crafted filters [62].

Information Postprocessing *Information postprocessing* is applied to improve the output of the information extraction step or to transform the output to represent the desired information. A segmentation map can be created from the output. For instance segmentation, each object in the image is depicted by another color in the segmentation map. Furthermore, small objects can be removed or holes in existing objects can be filled with morphological closing [30]. Furthermore, the output of neural networks used for instance segmentation may need to be post-processed to obtain an instance segmentation from semantic segmentations, seed points, distance maps, or others. A watershed transform can be used to obtain the final semantic segmentation [63]. If prior knowledge of the objects is available, under- or oversegmentation, resulting in merges or splits, can be corrected [64].

Information Analysis In the *Information analysis* step, the extracted information is analyzed. For segmentation tasks, properties such as the number of objects, size, or shape can be extracted and analyzed from the segmentation masks. Alternatively, an overlay of the segmentation mask can be added to the input image to support human analysis [65]. For 3D data like CT or MRI this can be done with renderings and visualizations of the captured structures [66], [67]. If images are recorded over time and objects are tracked, visualization can be used to analyze movements and correct them if necessary [68], [69].

1.2.2 Deep Learning-based Image Analysis

Deep learning describes the training of neural networks inspired by the human brain through an iterative optimization process with backpropagation and gradient descent [70], [71]. A neural network consisting of interconnected layers of neurons. Each neuron contains a set of parameters, such as a weight and a bias. Backpropagation calculates the gradient of the error with respect to the parameters of the network. Gradient descent adjusts the parameters to minimize the error, which improves the network's ability to learn and make accurate predictions. Supervised methods need ground truth provided with labels. Labels are the class to which the image belongs for image classification networks.

For segmentation networks, the labels are provided with a segmentation mask or polygons for each object. Instance segmentation is trained with a surrogate task, e.g. by a semantic segmentation mask together with a distance map. For all tasks, the network output is compared to the label with a metric called loss function. The loss function must be derivable to allow gradient descent. Different loss functions are explored in [72], [73]. The data are split into a training, validation, and test set. The training set is used to update the network parameters with an optimizer like Adam or stochastic gradient descent (SGD) [71], [74]. The validation set is needed to prevent overfitting to the training data and to stop training. If training loss and validation loss diverge, overfitting and poor generalization is observed. It is common to stop training after a fixed number of epochs without a decrease in validation loss. The training loss function is often used for the validation data, but any function can be used. Finally, the test data are used to test the generalization ability on unseen data and to obtain a final quality metric. The quality metric is often not derivable and differs from the training loss. The quality metric for instance segmentation can, e.g.: be the AJI^{+} score (see Section 1.3.5).

1.3 Generative Adversarial Networks

A Generative Adversarial Network (GAN) is a generative modeling approach that uses a deep neural network to create new samples in the same domain as the training dataset. The domain consists of all possible data points along with the associated probability distribution that describes the likelihood of each point occurring [75]. Generative modeling is used to model the probability distribution of the domain with the given training data and generate new samples from the modeled distribution. Data domains can be tabular data, time series, text, or images. In each domain, there are subdomains. For example, in the domain of images, the subdomain of images with cars exists.

1.3.1 Architectural Overview

GANs were first introduced in 2014 by Goodfellow *et al.* and are trained in a game-like setting, where NNs compete against each other in an adversarial setting [76]. One or more generators generate data. One or multiple adversarial networks (discriminators) try to distinguish between real samples and generated (also called synthetic or fake) samples. In an ideal setting, all networks increase their capabilities during training. The generators produce more realistic samples, while the discriminators increase their ability to distinguish between real and synthetic samples.

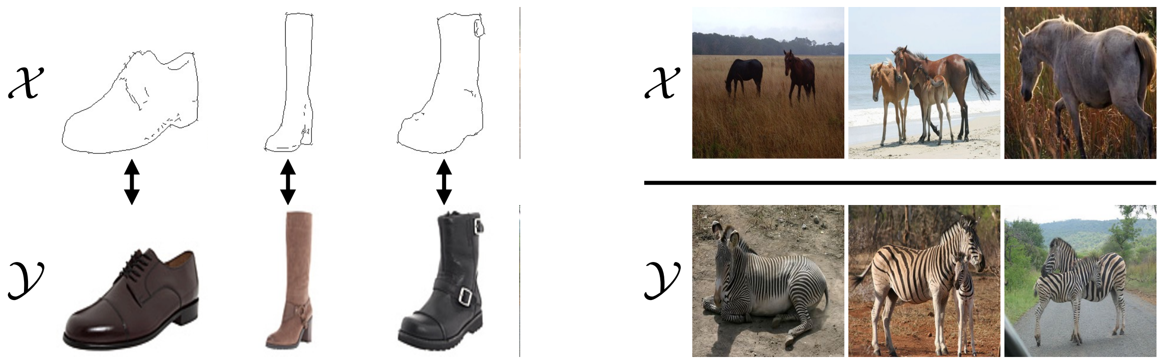
The initial architecture was based on a two-player-game, where one generator generates samples from a noise vector and one discriminator learns to distinguish real from fake samples. They compete against each other and the training goal is formulated with a minmax game with the value function $V \in [-\infty, 0]$, where the data $x \in X$ from domain $\mathcal{X}$ follows the probability density function (PDF) $p_{\text{data}}(x)$. This is denoted by $x \sim p_{\text{data}}(x)$. Consequently, the noise vector $z \in Z$ from domain $\mathcal{Z}$ follows the PDF $p_z(z)$. The generator is defined by $G(z, \theta_G)$ with the network parameters θ_G and the discriminator is defined by $D(x, \theta_D)$ with the network parameters θ_D. The expected value is denoted by $\mathbb{E}$ resulting in:

$$\min_{\theta_G} \max_{\theta_D} V(\theta_G, \theta_D) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x, \theta_D)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z, \theta_G), \theta_D))]. \tag{1.1}$$

For readability, the parameters θ_D and θ_G are often omitted from the equation formulation. The discriminator predicts the probability that a sample is real (one) or synthetic (zero). This can be achieved by using Sigmoid as an activation function, which limits the output value range between zero and one. If the discriminator is able to distinguish between real and synthetic samples, $D(x)$ will be one and $D(G(z))$ will be zero, resulting in Eq. 1.1 reaching the maximum value of 0. A generator that generates samples close to the given distribution p_{data} results in a discriminator output of one for $D(G(z))$, resulting in Eq. 1.1 tending towards $-\infty$.

The noise vector is arbitrarily set by the user, and therefore the PDF is known. Usually, the PDF of the data is unknown, but the empirical distribution of the training dataset $\hat{p}_{\text{data}}(x)$ is given. If the GAN is trained, Eq. 1.1 is used as a loss function $\mathcal{L}$. The complete equation is used to train the discriminator and the right half is used to train the generator. The expected value can be reformulated as $\mathbb{E}_{x \sim \hat{p}_{\text{data}}(x)}[\log D(x, \theta_D)] = \frac{1}{m} \sum_{i=1}^{m} \log D(x, \theta_D)$ with the empirical distribution. This results in iterating over the training dataset and sampling from $p_z(z)$ during the GAN training.

After training, during inference, new synthetic samples can be generated by applying $G(z)$ to varying input noise vectors. This type of GAN is called unconditional because targeted manipulation of the input to obtain a particular output is not possible. Instead,



a) Paired edge-to-shoe dataset

b) Unpaired horse-to-zebra dataset

Figure 1.3 | The paired edge-to-shoe dataset in (a) is used to train unimodal and multimodal GANs [79]. It is impossible to collect a paired dataset of horses and zebras and the dataset in (b) is therefore unpaired [82]. Unsupervised (also called unpaired) GANs are needed to learn the translation for unpaired datasets.

the input must be randomly modified until the desired output is obtained[1].

Conditional GANs enable control of the output. The GAN is conditioned during training by providing the generator and the discriminator with additional information on the desired output. During inference, the condition can be varied to control the output. The conditioning can range from class labels or attributes to text and images. Mirza *et al.* added class labels as a conditioning during training [78]. The GAN was trained on MNIST and thereby enabled the synthesis of distinct digits during inference. Later developed GANs like pix2pix refrained from adding noise and only conditioned the GAN with images [79]. Because the noise was removed, this type of GANs is called unimodal or single-modal. One input refers to one distinct output during inference. GANs able to map one input to multiple outputs are called multimodal, while training is still performed with paired data, as for the unimodal models. Prominent architectures are BicycleGAN or PixelNN [80], [81]. A widely recognized benchmark task for evaluating the performance of multimodal GANs is the edge-to-shoe task (see Fig. 1.3a). While a unimodal GAN maps exactly one image of a shoe to the edges of the shoe, BicycleGAN is able to synthesize different shoes (e.g.: in different colors) for the same input.

All above conditional GANs are called supervised or paired, because pairs of condition and sample are needed for the training of the discriminator with real samples. GANs such as CycleGAN or UNIT do not need paired training data and are called unsupervised or unpaired [82]–[85]. Instead of paired data from the input domains, unpaired data can be

[1]A GAN trained with the MNIST dataset, which contains images of digits, is capable of synthesizing images of digits from zero to nine during inference [77]. However, the synthesis of a specific digit is not possible and one has to vary z to find the right value by chance.

used. These GANs are used if collection of paired data is laborious or not possible. A well-known example for an unpaired dataset is the horse-to-zebra dataset (see Fig. 1.3b). It is not possible to collect data from zebras and horses with the same pose of the animal and the exact same background. Therefore, the transfer (the translation) from an image of a horse to an image of a zebra needs to be learned without paired data. One of the major challenges in unsupervised translation is to perform a meaningful translation. For the domains of zebras and horses, this means that during the translation of an image of a horse to an image of a zebra, only the properties different in both domains have to be changed, while e.g. the background and the pose of the animal need to be preserved. It is not desired to change an image of a horse to a random image of a zebra. In this case, the domain containing images of horses could be replaced by random noise, resulting in an unconditional GAN. Training of unsupervised GANs is unstable and a so-called mode collapse can occur, in which the generator only generates a small number of possible output images that are not sufficiently diverse to represent the real data distribution [80], [82], [84].

In the following, the CycleGAN architecture is thoroughly introduced. It enables unpaired image-to-image translation [21], [22], [86]–[89]. Although CycleGAN was introduced in 2017, it is still one of the prominent models used for unpaired image-to-image translation.

1.3.2 CycleGAN

CycleGAN is an unsupervised GAN able to learn the translation between two domains $\mathcal{X}$ and $\mathcal{Y}$ from unpaired data [82]. CycleGAN consists of two generators and two discriminators. The generator G_{XY} transforms images from domain $\mathcal{X}$ to domain $\mathcal{Y}$, while the generator G_{YX} transforms images from $\mathcal{Y}$ to $\mathcal{X}$. The discriminator D_X tries to distinguish between real images $x \in X$ and generated, also called synthetic, images $\hat{x}$. The discriminator D_Y is used to distinguish between real images $y \in Y$ and synthetic images $\hat{y}$.

Three different loss functions are used. For the adversarial loss (Fig. 1.4a), an input image from one domain is transformed to the other domain with a generator $\hat{y} = G_{XY}(x)$ and $\hat{x} = G_{YX}(y)$. The corresponding discriminator tries to distinguish between real and generated images. The cross-entropy loss for images y and $\hat{y}$ is formulated as:

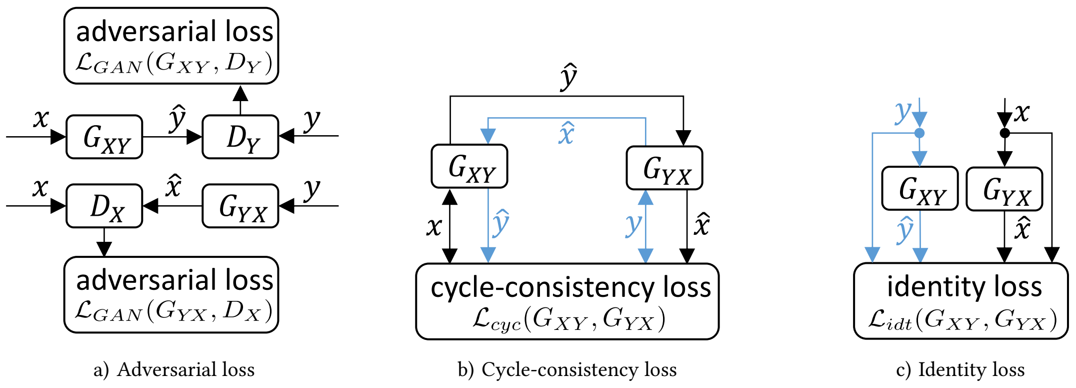


Figure 1.4 | Losses in CycleGAN training: (a) adversarial loss, (b) cycle-consistency loss, and (c) identity loss. Domain transfer is learned with the adversarial loss. The cycle-consistency loss is used to learn the mapping between domains, and the identity loss is added to preserve features present in both domains. The loss functions are applied to the blue inputs separately from the black inputs.

$$\begin{aligned}\mathcal{L}_{GAN}(G_{XY}, D_Y) = & \mathbb{E}_{y\sim p_{data}(y)}[\log D_Y(y)] \\ & + \mathbb{E}_{x\sim p_{data}(x)}[\log[1 - D_Y(G_{XY}(x))]].\end{aligned} \tag{1.2}$$

The loss $\mathcal{L}_{GAN}(G_{YX}, D_X)$ is formulated accordingly. Because the loss function suffers from vanishing gradients for samples on the correct side of the decision boundary, it is replaced by a least squares loss in practice [82], [90]:

$$\mathcal{L}_{GAN}(G_{XY}, D_Y) = \mathbb{E}_{y\sim p_{data}(y)}[D_Y(y)^2] + \mathbb{E}_{x\sim p_{data}(x)}[[1 - D_Y(G_{XY}(x))]^2]. \tag{1.3}$$

The discriminator is trained to generate the output one for real images and zero for synthetic images with Eq. 1.3. Just as in Eq. 1.1, $\mathcal{L}_{GAN}(G_{XY}, D_Y)$ needs to be minimized for the generator ($\mathcal{L}_{gen}$) and maximized for the discriminator ($\mathcal{L}_{disc}$). Using only the adversarial loss does not guarantee a meaningful translation between domains. Without further restrains, the input images serves only as a substitute for the random noise vector used in standard GANs.

To preserve important properties of the images in both domains, the mapping is further restrained by the cycle-consistency loss. As seen in Fig. 1.4b an image x is transformed to $\hat{y}$ by G_{XY}. Subsequently, the image is transformed back to $\hat{x}$ by G_{YX}. Ideally, x and $\hat{x}$ should be the same. The L1-norm is used to compare x to $\hat{x}$ in the standard implementation. The L1-norm of a $m \times n$ matrix $A(i,j)$ is defined by $\|A\| = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}|A(i,j)|$.

Also y is compared to $\hat{y}$ and the the cycle-consistency loss

$$\mathcal{L}_{cyc}(G_{XY}, G_{YX}) = \mathbb{E}_{x \sim p_{data}(x)}[\|G_{YX}(G_{XY}(x)) - x\|] + \mathbb{E}_{y \sim p_{data}(y)}[\|G_{XY}(G_{YX}(y)) - y\|], \tag{1.4}$$

can be defined.

The identity loss is added to further reduce the degree of freedom that generators have for transformation [82], [91]. The loss enforces an identity mapping, when feeding images from the output domain to the generator. An image y fed to the generator G_{XY}, which is normally used to transform an image x, should resemble itself. The loss is defined by:

$$\mathcal{L}_{idt}(G_{XY}, G_{YX}) = \mathbb{E}_{y \sim p_{data}(y)}[\|G_{XY}(y) - y\|] + \mathbb{E}_{x \sim p_{data}(x)}[\|G_{YX}(x) - x\|]. \tag{1.5}$$

Using the identity loss is not always possible out of the box. This is e.g. the case if grayscale images are in one domain and RGB images in the other domain, resulting in different numbers of channels for the input images. In this case, a possible workaround is to expand the grayscale images to three color channels with the same value in each channel.

The overall loss can be defined by:

$$\begin{aligned} \mathcal{L}(G_{XY}, G_{YX}, D_X, D_Y) = & \mathcal{L}_{GAN}(G_{XY}, D_Y) + \mathcal{L}_{GAN}(G_{YX}, D_X) \\ & + \lambda_{cyc}\mathcal{L}_{cyc}(G_{XY}, G_{YX}) + \lambda_{idt}\mathcal{L}_{idt}(G_{XY}, G_{YX}), \end{aligned} \tag{1.6}$$

where λ_{cyc} and λ_{idt} are used to scale the cycle-consistency and identity loss.

1.3.3 GANs for Image Analysis

GANs can be applied to a wide range of imaging applications [84], [92]. Many major architectures are introduced and evaluated with respect to their ability to synthesize meaningful, high-quality images [79], [82], [93], [94]. In contrast, GANs can be used as part of a complex image processing pipeline to solve segmentation and classification tasks. Usually, GANs are used to synthesize additional training data, but the GAN itself can also be used, e.g. to solve segmentation tasks. Because this work does not focus on the sole synthesis of images, but uses GANs integrated into image processing pipelines, the following sections focus on GANs used to solve real-world segmentation and classification tasks.

Classification

For classification tasks, GANs can be used to synthesize additional samples to train classifiers. GANs are often used in supervised classification pipelines if labeled data are scarce or a large class imbalance is present. Furthermore, the GAN can be used as a complex augmentation complementing standard image augmentation. Frid-Adar *et al.* use unconditional DCGANs (Deep Convolutional Generative Adversarial Networks) [96] to synthesize additional images for liver lesion computed tomography images [95]. For each of the three classes, cysts, metastases, and hemangiomas, a separate DCGAN is trained and used to synthesize images. Using the synthetic images increases the accuracy of a subsequent classifier by 7.1%, compared to standard augmentation. In addition, Kong *et al.* use a GAN to synthesize images of X-ray views of the facial bones [97]. ACGAN (Auxiliary Classifier GAN) [98], which is a conditional GAN, is used. In contrast to [95], the class label is provided as a condition during training and therefore a single GAN is capable of creating images from all classes. GANs are also used during the classification of hyperspectral images (HSIs) [99]–[101]. In [99], a conditional GAN is used to synthesize images. Instead of training an additional NN, the discriminator itself is used for the later classification of real images. The method is evaluated on three datasets of HSIs captured by a plane. Further applications are, for example, fruit quality and defect classification [102] or defect detection of cylinder surfaces [103].

Segmentation

GANs performing unsupervised image-to-image translation can be used for segmentation tasks if there are no paired labels for the target domain. There are two main approaches: one that utilizes synthetic labels to train a GAN, and another that leverages domain adaptation strategies.

Synthetic Labels A general pipeline how to use synthetic labels and GANs to omit manual labeling is deducted from previous work [22], [86]–[89], [104]–[107]. The pipeline is shown in Fig. 1.5. The GANs in this pipeline need to be able to perform unpaired image-to-image translation. Therefore, if GANs are mentioned, unsupervised GANs capable of performing unpaired image-to-image translation are meant in the remainder of this section. If different GANs are meant, it is explicitly mentioned.

All image segmentation pipelines start with a set of images from domain $\mathcal{Y}$, the target images which need to be segmented. The images contain e.g.: street scenes, manufactur-

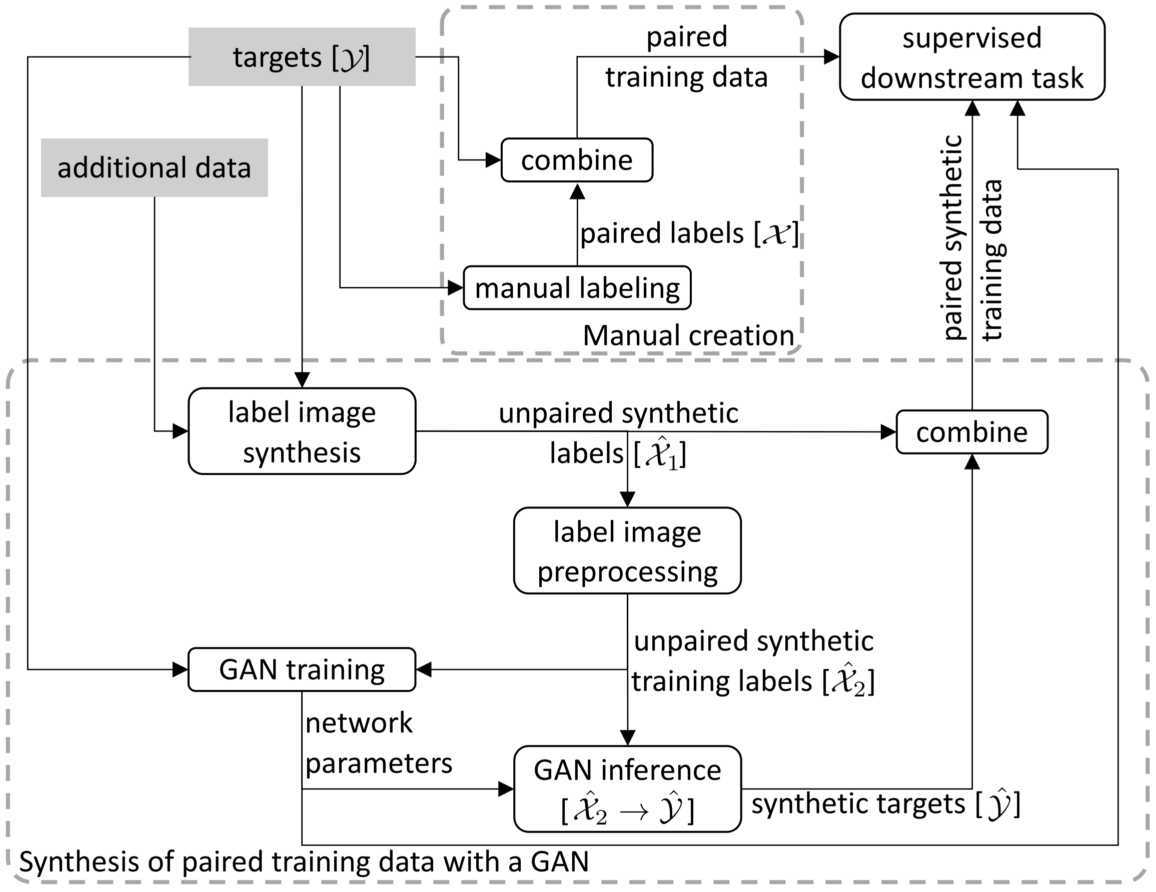

targets [$\mathcal{Y}$]
additional data
paired training data
supervised downstream task
combine
paired labels [$\mathcal{X}$]
manual labeling
Manual creation
paired synthetic training data
label image synthesis
unpaired synthetic labels [$\hat{\mathcal{X}}_1$]
combine
label image preprocessing
unpaired synthetic training labels [$\hat{\mathcal{X}}_2$]
GAN training
network parameters
GAN inference [$\hat{\mathcal{X}}_2 \rightarrow \hat{\mathcal{Y}}$]
synthetic targets [$\hat{\mathcal{Y}}$]
Synthesis of paired training data with a GAN

Figure 1.5 | Pipeline for the creation of paired training data from targets without paired labels for supervised downstream tasks. The top dashed gray box shows the steps for images from domain $\mathcal{Y}$ using manual labeling. The bottom dashed gray box shows the steps if unpaired synthetic label images together with a GAN to create synthetic training data are used while time consuming manual labeling is omitted.

ing images for quality control or microscopy images of cell nuclei. By default, no GANs are used, and the data is labeled manually. This is depicted in the top dashed gray box. Afterwards, the paired training data is used in supervised downstream tasks to train segmentation models [108]–[112].

The bottom dashed gray box shows the pipeline using GANs to omit manual labeling. All existing works have the common feature that unpaired synthetic label images from domain $\mathcal{X}$ need to be generated. The generated synthetic label images are exclusively an estimate of the domain $\mathcal{X}$ and are therefore denoted as $\hat{\mathcal{X}}_1$. Label image synthesis can be performed in many different ways and an overview is given in Section 1.4. If programming methods or modeling software are used, target information is included by inspecting the acquired images from domain $\mathcal{Y}$. Furthermore, shape information can be provided by additional data and even label images from previously acquired datasets can be used [89], [104], [105]. The unpaired synthetic label images need to be created with respect to the supervised downstream task. This could e.g. be instance labels for an instance segmentation task or semantic labels.

Because training GANs for unpaired image-to-image translation is a complex task, unpaired synthetic label images are often times preprocessed from domain $\hat{\mathcal{X}}_1$ to domain $\hat{\mathcal{X}}_2$. For example, instance labels are converted to semantic labels [86]. This is done because the instance label (e.g.: a number for each object) is arbitrary and not useful or even hinders the learning task. Furthermore, researchers reduce the domain gap between both domains to ease the learning task by simulating the image acquisition process. A point spread function (PSF) can be applied to simulate the response of a focused optical imaging system to a point source together with noise. As a result, the synthetic training images from domain $\hat{\mathcal{X}}_2$ are a good estimate of the real-world images from domain $\mathcal{Y}$ [89].

After training, the GAN is used to create a set of paired synthetic images from domain $\hat{\mathcal{X}}_2$ and domain $\hat{\mathcal{Y}}$ with an arbitrary number of pairs only dependent on the number of synthetic label images. Because the images in domain $\hat{\mathcal{X}}_2$ and domain $\hat{\mathcal{X}}_1$ are also paired, the pairs from domain $\hat{\mathcal{X}}_1$ and $\hat{\mathcal{Y}}$ can be used to create a set of paired synthetic training data. Afterwards, the synthetic training data can be used to train supervised methods for downstream tasks. Furthermore, many GAN architectures like UNIT and CycleGAN learn both domain translation directions and the GAN with the network parameters after training can be used directly for downstream segmentation [86].

The pipeline is used by [22] to segment 3D confocal microscopy images of cell nuclei in rat liver and kidney tissue. Also in [87], segmentation of cell nuclei of 3D confocal microscopy images is performed. A GAN called Aligned Disentangled Generative Adversarial Network (AD-GAN) is used. Instead of using two generators, a single generator is used. Adaptive Instance Normalization layers with one-hot encoded domain labels are used to encode the input and output image domains of the generator. Additionally, a single discriminator is trained to distinguish between reconstructed images (comparable to identity mapping in CycleGAN) and images mapped from one domain to the other. Instead of training an additional segmentation network, AD-GAN is used directly to create a semantic segmentation. Classical post-processing is applied afterwards for instance segmentation. Simulation of 3D data of Escherichia coli bacterial cells acquired by lattice light-sheet microscopy is performed in [88]. CycleGAN and a U-Net-like architecture for follow-up segmentation are used. A CycleGAN is used in [107] to generate targets from synthetic labels for bright-field images of cell cultures, assay images of live / dead Caenorhabditis elegans roundworms, and X-ray CT of metallic nanowire meshes. An additional histogram

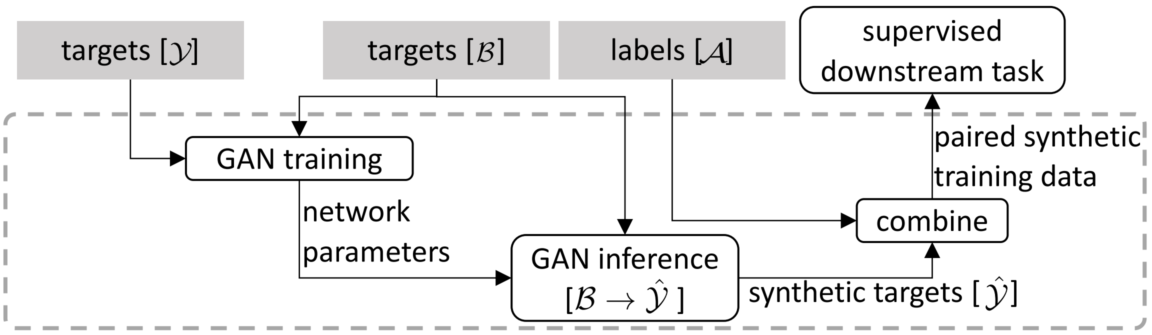

targets [$\mathcal{Y}$]
targets [$\mathcal{B}$]
labels [$\mathcal{A}$]
supervised downstream task
GAN training
network parameters
GAN inference [$\mathcal{B} \rightarrow \hat{\mathcal{Y}}$]
synthetic targets [$\hat{\mathcal{Y}}$]
combine
paired synthetic training data

Figure 1.6 | Domain adaptation with unsupervised GANs for segmentation tasks. No labels are present for targets from domain $\mathcal{Y}$, while paired labels from domain $\mathcal{A}$ exist for targets from domain $\mathcal{B}$. The GAN is used to transfer targets from domain $\mathcal{B}$ to domain $\hat{\mathcal{Y}}$. A segmentation network can be trained with synthetic targets from domain $\hat{\mathcal{Y}}$ and the paired labels from domain $\mathcal{A}$. Subsequently, the segmentation network can be used to infer labels for targets from domain $\mathcal{Y}$.

discriminator is added to help the generator learn the color distribution of real images. No additional segmentation network is used. An approach of [89] evaluates 3D data of human breast cancer cells imaged with confocal microscopy. The approach focuses on closing the domain gap between $\hat{\mathcal{X}}_2$ and $\mathcal{Y}$ to ease the learning task. Prototypes, which are single objects, are augmented and placed in an image. Afterwards, the image acquisition is simulated with, e.g. a convolution with a point spread function and noise simulation. The simulated image from domain $\hat{\mathcal{X}}_2$, is then refined with a 3D CycleGAN. Subsequently, a U-Net-like segmentation network developed by Scherr *et al.* is trained with the synthetic data [113]. This approach is closely related to domain adaptation introduced in the remainder of this section. Instead of a real-world paired dataset, the paired dataset is fully synthetic.

Most of the works utilize a CycleGAN adapted to the data. The pipeline is used mainly for biological and medical data and in material science. In these fields, manual label generation is often time consuming and complicated, whereas unlabeled data can be acquired fast and cost effective by automated acquisition systems. Therefore, the benefit of using GANs is large.

Domain Adaptation If labeled data from a target domain close to the unlabeled target domain are present, unsupervised domain adaptation can be used (see Fig. 1.6). Instead of synthesizing label images, a translation between two target domains is learned. Images of the labeled domain $\mathcal{B}$ are transformed into the unlabeled domain $\mathcal{Y}$. Synthetic images from domain $\hat{\mathcal{Y}}$ and the corresponding labels of domain $\mathcal{A}$ can be used to train segmentation networks. This is used, for example, for eye fundus vessel segmentation. The domain gap between different eye fundus vessel datasets is so large that a neural network trained on one data set produces poor results on the other data set. A CycleGAN is used to transfer images between domains and the segmentation quality is improved [114]. GANs can also be applied if the domain gap is not only present due to the image acquisition (e.g. scanner, illumination, resolution) but due to different acquisition methods. Many studies focus on the transfer between computed tomography (CT) and Magnetic Resonance Imaging (MRI) [115]–[117]. While CT is good at visualizing bones and dense structures, MRI provides good soft tissue contrast and is therefore used to visualize organs such as the brain or muscles. In addition, CT exposes the body to radiation. Therefore, it is desirable to replace one of the two recordings with GANs. Again, GANs are used to transfer labeled images to the unlabeled domain. However, it should be noted that a GAN can only use the information available in the respective image modality. Information that is missing in the input domain is randomly added during translation to the target domain. A CycleGAN is also used to transfer between renal tissue sections of histopathological whole slide images (WSIs) from two different centers [118]. Due to the large variation in staining, the synthetic training data created by the GAN improved the subsequent U-Net segmentation.

1.3.4 Advanced Tiling Strategy for Patch-based Inference

Bel et al. [118] performed domain adaptation with a CycleGAN to adapt the staining of histopathological 2D images from one medical center to the staining from a different medical center. Staining varies due to the different imaging modalities applied in different medical centers. Large histopathological images can be synthesized by processing the input image patch-by-patch during inference and merging the patches after they have been translated to the other domain by the GAN. This approach is called *simple tiling*. Using *simple tiling* resulted in a poor image quality, because the coloring of the single output patches differed and the transition between patches was clearly visible.

A novel tiling approach is introduced to mitigate artifacts commonly associated with *simple tiling*. This approach is called *weighted tiling*. The innovative approach involves several key adaptations to simple tiling:

1. Instead of using non-overlapping patches, large overlapping patches are processed. This adjustment results in a greater similarity between the mean and standard deviation of adjacent patches during inference. Consequently, the output in the overlapping areas is more similar, if standard instance normalization is used.

2. After processing by the GAN, overlapping patches are cropped to alleviate border effects caused by padding and variations in the receptive field between the border pixels and those in the center of the patch.

3. The cropped patches still maintain an overlap, and these overlapping regions are seamlessly blended together using a weight map, ensuring a smooth transition from one patch to the next.

Although *weighted tiling* has demonstrated its ability to generate high-quality output without requiring changes to the GAN training, it has two notable drawbacks. (i) The use of the proposed large overlap of 75% significantly increases the execution time for single-dimensional data by almost a factor of four. Furthermore, the inference time scales exponentially with the number of input dimensions. For 3D data, this results in a substantial 64-fold increase in inference time compared to *simple tiling*. (ii) *Weighted tiling*, is exclusively a post-processing method, and there is still no guarantee of a one-to-one mapping and a reliable output if an object spans two adjacent patches. This inherent limitation can lead to errors in the final image.

1.3.5 GAN Performance Metrics

Metrics are used to compare the performance of GANs. Therefore, several metrics have been proposed. In the following, metrics that need paired data and metrics that do not need paired data are discussed. In addition, a metric for downstream segmentation tasks is introduced. The evaluation of downstream tasks carried out with the help of GANs allows conclusions to be drawn about the performance of GANs.

Paired Image Metrics

Paired metrics can be used, if the GAN is trained in a supervised (paired) manner (see Section 1.3). The desired output for each input is known and the GAN output can be compared to it.

A simple quality measurement is the Peak Signal-to-Noise Ratio (PSNR). The squared maximum possible pixel value is divided by the averaged per pixel mean squared error (MSE) of two images. It is defined by:

$$\text{PSNR} = 10 * log_{10}\left(\frac{\text{max}_I^2}{\overline{\text{MSE}}}\right). \tag{1.7}$$

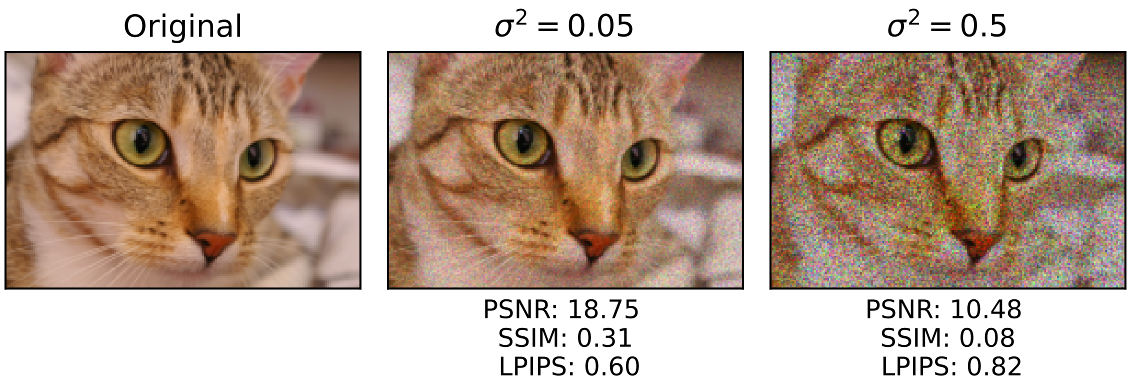


Figure 1.7 | Exemplary image and paired image metrics if the image is altered by speckle noise with increased variance and a mean of zero. The paired metrics PSNR, SSIM, and LPIPS are used to compare the altered images with the original image.

The $\overline{\text{MSE}}$ is the averaged per pixel MSE and $\max_I$ is the maximum possible pixel value. The PSNR assumes pixel-wise independence and ranges from zero to infinity, while a high value refers to more similar images. One drawback is that it does not reflect human visual perception. For example, blurring an image causes a small change in PSNR and a big change in human perception [119], [120].

Also the Structural Similarity Index Measure (SSIM) introduced by Wang *et al.* compares two images [120]. Each image is decomposed into luminance, contrast, and structure measurements. Afterwards a similarity index is calculated for each of the measurements, and a weighted combination of the three similarities is calculated. Furthermore, the SSIM is aggregated over local patches to mimic the human visual system. The final SSIM score is in the range $[-1, 1]$, with one for two identical images.

According to Zhang *et al.*, SSIM and PSNR still fail to account for the nuances of human perception due to their shallow functions [119]. They introduced the Learned Perceptual Image Patch Similarity (LPIPS), which utilizes NN activations to measure similarity. The features of a layer of the NN are normalized and scaled. Afterwards, the L2 distance between the features of both images is calculated. A small distance, and therefore a low value, means that the images are perceptual similar. They showed that the approach worked best on their data with a standard VGG network [121] pre-trained on ImageNet [3].

The scores of different paired metrics on an exemplary image altered with speckle noise are shown in Fig. 1.7. It shows that the scales and the scores of the PSNR, the SSIM and the LPIPS differ severely. Therefore, one cannot compare scores between different metrics. As a result, it is difficult to make a non-biased assessment of whether images are similar or not.

Unpaired Image Metrics

It is not possible to use the above metrics for unpaired data. In addition, it is a much more difficult problem to assess whether two datasets are similar if there are no image pairs, as direct comparison of pixels or image areas is not possible.

The Inception score (IS) is a metric which does not need paired data [122]. A neural network (the Inception-v3 classification model) pre-trained on ImageNet is used to classify the synthetic data. The output of the classification model is used to determine the quality of the synthetic data. Two conditions lead to a high IS score: High-quality images should have a conditional label distribution $p(y|x)$ with low entropy. This means that the output layer of the classification model should have a peak for a single class for each image. Additionally, the GAN should generate diverse images, corresponding to an even distribution across the output classes of the pre-trained classification model. An even distribution of the synthetic images across the output classes of the pre-trained classification model results in a high entropy of the marginal distribution of the output of the classification model. Combining both parts results in:

$$\text{IS} = \exp(\mathbb{E}_{x \sim p_{gen}} \text{KL}(p(y|x) || p(y))), \tag{1.8}$$

where KL is the Kullback-Leibler (KL) divergence. KL-divergence measures the difference between two probability distributions, with a smaller value indicating greater similarity. Because both parts of the score rely on an output class distribution, the IS is not suitable for GANs trained to predict outputs from a single class. This means that the IS cannot be used for GANs trained to synthesize only one distinct type of image, e.g., only cats or only dogs, while it can be used for a GAN able to synthesize images of cats and dogs.

The Fréchet Inception Distance (FID) is also used to compare two unpaired sets of images. It also uses a pre-trained classification model, which is the Inception-v3 network trained on ImageNet [123], [124]. Instead of using the classification output, multivariate Gaussians are fitted to the distribution of images in the feature space of the Inception-v3 network. The multivariate Gaussians are fitted to the distributions in feature space of the first set of images and the second set of images separately. The Fréchet distance is used to quantify the distance between the multivariate Gaussians of the different sets of images. A lower score indicates more similar images.

Like the FID, the Kernel Inception Distance (KID) also uses the feature space representations of images by utilizing an Inception-v3 model pre-trained on ImageNet [125]. The Maximum Mean Discrepancy (MMD) between the feature space representations of two sets of images using a polynomial kernel is measured. Also, for KID, a lower score refers to more similar image distributions.

Downstream Task Metrics

The paired and unpaired evaluation metrics described above aim to mimic human perception. It is unclear whether synthetic images close to human perception are necessary if synthetic images are used to train models for downstream tasks such as classification or segmentation. Additionally, the synthesis quality can be measured with metrics applied to the results of the trained downstream model if labeled data are present. There are a multitude of metrics for classification, detection, and segmentation [126].

A metric for instance segmentation is the advanced aggregated Jaccard index (AJI$^+$), which pairs ground truth objects with predicted objects. The intersections of all pairs are aggregated (I), and the unions of all pairs are aggregated (U). The number of pixels of all unmatched ground-truth and predicted objects are added to the sum of unions. At the end, the AJI$^+$ is calculated by AJI$^+$ $=$ I/U. The plus in AJI$^+$ represents the matching strategy. Each predicted object is paired with at most one ground-truth object. Furthermore, the pairing is carried out to maximize the AJI$^+$ score. A perfect segmentation results in a score of one, while zero is reached if no predicted objects match ground-truth objects [65], [127].

Network Metrics

A metric that does not need any additional data is the α-metric [128]. It uses only the weights of the individual layers in a neural network to determine whether a neural network is well-trained. The α-metric has been compared, e.g., to the test accuracy of classification networks and a strong correlation is observed. The calculation of α_i for a single layer i is based on the theory of heavy-tailed self-regularization. A singular value decomposition of the weight matrix W of a layer is performed. The histogram of eigenvalues is used to fit a power law function with the exponent α_i to the tail of the histogram. A lower α_i indicates a better trained layer. To evaluate an entire network, the average of the α_i values of all layers of the neural network can be calculated, which is then called the α-metric. The WeightWatcher (WW) tool can be used to calculate the α-metric[2].

[2]An implementation is available at https://github.com/CalculatedContent/WeightWatcher, accessed 26.01.2024.

1.4 Label Image Synthesis

As described in Section 1.3.3, GANs can be part of segmentation pipelines. If a related target domain is present, a GAN can be used to adapt the unlabeled target domain to the labeled target domain or vice versa. If this is not possible, synthetic labels from the synthetic label domain can be used to train the GAN together with the target images from the target domain to synthesize targets paired with the images from the synthetic label domain. This section focuses on methods for synthesizing label images that can be used to train a GAN for unpaired image-to-image translation. Instance segmentation labels can also be used for segmentation and detection tasks. While detection is also an important task in image processing, most studies that use synthetic data focus on segmentation because manual labeling is considerably more laborious for segmentation than for detection.

Label images for segmentation tasks are synthesized in different ways and can be grouped into programming methods, Statistical Shape Models (SSMs), phantom-based, and modeling software. A summary is given in Table 1.1. The table shows that label synthesis is most prominent for biological and biomedical use cases.

Table 1.1 | Label image synthesis methods. Synthetic segmentation datasets are most common in a biological and biomedical context. For biological data, programming and Statistical Shape Models (SSMs) dominate.

Data	Method	Tools	Dim.	First Author
cells, nuclei	programming	Circle distortion, spline interpolation	2D	A. Lehmussola [129]
nuclei	programming	PDE-based sphere distortion	3D	D. Svoboda [130]
cells	programming	Biophysical modelling	3D+t	T. Rudge [131]
plant root system	programming	L-system process	2D+t	L. Benoit [132]
nuclei	SSM	FSD, naive Bayesian sampling	2D	P. Malm [133]
nuclei	Phantom	Phantoms from existing programming	3D+t	J. Stegmaier [134]
street scenes	modelling software	Unity development platform	2.5D+t	G. Ros [135]

Continued on next page

Table 1.1 | Label image synthesis methods (Continued)

nuclei	programming	Ellipsoid	3D	C. Fu [106]
bell pepper	modelling software	Distribution sampling, procedural structure in PlantFactory 2015 Studio	2.5D	R. Barth [136]
cell membranes	programming	Voronoi Diagram	2D	D. Eschweiler [104]
nuclei with cytoplasm	SSM	Joint EFD, GMM sampling	2D	M. Scalbert [137]
neurons, worms, nanowires	programming	Different shapes, random walk	2D	S. Ihle [107]
kitchen objects	Phantom	Phantoms placed in kitchen images	2D	R. Sauges-Tanco [138]
nuclei	programming	2D Bézier Curves adapted to 3D	3D	A. Chen [139]
nuclei, microvilli	programming	Circles, rectangles deformed by CycleGAN	2D	Q. Liu [140]
cell membranes	SSM	Spherical harmonics	3D	D. Eschweiler [105]
echocardiography	SSM	PCA, random sampling	3D	A. Gilbert [141]
surgical tools	modelling software	Rendered surgical tools placed	2D	E. Colleoni [142]

Programming Methods

Programming methods are methods in which label images are synthesized by placing objects created algorithmically. Programming is the prominent method for cell synthesis, but it is also utilized in material science or botany. Lehmussola *et al.* synthesize cell nuclei by defining points on a circle that have the same angular distance from each other [129]. Afterwards, the points are distorted and a new object is created by spline interpolation of the points. 3D label images for HL-60 cell nuclei and granulocytes are created in [130] by placing ellipsoids, which are deformed with partial differential equations (PDEs). The object boundary is viewed as a deformable surface and the deformation is realized with fast level set methods and artificial noise as a speed function. Rudge *et al.* focus on modeling large amounts of 3D+t microbial biofilms, where more than 30.000 objects can be placed in under 30 minutes of computation [131]. They model the rod-shaped bacteria by a cylinder capped with hemispherical ends and establish a biophysical model for the cell properties like position, interaction, volume, and division. Fu *et al.* place 3D ellipsoids without further distortion [106]. Modeling of 2D cell membranes is performed by Eschweiler et al. by placing seed points and constructing a Voronoi diagram with them. Ihle *et al.* synthesize 2D label images for neurons, worms, and nanowires [107]. Neurons are modeled by placing ellipses, while worms are modeled by a random walk with momentum for the curvature. Furthermore, for the nanowires, rectangles are placed in the synthetic label images. Chen et al. use Bézier curves to model nuclei in 2D and adapt the object size to create slices of 3D nuclei. Liu *et al.* show, that the early stages of a trained CycleGAN can be used to distort objects in synthetic ground truth images [140]. They demonstrate the process on nuclei, where the initial objects are circles, and on microvilli, where the initial objects are rectangles. Finally, Benoit *et al.* model 2D+t plant root systems with the L-system process, which is used to model plant architecture and has also been applied to root systems in the past [132], [143].

Especially if simple objects are placed with programming methods, the influence of the shape on the segmentation results has not been investigated comprehensively in the past. Most of the time, distributions are assumed for the properties of the placed objects. The distributions of the properties of the objects have to be determined empirically. However, there is a loss of downstream segmentation quality if the domain of the synthetic labels does not coincide with the domain of the real labels (which do not yet exist at this time).

Statistical Shape Models

Statistical Shape Models are "geometric models that describe a collection of semantically similar objects in a very compact way" [144]. In contrast to programming methods, a set of existing labels must be available. If SSMs are used for label image synthesis, the workflow is as follows:

1. Extract annotation mask for each object from label images, and
2. apply SSM on each object.
3. Estimate the probability density function from the population of SSM parameters.
4. Sample new objects from the probability density function.
5. Convert the new object to an annotation mask.

In [133], Fourier Shape Descriptors (FSDs, [145]) are used as a SSM to model 2D cell nuclei, while naive Bayesian theory is used to generate new objects. Elliptic Fourier Shape Descriptors (EFDs, [146]) are used as a SSM to model 2D nuclei and the corresponding cytoplasm in [137]. A Gaussian Mixture Model (GMM) is used to sample from the joint FSD parameters for the cell nucleus and cytoplasm. Furthermore, a similarity tolerance is used to omit sampling of objects that are too different from a reference object. Spherical harmonics are used to model 3D cellular structures in [105]. Gilbert *et al.* use SMMs to simulate 3D echocardiography images [141]. They perform a statistical shape analysis with a Principal Components Analysis (PCA) of the existing labels and sample from the first nine modes within two standard deviations of the mean. Therefore, they are able to capture 90% of the total variation.

The inherent problem with SSMs is that a set of existing label images needs to be available. If the labels are from a related domain, one can often switch from synthesizing labels to domain adaptation and omit the label synthesis task. Therefore, usage is limited. On the other hand, most SSMs have a parameter to alter the achievable model complexity. Similarly to programming methods, it is unclear how to adjust this parameter. Using more simple SSMs reduces the number of labels needed.

Phantom Based Approaches

Phantom based approaches use existing masks acquired by manually labeling of single objects. The masks of the single objects are distorted to resemble the real-world distribution of objects. Distortion can e.g. be performed by rotation, scaling, or affine transformations. The distorted objects are then placed in the synthetic label images. In contrast to SSMs, no parameters are extracted and the masks are not converted to a geometric model. Stegmaier *et al.* use phantoms to create 3D+t data of cell nuclei [134]. Existing masks are extracted from the work in [147]. Phantoms are also used for synthesis of label images for kitchen scenes with complex objects like knifes, pots potatoes and more. The phantoms are placed in existing kitchen scenes [138]. Compared to programming methods, phantom based approaches require existing masks and the distortions are often times more complex. Both need to assume the property distributions of the objects in the real-world images.

Modeling Software

Modelling software is used mainly for complex objects where programming is not feasible and no label images are available for SSMs. Task-specific software is used to create 2D or 3D models of objects. Modelling software is usually used to create complex scenes for computer games, movies, art or more, but can also be used to synthesize label images. Colleoni *et al.* synthesize 2D surgical images by placing simulated surgical tools from a CoppeliaSim[3] da Vinci robot [142], [148]. Due to the diversity and the large amount of information present, street scenes are complex to synthesize. Ros *et al.* synthesize them using the Unity development platform[4], where they created a virtual city [135]. The model allows for extraction of 3D+t data, while 2.5D+t (stereoscopic images, depth information) is extracted in the article. Barth *et al.* simulate label images for bell pepper [136]. Properties regarding the nodes, sideshoots, leaf stem, and more are extracted from real-world bell pepper. Subsequently, the PlantFactory 2015 Studio software[5] is used to synthesize images. PlantFactory is usually used to model plants for video games. Although the models in PlantFactory are 3D, the authors extract 2D images with additional depth labels.

Although approaches using modeling software do not need labeled data, it is the most laborious approach. A modeling software is needed together with a skilled person creating the models. Although there are methods to partially automate model creation, many models are created manually.

1.5 Open Questions

The rise of supervised methods for image analysis results in an increasing need for labeled training data. Labeling is mainly performed manually, resulting in a time-consuming and tedious task with high inter-observer variability. Recently, ways to replace this manual process with unpaired image-to-image translation creating synthetic training data are explored. This was made possible by the progress in unpaired image-to-image translation with GANs. Usage of GANs for image synthesis increases rapidly, and new architectures and concepts are developed constantly. Nonetheless, several open questions regarding the use of GANs to omit manual labeling in the context of large-scale 2D and 3D image analysis remain:

- Performance of GANs for unpaired image-to-image translation is carefully evaluated if two real-world domains are transferred into each other. Meanwhile, label images are created synthetically if GANs are used to generate synthetic training

[3]Available at https://www.coppeliarobotics.com, accessed 13.03.2024.
[4]Available at https://unity.com, accessed 13.03.2024.
[5]Available at https://info.e-onsoftware.com/plantfactory/overview (current version), accessed 13.03.2024.

data for downstream tasks. The influence of these synthetic label images on the performance of GANs and subsequent downstream tasks is not considered and quantitatively evaluated. Furthermore, synthetic label images must match the underlying label image properties of the corresponding real-world images. This is a challenging task, since no label images are present for the real-world images. As a result, the synthetic label images have to be created on the basis of assumptions for the object properties. The influence of different properties needs to be analyzed. Furthermore, an approach to synthesize high-quality training data without a large hyperparameter study is missing. This could reduce the time needed and increase the quality of the synthetic training data.

- GANs that perform unpaired image-to-image translation are trained for a fixed number of epochs because no paired validation data are present. Furthermore, training GANs is unstable and the last epoch may not yield the best results compared to earlier epochs. Currently, instead of using an earlier epoch, the GAN is trained multiple times until the desired quality is reached. Training of multiple GANs is not desired as the training is computationally expensive. The computational expense and therefore the time needed to train the GANs as well as the uncertainty how many GANs to train until the desired results are acquired limit the usage of GANs performing unpaired image-to-image translation. It is still unclear whether and how validation metrics can be integrated into GAN training. In addition, there is no workflow to test metrics for their suitability as validation metrics for GANs that generate data for downstream segmentation.

- Existing approaches using GANs for unpaired image-to-image translation focus on generation of small images, while the demand for large-scale 2D or 3D data is increasing. Therefore, inference is performed primarily by splitting an image into individual patches, transferring the patches to another domain with the GAN, and reassembling the patches into one large image. Patch-based inference results in errors in the generated synthetic data. However, there are no studies that systematically assess how and under which circumstances errors occur due to patch-based inference. It is not clear how to quantitatively evaluate methods with respect to the errors that occur during patch-based inference. Furthermore, it is not clear how a benchmark dataset must be structured to be used for automated quantification of patch-based errors.

1.6 Objectives and Thesis Outline

The open questions in Section 1.5 lead to the following central objectives:

1. A coherent approach has to be developed to systematically assess the influence of synthetic label images on workflows that use GANs to omit manual labeling for downstream segmentation.

2. The approach has to include the development of benchmark datasets that can be used for quantification.

3. The influence of the individual label image properties on downstream segmentation performance has to be quantified and recommendations for the creation of synthetic label images have to be provided.

4. A workflow to incorporate validation metrics into GAN training has to be established.

5. The resulting workflow has to be used to quantitatively assess the suitability of metrics for early stopping or selection of the best epoch of GANs using multiple real-world benchmark datasets.

6. A new fully synthetic benchmark dataset that enables quantification of errors that occur during patch-based GAN inference has to be designed.

7. A fast and flexible method that can be integrated into existing GAN architectures to reduce errors that occur during patch-based inference needs to be developed. The method must have small additional computational requirements for training and inference.

8. The developed method must be quantitatively compared with existing methods regarding the errors introduced due to patch-based inference using the fully synthetic benchmark dataset.

In order to realize the objectives, a systematic approach to analyze the influence of synthetic label images on a downstream segmentation task is developed in Chapter 2 and the quantitative results on the benchmark dataset are used to provide practical guidance for label image synthesis. Chapter 3 examines different ways of integrating validation metrics into the training of GANs performing unsupervised image-to-image translation.

Different validation metrics are quantitatively compared regarding their ability to be used for early stopping and epoch selection. Subsequently, the patch-based inference using GANs is investigated in Chapter 4. Therefore, a benchmark dataset is presented and a new method to improve the image quality if inference is performed patch-based is developed and tested on the benchmark dataset. Finally, in Chapter 5, a conclusion to the work and an outlook on future possibilities to advance the usage of GANs in image processing pipelines is given.

Label Image Property Influence 2

If GANs are used to create fully synthetic paired segmentation datasets, all subsequent steps are directly or indirectly dependent on the synthetic label images. Therefore, this chapter evaluates the influence of synthetic label images and their properties on fully synthetic datasets created with GANs.

Although GANs are already used to synthesize training data, it is still an open question how much the properties of objects in the synthetic label images influence GAN training and the following downstream tasks. As described in Section 1.4, programming is the prominent method to create synthetic label images. A quantitative analysis of the influence of object properties can provide a better understanding of the requirements regarding the object properties in the synthetic label images, and therefore drastically reduce the development time of the label image synthesis pipeline.

A theoretical analysis of label image properties and their influence on synthetic datasets created with GANs is given in Section 2.1. For the first time, the theoretical analysis conceptually examines the possibilities that exist for the translation of object properties between domains and their influence on synthetic datasets used for downstream segmentation. The theoretical analysis is followed by the introduction of a new pipeline to quantify the label image property influence. Because there are no benchmark datasets for quantification, two benchmark datasets are designed and described in Section 2.3. Subsequently, the pipeline and the benchmark datasets are used to carry out experiments in Section 2.4. The quantitative results are shown in Section 2.5 and can be used to increase the quality of synthetic training data. A discussion in which the results are analyzed and with which the effort required to create synthetic label images can be reduced is given in Section 2.6[1].

[1]The following sections extend the methods and the results of [86].

2.1 Synthetic Label Image Properties and Implications for Synthetic Datasets Created with GANs

If programming methods are used to create the synthetic label images, it is necessary to model the properties of the object in the synthetic label images. Properties are divided into three categories to allow a theoretical and a subsequent quantitative evaluation [50]. In a real-world setting, properties resemble distributions over a given dataset. Properties can be subdivided into any desired granularity, depending on the data and the synthesis method, and the distributions can be arbitrarily complex. The three object property categories are:

Spatial properties P_{Sp} relating to the position of objects in an image are grouped in this category. The available properties depend on the target images. The spatial property can be divided into the location P_{Sp}^{L} of objects (which can be modeled by the center of gravity) and the total number of objects in an image P_{Sp}^{N}. Additional granularity can be achieved by adding properties such as crowding, which can be modeled by neighbors within a distinct radius or a placement probability map.

Shape properties P_{Sh} describe the contour of objects. For less complex shapes, ellipses or polygons can be used to describe the shape. For complex shapes, Statistical Shape Models (SSMs) can be used to describe the shapes. More complex properties must be defined, if the objects have holes.

Color properties P_{C} describe the visual appearance within an object, which also applies to grayscale or multichannel images.

An example of the properties of a car in the Cityscapes dataset is shown in Fig. 2.1 [149]. The center of gravity of the truck is marked with a green rectangle and corresponds to P_{Sp}^{L}. The number of trucks corresponds to P_{Sp}^{N}. The outline and therefore the shape is marked in blue. This property can, for example, be subdivided into the contour P_{Sh}^{C} normalized to surface area one and the surface area P_{Sh}^{A}. Segregation of both properties can reduce the complexity of the SSM. In case all objects are equally sized, this split is not needed, and both can be captured in a single property. The color property P_{C}, describes the pixel values inside the blue outline. For the depicted truck, this is a complex property which is not easily defined. One possible color property can be the prominent color, which is yellow in this case.

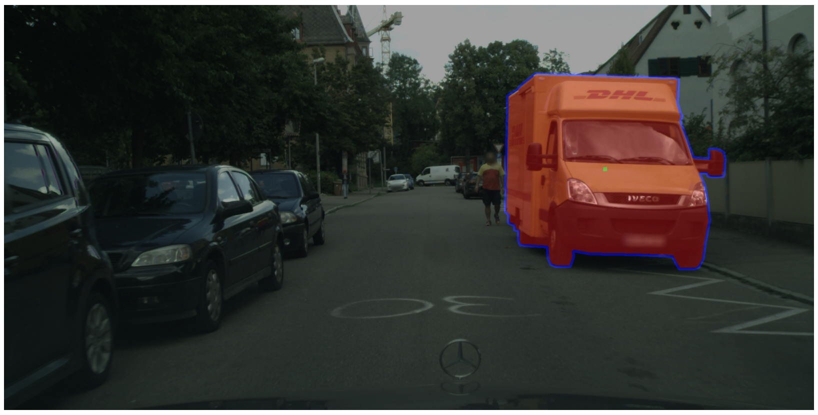


Figure 2.1 | Object properties: spatial (green dot, center of gravity), shape (blue contour) and color (everything inside the contour). Adapted from the Cityscapes dataset [149].

If a GAN is trained to perform unpaired image-to-image translation between two domains, four possible outcomes can occur for each property:

1. The property is equal in both domains, and the GAN does not change it, when transforming images between both domains.

2. The property is equal in both domains, but the GAN changes it when transforming images between both domains.

3. The property is not equal in both domains, and the GAN learns to change the property when transforming images between both domains.

4. The property is not equal in both domains, but the GAN does not learn to change the property correctly when transforming images between both domains.

Standard unpaired image-to-image GANs learn an entangled feature representation of the image properties. An entangled feature representation means that one property of the image cannot be changed without changing other properties. Entanglement limits the generator‘s ability to independently manipulate different properties of objects in an image because standard discriminators use all properties to classify images into real and synthetic ones and the generators are trained to transfer the entangled properties between both domains. Therefore, outcomes 2 and 4 do not occur for a successfully trained GAN.

Outcomes 1 and 3 result in restrictions on the use of GANs for segmentation data synthesis. These restrictions are theoretically analyzed in the following with a special focus on the different property categories P_{Sp}, P_{Sh} and P_C.

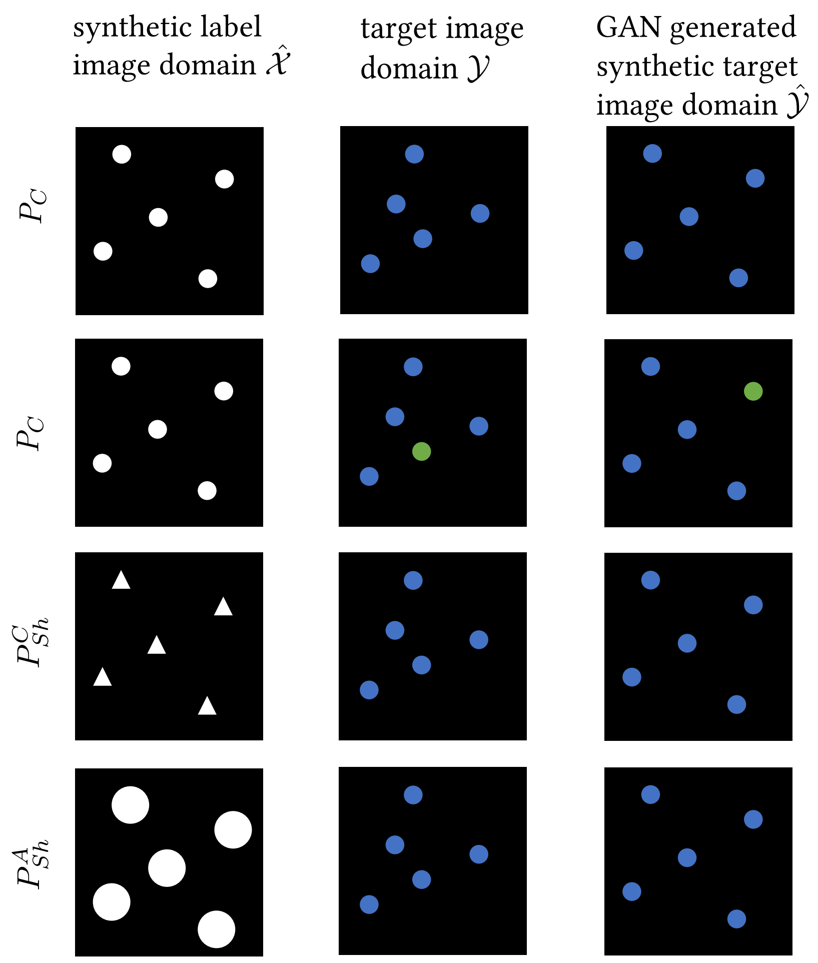

synthetic label image domain $\hat{\mathcal{X}}$
target image domain $\mathcal{Y}$
GAN generated synthetic target image domain $\hat{\mathcal{Y}}$
P_C
P_C
P_{Sh}^C
P_{Sh}^A

Figure 2.2 | Example images for synthetic label image domains $\hat{\mathcal{X}}$ and target domains $\mathcal{Y}$ with the corresponding synthetic target image domains $\hat{\mathcal{Y}}$. In the first and second row, only the color property P_C differs. In the third row, the contour P_{Sh}^C is varied, while the area of objects P_{Sh}^A is varied in the fourth row. The GAN learns to transfer the properties to the target domain $\mathcal{Y}$ for all cases. In the first row, all white objects are transferred to blue, while in the second row, the distribution of different colors is also considered. In the third and fourth row, the shape property (P_{Sh}^C or P_{Sh}^A) together with the color is transferred.

Fig. 2.2 shows example translations between the synthetic label image domain $\hat{\mathcal{X}}$ and the target domain $\mathcal{Y}$. The target domain $\mathcal{Y}$ consists of colored circles randomly placed in the images. Each row shows an example for a different combination of $\hat{\mathcal{X}}$, $\mathcal{Y}$ and the corresponding synthetic target image domain $\hat{\mathcal{Y}}$. The first row shows an example of the color property P_C. $\hat{\mathcal{X}}$ is designed to resemble all properties of $\mathcal{Y}$, except P_C. The color is set to white. If the GAN is properly trained, it learned to color the circles blue (outcome 3). All other properties are left as is (outcome 1). In the second row, the target domain $\mathcal{Y}$ is more complex and yields a distribution of colors. Therefore, the color distribution is the same in $\hat{\mathcal{Y}}$. The shape property is subdivided into P_{Sh}^C (third row), which describes

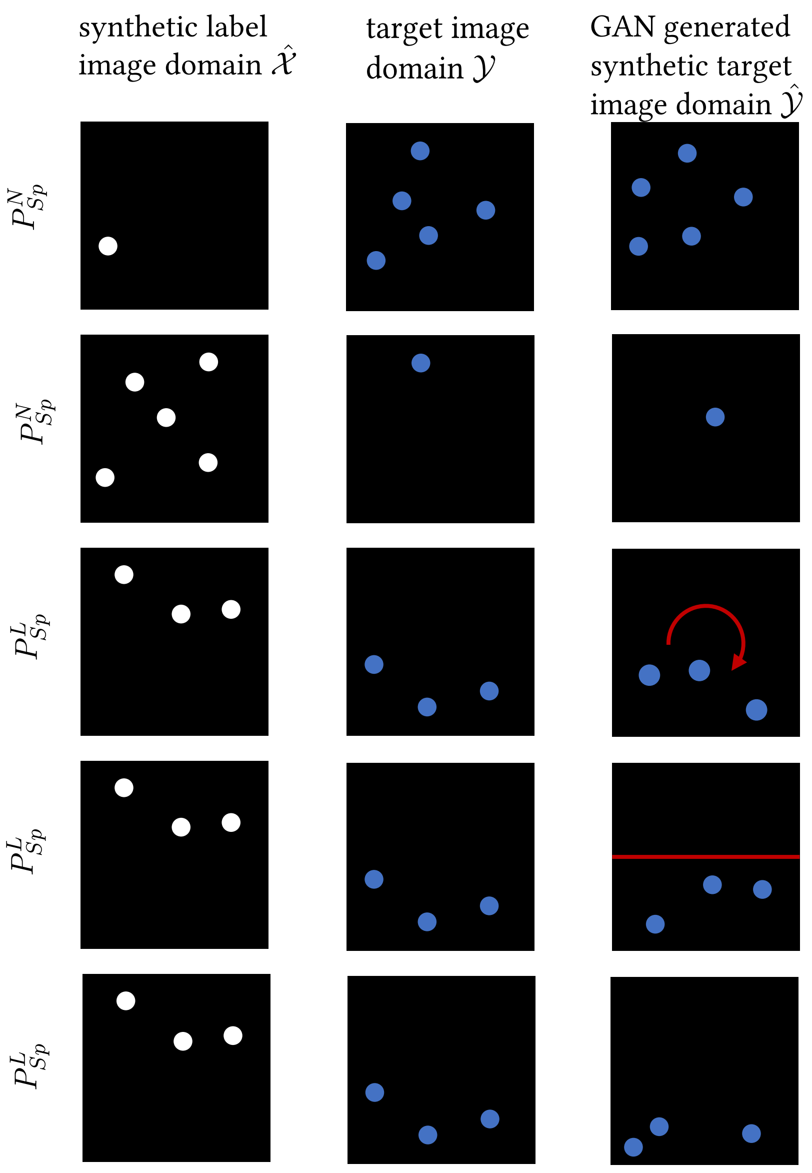

synthetic label image domain $\hat{\mathcal{X}}$
target image domain $\mathcal{Y}$
GAN generated synthetic target image domain $\hat{\mathcal{Y}}$
P_{Sp}^{N}
P_{Sp}^{N}
P_{Sp}^{L}
P_{Sp}^{L}
P_{Sp}^{L}

Figure 2.3 | Example images for synthetic label image domains $\hat{\mathcal{Y}}$ and target domains $\mathcal{Y}$. Different examples are shown for the spatial properties and the transfer. In the first row, the number of objects P_{Sp}^{N} is reduced in the synthetic label image domain. In the second row, the number of objects P_{Sp}^{N} is reduced in the target domain. In the last three rows, the spatial location P_{Sp}^{L} differs in both domains. For all cases, the GAN learns to adapt the properties to the target domain. This results in circles added in the first row and removed circles in the second row. While $\hat{\mathcal{X}}$ and $\mathcal{Y}$ are the same for the last three rows, the GAN can learn different translations. In the third row, the GAN rotates the input image to adapt to $\mathcal{Y}$. In the fourth row, the GAN mirrors the objects. Both translations are indicated in red. In the last row, the GAN randomly shifts all objects. While rotation and mirroring can be desired learning tasks, random shift is generally not desired. It cannot be controlled which circles are added or removed and how the shift of objects for P_{Sp}^{L} is performed. Because these changes cannot be controlled, the domain of synthetic label images has to be changed if the changes are undesired.

the area normalized contour and P_{Sh}^{A} (fourth row), which describes the area of objects. Varying P_{Sh}^{C}, is done by placing triangles in domain $\hat{\mathcal{X}}$. The GAN therefore learns the transfer for the color property P_C and object contour property P_{Sh}^{C}. This is depicted in the third row. In the bottom row, the circles are enlarged in the domain $\hat{\mathcal{X}}$ and P_{Sh}^{A} varies for both domains. It has to be denoted that keeping the center of gravity in both domains is only one possible solution for the domain transfer. A GAN can also learn to place the circles in the synthetic target domain $\hat{\mathcal{Y}}$ at the top of the triangles or at random positions inside the enlarged circles.

The same is true for the spatial properties. If there is a mismatch in the number of objects P_{Sp}^{N} in both domains, the objects are removed or added by the GAN. This is shown in Fig. 2.3. In the first row, too few objects are placed in domain $\hat{\mathcal{X}}$ and, therefore, the GAN hallucinates additional objects. In the second row, to many objects are placed in the domain $\hat{\mathcal{X}}$ and the GAN removes objects during the transfer from the domain $\hat{\mathcal{X}}$ to $\hat{\mathcal{Y}}$. Furthermore, the spatial location of objects P_{Sp}^{L} can differ. As shown in the three bottom rows, all circles in the domain $\hat{\mathcal{X}}$ are in the upper half. All circles in domain $\mathcal{Y}$ are in the lower half. Again, the discriminator is able to learn this distinction and, therefore, the generator will move the circles. However, it is not certain that the generator shifts the circles in a meaningful way for humans. In the first example, the shift is performed by rotating the input image, while in the second example, the shift is performed by mirroring the objects. Both are reasonable translations for humans. In the last row, the GAN shifts the objects to the lower edge of the image in a way that is incomprehensible to humans.

The theoretical analysis of the transfer of the different properties shows that a GAN learns to transfer all properties. While it is desired to learn the transfer of the color properties to create synthetic segmentation datasets, the transfer of shape and spatial properties leads to unaligned pairs of synthetic label image and synthetic target image. Unaligned shape and spatial properties can be compared with poor manual segmentation. An unaligned contour property P_{Sh}^{C} and an unaligned area property P_{Sh}^{A} correspond to a poor manual outline labeling of single objects. An unaligned number of objects P_{Sp}^{N} corresponds to a manual under- or oversegmentation. Unaligned spatial locations of objects P_{Sp}^{L} do not occur during manual labeling. As for poor manual labels, unaligned properties of the synthetic label images can lead to poor performance on supervised downstream tasks. The influence of shape and spatial properties on subsequent downstream tasks remains unclear. Therefore, the spatial and shape properties of an individual dataset must be matched when creating the corresponding synthetic label images of domain $\hat{\mathcal{X}}$. Automated quantitative assessment of whether a single property has been translated in the desired manner is not possible due to the complexity of most real-world datasets.

2.2 Evaluation of Synthetic Label Image Property Influence

Creating synthetic label images with exactly the same properties as the target images is often times not possible because no segmentation for the target domain is available for many real-world tasks, and therefore the distributions for the properties cannot be extracted. One can estimate the distributions with expert knowledge or measurements on the target images, but expert knowledge is often not available. Using measurements on the target images to estimate the distributions is time-consuming, and it is unclear how many measurements need to be performed. In general, the estimation of the properties is complex and it is important to know how much effort one has to put into the estimation of the properties and which properties are most relevant.

Currently, the influence of the properties of the synthetic label images on the GAN image synthesis and, thereby, on a supervised downstream task cannot be quantified. An evaluation pipeline, as well as a benchmark dataset is missing. Standard segmentation benchmark datasets cannot be used out of the box as they only contain the targets and manually created label images. However, different sets of synthetic label images with modified object properties are required for the analysis. Therefore, the pipeline in which synthetic label images are used to synthesize training data (see Section 1.3.3) is extended below by the necessary components (benchmark dataset creation and downstream evaluation) to perform the analysis based on any labeled benchmark dataset.

The extended pipeline shown in Fig. 2.4 consists of the following tasks:

1. Extraction of the properties needed for label image synthesis from the labels in domain $\mathcal{X}$
2. Variation of the properties to create different property sets
3. Synthesis of synthetic label images for each property set
4. Synthesis of paired synthetic training data with the GAN for each set of unpaired synthetic training labels
5. Training of machine learning model for the supervised downstream task with paired synthetic training data

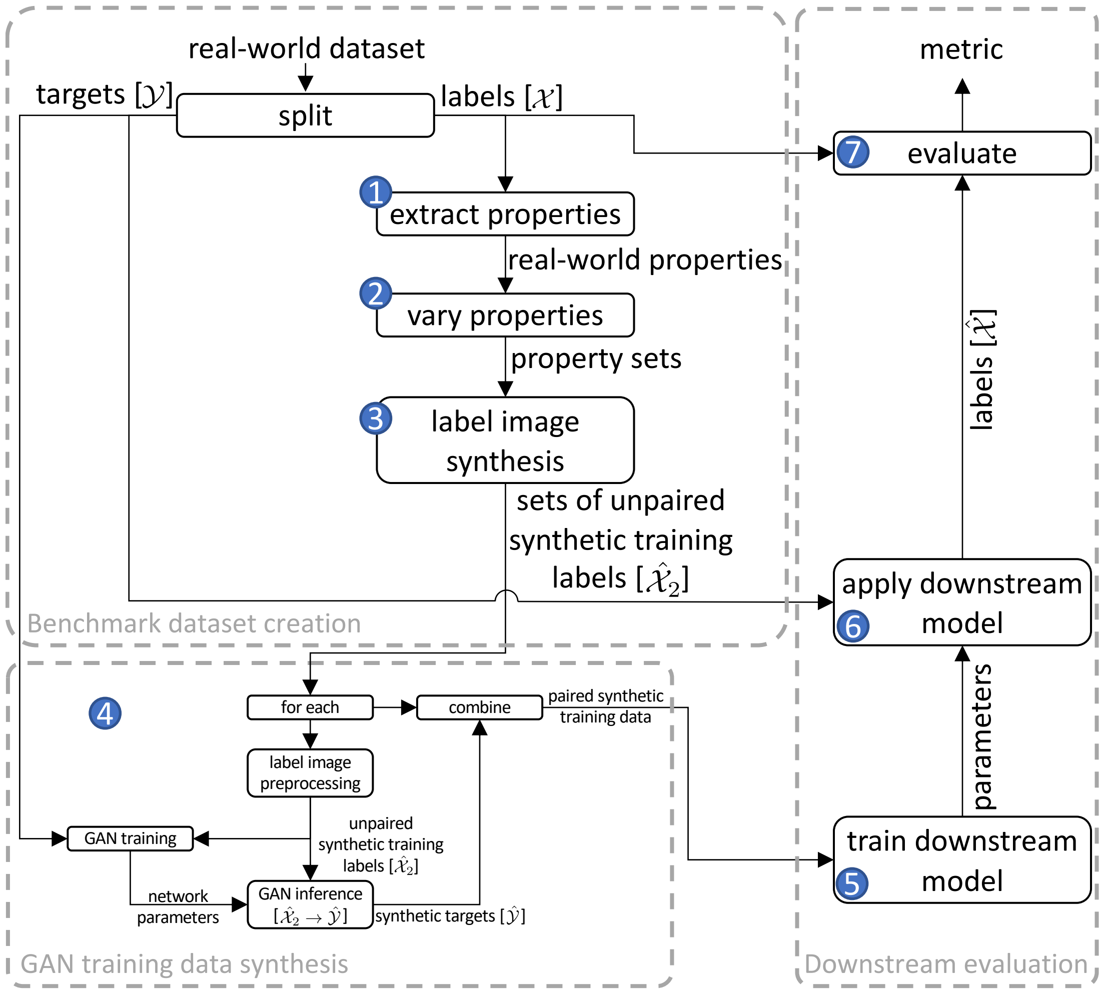

real-world dataset
targets [$\mathcal{Y}$]
split
labels [$\mathcal{X}$]
1
extract properties
real-world properties
2
vary properties
property sets
3
label image synthesis
sets of unpaired synthetic training labels [$\hat{\mathcal{X}}_2$]
Benchmark dataset creation
4
for each
combine
paired synthetic training data
label image preprocessing
unpaired synthetic training labels [$\hat{\mathcal{X}}_2$]
GAN training
network parameters
GAN inference [$\hat{\mathcal{X}}_2 \rightarrow \hat{\mathcal{Y}}$]
synthetic targets [$\hat{\mathcal{Y}}$]
GAN training data synthesis
metric
7
evaluate
labels [$\hat{\mathcal{X}}$]
apply downstream model
6
parameters
train downstream model
5
Downstream evaluation

Figure 2.4 | Pipeline to evaluate the influence of synthetic label image properties on a downstream task with a fully labeled real-world benchmark dataset. Properties of the real-world dataset are extracted (1). The properties are varied and different property sets are created (2). The property sets are used to synthesize sets of unpaired synthetic training labels (3). Each set is used for GAN training data synthesis (4) and downstream evaluation. Downstream evaluation consists of downstream model training (5), application of the downstream model on the real-world targets (6), and evaluation of the predictions (7).

6. Application of the downstream model to the real-world dataset
7. Evaluation with predicted labels and ground truth labels

The steps are marked with blue circles and the respective numbers in Fig. 2.4. At first, a benchmark dataset needs to be created (steps 1 to 3). The GAN training data synthesis and the downstream evaluation (steps 4 to 7) have to be performed for each set of unpaired synthetic training labels present in the benchmark dataset. The downstream metric (step 7) acquired for each set of properties used to synthesize label images can be used for the final evaluation of the influence of the properties of the synthetic label images.

It must be assumed that the properties that are varied during step 2 depend on each other. However, a multivariate variation of the synthetic label image properties results in many synthetic label image sets with different properties. Training GANs and downstream models for each set is computationally not feasible. Therefore, an univariate analysis of the synthetic label image properties, where each property is evaluated separately, has to be performed to limit the needed computation.

2.3 Synthetic Label Image Property Benchmark Datasets

The BBBC039v1[2] dataset, which contains roundish cell nuclei, is used as a basis to create the benchmark dataset for the pipeline introduced in Section 2.2 [150]. This is done because GANs are used mainly in biological and medical image analysis and most of the time roundish objects are present in the target images. The results obtained for cell nuclei can be transferred to a wide variety of datasets that contain round objects because the same programming methods can be used to create the synthetic images.

The BBBC039v1 dataset contains 200 images of U2OS cells acquired by fluorescence microscopy using the Hoechst staining. All images are 520 px $\times$ 696 px sized and stored as single channel 16-bit grayscale images. The cell nuclei in the instance segmentation ground truth are manually annotated. Example images and the corresponding ground truth are shown in Fig. 2.5.

Two benchmark datasets are created with the BBBC039v1 dataset. The first benchmark dataset is called the Contour Complexity Benchmark Dataset (CoCoBeDa). It is introduced in Section 2.3.1 and it is used to investigate the influence of the complexity of the contour of objects. Given that the size of an object can be modified when a contour is provided, the focus of this benchmark dataset is on the morphology of the contour of objects itself, irrespective of the absolute size of the objects. The second benchmark dataset, called Shape And Spatial Property Benchmark Dataset (SSPBeDa), is introduced in Section 2.3.2 and is used to perform a univariate analysis of the shape and spatial properties. To do so, the eccentricity of elliptic objects, the number of objects in an image, and the size of objects are modeled.

[2]The dataset is available at https://bbbc.broadinstitute.org/BBBC039/, accessed 20.03.2024.

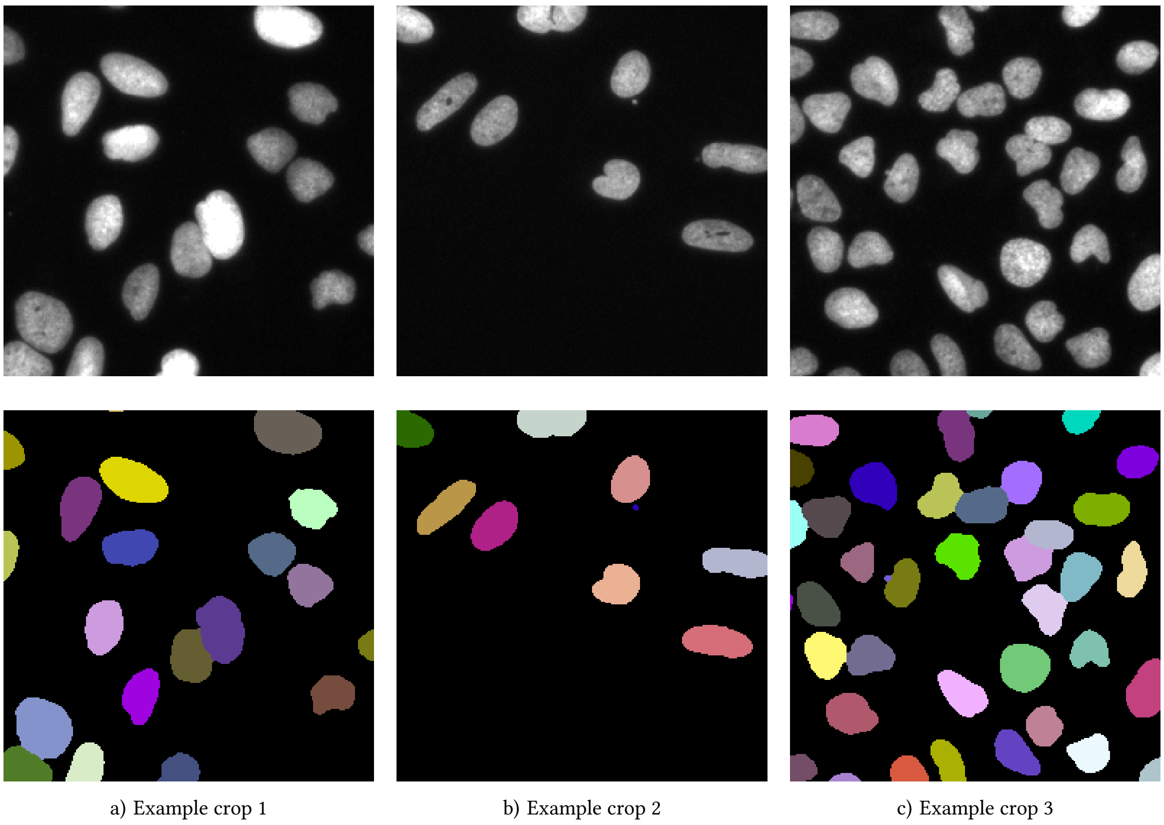


Figure 2.5 | Example crops sized 256 px × 256 px from the BBBC039v1 dataset and the corresponding instance segmentation. Images are normalized between the 1% and the 99% percentiles and then converted to 8-bit grayscale. The examples show that the number of nuclei differs significantly between images.

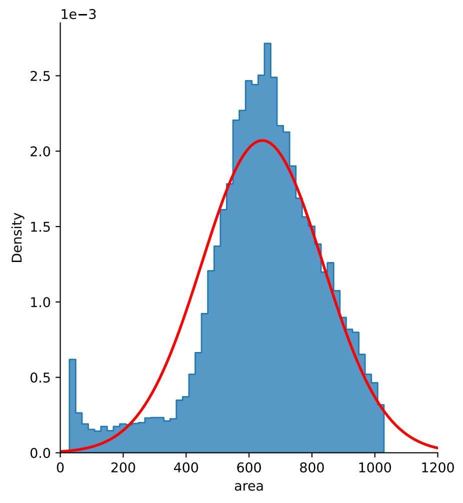
2.3.1 Contour Complexity Benchmark Dataset (CoCoBeDa)

The Contour Complexity Benchmark Dataset (CoCoBeDa) is introduced to evaluate the influence of the contour complexity if GANs are used to create synthetic segmentation datasets. If the influence of the contour complexity of the synthetic label images on the downstream segmentation performance is known, the contour complexity can be reduced to the minimal required level. A minimal contour complexity saves development time and expert knowledge needed. For evaluation, the contour complexity of the synthetic label images is varied in the pipeline (Fig. 2.4, step 2). Variation of the contour complexity during label image synthesis can be performed in several ways (see Section 1.4): With programming methods, complex objects can be modeled with higher-order splines. Because the BBBC039v1 ground truth is available, SSM methods can also be used. In contrast to programming methods, SSM methods do not have to be adapted for many different shapes, and new objects can be sampled from a distribution without additional knowledge of the objects' shape. Therefore, human bias is reduced. Furthermore, most of the SSM methods like spherical harmonics, Fourier shape descriptors (FSDs) and elliptic Fourier shape descriptors (EFDs) have a parameter which directly controls the complexity of resulting shapes. This is needed to evaluate the influence of the contour complexity. While spherical harmonics are used for 3D objects, FSDs and EFDs are used for 2D objects. Both are equally suitable for modeling spherical objects, and the number of Fourier modes used changes the contour complexity that can be mapped. EFDs are used in this study because U20S cell nuclei, like many other cell nuclei, are very similar to an ellipse and ellipses are modeled with EFDs of mode one. If the complexity of the EFD distribution is reduced, the quality of the objects drawn from the distribution increases because of a simpler Gaussian mixture model (GMM). To decrease the complexity of the EFD distribution, EFDs are normalized to the size of one and transformed to be rotation invariant by rotating the major axis of all objects to the x-axis.

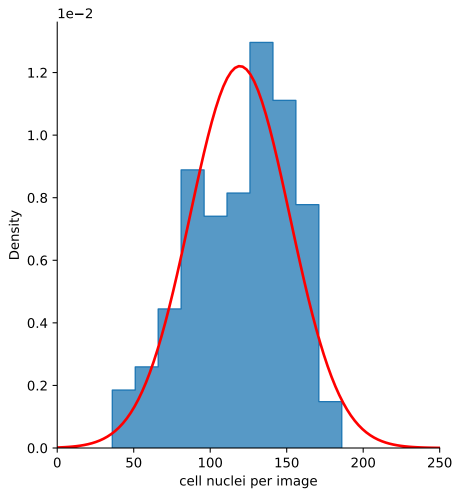
For the above reasons, normalized rotation invariant EFDs are used to model and vary the complexity of the shape's contour property P_{Sh}^{C}. Synthetic objects can be sampled in the following manner:

1. calculate EFDs for each ground truth object,
2. normalize the EFDs for each object to an area of one and convert to rotation invariant version,
3. train Gaussian mixture model (GMM) on distribution of EFDs,
4. sample from GMM,
5. rotate and scale sample,
6. transform to mask,
7. place mask in synthetic label image.

Distributions for the object size P_{Sh}^{A} and number of objects P_{Sp}^{N} are extracted from the ground truth and kept similar to the ground truth distributions. The spatial distribution of nuclei in the dataset is random. The distributions are normalized to a sum of one, and the corresponding histograms are shown in Fig. 2.6. The distributions are clipped between the 5% and the 95% percentiles to account for outliers and a normal distribution is fitted, which is indicated by a red line. Although normal distributions do not perfectly



a) Area of individual nuclei P_{Sh}^{A}



b) Number of cell nuclei per image P_{Sp}^{N}

Figure 2.6 | Histograms for the area of cell nuclei a) and number of cell nuclei per image b), extracted from the BBBC039v1 dataset. Histograms are clipped between the 5% and 95% percentiles and normalized to a sum of one. An overlay for the corresponding probability density function of a normal distribution is depicted in red. The corresponding normal distributions are $\mathcal{N}_{Sh}^{A}(642, 193^2)$ and $\mathcal{N}_{Sp}^{N}(119, 33^2)$.

reflect the real data, they provide a uniform approach with few parameters. Furthermore, normal distributions are used if the underlying distribution is unknown. This is the case for real-world datasets without annotated data and in-depth expert knowledge.

The EFDs together with the extracted distributions are used to create the unpaired synthetic label images for the BBBC0039v1 dataset to enable quantification of the contour complexity. Synthetic label image sets are created for the EFD modes $P_{Sh}^{C} \in [1, 2, 3, 5, 10, 15, 20]$. The distribution of P_{Sh}^{C} aims to range from a very simple representation of the objects to an exact reproduction. An empirically determined number of ten EFD modes was used to sample nuclei in [137]. Therefore, it can be assumed that a range from 1 to 20 is a sufficient variation of the contour complexity from simple contours to complex ones. For each image, the number of nuclei is drawn from the normal distribution $\mathcal{N}_{Sp}^{N}(119, 33^2)$. Each nucleus instance is sampled by the GMM. Because the orientation of the nuclei in the BBBC039v1 images is random, the nucleus is rotated randomly and later scaled to a size drawn from the normal distribution $\mathcal{N}_{Sh}^{A}(642, 193^2)$. To account for touching objects, an overlap of 5% is allowed. Because an unrealistic number of objects per image and an unrealistic object size can be drawn from the normal distributions, all normal distributions are clipped to the mean and standard deviation of $\mu \pm 2\sigma$.

Example images for different sets of synthetic label images are shown in Fig. 2.7. Visual

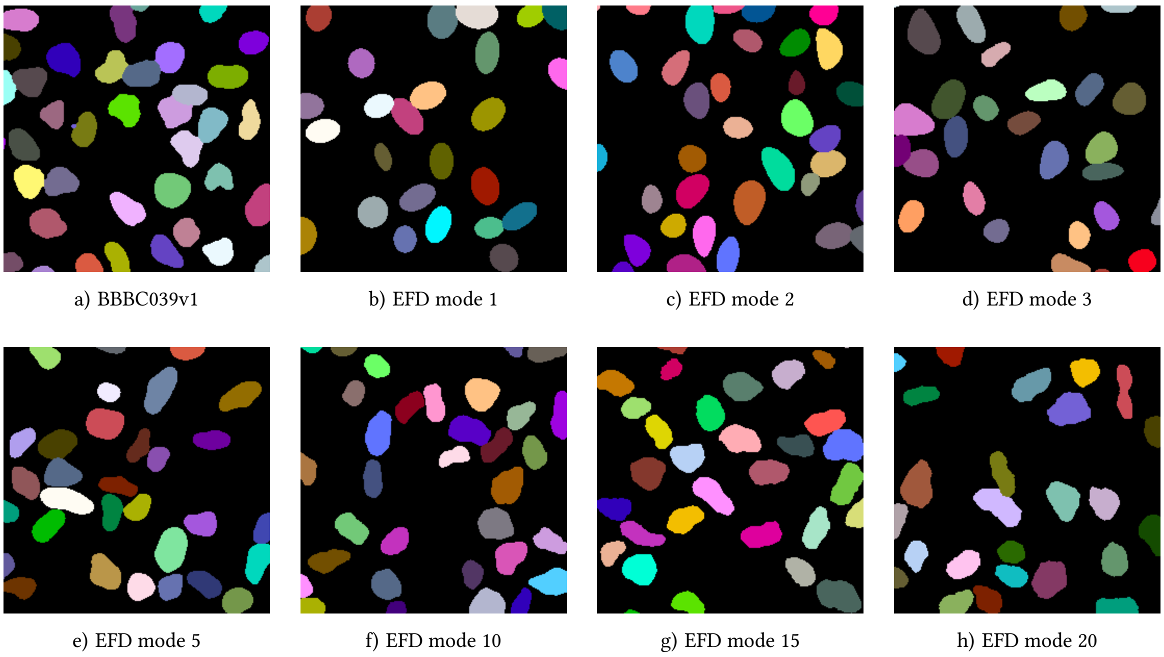


Figure 2.7 | Example crops sized 256 px × 256 px from the Contour Complexity Benchmark Datasets and a reference image from the BBBC039v1 dataset. The complexity of the shapes increases with the EFD mode. The objects are colored for visualization and the number of objects is random due to the underlying ground truth distribution P_N.

inspection shows that the complexity of the shapes increases with the EFD mode, while saturation is reached between modes 15 to 20.

The benchmark dataset creation, the GAN training data synthesis, and the downstream evaluation parts of the pipline introduced in Fig. 2.4 contain several steps in which images are synthesized or data are split. It is essential that no information used during the creation of the benchmark dataset or during the GAN training data synthesis leaks into the downstream evaluation. Therefore, a workflow to create the benchmark dataset and to use the dataset for GAN training data synthesis and downstream evaluation is designed and depicted in Fig. 2.8. Labels are marked with a blue dot, and targets are marked with a red dot. Paired labels and targets (real-world target microscopy images or synthetic ones) are marked by a blue/red dot. Synthetic images are marked by light blue and red. The real-world label domain is called $\mathcal{X}$ and the real-world target domain is called $\mathcal{Y}$. The synthetic label image domain is called $\hat{\mathcal{X}}$ and the synthetic target image domain synthesized by the GAN is called $\hat{\mathcal{Y}}$. Furthermore, the pairing of two domains is visualized with the union symbol $\cup$. At first, the BBBC039v1 test data (20%, 40 paired images) are separated from the training and validation data (80%, 160 paired images). Properties are extracted from the training and validation labels. The properties are used to synthesize label images

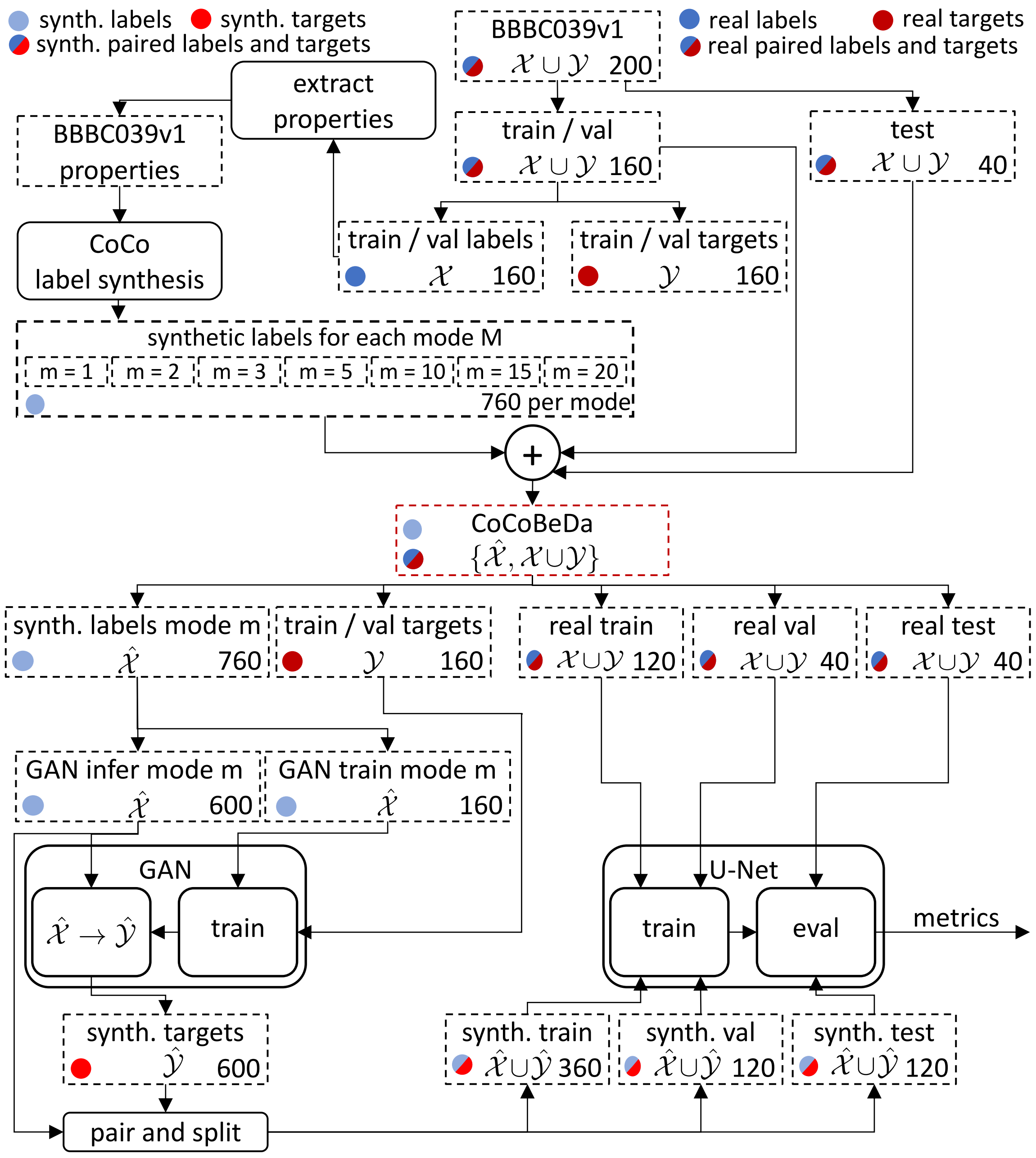

synth. labels
synth. targets
synth. paired labels and targets
BBBC039v1 $\mathcal{X} \cup \mathcal{Y}$ 200
real labels
real targets
real paired labels and targets
extract properties
BBBC039v1 properties
train / val $\mathcal{X} \cup \mathcal{Y}$ 160
test $\mathcal{X} \cup \mathcal{Y}$ 40
CoCo label synthesis
train / val labels $\mathcal{X}$ 160
train / val targets $\mathcal{Y}$ 160
synthetic labels for each mode M
m = 1
m = 2
m = 3
m = 5
m = 10
m = 15
m = 20
760 per mode
+
CoCoBeDa $\{\hat{\mathcal{X}}, \mathcal{X} \cup \mathcal{Y}\}$
synth. labels mode m $\hat{\mathcal{X}}$ 760
train / val targets $\mathcal{Y}$ 160
real train $\mathcal{X} \cup \mathcal{Y}$ 120
real val $\mathcal{X} \cup \mathcal{Y}$ 40
real test $\mathcal{X} \cup \mathcal{Y}$ 40
GAN infer mode m $\hat{\mathcal{X}}$ 600
GAN train mode m $\hat{\mathcal{X}}$ 160
GAN
$\hat{\mathcal{X}} \rightarrow \hat{\mathcal{Y}}$
train
U-Net
train
eval
metrics
synth. targets $\hat{\mathcal{Y}}$ 600
synth. train $\hat{\mathcal{X}} \cup \hat{\mathcal{Y}}$ 360
synth. val $\hat{\mathcal{X}} \cup \hat{\mathcal{Y}}$ 120
synth. test $\hat{\mathcal{X}} \cup \hat{\mathcal{Y}}$ 120
pair and split

Figure 2.8 | Data workflow, where the BBBC039v1 dataset is used to create the CoCoBeDa which is subsequently used for GAN training data synthesis and downstream evaluation. The BBBC039v1 training and validation data properties are used to create 760 synthetic labels for each contour complexity (CoCo) by variation of the EFD modes M. Subsequently, the BBBC039v1 training and validation targets and 160 synthetic labels are used to train a GAN. The remaining 600 synthetic labels are used after training to create the paired synthetic images. The synthetic images are used to train a U-Net and the U-Net is evaluated on real-world test data. Furthermore, a U-Net is trained and tested on real-world data as a reference. The number of images is displayed on the bottom right of each square. A blue dot marks label images, and a red dot marks target microscopy images. Shared blue and red dots mark paired labels and targets. Light colors refer to synthetic images.

for each of the different EFD modes M in the label synthesis step of the CoCoBeDa. For each mode, a total of 760 synthetic labels are created. The CoCoBeDa, marked with a red box, consists of the synthetic label images for each mode and the BBBC039v1 training, validation, and test data.

If the CoCoBeDa is used to evaluate the influence of the contour complexity on downstream segmentation, the synthetic labels of each mode are used separately for GAN training data synthesis and downstream evaluation. 160 synthetic label images of a mode m are used for GAN training together with the 160 targets from the BBBC039v1 training and validation data. The 160 labels of the BBBC039v1 training and validation data are not used for GAN training. After training, the remaining 600 synthetic labels (GAN infer mode m) are used to create 600 synthetic target images. The 600 synthetic targets are paired with the 600 synthetic labels and the pairs are split into 60% training, 20% validation, and 20% test data, resulting in 360 pairs of synthetic training images, 120 pairs of synthetic validation images, and 120 pairs of synthetic test images. Each of the synthetic splits (training, validation, and test) contains three times as many images as the corresponding real-world split. Using more synthetic images is common because synthesizing more images requires almost no computational overhead. The paired synthetic data is used for U-Net training and validation. Evaluation is performed on the synthetic test data and the real-world test data. Finally, the BBBC039v1 training, validation, and test data can be used to compare the results of U-Nets trained on synthetic data with the maximum achievable performance.

2.3.2 Shape And Spatial Property Benchmark Dataset (SSPBeDa)

For many real-world applications, no ground truth is available. This makes many SSM methods, such as EFDs, unusable. Programming methods (see Section 1.4) with standard shapes like circles or polygons have to be used to model objects. One has to estimate the properties of the real-world data for P^N_{Sp}, P^A_{Sh}, and the standard shapes P^C_{Sh}. The SSPBeDa is designed to enable the evaluation of the influence of individual properties of the synthetic label images on the downstream segmentation performance. Evaluation of the benchmark dataset enables scientists to assess the influence of these properties on their own data, and experiments and pipelines can be carried out accordingly. For example, it can be beneficial to spend more time gathering an accurate representation of the number of objects in an image than to model the size of objects.

In contrast to real-world applications where no ground truth is available, ground truth is needed for the evaluation, and therefore the BBBC039v1 dataset is used again. Since

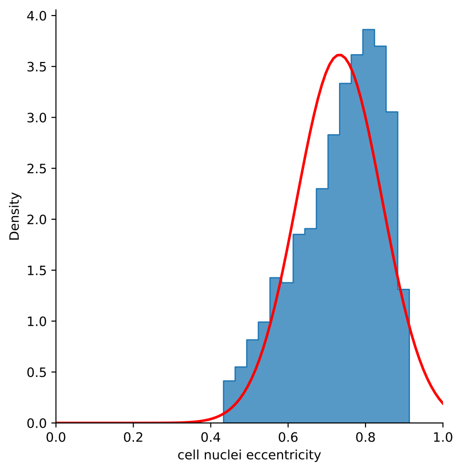


Figure 2.9 | Histogram for the eccentricity of ellipses with the same second moment as the cell nuclei in the BBBC039v1 ground truth. The histogram is clipped between the 5% and 95% percentiles and normalized to a probability density. An overlay for the probability density function of a normal distribution is depicted in red. The corresponding normal distribution is $\mathcal{N}^{E}_{Sh}(0.73, 0.11^2)$.

U2OS nuclei are very similar to ellipses, ellipses are placed in the images. Similarly to the Contour Complexity Benchmark Dataset, the distributions for the number of objects in an image P^N_{Sp} and the area of an object P^A_{Sh} are extracted. Furthermore, the distribution for the eccentricity of ellipses that have the same second moment as the nuclei in the ground truth P^E_{Sh} is extracted. The distribution is shown in Fig. 2.9. The properties and thus the distributions can depend on each other, but a multivariate analysis is not possible because of the computational effort involved. Therefore, the mean value for each property is varied separately with a scaling factor v. The scaling factor must reflect the entire range of plausible mean values. A mean value is considered plausible if it can be chosen during the data synthesis process such that it is not apparent that the selected mean value significantly diverges from the expected value observed in the real-world target data. Because this selection depends on the existing expert knowledge and the person carrying out the work, scaling factors that go beyond the range of apparent values are selected. Scaling factors between 0.4 and 1.6 are selected for P^N_{Sp} and P^A_{Sh}. For P^E_{Sh}, the range of scaling factors is set between 0.3 and 1.3. A scaling factor of 0.3 results in nearly round objects, while a scaling factor of 1.3 already results in objects that are much more elliptical than the real-world objects. Furthermore, in contrast to the other properties, the eccentricity must be in the range $[0, 1]$.

The extracted distributions and the scaling factors are as follows:

- $\mathcal{N}^N_{Sp}(\mu_N * v_N, \sigma^2_N)$, with $v_N \in [0.4, 0.6, 0.8, ..., 1.6]$,

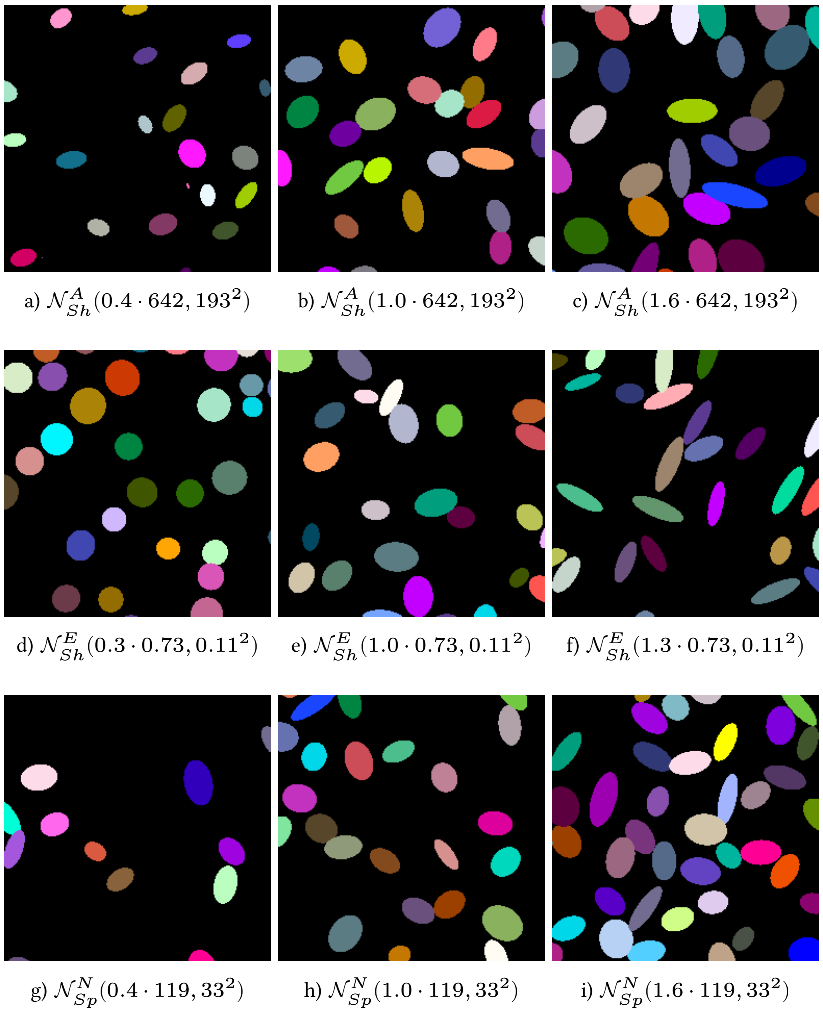

a) $\mathcal{N}_{Sh}^{A}(0.4 \cdot 642, 193^2)$
b) $\mathcal{N}_{Sh}^{A}(1.0 \cdot 642, 193^2)$
c) $\mathcal{N}_{Sh}^{A}(1.6 \cdot 642, 193^2)$
d) $\mathcal{N}_{Sh}^{E}(0.3 \cdot 0.73, 0.11^2)$
e) $\mathcal{N}_{Sh}^{E}(1.0 \cdot 0.73, 0.11^2)$
f) $\mathcal{N}_{Sh}^{E}(1.3 \cdot 0.73, 0.11^2)$
g) $\mathcal{N}_{Sp}^{N}(0.4 \cdot 119, 33^2)$
h) $\mathcal{N}_{Sp}^{N}(1.0 \cdot 119, 33^2)$
i) $\mathcal{N}_{Sp}^{N}(1.6 \cdot 119, 33^2)$

Figure 2.10 | Example crops sized 256 px × 256 px from the Shape and Spatial Property Benchmark Dataset. The mean of the distributions for the area $\mathcal{N}_{Sh}^{A}$, the eccentricity $\mathcal{N}_{Sh}^{E}$, and the number of objects per image $\mathcal{N}_{Sp}^{N}$ is varied. For $\mathcal{N}_{Sp}^{N}$, a crop of an image with the number of objects equal to the mean is shown.

- $\mathcal{N}_{Sh}^{A}(\mu_A * v_A, \sigma_A^2)$, with $v_A \in [0.4, 0.6, 0.8, ..., 1.6]$, and
- $\mathcal{N}_{Sh}^{E}(\mu_E * v_E, \sigma_E^2)$, with $v_E \in [0.3, 0.5, 0.7, 1.0, 1.1, 1.2, 1.3]$.

The means and standard deviations correspond to $\mu_N = 119$ and $\sigma_N = 33$, $\mu_A = 642$ and $\sigma_A = 193$, and $\mu_E = 0.73$ and $\sigma_E = 0.11$.

When drawing from the distributions during the creation of the synthetic label images, the distributions are clipped to $\mu \pm 2\sigma$, similar to the CoCoBeDa. Additionally, the following limits for drawn values are added: A lower bound of 1 is added to P_{Sp}^{N} and P_{Sh}^{A}, because empty images and objects with a size of less than one pixel are not reasonable samples from the distribution. For P^E_{Sh}, the output is clipped to the range [0.05, 0.95]. These additions ensure reasonable outputs for the extreme variations of v.

Fig. 2.10 shows example images for the variation of each of the properties, where v increases for each property. The data splits are generated in exactly the same way as for the CoCoBeDa, while the shape and spatial properties are varied instead of the EFD mode.

2.4 Experimental Setup

The pipeline described in Section 2.2 is used to evaluate the influence of synthetic label image properties on downstream segmentation. The CoCoBeDA (Section 2.3.1) is used to evaluate how precisely the shapes in the synthetic label images have to be reproduced compared to the real-world targets. The SSPBeDa (Section 2.3.2) is used to assess which properties are most relevant for the performance of the downstream task.

CycleGAN is used for the experiments because CycleGAN is a well-established unpaired image-to-image GAN and it is frequently used for data synthesis. The ResNet architecture with instance normalization, 96 initial generator feature maps, and nine ResNet blocks in the feature space is used for the generators. The PatchGAN-Discriminator architecture with instance normalization is used for the discriminators. The mean squared error (MSE) is used for the cycle-consistency loss ($\mathcal{L}_{cycle}$), the identity loss ($\mathcal{L}_{idt}$), and the discriminator loss ($\mathcal{L}_{disc}$), which is also used to optimize the generators ($\mathcal{L}_{gen}$). Because GAN training can be unstable, each GAN is trained three times. Therefore, there are three different versions of the synthetic images for the same synthetic labels. The synthetic label images are augmented during GAN training. First, the semantic labels are converted to a binary instance segmentation mask because the instance information is not needed for GAN training. Afterwards, Gaussian noise with a mean of zero and a variance of 0.1 is applied to add variation to the label images.

The well-established U-Net architecture is used to create the results for the downstream instance segmentation. The U-Net is trained on the synthetic training data (bottom of Fig. 2.8) and it is trained separately for each version. Furthermore, the U-Net is trained on real-world training data to compare the results achievable on synthetic data to the results achievable with real-world training data. The U-Net encoder consists of a ResNet-50 pre-trained on ImageNet and SmoothL1Loss with the Adam optimizer is used. The network is trained for a maximum of 200 epochs with early stopping after no decrease in the loss on the validation data for 50 consecutive epochs. Furthermore, the starting learning rate of

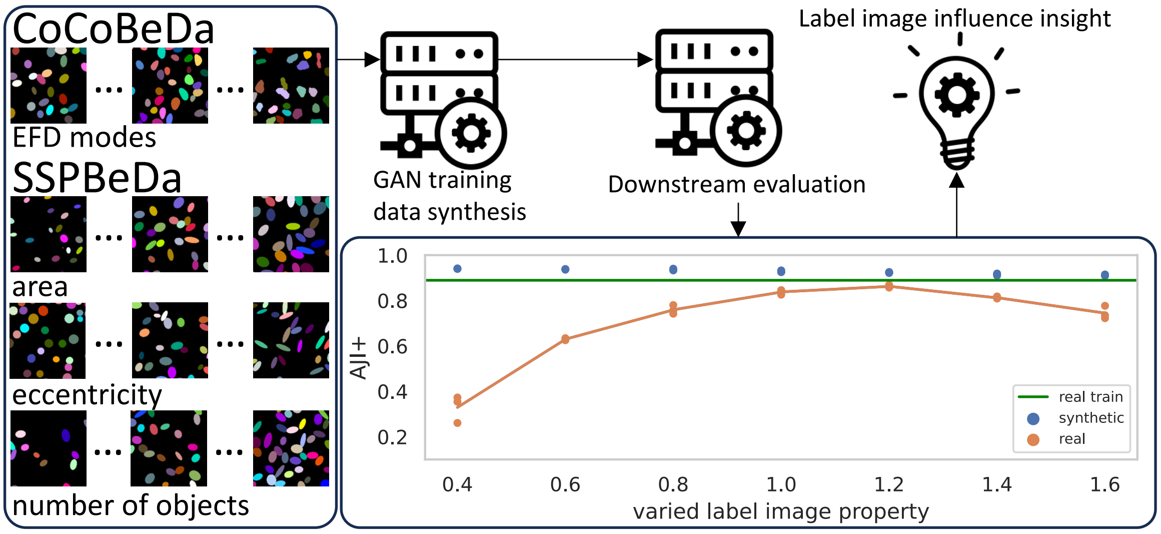


Figure 2.11 | The benchmark datasets CoCoBeDa and SSPBeDa are used to train GANs and the data synthesized by the GAN is subsequently used to train and evaluate U-Nets. The AJI$^+$ metric is used to evaluate the coherence between varied synthetic label image properties and the downstream segmentation performance. The same plot is created for each of the different properties. The green line shows the results if a U-Net is trained and tested with the real-world data. The orange dots show the results for U-Nets trained with the data synthesized for different variations of a synthetic label image property and tested on real-world data. Three U-Nets are trained per variation. The blue dots show the AJI$^+$ metric of the U-Nets applied to the synthetic test data. A linear interpolation of the mean values for each varied label image property tested on the real-world data is represented by an orange line.

0.001 is multiplied by a factor of 0.5 after 20 consecutive epochs with no decrease in loss on the validation data.

2.5 Results

The workflow to extract information from the data is shown in Fig. 2.11. The CoCoBeDa is used to find out how complex shapes in the synthetic label images need to be and whether there is a point beyond which a more accurate representation of the shapes in the real-world target images is not reflected in a further increase of the segmentation quality. The SSPBeDa is used to evaluate shape and spatial properties when using programming methods to synthesize the synthetic label images.

A schematic description of the results results acquired for each property is depicted at the bottom right of Fig. 2.11. The x-axis shows the variation of the scaling factor of a property. The y-axis shows the AJI$^+$ metric of the downstream segmentation. The AJI$^+$ metric of a U-Net trained with the data synthesized by a GAN and evaluated on the real-world test data is represented by an orange dot. Three orange dots are present for each scaling factor because three GANs are trained for each of the 7 scaling factors. A linear interpolation of the mean values is represented by an orange line. Each of the 21 trained U-Nets is also evaluated on test data synthesized by the GAN for each scaling factor. In contrast to the real-world test data, the synthetic test data differ for each scaling factor. Furthermore, the green line indicates a U-Net trained on real-world data and tested on real-world data. This data representation is used in the following to visualize the results and gather insight for the CoCoBeDa (see Section 2.5.1) and the SSPBeDa (see Section 2.5.2).

2.5.1 Contour Complexity

The experiments on the CoCoBeDa are carried out to determine the relationship between the contour complexity in the synthetic labels and the subsequent downstream segmentation quality. A more complex contour in the synthetic label images corresponds to a more accurate representation of the contours in the real-world images. The results can be used to estimate the minimum required complexity of the contours. Using less complex contours leads to less development effort and saves time. First, a general analysis of the results is carried out. This is followed by a detailed examination of why the segmentation quality does not continue to increase as the EFD mode increases.

General analysis

The results on the CoCoBeDa are shown in Fig. 2.12. The performance on the synthetic test data (blue dots) is stable, and AJI$^+$ ranges from 0.92 to 0.94 for all EFD modes. The synthetic test data for each EFD mode is from the same EFD mode. In contrast, the real-world test data is the same for all EFD modes and the AJI$^+$ ranges from 0.82 to 0.87. The maximum mean AJI$^+$ is achieved for modes 2, 3, and 20 with a value of 0.86. A network trained on the real-world training data achieves an AJI$^+$ of 0.89. The best model trained on synthetic data performs 2.2% worse than the model trained on real-world data. While this difference is small, it shows that there is still a domain gap between synthetic training data and real-world training data.

Comparing the blue and orange dots shows that the results on the synthetic test data cannot be used to estimate the minimum required complexity of the contours. The U-Net performs well on the synthetic test data for all scaling factors. The good performance can be explained with the learning task of a GAN. The cycle-consistency loss enforces the GAN to learn the transformation from one domain to the other and back during the GAN training. This can only be done if the information from the first domain is kept in the syn-

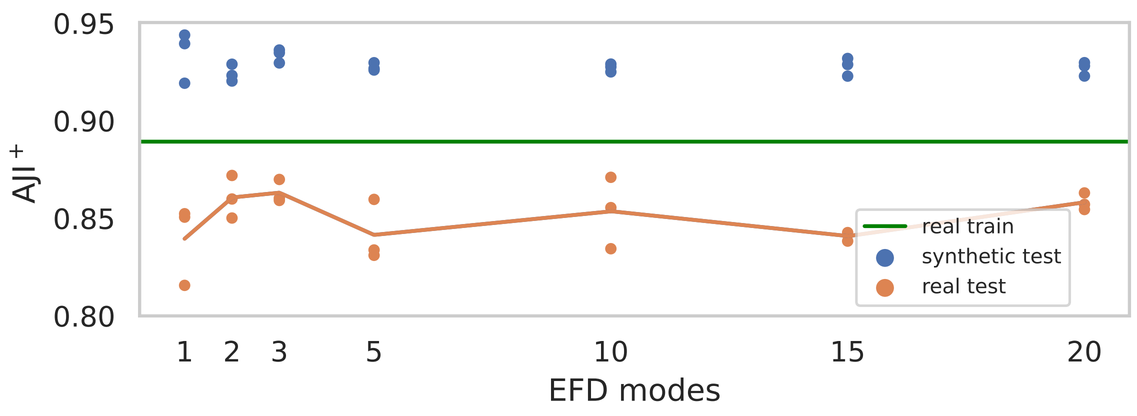


Figure 2.12 | Results for the downstream AJI$^+$ metric achievable for data synthesized with varying number of EFD modes. The green line shows the results if a U-Net is trained and tested with the real-world data. The orange dots show three U-Nets trained with the data synthesized for each EFD mode and tested on real-world data. The blue dots show the test score of the U-Nets applied to the synthetic test data. A linear interpolation of the mean values for each mode tested on the real-world data is represented by an orange line.

thesized images of the second domain. Therefore, the GAN learns to transform synthetic label images to synthetic microscopy images, and back. If the GAN is able to perform this segmentation task during training, the U-Net is also able to learn this transformation with the data created by the GAN. However, the GAN only performs semantic segmentation, and the U-Net performs instance segmentation in the experiments.

In depth contour analysis

If there is a correlation between segmentation performance and EFD mode, an increase, decrease, or a platau for different EFD modes is expected. The lowest mean AJI$^+$ is achieved for modes 1, 5 and 15 with 0.84, the highest for modes 2, 3 and 20 with 0.86. Looking at the real-world cell nuclei images and their corresponding representations through various EFD modes shown in Fig. 2.13 shows, that most cell nuclei can already be described by an EFD mode of three. These visual findings can also be quantified by quantifying the ability of different EFD modes to capture the shape of the cell nuclei. Therefore, each cell nucleus in the training and validation data from the BBBC039v1 dataset is converted to an EFD and back to its mask. Subsequently, the intersection and union are calculated for each nucleus. All intersections are summed and divided by the sum of all unions. This results in the following values for modes 1, 2, 3, 5, 10, 20 over all images in the dataset: 0.877, 0.895, 0.930, 0.939, 0.941, 0.941. A value of 1 cannot be reached because of discretization errors during the transformation of the contour described by the EFD to a pixel mask.

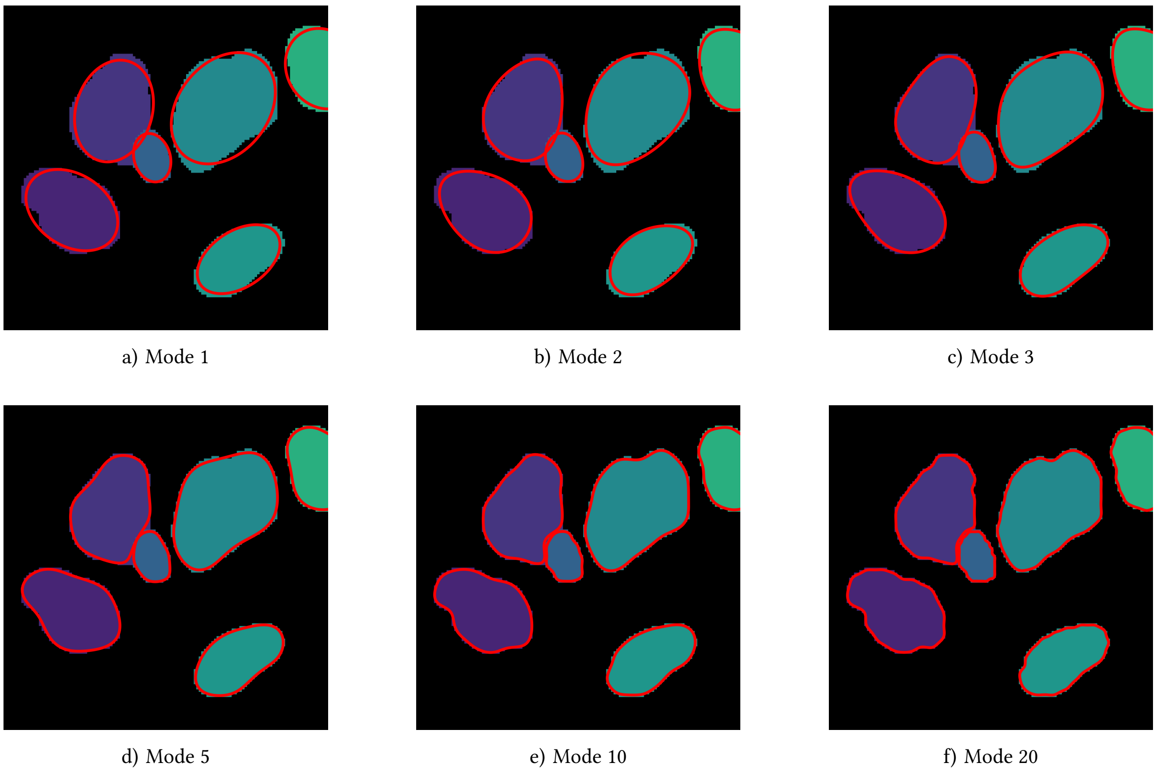


Figure 2.13 | Crop of a segmentation mask of an image from BBBC039v1 with reconstruction overlays of different EFD modes. Plain ellipses described by a mode of one do not capture the shape of the ellipses completely. It can be seen that there are nearly no differences between modes five to twenty.

The score stops to increase between mode 10 and mode 20. This leads to the conclusion that the differences in AJI$^+$ achieved for modes 10, 15, and 20 are not due to the contour complexity but due to the randomness during training and there is no need to choose high modes for simple shapes.

The question of why modes one and two do not perform substantially worse than other modes on real-world data remains. This can be due to the following hypotheses:

1. The GAN changes all shapes slightly when transferring from the synthetic label image domain to the target image domain, regardless of the complexity of the shape in the synthetic label images. Therefore, it does not matter whether the shape in the synthetic label image domain is a perfect representation of real-world shapes.

2. Errors like merges or splits introduced during inference with the U-Net exceed the errors introduced by the simpler shapes.

3. The objects in the synthetic label images are created by drawing from a distribution.

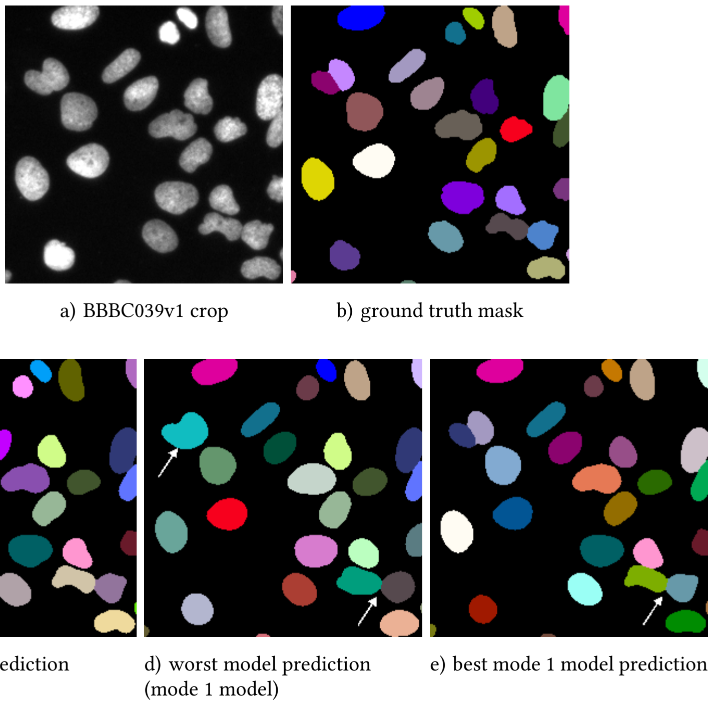

a) BBBC039v1 crop
b) ground truth mask
c) best model prediction (mode 2 model)
d) worst model prediction (mode 1 model)
e) best mode 1 model prediction

Figure 2.14 | Example crop from the BBBC039v1 dataset, corresponding ground truth mask and different model predictions. The prediction from the best model (a mode 2 model) shown in c) is accurate, while the prediction from the worst model (a mode 1 model) shown in d) has a merge in the top left, and the objects are too elliptical, which is shown in the bottom right. The best model for mode 1 is shown in e). In contrast to d) it has no merge, and the object shapes in the bottom right are predicted correctly.

> The more modes there are, the more complex the distribution. This can result in unrealistic objects being drawn for high modes. Therefore, the errors introduced by drawing unrealistic complex objects outweigh the errors introduced by simple shapes.

Hypothesis three is unlikely, because if high modes introduce errors, a decrease of AJI$^+$ has to be expected for mode 20, which is not the case. Unfortunately, it is not possible to conclusively evaluate whether hypothesis one or hypothesis two applies without a laborious manual segmentation of many synthetic target images. However, it is likely that a mix of both is responsible for the fact that mode 1 and mode 2 do not perform worse than the higher modes.

Fig. 2.14 shows example predictions for the best-performing model (a model from mode 2), and the worst-performing model (a model from mode 1). Furthermore, the best model is shown for mode 1. The differences between the best model Fig. 2.14c and the ground truth

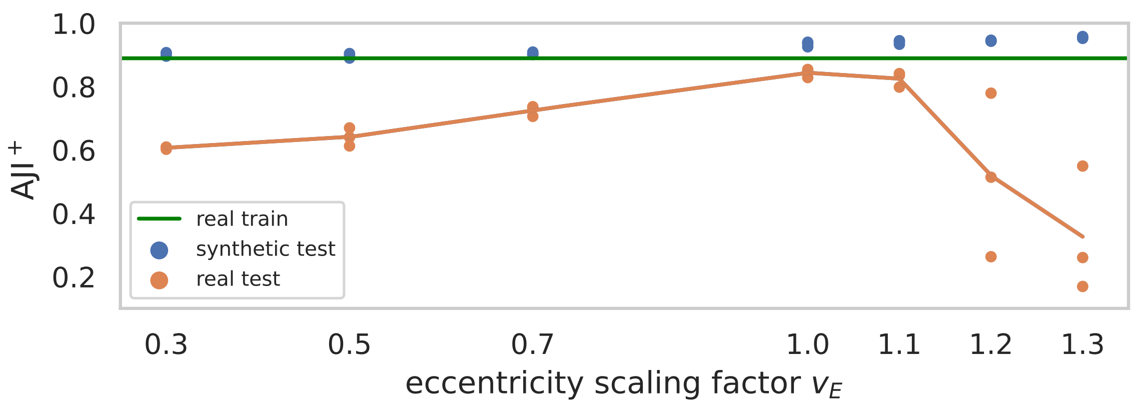


Figure 2.15 | Results for data synthesized with varying scaling factors for the eccentricity. The baseline result for a U-Net trained and tested on real-world data is indicated by a green line. The dots show the same three networks trained for each mean eccentricity evaluated on synthetic test data (blue) or on real-world test data (orange). The orange line connects the mean values for U-Nets trained on the synthetic data and evaluated on the real-world data for each scaling factor.

mask (Fig. 2.14b) are small. Visual inspection reveals that for the worst model (Fig. 2.14d mode 1), most objects are too round and a merge is present at the top left. In contrast to that, the best model from mode 1 (Fig. 2.14e) shows no merge and is able to predict objects that differ from the ellipses on which it was trained.

It can be concluded that ellipses are a good enough representation of cell nuclei in the BBBC039v1 dataset to achieve high-quality segmentation results. If the additional segmentation quality is needed, a mode of 2 can be used.

2.5.2 Shape and Spatial Properties

In this section, the results of the SSPBeDa are evaluated. The mean values of normal distributions fitted to object properties are varied and the influence on the downstream segmentation task is measured to assess the importance of correctly estimating the different distributions. First, the influence of the eccentricity on the downstream segmentation performance is evaluated, followed by the size of the objects and the number of objects.

The AJI$^+$ scores for different eccentricities are shown in Fig. 2.15. The data show that the metric on the real-world test data peaks for the scaling factor of 1.0. The worst mean performance of the evaluated scaling factors is achieved for a scaling factor of 1.3, with AJI$^+$ = 0.33. The steep decrease for large scaling factors could be due to the normal distribution and the marginal conditions used to determine the eccentricity of objects in the synthetic labels. Two marginal conditions are applied for the synthesis of eccentric

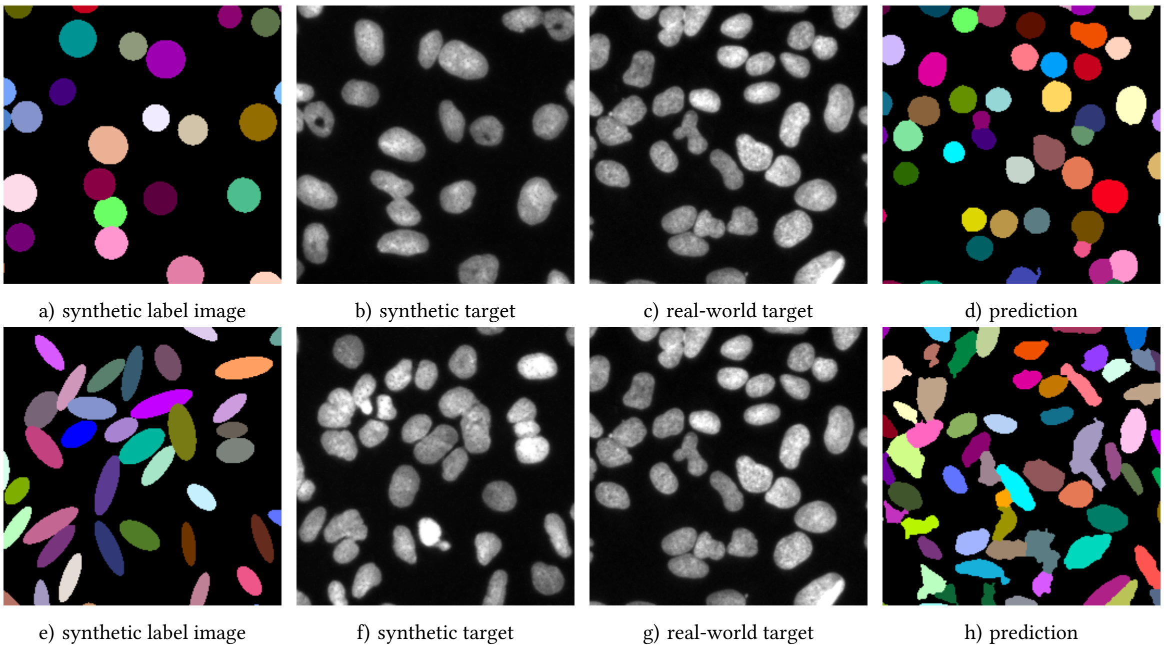


Figure 2.16 | The figure shows synthetic labels (a, e) and the corresponding synthetic images (b, f), real-world target images (c, g) and the corresponding instance segmentation (d, h). The top row shows images for a scaling factor of 0.3 and the bottom row shows images for a scaling factor of 1.3. Images (b) and (f) show that the GAN learned to adapt the shapes in (a) and (e) to the real-world shapes. Images (c) and (d) show that the U-Net trained on the synthetic data learned to transfer the distorted shapes back to circles. The shapes are too round compared to the shapes present in the real-world target. The same is true for images e to h. The U-Net learns to transfer the shapes in the real-world targets to elliptic labels present in the synthetic label images, while the prediction quality of h is severely reduced.

objects. First, the eccentricity is drawn from the distribution $\mathcal{N}^{E}_{Sh}(0.73 * v_E, 0.11^2)$ and the eccentricity is clipped between $\mu - 2\sigma$ and $\mu + 2\sigma$. In addition, the values are clipped a second time if they fall below or exceed the limits of 0.05 and 0.95. This leads to the marginal condition $0.05 \leq \mu{-}2\sigma \leq e \leq \mu{+}2\sigma \leq 0.95$. The mean of the used distribution is $\mu = 0.95$ for $v_E = 1.3$ and therefore half of the distribution used to synthesize objects is clipped to the maximum eccentricity of 0.95. This reduced variety in the synthetic labels results in a larger domain gap than a scaling factor of $v_E = 0.7$. Clipping the distribution is part of the experimental setup. It is also possible to allow for the entire range of eccentricity values. However, it can be seen in Fig. 2.16e that even with a limit of 0.95 most of the objects in the synthetic label images are too eccentric in relation to the real-world images. Therefore, it can be concluded that both the lower limit of 0.95 and the limit of 1.0 for the eccentricity e of objects do not represent a value range that leads to good results, while increasing the clipping value can be beneficial to increase variety.

A scaling factor v_E below 1.0 results in synthetic labels with objects closer to a circle.

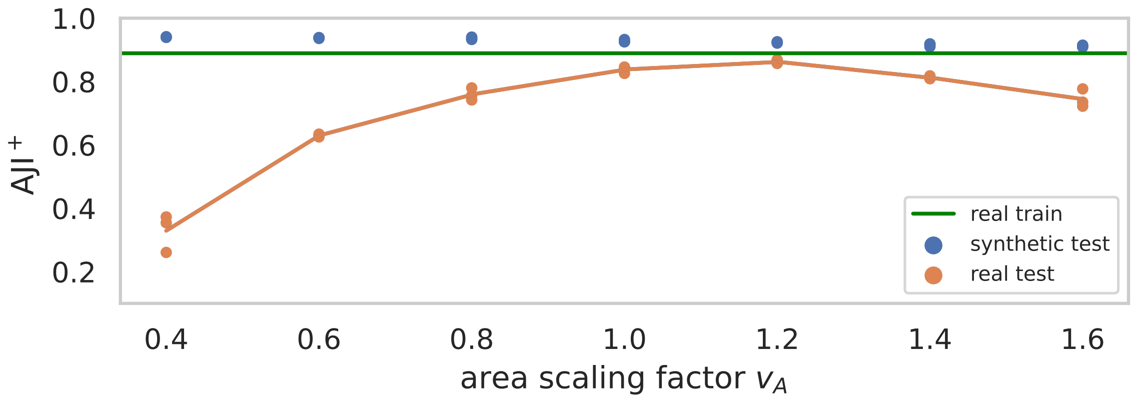


Figure 2.17 | Results for data synthesized with varying scaling factors for the area. The green line shows the results if training and testing are performed on real-world data. The orange and blue dots show the same three networks trained for each mean area evaluated on synthetic test data (blue) or on real-world test data (orange). A linear interpolation of the mean values for each mean area tested on the real-world data is represented by an orange line.

This reduces the AJI$^+$ score on the real-world data. For a scaling factor of 0.3, the mean AJI$^+$ score is 0.61 compared to 0.84 for a scaling factor of 1.0. If the objects in the synthetic labels are too circular, the GAN learns to change the shape during the translation. This translation is then learned by the U-Net and finally results in too round objects after predicting the labels for real-world images. An example is shown in Fig. 2.16 in the top row. The same behavior is observed if the eccentricity of objects in the synthetic labels is too high. This is shown in Fig. 2.16 in the bottom row.

The evaluation of the area property is shown in Fig. 2.17. The largest mean values for the performance on the synthetic test data (blue dots) is achieved for the scaling factor of 0.4 with AJI$^+$ $= 0.94$ and the lowest for a scaling factor of 1.6 with AJI$^+$ $= 0.91$. It is interesting to see that the performance on the real-world test data does not peak for a scaling factor of 1.0 (AJI$^+$ $= 0.84$), but for a scaling factor of 1.2 (AJI$^+$ $= 0.86$). This could be partly due to the 5% overlap allowed throughout the placement of the objects during the creation of the synthetic labels. The overlap reduces the size of objects already placed. Furthermore, it could be due to the design of the AJI$^+$ metric. The correct prediction of large objects is more important for the metric. Having slightly more large objects in the synthetic label images could help the U-Net to focus more on the prediction of these objects. In contrast to eccentricity, the AJI$^+$ is relatively stable for a scaling of 0.8 (AJI$^+$ $= 0.76$) to 1.6 (AJI$^+$ $= 0.74$). An explanation can be the larger standard deviation of $\mathcal{N}^{A}_{Sh}(642, 193^2)$ compared to $\mathcal{N}^{E}_{Sh}(0.73, 0.11^2)$. Compared to the mean, the standard deviation of $\mathcal{N}^{A}_{Sh}$ is twice as large as the standard deviation of $\mathcal{N}^{E}_{Sh}$. If the object size

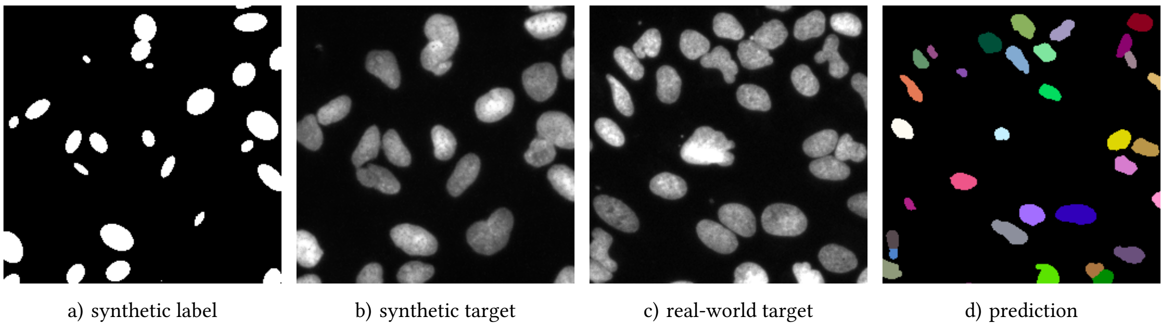


Figure 2.18 | The figure displays a synthetic label image (a) alongside its corresponding synthetic target image (b). Additionally, it presents a real-world target image (c) and its associated instance segmentation (d). The mean object area in the synthetic labels has been scaled by a factor of 0.4. The GAN increases the size of objects during the transformation from synthetic labels to synthetic targets and the U-Net reduces the size of objects during the prediction of real-world targets.

in the synthetic label images is chosen too small, the GAN starts to enlarge the objects when transferring between the synthetic label images and the real-world target images. The U-Net subsequently learns to reduce the size of the objects when transferring from real-world target images to label images. This behavior is shown in Fig. 2.18.

The results for the number of objects on the synthetic labels are shown in Fig. 2.19. The AJI$^+$ scores on the synthetic test data (blue dots) is stable throughout the experiment, ranging from 0.92 to 0.93. The highest mean AJI$^+$ score on the real-world test data is achieved for a scaling factor of 1.0 with AJI$^+$ $= 0.85$. The quality is good for scaling factors from 0.8 (AJI$^+$ $= 0.82$) to 1.4 (AJI$^+$ $= 0.81$). Similarly to the area property, the ratio between the standard deviation and the mean value for $\mathcal{N}_{Sh}^{N}$ is twice as high as for $\mathcal{N}_{Sh}^{E}$.

The GAN adds (hallucinates) objects not present in the synthetic labels to the synthetic targets if too few objects are placed in the synthetic label images. This is shown in Fig. 2.20. The U-Net learned to remove objects during the translation from target images to label images because of the unaligned synthetic training data. This behavior is not only present on the synthetic test data, but also on the real-world test data.

Interestingly, the GAN learned to add objects to the synthetic target images, while the existing objects are still transferred correctly. The same behavior can be observed by the U-Net the other way around. The predicted objects in the label images are not placed randomly, but refer to an object in the real-world target images, whereas many objects are not predicted at all. Fig. 2.20b shows that the GAN placed slightly brighter objects in the synthetic targets, at places where objects are present in the synthetic labels. This behavior was learned by U-Net. The objects on the right-hand side of the real-world tar-

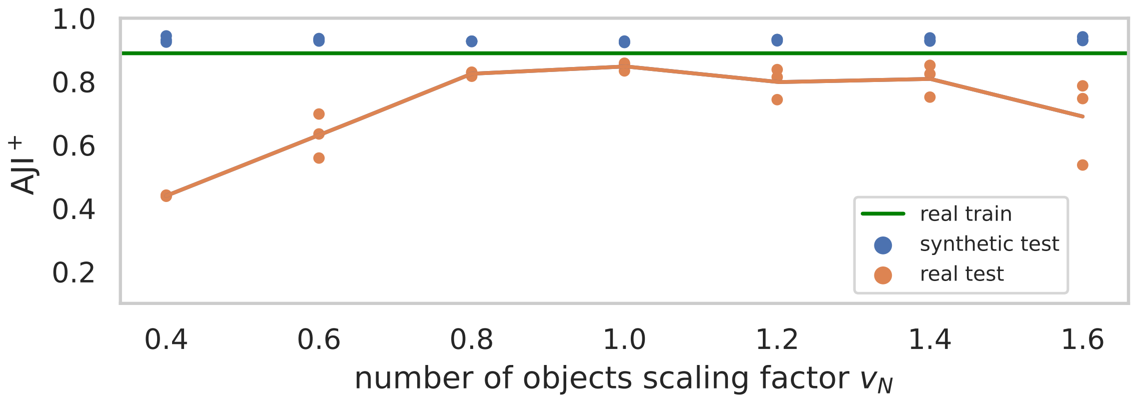


Figure 2.19 | Results for data synthesized with varying scaling factor for the number of objects. The U-Nets evaluated on the synthetic data yield stable results for all scaling factors. The AJI$^+$ scores for U-Nets evaluated on the real-world test data are stable for scaling factors between 0.8 and 1.4.

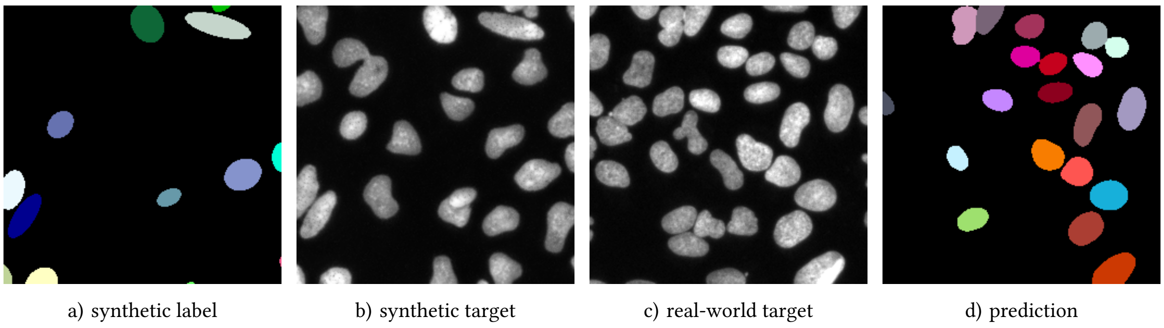


Figure 2.20 | The figure shows a synthetic label image (a) and the corresponding synthetic image (b), real-world target image (c) and the corresponding instance segmentation (d). The mean of the number of objects present in the synthetic labels is scaled by 0.4. The GAN adds objects during the transformation from synthetic labels to synthetic targets and the U-Net removes objects from the real-world targets during prediction.

get (Fig. 2.20c) are brighter and, therefore, are predicted by the U-Net. The less bright objects on the left side of the real-world target are mostly not predicted. As described in Section 2.1, the GAN does not have to use the brightness of the objects in order to encode the added objects. The property used and how the transformation is performed is random and depends, e.g. on the dataset, the GAN itself, and the random initialization of the GAN. Other color properties like a local pattern or shape properties like larger or smaller objects can be used by the GAN to encode added objects.

2.6 Discussion

The results confirm the theoretical conclusion made in Section 2.1, that the discriminator used all of the image properties to differ between the real-world target images and the synthetic ones. This caused the generator to change all properties between the two domains. While it is desired for the color properties P_C, it is not desired for the shape properties P_{Sh} and the spatial properties P_{Sp}.

While the experiments are performed on datasets containing roundish objects, they can be used for different kinds of objects, such as angular shaped objects for process control in production or elongated objects in material science. If the object properties differ severely from the object properties in the examined benchmark datasets, the proposed pipeline can be used to conduct additional experiments. However, this is only possible if a labeled dataset is available.

The results lead to the following findings:

- The contour P_{Sh}^C is varied with the CoCoBeDa and it has been shown that a mode of two is already sufficient to reach very good scores. Therefore, researchers should not invest too much time in object synthesis if the real-world data consists of roundish objects.
- The results for the eccentricity on the SSPBeDa show that the variation of the shape is much more important than matching the mean of the fitted normal distribution with the mean of the real-world images. The decline for too small eccentricities is slow and steady, and the clipping introduced for too large eccentricities made the decrease steep. Therefore, it should be omitted to clip large parts of the synthetic distribution to one value.
- Both, the relative area and the number of objects have a zone between a scaling factor of 0.8 and 1.2, where the results are still close to the optimum. This is due to the large standard deviation in the real-world data. Therefore, broad distributions in real-world data reduce the need to match the distribution in the synthetic data. Unfortunately, the distributions of objects in real-world data are usually unknown for real-world tasks.
- Unknown real-world distributions force the researcher to make assumptions. The results show that no large hyperparameter study has to be performed to find matching distributions for the synthetic data. Instead, visual inspection of the predictions

> of the U-Net on the real-world test data can be used. If many objects in the U-Net segmentation are missing, there are not enough objects in the synthetic data. If many objects in the U-Net segmentation are hallucinated, there are too many in the synthetic data. The same is true for the shape. Too round predictions arise from too round objects in the synthetic label images. While this is a time-consuming and manual process, there is no feasible alternative if there is no ground truth available for the real-world targets.

Several options to accelerate this manual process could be explored in the future. Instead of retraining the GAN, one could use a selection of synthetic data to train the U-Net. For example, if there are too few objects on the synthetic labels, the U-Net could only be trained on the 50% of images with the most objects in it. Furthermore, images with less eccentric objects can be synthesized, and the GAN can be used to infer synthetic targets without retraining. Using the GAN on images that are not in the training domain could reduce the synthesis quality. If this is the case, retraining on the new synthetic labels is needed. Instead of training the GAN from scratch, the weights of the GAN already trained can be used as an initialization. This could result in a shorter training and therefore less computation. The same is true for U-Net training. Furthermore, it can be explored whether the manual inspection of synthesized images can be automated with a very low number of manually annotated images. The manual annotations can be used to derive a feedback loop altering the properties used to synthesize the labels.

GAN Training Stopping and Epoch Selection 3

3.1 Overview

Many studies on GANs[1] for unsupervised image-to-image translation investigate methods to improve the quality of the results for a particular dataset or introduce new methods which are benchmarked on a common set of benchmark datasets. Common architectures share the feature that GANs used to synthesize training data for downstream segmentation are trained for a fixed number of epochs, and the last epoch is used as the final model. Using the last epoch without early stopping or epoch selection has two substantial drawbacks. (1) Training is unnecessarily prolonged, which slows down progress and increases cost. (2) The last epoch is not necessarily the best one and using the last epoch can reduce the quality of the output. This is because training a GAN is a two- or more-player game and, therefore, unstable if the players learn at a different pace. If the quality of the output from the last epoch is not good enough, another GAN needs to be trained. This further increases the computational cost of using GANs and reduces usage. Recently, a first approach for early stopping during training of GANs has been published [151]. The approach uses an energy distance as the generator loss function and for early stopping. However, the loss function is used to train unconditional GANs, which are used in the work to generate new digits (with the MNIST dataset [77]) or faces (with the CelebA face dataset [152]) from noise.

Supervised segmentation approaches use validation data to stop training and, therefore, avoid both drawbacks (see Section 1.2.2). Transferring the concepts of early stopping and selecting the best epoch after training from supervised segmentation approaches to GANs can remove both drawbacks. However, the transfer is not trivial because GANs are trained in an unsupervised manner and no ground truth is available to calculate the validation

[1] In this chapter, GANs always refer to GANs for unsupervised image-to-image translation. If a different task is meant, this is explicitly stated.

metric. Therefore, this work investigates the possibilities to select the best epoch after training or to stop training before reaching the maximal number of epochs for the first time. A comparison with supervised segmentation methods is given in Section 3.2. The new method is introduced in Section 3.3 and the experiments with the results are presented in Section 3.4 and Section 3.5. Finally, a discussion and a workflow to select the best epoch after GAN training is given in Section 3.6[2].

3.2 Quality Assessment Comparison

Loss functions and quality metrics after GAN training cannot be simple comparisons between an image and the corresponding ground truth. Nevertheless, the loss functions acquired during training can be used for a set of validation images. However, there is a large difference between the loss functions used for supervised models and GANs. Loss functions for supervised NNs emulate the metric used to measure the quality of the NN output on the test data. This is often done because the metric is not derivable and therefore cannot be used for gradient descent. For complicated tasks like instance segmentation, the loss functions are deduced from an intermediate surrogate task like semantic segmentation together with distance maps. The instance segmentation is finally derived from both in an *Information post-processing* step (see Section 1.2.1). Despite the different tasks, there is a strong correlation between the loss during training and the final metric. Loss functions used during training of a GAN are not directly linked to the synthesis quality, but it turned out that these loss functions, for example, stabilize the training.

Exemplarily, the loss functions of CycleGAN and their link to synthesis quality are considered. One loss function not related to the quality of the synthesis is the identity loss. Identity loss stabilizes training, but the identity mapping ($\mathcal{X} \rightarrow \mathcal{X}$ and $\mathcal{Y} \rightarrow \mathcal{Y}$) does not guarantee a good translation $\mathcal{X} \rightarrow \mathcal{Y}$ and vice versa. Also the cycle-consistency loss, $\mathcal{X} \rightarrow \mathcal{Y} \rightarrow \mathcal{X}$ and vice versa, does not guarantee a good translation. It ensures that the input image is resembled after a full circle, but the image from the intermediate domain can be arbitrary. The only loss that measures the quality of the output is the adversarial loss. However, the adversarial loss is the loss derived from the discriminator output, and the discriminator is trained in parallel to the generator. Therefore, the output of the discriminator for the same image changes during training. A low adversarial loss can mean that the generator synthesis quality is high or that the discriminator is poor at differentiat-

[2]The following sections extend the methods and the results of [153].

ing between real and synthetic images. The nature of the training task being a two-player game between generator and discriminator makes the adversarial loss unusable to determine the synthesis quality and therefore as a stopping criterion or for epoch selection. For the reasons mentioned above, the loss functions are not suitable for ending the training or selecting the best epoch, and GANs are still trained for a fixed number of epochs while using the last epoch as the final model.

The synthesis quality during training is currently not evaluated, but it is quantified after training for the final GAN model. Quantification has to be performed by comparing two sets of images (real images and those synthesized by the GAN). Therefore, it must be quantified how different the domains are from which the sets are sampled. Methods to solve this task, such as the Fréchet Inception Distance (FID) or the Kernel Inception Distance (KID), exist (see Section 1.3.5).

A metric that is used as a validation metric during GAN training must have both a high correlation with the downstream metric and a low computational effort. Downstream task metrics such as the AJI^{+} of a trained instance segmentation model cannot be used because it is computationally not feasible to train a supervised segmentation model model after each epoch. As of now, it is unclear whether metrics exist that meet the above requirements. Therefore, for the first time, a procedure is developed to test any metric for their use as GAN validation metrics, and experiments are carried out on two benchmark datasets.

3.3 Method

The workflow shown in Fig. 3.1 is developed to examine metrics with respect to downstream instance segmentation, while it can also be used for other downstream tasks, such as classification. The aim of the workflow is to enable a comparison between metrics acquired during the GAN training and a segmentation metric obtained after the GAN training by a downstream segmentation model.

To do so, a paired labeled dataset $D = (D^L, D^T)$ with labels D^L and target images D^T is split into training data D_{train}, validation data D_{val}, and test data D_{test}. The labels of the training data D^L_{train} are used to synthesize new unpaired synthetic labels S^L. Because ground truth is available, Statistical Shape Models (SSMs) can be used in addition to programming methods to synthesize new labels (see Section 1.4). The synthetic labels are split into training labels S^L_{train}, validation labels S^L_{val}, and test labels S^L_{test}. A GAN

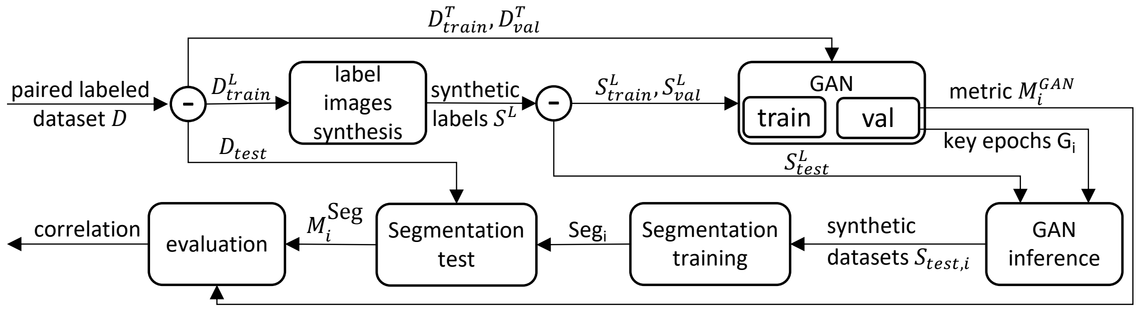


Figure 3.1 | Workflow to compare validation metrics acquired during GAN training to downstream segmentation metrics. If GAN validation metrics correlate with downstream segmentation metrics, they can be used for GAN epoch selection and early stopping. A paired labeled dataset D is needed for the evaluation. The minus sign indicates a data split. Figure 7 from [153] licensed under CC BY 4.0; Changed variable naming.

is trained with the targets D^T_{train} and the unpaired synthetic labels S^L_{train}. The validation metrics M_i^{GAN} are calculated after each epoch i. The generator of the GAN is used to synthesize synthetic validation targets S^T_{val} from S^L_{val} if a validation metric compares S^T_{val} to the real-world validation targets D^T_{val}. This is the case for the KID and the FID metric. No synthetic validation targets need to be synthesized during training for network metrics like the α-metric, because the layers of the generator are used for the calculations. Model checkpoints are extracted for key epochs G_i and the checkpoints are used to infer synthetic test targets S^T_{test} from S^L_{test}. Each paired synthetic dataset $S_{\text{test},i}$ is used to train a segmentation network Seg_i. In this work, a U-Net is utilized as the segmentation network. Each segmentation network is then used to predict labels from the test targets D^T_{test} and compare them with the test labels D^L_{test} resulting in the instance segmentation metric M_i^{Seg}. The AJI$^+$ is used as the instance segmentation metric. Finally, the metrics acquired during GAN training M_i^{GAN} are compared to M_i^{Seg} in the evaluation step.

3.4 Experimental Setup

The FID, the KID, and the α-metric are used as validation metrics (see Section 1.3.5) in the experiments. FID and KID are metrics to compare two unpaired sets of images regarding their similarity. A lower score means, that the sets are more similar. The α-metric is used to assess whether a NN is trained well by evaluating the neurons of the single layers. A low α-metric means that the NN is trained well. Paired image metrics cannot be used, since no paired data are available. To reduce the influence of the random initialization of neural networks, 3 GANs are trained for each dataset and 3 U-Nets are trained for each key epoch. The neural networks and datasets used for the experiments are described in the remainder of this section.

3.4.1 Neural Networks

The CycleGAN architecture is used for the unsupervised image-to-image translation and the mean squared error (MSE) is used for all loss functions together with the Adam optimizer. The scaling factors λ_{cycle} and λ_{idt} for the cycle-consistency loss $\mathcal{L}_{cycle}$ and the identity loss $\mathcal{L}_{idt}$ are set according to the original implementation of CycleGAN to λ_{cycle}=10 and λ_{idt}=5. Each generator utilizes a ResNet-Generator with 96 feature maps in the initial convolutional layer, instance normalization, and nine ResNet blocks in the feature space. Additionally, a PatchGAN-Discriminator with instance normalization is employed for each discriminator. Each CycleGAN is trained for a total of 400 epochs, while the initial learning rate is 0.0002 and is linearly decreased towards zero after 200 epochs trained with the initial learning rate. A key epoch G_i is extracted every 50 epochs, resulting in eight key epochs. Every 50th epoch is chosen in order to limit computational expense. In addition, the epochs in which FID, KID, or the α-metric are the smallest are extracted[3].

The Karlsruhe Image Data Annotation (KaIDA) tool with the U-Net architecture is used for the segmentation task [154]. The U-Net encoder consists of a ResNet-50 pre-trained on ImageNet. SmoothL1Loss is used together with the Adam optimizer. Each network is trained for a maximum of 200 epochs, with early stopping implemented after 50 consecutive epochs with no decrease in the validation data metric. Furthermore, the initial learning rate of 0.001 is halved after 20 consecutive epochs with no decrease in the validation data metric. The AJI$^+$ is used for the validation metric and the AJI$^+$ is also used for the test metric. Both datasets are augmented with the Albumentations framework during the training of the U-Net [155]. To reduce the influence of the augmentations on the results, a simple augmentation pipeline is used. Each image is normalized to the range $[0, 1]$ with the minimum and maximum value present in the image. Afterwards, the images are flipped with a probability of 50%. Gaussian blur with a random $\sigma \in [0.001, 0.2]$ is applied with a probability of 30%. The image is shifted, scaled, and rotated with a probability of 50%. The random shift and scaling are in the range of $[-5\%, 5\%]$ of the image size, and the random rotation is between -5 and 5 degrees. Finally, contrast and brightness are adjusted with a probability of 50% with random brightness and contrast adjustments in the range

[3]Details of the implementation can be found at the link provided under https://github.com/MoritzBoe/BMT_GAN_stopping.

of $[-0.1, 0.1]$.

3.4.2 Datasets

The datasets must be selected with respect to the following two specifications:

1. It must be evaluated whether the complexity of the GAN training task influences the possibility of using the metrics during GAN training.

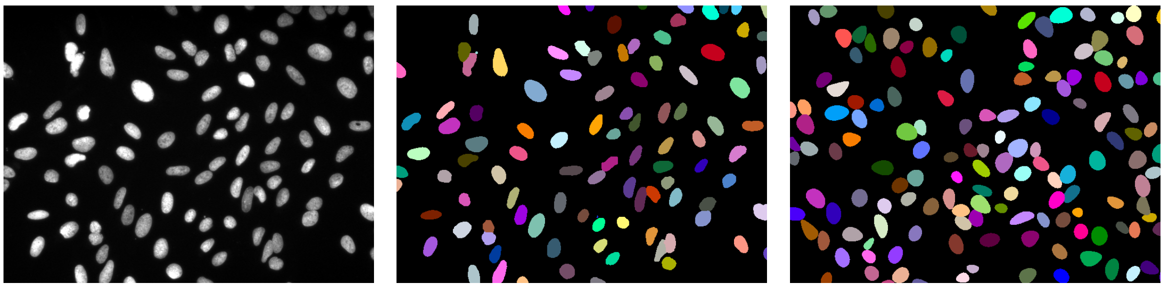
2. KID and FID are developed for photographic RGB images, and it is unclear whether they work on grayscale microscopy images. Therefore, it must be evaluated whether the representation of the image data has an influence on the applicability of the various metrics.

Due to the requirements described above, the experiments are carried out on two different datasets. The first dataset is the grayscale BBBC039v1 dataset, which is a low diversity fluorescence microscopy dataset. The second dataset is the Lizard dataset, which is a dataset of high diversity histopathological RGB images. High diversity results in a more difficult training task for the GAN.

BBBC039v1 Dataset

The BBBC039v1 dataset[4] contains 200 images of U2OS cells acquired by fluorescence microscopy using Hoechst staining [150]. Each of the 16-bit grayscale images has a size of 520 px × 696 px. Each image is normalized between the 1% and the 99% percentiles of the image. The values below and above the threshold are set to the minimum or maximum pixel value. The ground truth instance segmentation of the cell nuclei is manually annotated. The dataset is split into 120 images for GAN training D^T_{train}, 40 validation images D^T_{val} for the evaluation of the GAN metric, and 40 image and label pairs D_{test} for the downstream segmentation metric. A total of 640 synthetic labels S^L are created using EFDs and the existing training labels D^L_{train}. For information on how EFDs are used to synthesize label images in this work, see Section 2.3.1. 120 synthetic label images S^L_{train} are used for GAN training and 40 synthetic label images S^L_{val} are used for the evaluation of the GAN metrics. Another 480 images S^L_{test} are synthesized for GAN inference to create the synthetic datasets $S_{\text{test},i}$, where 360 synthetic labels are used for U-Net training and 120 synthetic labels are used for U-Net validation. Example images are shown in Fig. 3.2.

[4]The dataset is available at https://bbbc.broadinstitute.org/BBBC039/, accessed 20.03.2024.



a) BBBC039v1 target b) BBBC039v1 label c) BBBC039v1 synthetic label

Figure 3.2 | Example image from the BBBC039v1 dataset (a) with the corresponding label (b) and a synthetic label (c).

The synthetic label images are augmented during GAN training. First, the label is binarized, where 0 represents a background pixel and 1 a foreground pixel. Second, Gaussian noise with a variance of 0.1 and a mean of zero is applied and the resulting image is clipped to the range [0, 1]. Finally, the image is normalized to the range [-1, 1]. The targets are also normalized to the range [-1, 1].

Lizard Dataset

The CoNiC (Colon Nuclei Identification and Counting) Challenge data preprocessing of the Lizard dataset is used in this work [54]. The dataset contains Hematoxylin and Eosin (H&E) stained histopathological images of colon tissue from 16 medical centers. Having different people processing tissue samples and using different scanners results in high diversity. The dataset contains 4981 image pairs of size 256 px $\times$ 256 px with a resulution of $0.5 \mu m$ per pixel. The label images are created semi-automatically with a manual refinement step. Example images are shown in Fig. 3.3.

The spatial properties and background structures in histopathological images are complex and the spatial properties of the nuclei are diverse due to the variety of patients. In contrast to the BBBC039v1 dataset, nuclei in histopathological images are e.g. crowded or aligned around background structures. Using wrong spatial properties, e.g.: by placing the nuclei randomly in an image, reduces the synthetic image quality and therefore can potentially influence the results of the experiments. To avoid these drawbacks, provided labels are used instead of the synthetic ones. To ensure that no information of paired images from the Lizard dataset is available in the complete pipeline, the corresponding targets are removed whenever the provided labels are used and vice versa. This results in the following data split according to the workflow depicted in Fig. 3.1: All images are shuffled, and images without nuclei are removed. A total of 70% of the data is used for

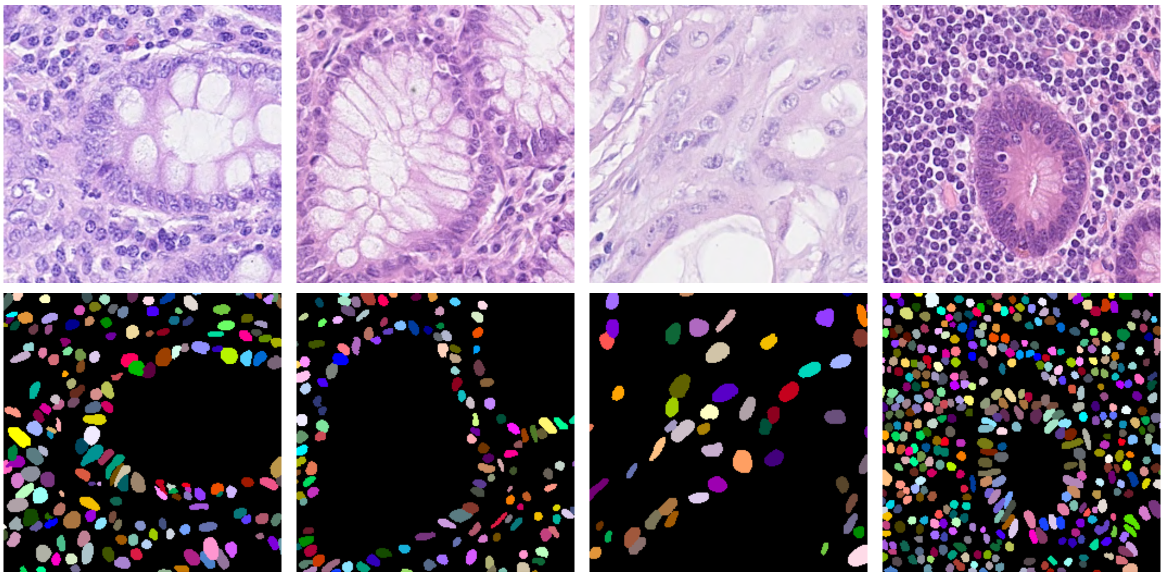


Figure 3.3 | Example images from the Lizard dataset with targets (top row) and corresponding labels (bottom row). The dataset shows high diversity due to the different patients, scanners, and medical personnel processing the samples. No label images are synthesized, instead a set of real-world labels is used for GAN training.

GAN training and validation. 80% of those data are used for GAN training, while the labels of the first half are used as the synthetic labels S^L_{train} and the targets of the second half are used as D^T_{train}. The other 20% of the data are used for GAN validation, and the labels and images are divided as before into S^L_{val} and D^T_{val}. The remaining 30% of the complete dataset are split into the synthetic test labels S^L_{test} (80%, only labels) for the GAN inference and D_{test} (20%, labels and targets) for the downstream segmentation metric.

The label images of the Lizard dataset are also augmented during GAN training. The grayscale labels are converted into a binary segmentation mask and then into an RGB image by repeating the values for each color channel. To ease the learning task, a novel color augmentation technique was implemented for the Lizard dataset. As shown in Fig. 3.3, the color (staining) of the targets differs greatly. Therefore, information on which staining to synthesize must be added to the labels. Without this augmentation, normalization of the staining is necessary because the GAN is unable to synthesize the diverse color properties present in the dataset [153]. The stain information is added by extracting the 10 most prominent colors from the histogram of a randomly sampled target image. Subsequently, the mean value of the 10 colors is calculated for each channel. The background of the label image is set to the mean value and the foreground is set to the mean value times 0.5, resulting in darker foreground cell nuclei.

Algorithm 1 Pseudocode to augment the color of an instance segmentation label (label) according to a reference image (target). The method ones_like(a) creates an array of ones of the same shape as a and stack(a, times, axis) stacks an array a for a number of times along a given axis.

```
Require: label, target
Ensure: new_label
  label ← stack(label, times = 3, axis = −1)
  top_colors ← extract_top_colors(target, 10)
  mean_color ← mean(top_colors)
  new_label ← ones_like(label) ∗ mean_color
  new_label[label > 0] ← 0.5 ∗ mean_color
```

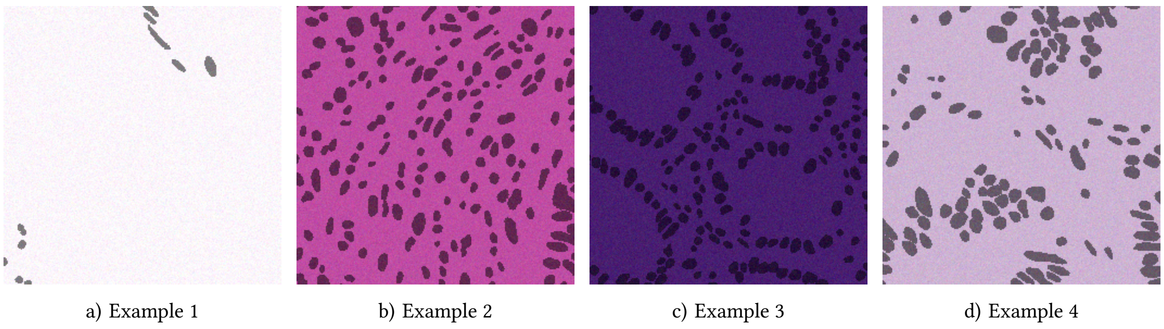


Figure 3.4 | Example label images with applied augmentations for the Lizard dataset. The color of the binarized labels is adapted according to a random reference image. This enables the GAN to synthesize target images with different stainings. The image in (a) is created by using a label image with few nuclei and a reference image that contains a very light purple as the mean color. The image in (b) is created by using a label image with many nuclei and a reference image that contains a darker purple as the mean color. Images (c) and (d) show that the nuclei are not randomly distributed in all images. Both are also created with different reference images.

The pseudocode to create the color-augmented labels from the instance segmentation mask is given in Algorithm 1.

After adjusting the color, Gaussian noise with a variance of 0.001 and a mean of zero is applied and the resulting image is clipped to the range [0, 1]. Finally, the image is normalized to the range [-1, 1]. The targets are also normalized to the range [-1, 1]. Example label images with applied augmentations are shown in Fig. 3.4.

3.5 Results

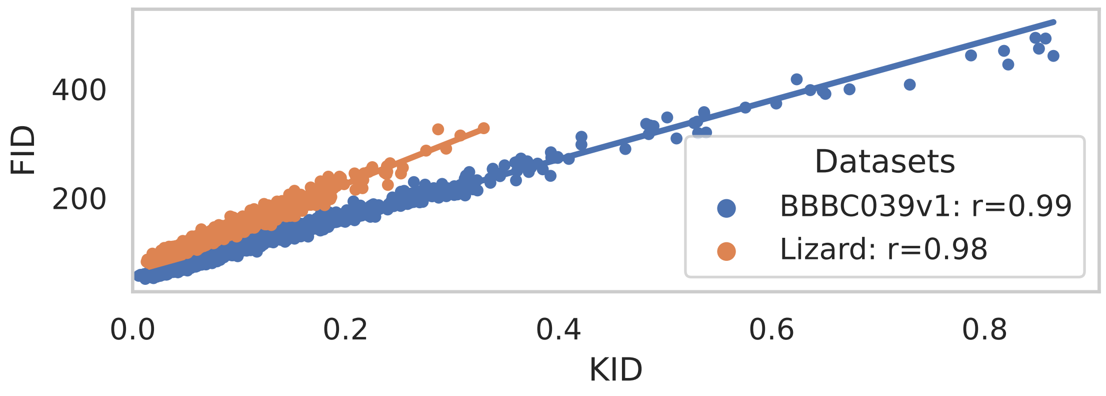
Correlations between the metrics used during GAN training are examined in Section 3.5.1. Subsequently, the correlation between the final AJI$^+$ metric acquired after training a U-Net and the metrics acquired during GAN training are examined for the BBBC039v1 dasetaset in Section 3.5.2 and the Lizard dataset in Section 3.5.3. A total of 6 GANs and 198 U-Nets (3 U-Nets per key epoch with 8 fixed key epochs and key epochs for the best KID, FID, and α-metric for each GAN) were trained. Training was carried out on the three different types of GPU, which are the NVIDIA RTX 6000 Ada Generation, the NVIDIA TITAN RTX, and the NVIDIA TESLA V100. The total training time is 438 hours.

3.5.1 Metrics

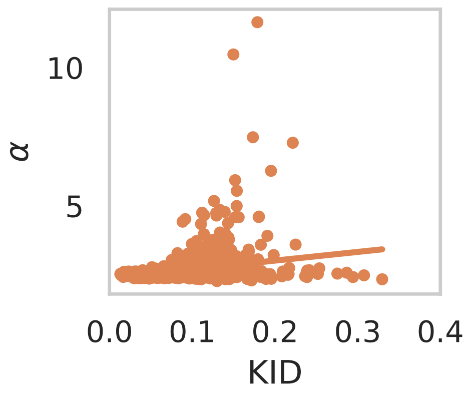
One metric can be replaced by the other if they are highly correlated. This section therefore examines whether it is possible to abstain from using one of the metrics due to the correlation with another metric. Both FID and KID compare distances in the feature space of a pre-trained neural network (see Section 1.3.5). A strong Pearson correlation can be observed for the Lizard dataset with r = 0.98 and for the BBBC039v1 dataset with r = 0.99. A scatter plot showing FID over KID for all epochs and all trained GANs is shown in Fig. 3.5a. In addition, a linear regression model is added for each of them separately. It can be observed that the linear regression leads to different models for both datasets. For the Lizard dataset, the FID gets worse faster than for the BBBC039v1 dataset compared to the KID. This behavior is not important for the experiments carried out in this work. As long as both metrics are highly correlated for the same dataset, either of the two can be used. Therefore, the KID is used for the remainder of this work.

The calculation of the α-metric differs from the KID and FID, as it is not image-based. Instead, it evaluates whether the layers of a neural network are well trained. Plots comparing the KID to the α-metric are shown in Fig. 3.5b and Fig. 3.5c. The Pearson correlation between KID and the α-metric on the Lizard dataset is r = 0.28. The Pearson correlation on the BBBC039v1 dataset is r = 0.46. The correlation between KID and the α-metric for both datasets is not high enough to substitute one for the other. Therefore, the α-metric must be examined in addition to the KID in the remainder of this chapter.

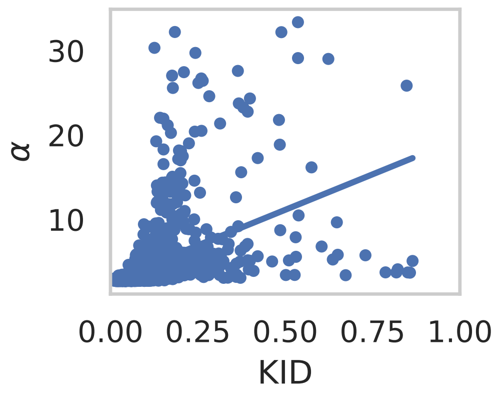
The computational effort of the individual metrics must also be examined. If the calculation of a metric takes too long in relation to the duration of an epoch, this metric cannot be used effectively. Using the KID metric for validation on the Lizard dataset results in an additional computation expense of 4 seconds per epoch on a computer with an Intel Xeon Gold 6136 CPU and a NVIDIA V100 16GB GPU, while a GAN epoch lasts 229 seconds. Using the FID metric results in an additional computational expense of 8 seconds per epoch, whereas the α-metric calculations for the generator take 7 seconds.



a) KID and FID for both datasets



b) KID and α for the Lizard dataset

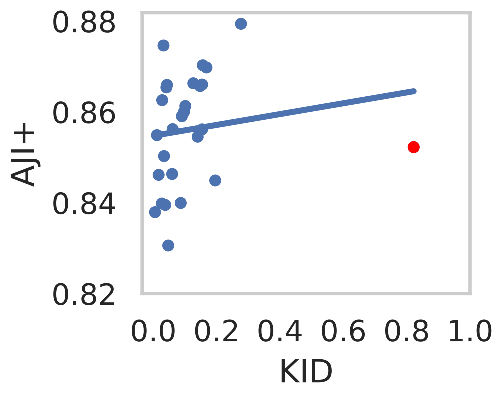


c) KID and α for the BBBC039v1 dataset

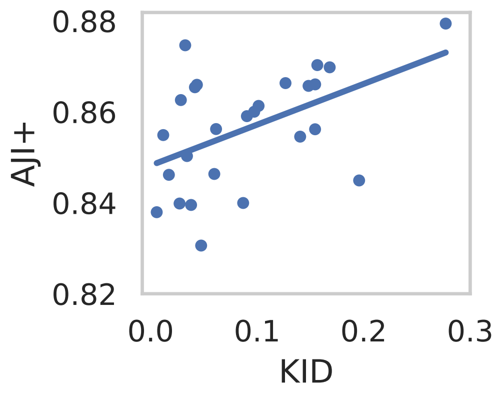
Figure 3.5 | Scatter plots and linear regression (straight lines) for the Lizard and the BBBC039 dataset. KID is compared to FID in (a) for the BBBC039v1 dataset in blue and the Lizard dataset in orange. The Pearson correlation for both datasets is strong with r = 0.98 for the Lizard dataset and r = 0.99 for the BBBC039v1 dataset. The linear regression between KID and FID differs for both datasets. KID is compared to the α-metric for the Lizard dataset in (b) and for the BBBC039v1 dataset in (c). The Pearson correlation r between KID and the α-metric for both datasets is low with r = 0.28 for the Lizard dataset and r = 0.46 for the BBBC039 dataset. The KID cannot be used to substitute the α-metric, but it can be used to substitute the FID.

3.5.2 BBBC039v1 Dataset

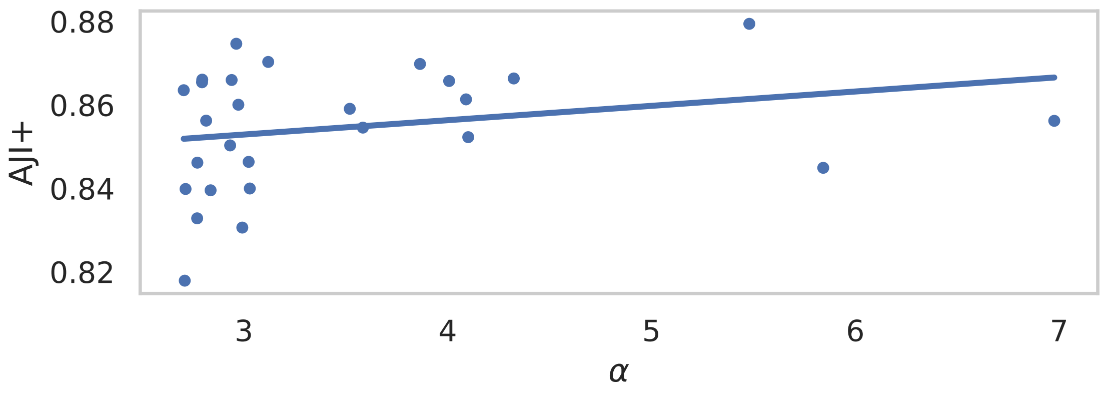
First, it is examined whether the segmentation metric (AJI$^+$) correlates with the GAN metrics (KID or α). To evaluate whether the AJI$^+$ correlates with the KID, the eight fixed epochs (epochs [50, 100, 150, ..., 400]) and the epoch with the lowest KID are examined and 3 GANs are trained, resulting in 27 key epochs G_i. Furthermore, 3 U-Nets are trained for each key epoch and the U-Net with the median AJI$^+$ for each key epoch is used to reduce the impact of outliers. A plot comparing AJI$^+$ to KID is shown in Fig. 3.6a. The



a) KID and AJI$^+$ with outlier in red, r = 0.15



b) KID and AJI$^+$ without outlier, r = 0.48



c) α and AJI$^+$, r = 0.25

Figure 3.6 | Scatter plots and linear regression for KID and AJI$^+$ (a, b) and the α-metric and AJI$^+$ (c). The outlier marked in red in (a) is removed in (b) resulting in a Pearson correlation of r = 0.48 with a p-value of 0.012. The Pearson correlation of the α-metric and AJI$^+$ is small with r = 0.25 and p = 0.206.

Pearson correlation of r = 0.15 is low, but a strong outlier, which is marked in red, can be observed on the right side. If the outlier is removed (Fig. 3.6b), r rises to r = 0.48 with a p-value of 0.012. The p-value for a given sample with the Pearson correlation r is the probability that abs(r')>abs(r), where r' is the Pearson correlation of a random sample x' and y' where x' and y' are both drawn from a population with zero correlation[5].

Although the correlation is stronger, a positive correlation was not to be expected. A low KID means high-quality synthetic images and a high AJI$^+$ means good segmentation results. The positive Pearson correlation implies that the segmentation results were better if the synthetic image quality was lower. However, the difference between the best median

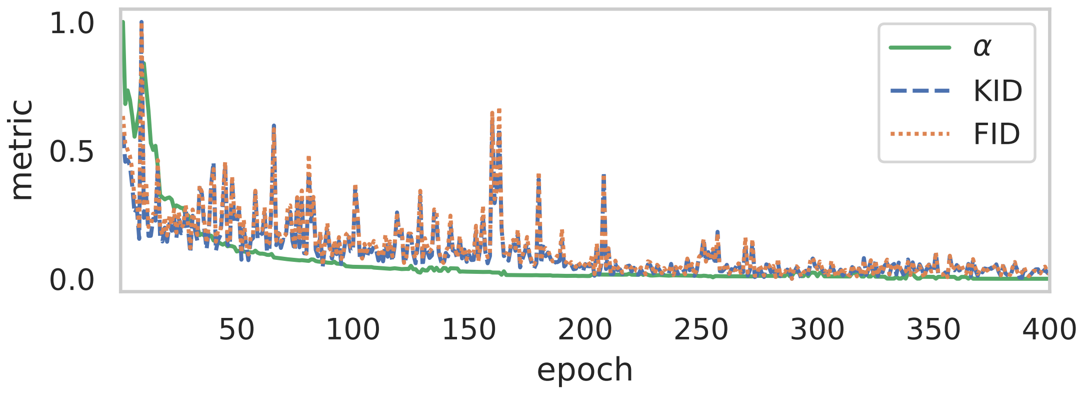
[5]The Pearson correlation and the p-value are calculated with SciPy [156]. The implementation can be found at https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html, accessed 12.02.2024

AJI$^+$ of 0.88 and the worst of 0.83 is small and the segmentation results are of high quality for all key epochs.

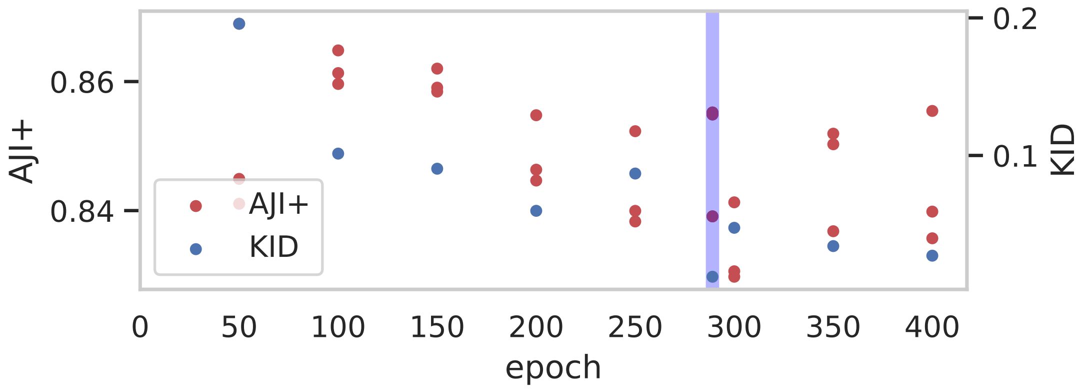
The same procedure is carried out for the α-metric, but the epoch with the lowest α-metric is used instead of the epoch with the lowest KID. A scatter plot with the corresponding linear regression is shown in Fig. 3.6c. The Pearson correlation of $r = 0.25$ with $p = 0.206$ is low and no sufficient correlation between the α-metric and the performance of downstream segmentation was observed. Therefore, the α-metric cannot be used to draw conclusions regarding downstream segmentation.

An in-depth analysis of the metrics for a single GAN is performed to evaluate why AJI$^+$ and KID are positively correlated and why the α-metric does not correlate with the performance of downstream segmentation. A plot is shown in Fig. 3.7. The plots and an evaluation for the other two GANs are provided in Section A.1. Fig. 3.7a shows the metrics acquired during the GAN training for all epochs. It shows that KID and FID yield a strong correlation. Both decrease during training, but the swings are large, with strong and rapid changes. The large swings and rapid changes omit the use of the metrics for early stopping of GAN training, while using the metric to select the best epoch is still possible. For later epochs, the changes are smaller. This is due to the decreased learning rate between epoch 200 and 400. For all three GANs, the decrease of the α-metric after epoch 200 is small, while the other two metrics show that the image quality still changes during training. This is another indication that the α-metric cannot be used to select the best GAN epoch. A well-trained generator with a low the α-metric does not necessarily lead to the best images for downstream segmentation tasks.

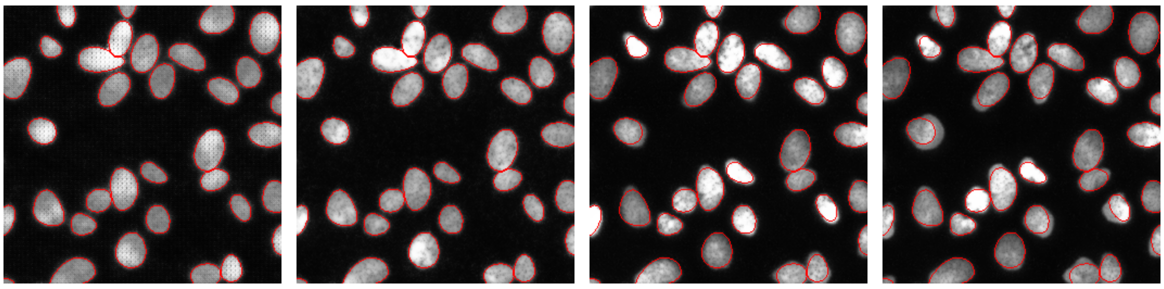
The KID for the nine key epochs together with the AJI$^+$ for the three U-Nets trained for each key epoch are shown in Fig. 3.7b. In general, KID decreases during training. The key epoch with the best segmentation results is epoch 100 and the epoch with the lowest KID is epoch 289. Both are far from the last epoch, which is normally used for inference after GAN training. Exemplary images synthesized by the GAN for epochs 50, 100, 289 and 400 are shown in Fig. 3.7c. The contours of the objects in the label image used to synthesize the target images are depicted with a red overlay. The image synthesized after epoch 50 shows that the synthesis quality is still minor and that the nuclei yield checkerboard artifacts. Training with these synthetic images results in a reduced AJI$^+$ score. The image synthesized with epoch 100 looks visually appealing and the U-Nets trained on images of epoch 100 yield the best AJI$^+$ scores. The nuclei in the image of epoch 289 have a more distinctive color pattern compared to the earlier epochs, but the GAN started to change the shape properties. Some nuclei are deformed compared to the red overlay. This is



a) Metrics acquired during GAN training



b) KID for key epochs and corresponding AJI^{+} scores



c) Crops of the synthetic images for epochs 50, 100, 289 and 400

Figure 3.7 | The metrics acquired during GAN training for all epochs are shown in (a). Each metric is normalized to the range [0, 1]. KID and FID are highly correlated and have strong peaks. The α-metric has a more steady decrease. The learning rate decreases from 0.0002 at epoch 200 to 0 at the end of epoch 400. The KID for the nine key epochs and the corresponding AJI^{+} scores of the U-Nets are shown in (b). Epoch 289 has the lowest KID and is marked in purple. Crops of target images synthesized from the same label image for different epochs are shown in (c). The contours of the objects in the synthetic label image are depicted with a red overlay. The GAN starts to alter the shape properties in late epochs.

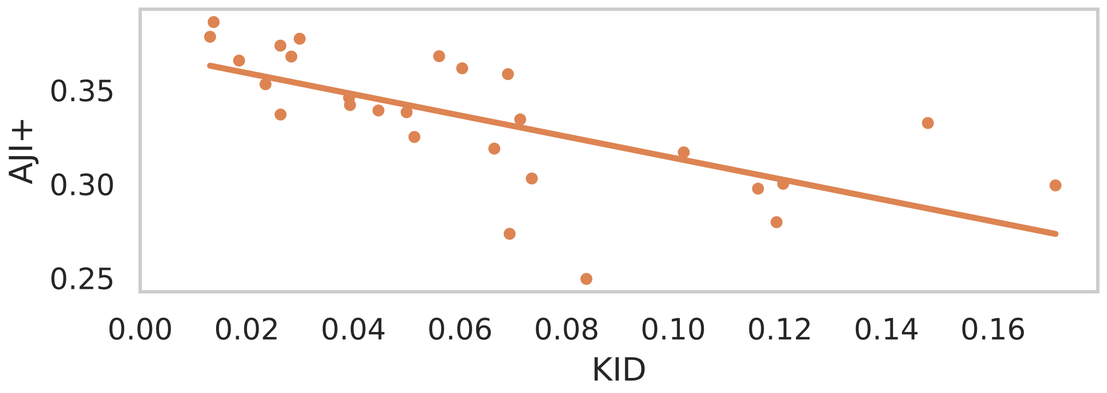
even more prominent for the image synthesized with epoch 400. The shape deformation can have several reasons. One reason could be that the discriminator has memorized the complete training data. Therefore, the generator tried to trick the discriminator by adding changes to the images. This is unlikely because the generator did not change the place or add or remove objects, and the discriminator could still learn the position of objects to determine whether the image is in the set of real target images or not. The most likely reason is that the ellipses do not perfectly resemble the shape of the nuclei, and therefore the generator needs to change the shape in the synthetic target images to match the shape in the real-world images. Although this is beneficial for the KID, it is not beneficial for downstream segmentation. The problem can be eliminated with better synthetic label images, but researchers are often unable to synthesize better synthetic label images, and ellipses are still the state-of-the-art for cell nucleus synthesis (see Section 1.4).

The positive correlation between KID and AJI^{+} can be explained by the above behavior. A U-Net trained with the images of epoch 100 is capable of learning the segmentation task. The change of the shape properties for later epochs reduces the ability of the U-Net to adequately predict the shapes for the test target images, resulting in a small decrease of the AJI^{+} metric. It can be compared to low-quality manual labeling. Although all objects are labeled, the shape is imperfect.

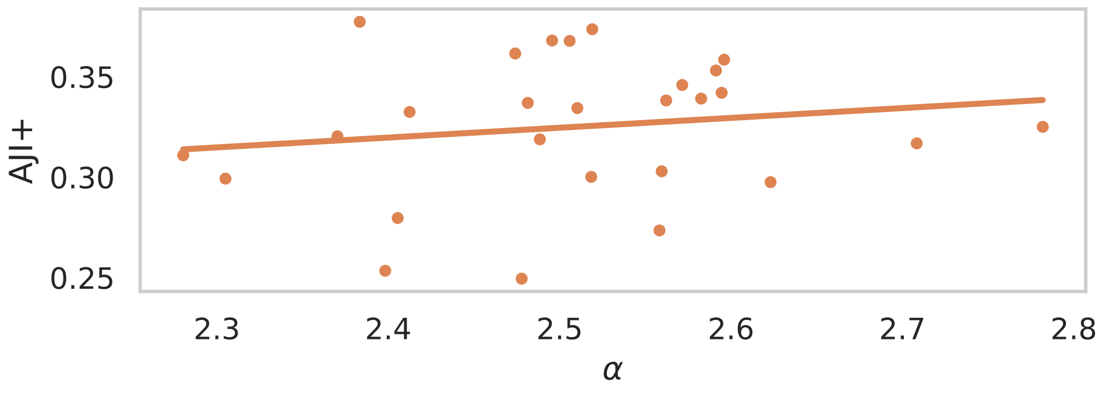
3.5.3 Lizard Dataset

It is first evaluated whether there is any correlation between the GAN metrics and the AJI^{+}. Fig. 3.8a compares the KID score of an epoch with the median AJI^{+} score of the U-Nets trained for downstream segmentation. The eight fixed key epochs [50, 100, 150, ..., 400] and the best KID epoch for each of the three GANs trained are shown. The Pearson correlation is $r = -0.67$ with $p = 0.00013$. Therefore, there is a correlation between synthetic images similar to the real-world images and good segmentation results. For high KID values, the variation of the AJI^{+} compared to the linear regression increases. Comparing the α-metric to AJI^{+} results in $r = 0.16$ and $p = 0.43$. Again, as with the BBBC039v1 dataset, there is no correlation between the α-metric and the downstream segmentation results. The differences in the median AJI^{+} score are greater compared to the BBBC039v1 dataset. The highest value is 0.38 and the lowest value is 0.25. Furthermore, the AJI^{+} scores are generally lower, due to the more complex instance segmentation task and the more diverse data.

Exemplarily, the training of one GAN is displayed in Fig. 3.9. The other two trained



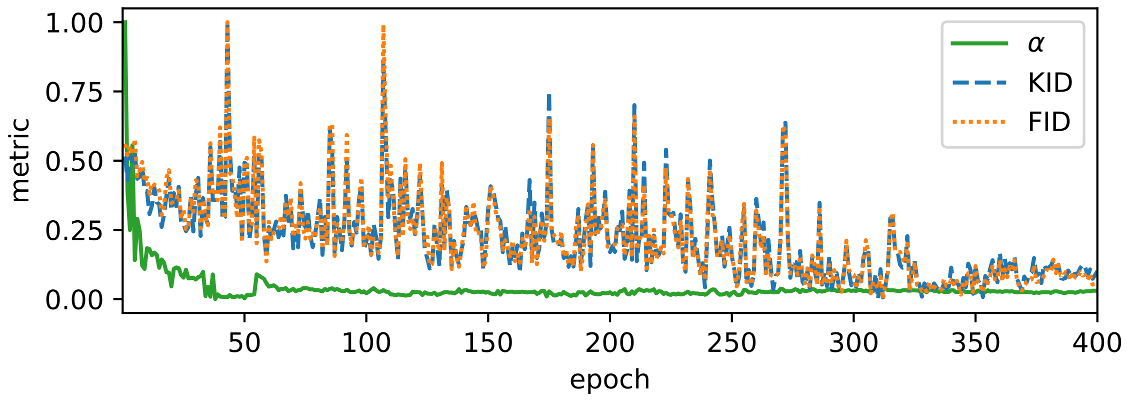
a) α and AJI$^+$, r = -0.67



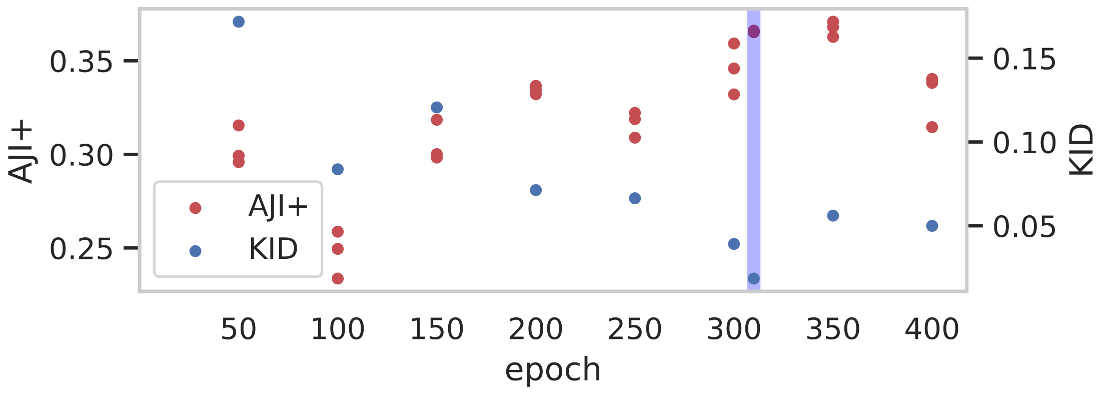
b) α and AJI$^+$, r = 0.16

Figure 3.8 | Scatter plots and linear regression for KID and AJI$^+$ (a) and the α-metric and AJI$^+$ (b). The Pearson correlation in (a) is r = -0.67 with a p-value of 0.00013. The Pearson correlation of the α-metric and AJI$^+$ is small with r = 0.16 and p = 0.43.

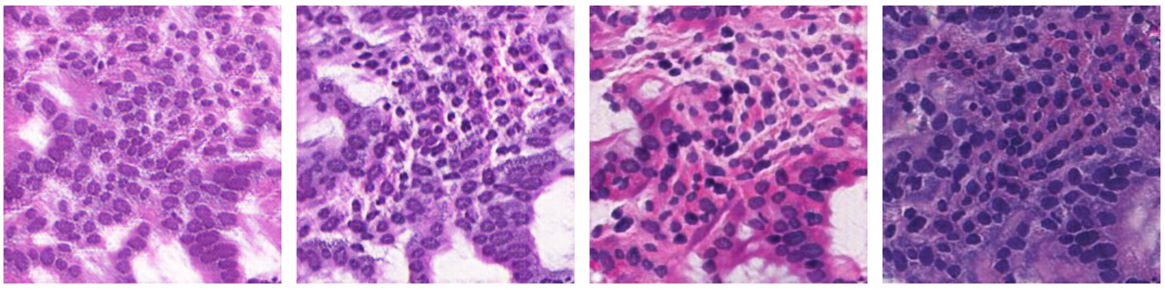
GANs are shown and described in Section A.1. The metrics during training, each normalized to the range $[0, 1]$, are displayed in Fig. 3.9a. As for the BBBC039v1 dataset, the α-metric decreases and is stable. It stabilizes after epoch 50 compared to epoch 200 for the BBBC039v1 dataset. However, a direct comparison is difficult because the images in the datasets differ and the number of images per epoch differs significantly with 1355 images per epoch in the Lizard dataset and 360 images per epoch in the BBBC039v1 dataset. Again, there is no correlation between the α-metric and the synthetic target image quality or the downstream AJI$^+$ score. The FID and KID scores have strong sudden peaks, while they decrease throughout training. Strong peaks occur even after the learning rate starts to decline in epoch 200. The peaks are only small for the last 70 epochs. These peaks



a) Metrics acquired during GAN training



b) KID for key epochs and corresponding AJI^{+} scores



c) Crops of the synthetic images for epochs 50, 100, 310 and 400

Figure 3.9 | The metrics acquired during GAN training for all epochs are shown in (a). Each metric is normalized to the range [0, 1]. KID and FID are highly correlated and have strong peaks. The α-metric has a more steady decrease. The KID for the nine key epochs and the corresponding AJI^{+} scores of the U-Nets are shown in (b). Epoch 310 has the lowest KID and is marked in purple. All three U-Nets trained for epoch 310 have an AJI^{+} close to 0.37. Crops for the same label image synthesized for different epochs are shown in (c). The overlay of the contours of the object in the label image is not shown to allow a better visual assessment of the image quality.

Table 3.1 | Median AJI$^+$ and the KID for the last epoch and the epoch with the lowest KID for each trained GAN. The epoch with the lowest KID always outperforms the last epoch.

GAN	epoch	median AJI+	KID
1	310	**0.37**	0.019
	400	0.34	0.050
2	319	**0.39**	0.014
	400	0.35	0.024
3	317	**0.38**	0.013
	400	0.37	0 028

can be an indicator of unstable training or an initial learning rate that is too high for this complex and diverse dataset. The strong peaks also prevent the metrics from being used for early stopping of the training on this dataset.

The KID is compared with the AJI$^+$ in Fig. 3.9b. The last epoch is not the epoch with the best KID score. This is the case for all 3 trained GANs. The KID of 0.08 at epoch 100 decreases by a large margin compared to the KID of 0.17 at epoch 50, but the mean AJI$^+$ is also lower (0.25 to 0.30). This opposes the negative correlation observed for all three trained GANs. Fig. 3.8a shows that this data point is the data point with the largest distance from the linear regression line and can be treated as an outlier. The best KID of 0.019 is achieved for epoch 310 with a median AJI$^+$ of 0.37.

Samples of targets synthesized for different epochs are shown in Fig. 3.9c. Unlike the BBBC039v1 dataset, no changes in shape properties can be observed. The generator did not need to change the shape to fool the discriminator since real-world labels have been used to train the GAN. Visual inspection shows that the image quality for epochs 50 and 100 is worse than the one for epochs 310 and 400. Because the GAN changes the color properties during training and the data is diverse, it is difficult to manually select the better epoch comparing epochs 50 and 100 and epochs 310 and 400.

The U-Net with the median AJI$^+$ trained on the data synthesized by the GAN with the best KID (epoch 310) outperforms the U-Net with the median AJI$^+$ trained on the last epoch. This is the case for all three trained GANs. The AJI$^+$ scores of the epochs with the best KID scores are compared to the last epochs in Table 3.1.

3.6 Discussion

Training downstream NNs as a validation metric is computationally not feasible, and metrics that require much less computation are needed. Two types of metrics, the KID and FID image metrics, which can quantify the difference between two sets of images, and the α-metric that focuses on the weights of a NN, have been evaluated in terms of their usability as validation metrics. The KID and FID are highly correlated, whereas there is no correlation to the α-metric. Therefore, there is no benefit in calculating KID and FID in parallel during training, while calculating only one further decreases the computational cost.

No correlation between the α-metric and the downstream segmentation performance could be detected on the BBBC039v1 dataset and the Lizard data. Therefore, it cannot be used as a validation metric. This is surprising since it has been shown that the α-metric correlates with the classification accuracy for a variety of different NNs [128].

The results for KID and FID differ for both datasets. For the simpler and less diverse BBBC039v1 dataset, there is a positive correlation between the GAN metrics and the AJI$^+$. This is counterintuitive because this means that a downstream segmentation NN trained with synthetic images less similar to the real-world images performs better than a downstream segmentation NN trained with more similar synthetic images. Having a closer look at the images synthesized for different GAN epochs reveals that the GAN adapts the shape of the objects in the synthetic label images to the shape of the objects in the real-world images. Although this decreases the difference between the image sets and results in better KID and FID scores, it is not desirable for downstream segmentation. Image-based metrics cannot detect whether the GAN alters the properties of objects. Usually, GANs start to focus on fine-grained properties, such as the exact shape of objects, during later epochs. If a high-quality segmentation of the shape is required, one has to manually inspect the synthetic target images together with the synthetic labels and choose an earlier epoch, in which the GAN did not start to change the shape. Alternatively, these changes indicate suboptimal synthetic label images, which can be adapted, if possible. The results show that training an additional GAN is no solution to this problem, since all three GANs showed this behavior.

The results for the complex and more diverse Lizard dataset yield a different behavior. There is a negative correlation between KID, FID and the AJI$^+$. Therefore, the metrics can be used for validation during GAN training. This is particularly remarkable as the metrics use neural networks that have been trained on fundamentally different data.

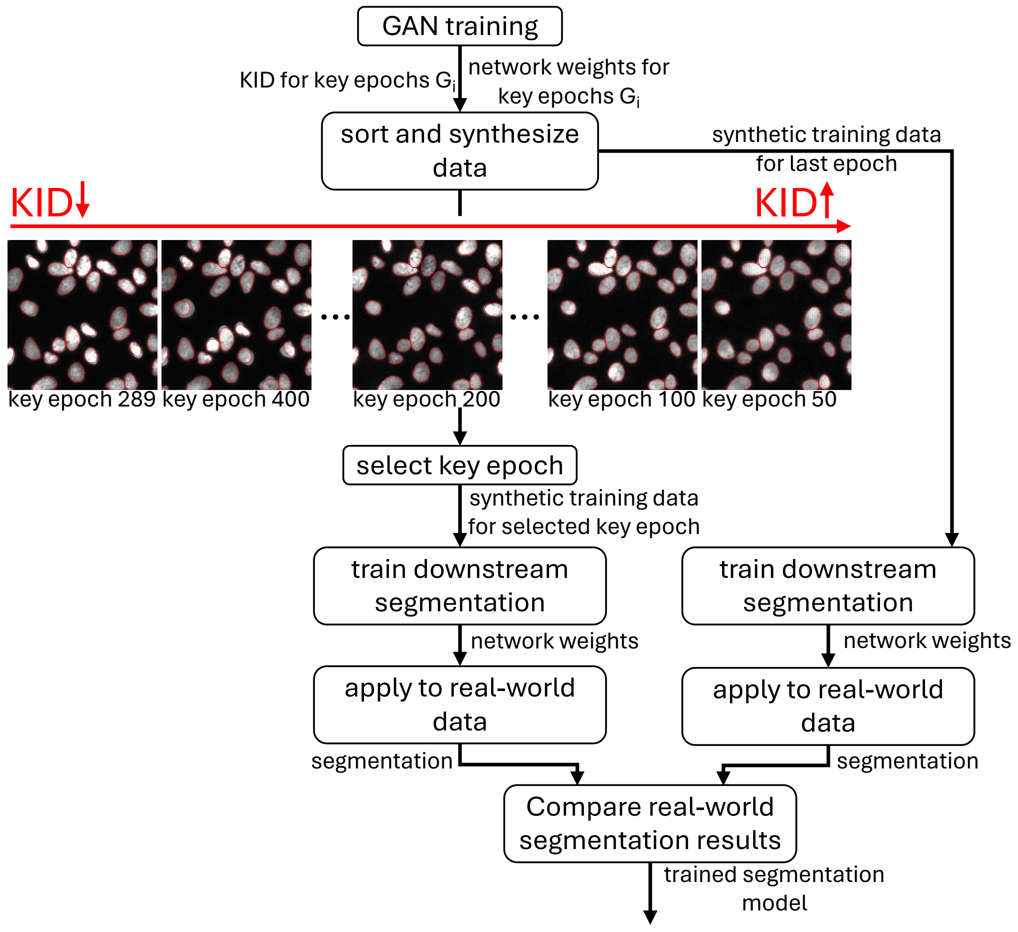


Figure 3.10 | Semiautomatic workflow to select the best epoch after GAN training. The KID and the network weigts are saved for key epochs during GAN training. After GAN training, data is synthesized and sorted by the KID in an ascending order. The epoch with the lowest KID and matching object contours is selected. The synthetic training data of the selected key epoch is used to train a segmentation model and the segmentation model is applied to real-world data. The segmentation results are compared to segmentation results of a segmentation model trained with data synthesized by the GAN after the last epoch. The example images for the key epochs are from a GAN trained on the BBBC039v1 dataset. For this GAN, epoch 200 is the key epoch with the lowest KID, where the object contours in the synthetic label images and the synthetic targets match.

The GAN does not change the object contours for later epochs because the label images used for GAN training are real-world labels that yield high quality. Using the epoch with the lowest KID metric resulted in better U-Net performance than using the last epoch for all three trained GANs. This is especially important because a purely visual inspection of the images does not make it possible to determine which epoch can be used to synthesize images that look most similar to the real-world images.

The different results on both datasets lead to the conclusion that the metrics cannot be used on new unknown datasets without further adjustments. This is because it is not possible to determine in advance whether a GAN will start to change the object shapes (BBBC039v1 results) or not (Lizard results). This would result in KID being a poor metric to select the best epoch. In order to be able to apply the KID as a validation metric to an unknown dataset, the following semiautomatic procedure (see Fig. 3.10) that includes visual inspection of the synthetic target images is proposed:

The KID and the network weights for key epochs G_i during GAN training are saved. After GAN training, data are synthesized for each key epoch and the data are sorted by the KID in ascending order. The synthetic data are used to select the key epoch with the lowest KID that has matching object shapes. This can be done by using synthetic target images with an overlay of the contour of the corresponding synthetic label images for visual inspection. After the key epoch is selected, a downstream segmentation model is trained with the data and the trained model is applied to real-world data. Additionally, the last epoch is used to train a segmentation model and the trained model is applied to real-world data. Finally, both segmentation results on real-world data are compared, and the best segmentation model is selected.

The proposed workflow mitigates the problems that arise due to altered shapes for late epochs (as for the BBBC039v1 dataset), while for datasets without altered shapes (as for the Lizard dataset) the best KID epoch is chosen. However, a manual assessment of the synthesis is required with regard to the shape of the translated objects in the synthetic target images. As a fallback scenario, a segmentation model is also trained with data synthesized with the GAN after the last epoch. The computational overhead is reduced comparing the new workflow to the current state-of-the-art where multiple GANs and multiple segmentation models are trained, while the downstream performance is increased.

Both KID and FID fluctuate greatly during GAN training. Therefore, they cannot be used for early stopping since it is not guaranteed whether the GAN produces better images for later epochs. However, this is not due to the metrics, but to the unstable training procedure of GANs itself. It is possible that they can be used for early stopping if GAN architectures with stable training are developed in the future. Furthermore, elaborate methods for key epoch selection can be explored. For example, it may be beneficial to pick a key epoch just before a strong upward peak occurs.

Patch-based Unpaired Image-to-Image Translation 4

4.1 Overview

GANs are capable of synthesizing high-quality data that can be used for downstream tasks (see Section 1.3.3). The existing literature shows that GANs are mainly used for applications in biology, medicine, and material science. The benefits of GANs for solving segmentation tasks are the greatest in these fields. However, high-quality 2D and 3D data need to be synthesized often.

The last step during synthesis is the inference highlighted in Fig. 4.1. Errors that occur during inference can cancel out improvements in the synthesis process of the previous steps like label image synthesis, or GAN training. For small images, inference is straightforward and the unpaired synthetic training labels $\hat{\mathcal{X}}_2$ can be transformed to $\hat{\mathcal{Y}}$ directly by the generator. If large-scale 2D or 3D images are transformed, inference needs to be performed patch-based with a tiling strategy because of VRAM limitations. Patch-based inference leads to errors on the edges of adjacent patches, reducing the quality of the final large-scale image. *Weighted tiling* was introduced to reduce these errors (see Section 1.3.4) by using large overlapping patches during inference [118]. However, there are no quantitative studies that compare existing tiling strategies. Furthermore, while *weighted tiling* is a sound method, it cannot eliminate all tiling-related errors. This results in the need for a quantitative comparison of existing methods and a new advanced method that can further reduce tiling-related errors.

A theoretical discussion that explains when and why errors occur at the edges of adjacent patches is given in Section 4.2. In Section 4.3, a new benchmark dataset is introduced to quantify the quality of different tiling strategies. Afterwards, a new method is introduced in Section 4.4. Finally, the results are presented in Section 4.5 and a discussion is given in Section 4.6[1].

[1]The following sections extend the methods and the results of [157].

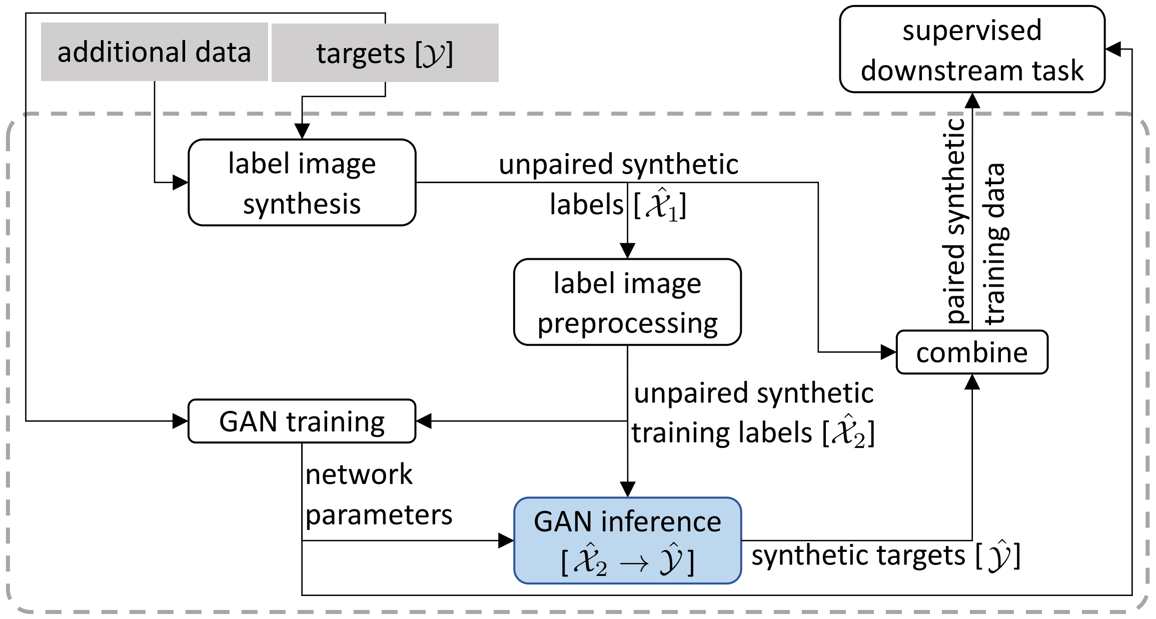

additional data
targets [$\mathcal{Y}$]
supervised downstream task
label image synthesis
unpaired synthetic labels [$\hat{\mathcal{X}}_1$]
label image preprocessing
paired synthetic training data
combine
GAN training
unpaired synthetic training labels [$\hat{\mathcal{X}}_2$]
network parameters
GAN inference [$\hat{\mathcal{X}}_2 \rightarrow \hat{\mathcal{Y}}$]
synthetic targets [$\hat{\mathcal{Y}}$]

Figure 4.1 | Pipeline for the creation of paired training data for supervised downstream tasks. GAN inference is a key component during the creation of paired synthetic training data. If images are too large to be inferred as a whole, patch-based inference is needed and a tiling strategy is applied. Existing tiling strategies lead to errors at the borders of adjancent patches.

4.2 Patch-based Errors

If a GAN is used to perform patch-based inference, errors appear at the edges of adjacent patches. The errors appear if there is no one-to-one mapping between images in both domains but a one-to-many mapping or a many-to-many mapping. An example of a one-to-many mapping is shown in Fig. 4.2a, and the mapping is visualized in Fig. 4.2b. A label domain $\mathcal{X}$ is transferred to a target domain $\mathcal{Y}$. No paired data are given, and the spatial properties P_{Sp} and the shape properties P_{Sh} of both domains are the same, while the color properties P_C differ. P_{Sp} represents the number of objects (5 per image) and the spatial distribution (random). P_{Sh} is used to determine the radius of the circles (for example, 20 px). Finally, P_C describes the color of the circles, which is white for $\mathcal{X}$ and green or blue for $\mathcal{Y}$. Therefore, the translation task from $\mathcal{X}$ to $\mathcal{Y}$ consists of coloring the circles. An image x can be transferred to different images $\hat{y}_N$ because the information on which color to choose (blue or green) is missing in $\mathcal{X}$ (see Fig. 4.2c), resulting in a one-to-many mapping. If an image is processed as a whole, all possible outcomes $\hat{y}_N$ are correct translations. If the synthesized data are used for downstream segmentation tasks, it is not important which of the images is predicted by the GAN, as long as enough input images x_i are transferred to $\hat{y}_i$ to cover the distribution of $\mathcal{Y}$. If the distribution of $\mathcal{Y}$ is covered,

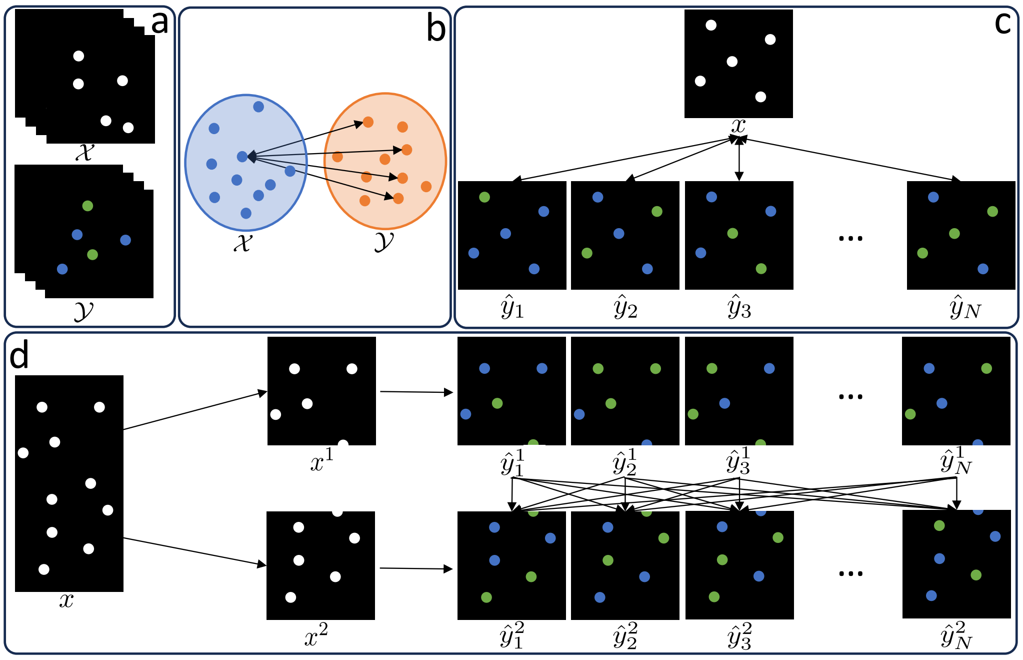


Figure 4.2 | Theoretical analysis of errors arising during patch-based inference. An example task coloring the white circles in the label domain $\mathcal{X}$ in blue or green in the target domain $\mathcal{Y}$ is shown in (a). (b) shows the one-to-many translation present, if one domain lacks information. In this case, the color property differs and color information is missing in domain $\mathcal{X}$. Possible correct translations are shown in (c). Finally (d) shows, how individual correct predictions can lead to an incorrect overall prediction. The circle present in both patches needs to have the same color in both predictions. This is e.g. the case if $\hat{y}_1^1$ and $\hat{y}_1^2$ are matched, but this is not the case if $\hat{y}_1^1$ and $\hat{y}_3^2$ are matched.

a downstream segmentation neural network (NN) can learn the task independently of the individual images $\hat{y}$ present in the training data. Therefore, unimodal GANs are used, which reduce the problem to a one-to-one mapping during training. Different versions of $\hat{y}$ for the same input x can only be acquired by retraining with modified hyperparameters or network initialization.

A one-to-many task exists if the probability distribution for at least one of the properties does not match in both domains. For the above example, the marginal probability for the property P_C and the color white is one in domain $\mathcal{X}$, while the marginal probability is 0.5 for each blue and green. If the domain $\mathcal{X}$ consists of red and orange circles, each with a marginal probability of 0.5, the GAN can learn to transform red to blue and orange to green, or orange to blue and red to green resulting in a one-to-one mapping. However, it would also be a one-to-many mapping if the marginal distribution would be 0.3 for red and 0.7 for orange. The GAN inevitably has to change some orange circles to green and some to blue to match the marginal distribution in $\mathcal{Y}$.

Errors appear for one-to-many or many-to-many mappings between the image domains, if an image is processed patch-wise (see Fig. 4.2d). The image is partitioned into non-overlapping tiles. Each of the tiles is predicted separately, without information on the adjacent patches. In this work, this method is called *simple tiling*. There is a wide variety of possible target images $\hat{y}_N^1$ and $\hat{y}_N^2$ for x^1 and x^2 in Fig. 4.2d. Each combination of the synthetic target images is possible. The part of the circle at the bottom of x^1 can be predicted in blue or green. The same is true for the part of the circle at the top of x^2. This results in an error, for example, if $\hat{y}_1^1$ and $\hat{y}_3^2$ are combined. In this case, the circle present in both tiles has two colors and is not part of the target domain $\mathcal{Y}$. For this simple translation problem, errors can be fixed by adding more domain knowledge to $\mathcal{X}$ to reduce the problem to a one-to-one mapping. For example, 50% of the circles can be colored gray instead of white in domain $\mathcal{X}$. The GAN is then able to transfer the white circles to green and the gray circles to blue or vice versa. This is not possible for real-world data, because real-world data are more complex, while the underlying distributions for the properties (e.g.: what is the percentage of blue objects) are unknown, and therefore it is difficult to synthesize label images with matching properties.

A post-processing step or multimodal GANs can be used instead of more domain knowledge. A multimodal GAN can synthesize new versions of $\hat{y}^2$ until the image matches $\hat{y}^1$. Although this is possible for simple benchmark datasets, this is not feasible for real-world data because there are no automated measures of whether adjacent patches match or not. Using a post-processing step is also not possible for real-world data because real-world data are typically too complex to apply programming methods to fix the errors. Also the *weighted tiling* strategy (see Section 1.3.4) is not capable of removing all errors that occur during patch-based inference. Therefore, an improved method for patch-based unpaired image-to-image translation, as well as a benchmark dataset to compare the different approaches, are needed.

4.3 Tiling Strategy Benchmark Dataset

A benchmark dataset is needed to quantitatively compare different tiling strategies. Real-world datasets cannot be used for this purpose because of the following reasons:

1. Automated error quantification is not possible due to the complex properties of objects present in real-world data.

2. A large number of objects have to be examined for a large number of networks for each tiling strategy, to obtain accurate results

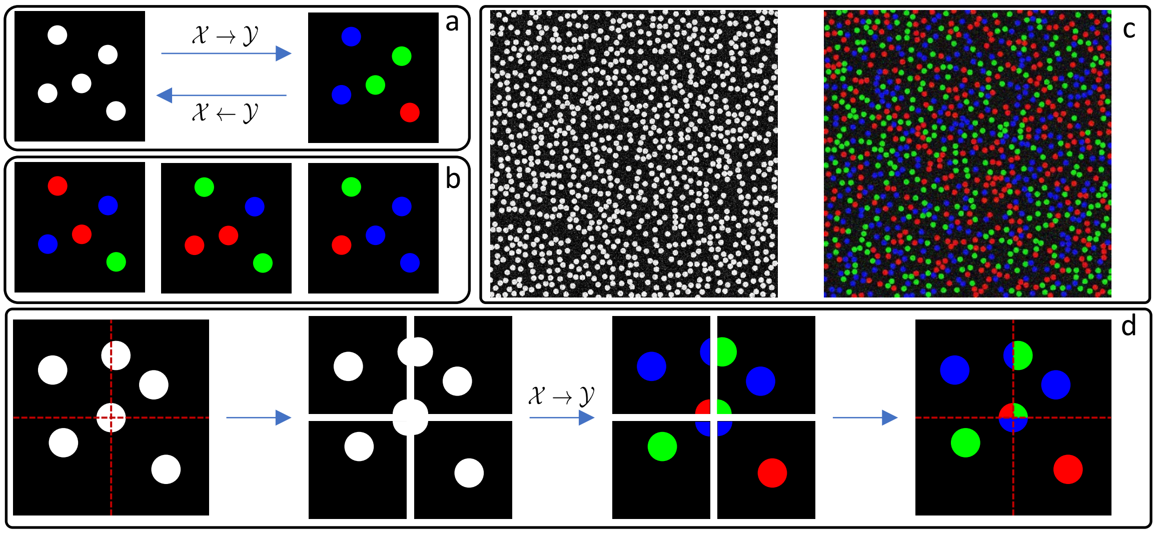

a
$\mathcal{X} \to \mathcal{Y}$
$\mathcal{X} \leftarrow \mathcal{Y}$
b
c
d
$\mathcal{X} \to \mathcal{Y}$

Figure 4.3 | The *tiling strategy benchmark dataset* dataset consists of a domain $\mathcal{X}$ with white circles and a domain $\mathcal{Y}$ with colored circles, which are green, red and blue (a). The different colors are evenly distributed. The one-to-many translation task is visualized in (b). Every image in (b) is a reasonable translation for the image on the top left in (a). An example for the images in the dataset is shown in (c). 512 images are created for each domain, and there is no pairing between both domains. The images are sized 2048 px × 2048 px and 1000 circles with a diameter of 40 px are placed in each image. An example error using the *simple tiling* strategy is shown in (d). Although the individual patches are inferred correctly, the circle in the middle of the final image consists of three colors and is therefore erroneous. The quantity of these errors is used to evaluate tiling strategies. Figure 2 from [153] licensed under CC BY 4.0.

3. It can be challenging to manually detect small local errors in large images.

4. Therefore, manual evaluation is also not feasible due to the time required.

The fully synthetic *tiling strategy benchmark dataset* is introduced to allow easy visual assessment and automated quantification of the tiling related errors.

A coloring task, shown in Fig. 4.3a is used. Domain $\mathcal{X}$ consists of white circles and domain $\mathcal{Y}$ consists of red, blue, and green circles. The shape properties are the same in both domains, with a fixed radius of 40 px for each circle. Also the spatial properties are the same with 1000 circles placed randomly in each 2048 px × 2048 px image. The total number of images created for each domain is 512. The transformation $\mathcal{X} \to \mathcal{Y}$ is a one-to-many mapping (see Fig. 4.3b), because no color information is present in $\mathcal{X}$. The color of each circle in domain $\mathcal{Y}$ is chosen randomly and the different colors are selected with the same probability. An exemplary translation with *simple tiling* and a total of four patches is shown in Fig. 4.3d. The circle present in all four patches is inferred to be of different colors, and therefore the final prediction yields one error.

Noise has been used in the past to ease the learning task [107]. It is added because convolutional neural networks (CNNs) are deterministic functions, and a low overall entropy in the input image can lead to repetitive patterns in translated images. Furthermore, a low overall entropy can hinder the cycle-consistency for one-to-many translation tasks. For our benchmark dataset, the noise enables the GAN to use it in $\mathcal{X}$ to encode the color information of $\mathcal{Y}$ for the cycle $\mathcal{Y} \rightarrow \mathcal{X} \rightarrow \mathcal{Y}$ while synthesizing images that do not differ from domain $\mathcal{X}$. Therefore, Gaussian noise is added to the images in both domains to increase the variety in the domains.

Images in the domain $\mathcal{X}$ are converted to the RGB color space to match the dimensions of the images in domain $\mathcal{Y}$. Matching input dimensions allows the usage of the identity loss, which improves stability during training. The images are encoded with 8 bits for each color channel. An exemplary image for each domain is shown in Fig. 4.3c.

The simple RGB output domain facilitates computational quantification and visual analysis of the synthesized images. If a circle consists exclusively of one of the colors red, blue, or green, the prediction is correct. If more than one color is present, the circle is not predicted correctly. However, simple thresholding cannot be used to assess whether a circle is translated correctly. Parts of a circle can change color by chance due to noise. To avoid identifying these circles as erroneous, a circle is deemed erroneous if there is a connected area of more than 30 px^2 (7.7% of the circle area) with color values above the threshold of 60 in more than one color channel of the circle.

Algorithm 2 Pseudocode to calculate the percentage of erroneous circles in a synthetic target image (target) given the label. The pixels of a circle in the label image are denoted by c_i and the total number of circles is len(c). The maximum connected area in pixels present in each color channel for a circle c_i are calculated with max_connected_area(target$[c_i]$). If the $+$ operator is used for Boolean values, false is treated as zero, and true as one.

```
Require: label, target
Ensure: error_percent
  target ← target > 60
  errors ← 0
  for $c_i$ in label do
    $a_r, a_g, a_b$ ← max_connected_area(target[$c_i$])
    error ← $(a_r > 30 + a_g > 30 + a_b > 30) > 1$
    errors ← errors + error
  end for
  error_percent ← $\frac{\text{errors}}{\text{len}(c)} * 100$
```

The values are chosen heuristically and yield a good balance between the detection of real errors and the detection of errors due to noise. Pseudocode to calculate the error in percent is given in Algorithm 2.

4.4 Stitching Aware Training and Inference

The current state-of-the-art method, *weighted tiling*, is a post-processing method. The correct continuation of objects present in multiple patches is not learned during training. In this work, the training and inference procedure is modified and, for the first time, information on previous predicted patches is included in the next patch to increase the quality of the prediction. The newly developed method is called *Stitching Aware Training and Inference (SATI)*.

GAN architectures and training routines often require modifications to suit specific problem domains. Consequently, the adoption of new architectures is hindered, as researchers tend to favor the architectures they have already customized for their primary image analysis applications. Due to this, no new GAN architecture is developed, but instead *SATI* can be integrated into existing architectures and is exemplarily integrated into CycleGAN.

SATI is introduced in Section 4.4.1. For each sub-step of *SATI*, procedures must be defined and adaptations can be integrated. These adaptations must be analysed to determine whether they improve the synthesis quality of the fundamental procedure. Therefore, the adaptations *overlap sampling*, *domain encoding*, *loss ramping*, the *stitching strategy* during inference, and *pixel overlap weighting* are introduced after the introduction of the method and the usefulness in terms of synthesis quality is quantified in an ablation study.

4.4.1 Training and Inference

Patch-based inference using simple tiling is a straightforward process. This means that images are processed patch-by-patch, e.g. from left to right from top to bottom. As described in Section 4.2, adjacent patches must be compatible with each other, and the patch to predict next depends on the adjacent patches already predicted. Therefore, the GAN needs information about objects that are in patches already predicted as well as in the patch that will be predicted next. For the example in Fig. 4.2d, only the circle at the bottom right of $\hat{y}_n^1$ is relevant for translating x^2. All other circles are not relevant. The new method *SATI* is developed to provide this required information to GANs during the trans-

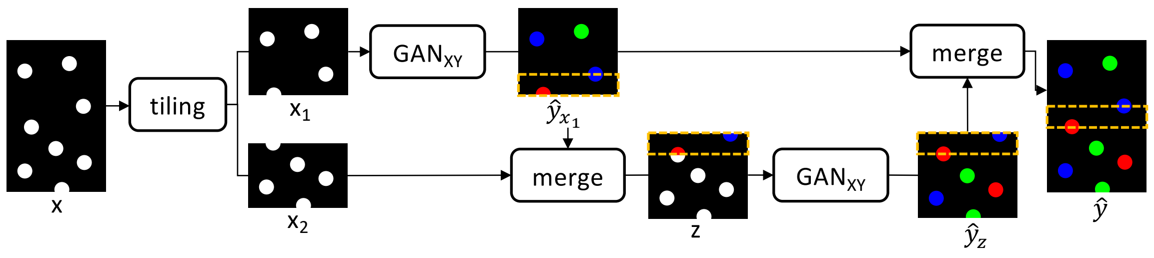


Figure 4.4 | Inference workflow used for *SATI*. An image x is tiled into patches. The first patch x_1 is processed by the generator GAN_{XY} and $\hat{y}_{x_1}$ is synthesized. x_2 is merged with the bottom part of $\hat{y}_{x_1}$ and z is created. The overlapping area between $\hat{y}_{x_1}$ and z is marked with an orange box in all subsequent steps. The same generator GAN_{XY} is used to synthesize $\hat{y}_z$ from z. The GAN can correctly continue the objects because the information from $\hat{y}_{x_1}$ is present in z. In this case, the circle at the bottom left of $\hat{y}_{x_1}$ continues in red at the top of x_2. Finally, $\hat{y}_{x_1}$ and $\hat{y}_z$ are merged to $\hat{y}$. The patch size is set by the user, whereby the patches should be as large as possible and the overlap as small as possible to reduce computation. However, the overlap must contain at least the object properties required to predict the next patch. Figure 3 from [153] licensed under CC BY 4.0; added orange rectangles.

lation between domains. Adding information of all adjacent patches to the patch, which needs to be predicted next, e.g. by stacking, vastly increases input size and is not desirable and needed. Hence *SATI* includes only the relevant information for the prediction of the next patch. Moreover, the additional VRAM required must be small. *SATI* is therefore developed in order to avoid the need for additional generators and discriminators.

The new tiling strategy and inference workflow *SATI* is depicted in Fig. 4.4. An input image x is tiled into two non-overlapping patches x_1 and x_2. For the first patch, no information about neighboring tiles exists and, therefore, no additional information about neighboring tiles is needed. After using the generator GAN_{XY} to predict $\hat{y}_{x_1}$, the bottom part of the prediction is merged to x_2. The adjacent patch information is included in the newly created image z with the merge. The new domain of images z, which consists of parts of the two domains $\mathcal{X}$ and $\mathcal{Y}$, is called $\mathcal{Z}$. The same generator can be used to translate $\mathcal{Z} \to \mathcal{Y}$ due to the similarity of the task and z is transferred to $\hat{y}_z$. In the example, the GAN can use the parts of the circle at the bottom left of $\hat{y}_{x_1}$ present in z to infer the red color of the part of the same circle at the top left of x_2.

The inference workflow introduced adds the translation $\mathcal{Z} \to \mathcal{Y}$ in addition to the standard GAN translations $\mathcal{X} \to \mathcal{Y}$ and $\mathcal{Y} \to \mathcal{X}$. The new translation has two constraints: (i) The areas of $\mathcal{Y}$ integrated into $\mathcal{Z}$ must remain constant during the translation $\mathcal{Z} \to \mathcal{Y}$, and (ii) the areas of $\mathcal{X}$ integrated into $\mathcal{Z}$ need to be transferred to $\mathcal{Y}$ with respect to the areas of $\mathcal{Y}$ present in $\mathcal{Z}$.

Both constraints must be learned during training. Therefore, the training procedure of

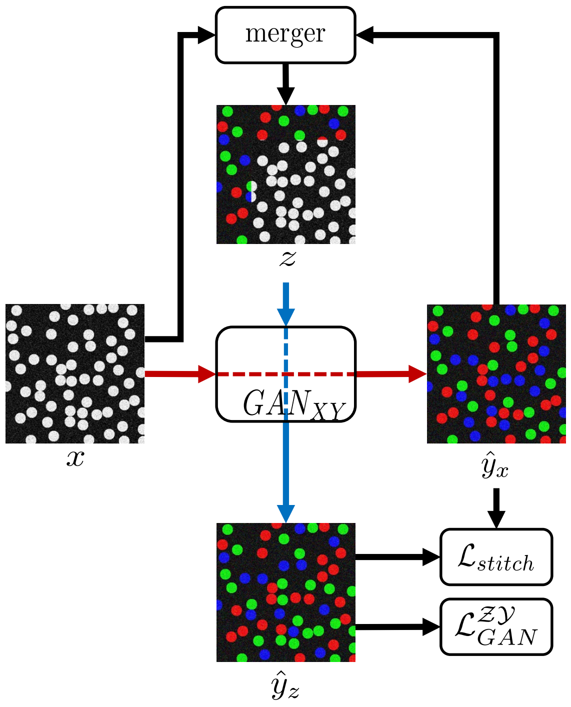


Figure 4.5 | Training steps added to a GAN if *SATI* is used. An image x is transferred to $\hat{y}_x$. The image x and $\hat{y}_x$ merge to z and z is transferred to $\hat{y}_z$. The adversarial loss $\mathcal{L}^{zy}_{GAN}$ is used to ensure the overall quality of $\hat{y}_z$. A stitching loss $\mathcal{L}_{stitch}$ is introduced to enforce the GAN to keep the areas of $\hat{y}_x$ in z constant during the transformation to $\hat{y}_z$. The blue and red lines illustrate the flow of information through the generator GAN$_{XY}$. Derived of Figure 4 from [153] licensed under CC BY 4.0; changed images and $\mathcal{L}_{adv}$ to $\mathcal{L}^{zy}_{GAN}$.

an arbitrary GAN is adapted as shown in Fig. 4.5. An image x is transferred to $\hat{y}_x$ with the generator GAN$_{XY}$. Subsequently, the border regions of $\hat{y}_x$ merge into x and z is created. In this case, the top and left border regions of $\hat{y}_x$ are used. The same generator is used to transfer z to $\hat{y}_z$ to reduce VRAM needed. The first constraint is enforced by the stitching loss $\mathcal{L}_{stitch}$. This is done by comparing the areas of $\hat{y}_x$ in z with the same areas in $\hat{y}_z$. The output of the comparison needs to be minimal if both are equal. Therefore, $\mathcal{L}_{stitch}$ can, for example, be defined with the L2-norm. The indices of all pixels from $\hat{y}_x$ in z are defined by M which leads to:

$$\mathcal{L}_{stitch} = \mathbb{E}_{z\sim p_{data}(z)}\Big[||\text{GAN}_{XY}(z)(M) - z(M)||_2\Big]. \tag{4.1}$$

The adversarial loss $\mathcal{L}^{zy}_{GAN}$ is used to satisfy the second constraint, ensuring the overall quality of the image. Most GANs already have a discriminator D_Y trained to differentiate between real images y and synthetic images $\hat{y}_x$. In addition to the images $\hat{y}_x$, D_Y is also trained with the images $\hat{y}_z$. The discriminator can then be used to train the generator

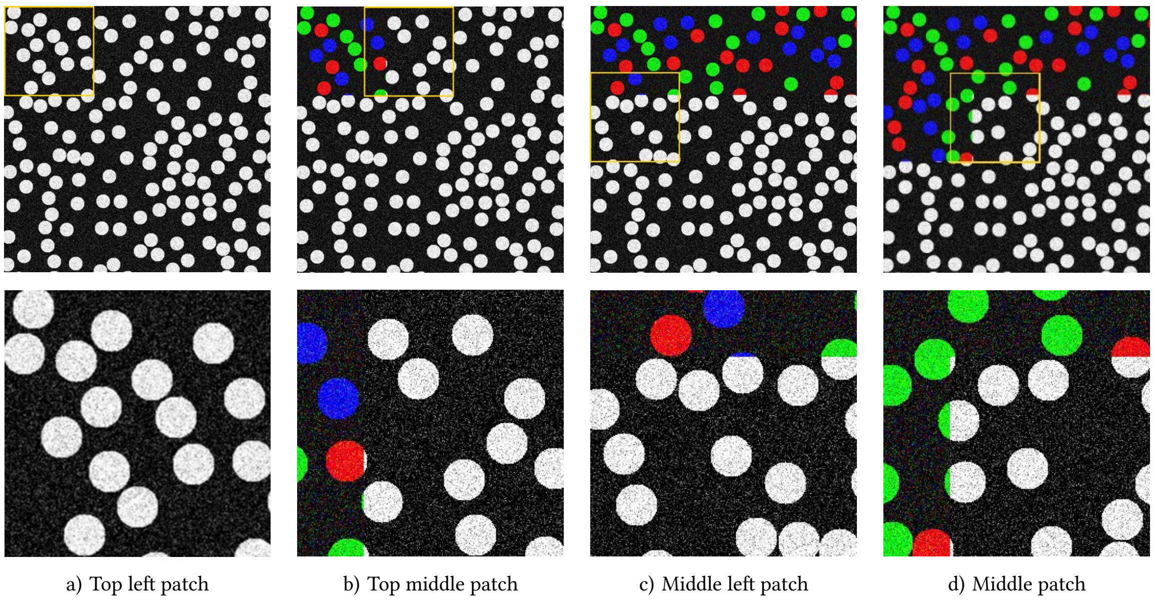


Figure 4.6 | Different patches during inference with *SATI*. The top row shows different points in time during the inference. The patch to be predicted next is marked in yellow. The bottom row shows the magnification of each yellow square. The patch in (a) is a patch learned during standard GAN training. The patches in (b) - (d) are learned if *overlap sampling* is enabled. If *overlap sampling* is disabled, only (d) is learned during the training with *SATI*.

GAN$_{XY}$ to ensure the overall quality of $\hat{y}_z$. As for images of domain $\mathcal{X}$, the adversarial loss for images $\hat{y}_z$ can be defined as:

$$\mathcal{L}_{GAN}^{\mathcal{Z}\mathcal{Y}} = \mathbb{E}_{z\sim p_{data}(z)}\Big[||[1 - D_Y(\text{GAN}_{XY}(z))]||_2\Big]. \tag{4.2}$$

The discriminator is trained to output zero for fake images and one for real images. Therefore, Eq. 4.2 is minimized for the generator and maximized for the discriminator.

4.4.2 Overlap Sampling

Several adaptations can be made to the basic training and inference workflow. The first adaptation to the training workflow is *overlap sampling*. The basic setup shown in Fig. 4.5 uses the top and left regions of $\hat{y}_x$ to create z, but not all images during inference include the top and left regions from $\hat{y}_x$. If a 2D image x is processed row by row during inference, the images from domain $\mathcal{Z}$ must be assembled differently depending on the row and column. Examples for all possible cases are shown in Fig. 4.6. The first image processed has no regions from $\hat{y}_x$. All other images in the first row contain the left border region of $\hat{y}_x$. The first image in each subsequent row includes the top border regions of $\hat{y}_x$. All other images contain the top and left border regions of $\hat{y}_x$. Initiating the process from the lower right corner leads to overlaps on both the bottom and right sides, thereby presenting a training task of equivalent complexity. Image statistics for the mean and variance of an image z can differ greatly depending on the two domains $\mathcal{X}$, $\mathcal{Y}$, and which parts of $\hat{y}_x$ are combined into z. If instance normalization without running mean and variance is used, training on one of the three cases and inferring on all of them results in a reduced quality for the cases not used for training. To ensure high quality for all cases, the merger samples randomly from the cases during training.

4.4.3 Domain Encoding

If *SATI* is used, the GAN has to learn which areas in an image z are from $\hat{y}_x$ and must not be changed and which areas are from x and must be changed. It seems to be an easy task that can be learned without further adjustments. But CNNs work locally and the receptive field is limited in the early layers. Therefore, a CNN has to infer the information if a pixel needs to be changed or not solely from the pixel and its surrounding pixels without additional positional information. This can be a challenging task, especially if the domains $\mathcal{X}$ and $\mathcal{Y}$ are similar, which is often the case for background areas. This is also the case for the *tiling strategy benchmark dataset* where areas without circles are the same in both domains. A straightforward way to add information about whether or not a pixel needs to be transformed is to add an additional layer encoding the domain of origin for each pixel. However, this increases the input size. Instead, the input domain is directly encoded in the input image. The pixels in domain $\mathcal{X}$ are normalized to the range $[-1, 0]$ and the pixels in domain $\mathcal{Y}$ are normalized to the range $[0, 1]$. The GAN can use these ranges to identify the domain and, therefore, if it needs to transform the pixel or not. Whether *domain encoding* eases the learning task or not is examined in the ablation study.

4.4.4 Loss Ramping

Learning the unsupervised image-to-image translation between two domains is a challenging task. Thus, the synthesis quality is low for the first epochs. It may hinder the GAN training to enforce that the low-quality parts of $\hat{y}_x$ in z remain constant during translation to $\hat{y}_z$. Therefore, the stitching loss $\mathcal{L}_{stitch}$ is scaled from zero to the final scaling factor λ_{stitch} throughout training.

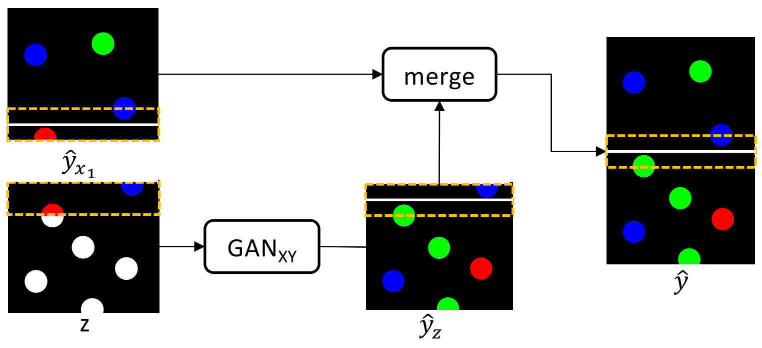


Figure 4.7 | Non-critical error, because the stitching strategy is applied. An image z is transferred to $\hat{y}_z$. The partially red circle at the border of the area from $\hat{y}_{x_1}$ in z (orange rectangle) is transferred to green. Without the stitching strategy, this would result in a red and green circle in $\hat{y}$. With the stitching strategy, the circle is completely green because the middle of the overlapping area is used to merge patches (white line).

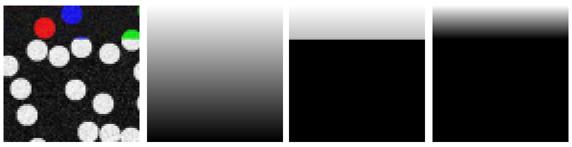
4.4.5 Stitching Strategy

As shown in Fig. 4.4, the overlapping parts of the patches are processed twice by the GAN during inference. Intuitively, the overlapping part of the first image $\hat{y}_{x_1}$ would be used when merging $\hat{y}_{x_1}$ and $\hat{y}_z$. Using the overlapping parts of the first or second image resulted in errors for objects just starting or ending at the borders of the overlapping regions. These errors are prevented by using the first half of the overlap of the first image and the second half of the second one. An example is shown in Fig. 4.7.

Weighted tiling is not used as a stitching strategy, because it is no longer needed, since the GAN has learned how to continue image in the next patch during training.

4.4.6 Pixel Overlap Weighting

Cutting in the middle of the overlapping area using the *stitching strategy* slightly changes the training target. Pixels in the overlapping area of z that are close to the edge of the image still need to maintain their values. Pixels of domain $\mathcal{Y}$ in the overlapping area of z that are close to the areas of domain $\mathcal{X}$ can potentially change. For the example shown in Fig. 4.7, this means that pixels in z that are close to the top edge of the orange rectangle must not change, while pixels in z that are close to the bottom edge of the orange rectangle may change. If only a small part of an object is in the patch already predicted, it may be useful to change it during the translation of the next patch since more information about the object may be given in the next patch. Changing objects that only have small parts in the patch already predicted can be enabled by weighting the pixels used in $\mathcal{L}_{stitch}$ according to their location. Example weight maps and the procedure to create them are


a) Weight map for image z with top area from domain $\mathcal{Y}$

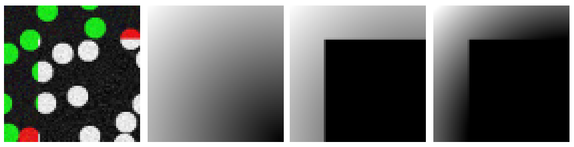

b) Weight map for image z with top and left area from domain $\mathcal{Y}$

Figure 4.8 | Weight maps if *pixel overlap weighting* is enabled. The left image in (a) shows an image with the top area from domain $\mathcal{Y}$ and the left image in (b) shows an image with the top and left area from domain $\mathcal{Y}$. The weight maps are created by calculating the Euclidean distance to the pixel with the largest distance to pixels of domain $\mathcal{Y}$ (second image in (a) and (b)). For (a), this is the bottom row. For (b), this is the pixel at the bottom right. Afterwards, all pixels of domain $\mathcal{X}$ in z in the weight map are set to zero (third image in (a) and (b)). Finally, the distance is normalized to the range $[0, 1]$ resulting in the weight maps used during training (right image in (a) and (b)).

shown in Fig. 4.8. An example weight map for an image with the top and left area from domain $\mathcal{Y}$ is shown in Fig. 4.8b. First, all pixels are weighted according to the shortest Euclidean distance to the pixel with the largest distance to pixels of domain $\mathcal{Y}$. For the example, this is the bottom right pixel. Second, all pixels of domain $\mathcal{X}$ are weighted zero and therefore can change. Finally, the distances are linearly scaled between zero and one. The same routine is performed for an image with the top area from domain $\mathcal{Y}$, which is shown in Fig. 4.8a.

4.5 Experiments and Results

SATI is integrated into CycleGAN because CycleGAN is the leading method for the synthesis of biomedical data and in materials science [21], [22], [86], [88], [89], [107]. The mean squared error (MSE) is used for all loss functions. CycleGAN uses scaling factors λ_{cycle} and λ_{idt} for the cycle-consistency loss $\mathcal{L}_{cycle}$ and the identity loss $\mathcal{L}_{idt}$. They are set according to the original implementation of CycleGAN[2] to λ_{cycle}=10 and λ_{idt}=5. Furthermore, the scaling factor for the stitching loss $\mathcal{L}_{stitch}$ is set to λ_{stitch}=10. The overall loss is calculated as follows:

$$\mathcal{L} = \mathcal{L}_{GAN} + \lambda_{cycle}\mathcal{L}_{cycle} + \lambda_{idt}\mathcal{L}_{idt} + \mathcal{L}_{GAN}^{\mathcal{Z}\mathcal{Y}} + \lambda_{stitch}\mathcal{L}_{stitch}. \tag{4.3}$$

A ResNet-generator with 96 feature maps in the first convolutional layer, instance normalization and nine ResNet blocks in the feature space is used for each generator. A

[2]The implementation https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix (accessed: 24.01.2024) is adapted for this work.

PatchGAN-discriminator with instance normalization is used for each discriminator[3].

4.5.1 Tiling Strategy Benchmark Dataset Results

SATI is compared to *simple tiling* and *weighted tiling* with the *tiling strategy benchmark dataset*. Furthermore, it is compared to the maximum achievable performance if the image is processed without tiling (*no tiling*). This is possible because the image size of 2048 px × 2048 px in the benchmark dataset is small enough to be processed as a whole during inference. In addition, an ablation study is conducted that quantifies the performance of the different adaptations made to *SATI*. All networks are trained on 256 px × 256 px crops and the initial inference crop size for all tiling strategies is 512 px × 512 px. 50 images not present in the training data are used for evaluation, which results in 50.000 objects on the images. A NVIDIA GeForce RTX 3090 GPU is used for the experiments. Training a standard CycleGAN takes approximately 24h on the *tiling strategy benchmark dataset*, while training with *SATI* takes around 27:30h. 10 networks per tiling strategy are trained to limit computational expense. Because *simple tiling* and *weighted tiling* are post-processing methods, the same 10 networks can be used for the standard CycleGAN without tiling (*Standard + no tiling*), standard CycleGAN with simple tiling (*Standard + simple tiling*), and the standard CycleGAN with weighted tiling (*Standard + weighted tiling*). All adaptations to *SATI* except the *stitching strategy* are training adaptations. Therefore, 50 networks must be trained for *SATI* resulting in a total of 60 trained networks and approximately 67 days of training.

Benchmark Method Comparison

SATI is compared to the benchmark methods in the first four rows of Fig. 4.9. Training GANs is a challenging task. This is especially the case for unsupervised models. Training can be unstable, convergence is not guaranteed, and mode collapse can occur [84]. This also happened for some of the runs shown in Fig. 4.9, where the error is much higher than for the other models. To reduce the influence of collapsed runs on the metric, the median scores are used instead of the mean scores. While one of the runs training a standard CycleGAN collapsed, none of the *SATI* runs with all adaptations collapsed, but the number of networks trained is not large enough to draw conclusions regarding the training stability.

[3]An implementation for conducting the experiments is available at https://github.com/MoritzBoe/patch_based_image_translation.git, accessed 24.01.2024.

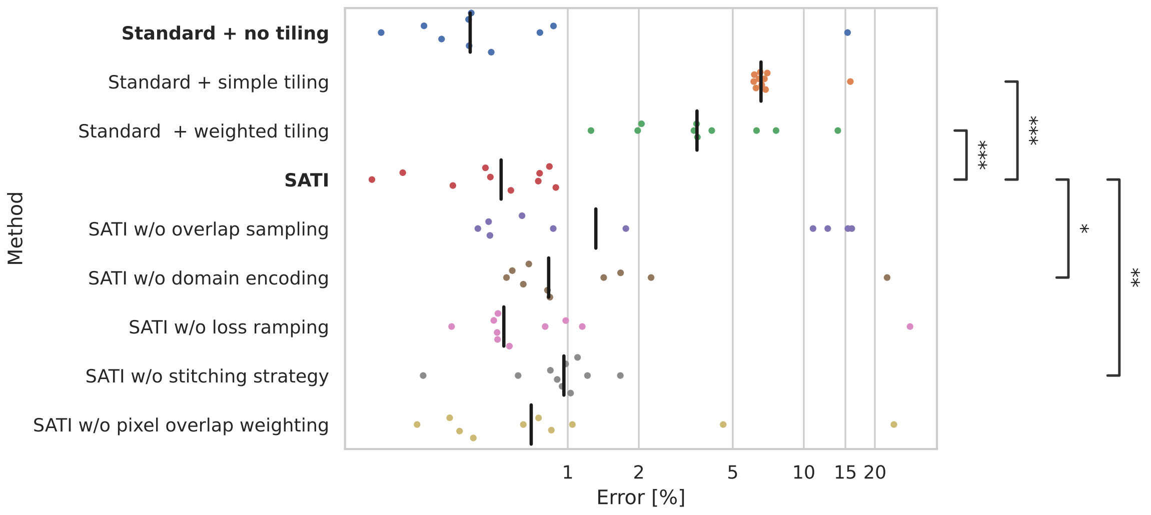


Figure 4.9 | Results for *SATI* benchmarked against *Standard + no tiling*, *Standard + simple tiling*, and *Standard + weighted tiling*. An ablation study shows the change regarding the error if one of the adaptations made to *SATI* are deactivated (*SATI w/o [adaptation]*). The error rate is displayed in percent (see Algorithm 2) on a logarithmic scale with the median for each method indicated by a black line. The best case scenario is *Standard + no tiling*, where the whole image is transferred without the need for tiling. This is only possible for small 2D and 3D images. The significance levels of a Mann-Whitney U test are calculated for *SATI* against the other methods and *SATI* without each of the single adaptions. The significance levels are: *: $p \leq 5 \cdot 10^{-2}$, **: $p \leq 1 \cdot 10^{-2}$, ***: $p \leq 1 \cdot 10^{-3}$. Figure 7 from [157] licensed under CC BY 4.0; Changed method order and added Mann-Whitney U test.

The results for *Standard + simple tiling*, *Standard + weighted tiling* and *Standard + no tiling* only differ in the inference strategy. The results for *no tiling*, which correspond to the minimum achievable error rate, show that *no tiling* is superior to all tiling strategies, with a median error of 0.4%. If *Standard + simple tiling* is used, the median error increases to 6.6%. Using *Standard + weighted tiling* reduces the median error to 3.5%. This is still a large increase compared to *Standard + no tiling*, but also a large decrease compared to *Standard + simple tiling*, while the training is not changed. If *SATI* is integrated into training and inference, the median error is reduced to 0.5%, yielding a reduction of 92% compared to *Standard + simple tiling* and a reduction of 85% compared to *Standard + weighted tiling*. A Mann-Whitney U test comparing *SATI* to *Standard + weighted tiling* results in a p-value of $1.8 \cdot 10^{-4}$. The best *SATI* model results in an error of 0.15%, while the best *Standard + no tiling* model has an error of 0.16%. It can be concluded that it is possible to achieve performance comparable to *Standard + no tiling* with *SATI* and no significant difference for the error rate between *SATI* and *Standard + no tiling* was found.

Exemplary errors using the different tiling strategies are shown in Fig. 4.10. The errors at the border of the patches are clearly visible for *Standard + simple tiling* in Fig. 4.10a.

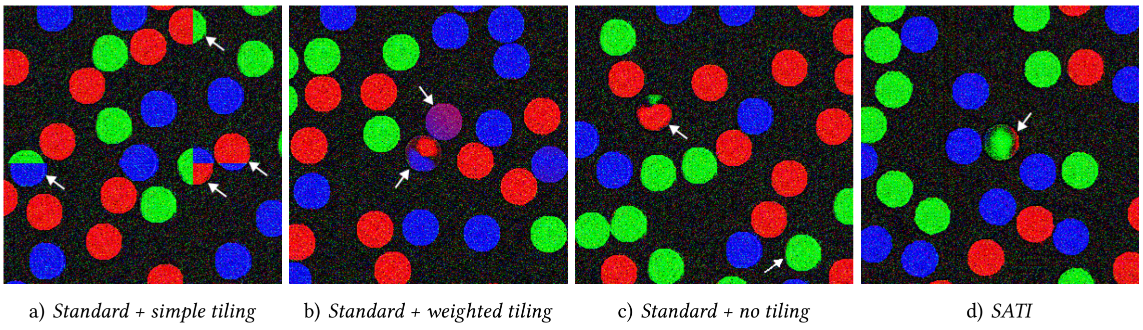

a) *Standard + simple tiling*
b) *Standard + weighted tiling*
c) *Standard + no tiling*
d) *SATI*

Figure 4.10 | Exemplary errors for different inference strategies (a-c) and *SATI* (d). Errors marked with a white arrow in (a) exist because of the *simple tiling* strategy. The purple circle in (b) (top white arrow) originates from the weighted overlap. A red and a blue circle are merged to purple. The arrow to the lower left points on an error that is comparable to the errors in (c) and (d) and is not related to tiling. These errors exist due to a faulty translation $\mathcal{X} \rightarrow \mathcal{Y}$ and can possibly be reduced with GANs better adapted to the translation task. Because no tiling related errors are present for *Standard + no tiling* and *SATI*, the remaining non tiling related errors are shown in (c) and (d). Figure 8 from [157] licensed under CC BY 4.0; Added arrow in (c).

The errors for *Standard + weighted tiling* are visually less prominent. This is due to the weighted overlap. Instead of a sharp transition between colors, the circles yield a mixture of colors. An example is the purple circle in the middle of Fig. 4.10b. The quantitative results show that there are still errors for *Standard + no tiling*. An example of the remaining errors is shown in Fig. 4.10c. These errors are not related to tiling and can be reduced by adapting GAN parameters, more training data, longer training, or using a better suited GAN architecture for the translation task. The same is true for *SATI*. The errors shown in Fig. 4.10d look very similar to the errors if no tiling is applied. An example of these errors not related to tiling is also shown in Fig. 4.10b, where a circle is partially blue and red.

All images in Fig. 4.10 include circles with distorted edges that have no influence on the error criteria defined in Section 4.3. These distortions are not related to tiling. The spatial and shape properties are the same in both domains and tiling does not change these properties. The distortions are also present if no tiling is applied. This can be seen by looking at the green circle on the bottom right for *Standard + no tiling* in (c). The distortions arise due to imperfect translations from $\mathcal{X}$ to $\mathcal{Y}$. The distortions of the shape properties can be reduced by adding a spatial constraint to CycleGAN [106].

The results can be used to qualitatively compare *SATI* to the benchmark methods according to the criteria implementation effort, inference VRAM, training computation, inference computation, and inference quality (see Table 4.1). The implementation effort of *SATI* is the highest, because GAN training needs to be adapted. The VRAM required during Inference is the highest for *Standard + no tiling*. The patch size can be adapted to

Table 4.1 | Qualitative comparison of patch-based inference methods. *SATI* has to be used if the inference quality needs to be high and VRAM is limited.

Method	Implementation effort	Inference VRAM	Training computation	Inference computation	Inference quality
Standard + no tiling	++	−−	++	++	++
Standard + simple tiling	++	++	++	++	−−
Standard + weighted tiling	+	++	++	−	+
SATI	0	++	+	+	++

the VRAM available for all other methods. The training computation of *SATI* exceeds the computation of standard GAN training, while *Standard + weighted tiling* needs more computation during inference because many patches need to be predicted due to the large overlap. *Standard + no tiling* and *SATI* achieve the lowest error rates during inference, while *Standard + weighted tiling* performs better than *Standard + simple tiling*. Therefore, *SATI* should be used if the inference quality of *Standard + weighted tiling* is not sufficient or less inference computation is needed.

Ablation Study

Several adaptations are made to *SATI* and the results of the ablation study are shown in the bottom rows of Table 4.2 which summarize the results of Fig. 4.9.

The ablation study examines whether an adaptation leads to better performance, while *overlap sampling* has the greatest impact on performance. If *overlap sampling* is disabled, the GAN is trained only with images z where the top and left areas are from domain Y. The median error increases by 151.9%. Looking at the final images shows great deficits for images where only the top or left are from domain Y. An example is shown in Fig. 4.11. The inference is performed from left to right, from top to bottom. The GAN is able to infer a high-quality patch for the first patch with regions from domain $\mathcal{Y}$ at the top and the left (see yellow square). The ability to generate a good prediction despite a previous bad prediction is an important part of the method. The GAN can recover if a previous prediction yields poor quality, which is a key requirement to process arbitrary large images. Although the Mann-Whitney U test did not show statistical significance, *overlap sampling*

Table 4.2 | Results of the tiling strategies *Standard + [post-processing]* and *SATI* as well as the ablation study (*SATI w/o [adaption]*) on the *tiling strategy benchmark dataset*. *Standard + no tiling* (best case), and *SATI* are highlighted. The values correspond to the error in percent (see Algorithm 2). The low median values and the high mean values occur due to collapsed GAN runs which yield a high amount of erroneous objects.

Method	mean	median	std	best
Standard + no tiling	**1.93**	**0.39**	**4.71**	**0.16**
Standard + simple tiling	7.46	6.59	2.92	6.13
Standard + weighted tiling	4.77	3.53	3.76	1.25
SATI	**0.54**	**0.52**	**0.27**	**0.15**
SATI w/o overlap sampling	5.95	1.32	6.84	0.42
SATI w/o domain encoding	3.20	0.83	6.81	0.55
SATI w/o loss ramping	3.40	0.54	8.71	0.32
SATI w/o stitching strategy	0.95	0.96	0.37	0.24
SATI w/o pixel overlap weighting	3.32	0.70	7.40	0.23

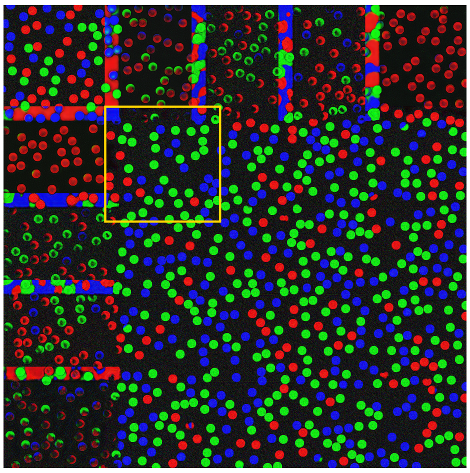


Figure 4.11 | Errors if *overlap sampling* is disabled for *SATI*. The image is created from 25 individual patches. All predictions not learned during training are erroneous (top and left row). Only the predictions where top and left are from domain $\mathcal{Y}$ yield good quality. The first patch with areas from domain $\mathcal{Y}$ at the left and top is marked with a yellow square. All patches to the right and the bottom of the yellow square also yield areas from domain $\mathcal{Y}$ at the left and top. Figure 9 from [157] licensed under CC BY 4.0; Added a yellow square.

should not be disabled, because four runs yielded error rates of more than 10% if *overlap sampling* is disabled, while none of the runs of *SATI* with overlap sampling had that high error rates. Furthermore, enabling *overlap sampling* has no drawbacks.

The *domain encoding* is integrated to facilitate the learning task by providing information on whether a pixel needs to be transformed or not. Disabling *domain encoding* reduces the performance by 59.0%. It can be concluded that the encoding enables the

GAN to differentiate between pixels from domain $\mathcal{X}$ and $\mathcal{Y}$ in the input images of domain $\mathcal{Z}$. The Mann-Whitney U test comparing *SATI* to *SATI* without *domain encoding* resulted in a p-value of $2.6 \cdot 10^{-2}$.

Disabling *loss ramping* yields the lowest reduction, increasing the median error by 2.7%. However, one of the ten runs collapsed, which did not happen to *SATI* with *loss ramping*.

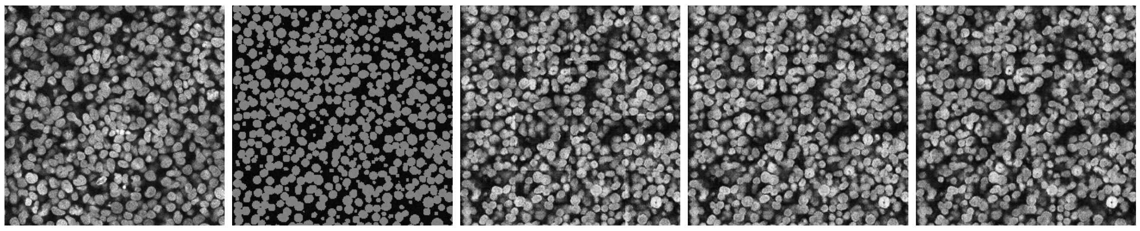
Removing the *stitching strategy* increases the error by 84.5%. The GAN has problems with objects at the border of images and at the transition between the domains $\mathcal{X}$ and $\mathcal{Y}$ in images of domain $\mathcal{Z}$. The errors are reduced by using the *stitching strategy*, while no adaptations to training are needed. The Mann-Whitney U test comparing *SATI* to *SATI* without the *stitching strategy* resulted in a p-value of $5.6 \cdot 10^{-3}$.

If the *pixel overlap weighting* is disabled, the error increases by 34.0%. The reduced performance proves that it is necessary to allow the GAN to change pixels in the boundary area between the domains. Inferring the majority of an object from a small area is a complex task for the GAN and *pixel overlap weighting*, and therefore the ability to change the small area instead of the large area increases performance. Although the Mann-Whitney U test does not show statistical significance, there is no need to disable it. The additional computational expense is negligible and deactivating *pixel overlap weighting* results in two runs with severely higher error rates.

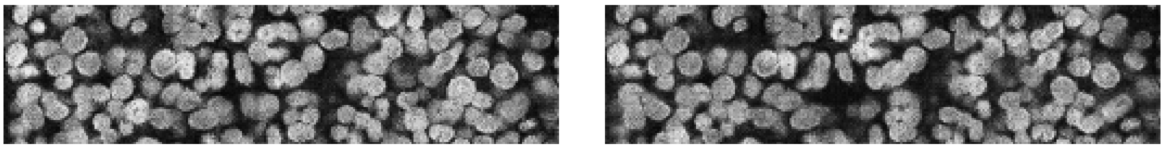
4.5.2 Real-world Dataset Results

The results on the *tiling strategy benchmark dataset* show, that *SATI* is able to reduce errors introduced by patch-based inference. In this section, *SATI* is used on a real-world 3D dataset to prove that (1) *SATI* can be expanded to 3D and (2) *SATI* can be applied to complex real-world data. A real-world dataset of KP-4 cells with Draq5 stained nuclei is used. The images are recorded with a Leica SP8 confocal microscope (Leica Microsystems, Wetzlar, Germany)[4]. A total of four images with a voxel size of 568 nm×568 nm×1000 nm are recorded. The images are grayscale images with a resolution of 8 bit. All images are cropped to remove areas without cell nuclei. The crop size ranges from 380 px to 550 px in the XY-plane and 140 px to 190 px in the Z-direction. The crops are downscaled by the factor of 2 in the XY-plane to match the Z-resolution, resulting in a final range of 190 px to 275 px in the XY-plane. Furthermore, downscaling increases the number of objects in a single volume during training, easing the learning task. A crop of an XY-slice is shown

[4]Thanks to Mario Vitacolonna from the Institute of Molecular and Cell Biology, Mannheim University of Applied Sciences, Germany for providing the microscopy images.



a) Real-world crop b) Synthetic label image c) *Std. + simple tiling* d) *Std. + weighted tiling* e) *SATI*



f) Magnification for *Std. + weighted tiling* g) Magnification for *SATI*

Figure 4.12 | XY-slices of 3D volumes cropped to 256 px × 256 px. A real-world crop is shown in (a). A synthetic label image is shown in (b). The crops of images synthesized for the 3D volume of the crop shown in (b) are shown in (c-e). The same trained network is used for the crops in (c - e). Although differences between patches are visible in (c), they are not visible in (d) and (e). The crop of an image created with *SATI* consists of 25 individual patches. Magnifications of (d) and (e) are shown in (f) and (g) by extracting an area of 256 px × 64 px from the crops. Figure 10 from [157] licensed under CC BY 4.0; Added magnifications.

in Fig. 4.12a. And a XY-crop of a synthetic label image used to train the GAN is shown Fig. 4.12b. The procedure for creating the synthetic label images is described in Section A.2.

The CylceGAN is trained for 1120 epochs with 256 random crops per epoch. The volume of a crop is 64 px × 64 px × 64 px and a batch size of 12 is used. The overlap is set to 16 px and the stitching loss is scaled with λ_{stitch}=20. All other scaling factors are the same as for the GANs trained on the *tiling strategy benchmark dataset*. Training took 31 hours on an NVIDIA A100 GPU. The patch size for inference is set to 64 px × 64 px × 64 px with an overlap of 16 px.

A XY slice of the inferred image after training is shown in Fig. 4.12e. The crop consists of 25 individual patches and there is no sharp transition between patches. In contrast, an image inferred with the *simple tiling* strategy (Fig. 4.12c) shows clearly visible differences between the single patches. The same trained network was used for inference with *Standard + simple tiling* and *Standard + weighted tiling* (Fig. 4.12d) to omit quality differences due to training. The visual results for *Standard + weighted tiling* are comparable to (e). The results show that *SATI* can be expanded to 3D and the GAN can learn the complex mapping between two domains for real-world data. However, it should be noted that there is still a domain gap between the synthetic images in Fig. 4.12c, Fig. 4.12d, and Fig. 4.12e and the real-world images. The domain gap can be attributed to the spatial differences in real-world images. The brightness at the edges of XY-slices and for lower Z-slices is reduced. The reduced brightness is characteristic for confocal microscopy. Standard GANs cannot learn these differences because spatial information is not given during training and inference. It is possible to add spatial information to reduce the domain gap, while still using *SATI* [89], [105].

The large overlap of *Standard + weighted tiling* increases the inference time by the factor of 4 for each dimension compared to *Standard + simple tiling*. *SATI* does not need a big overlap, because the transfer between patches is learned explicitly. The overlap depends only on the spatial extent of the information needed for the prediction of the next patch. In the experiments, the inference time is increased by a factor of 1.25 for each dimension. Therefore, *SATI* increases inference time by 1.95 for 3D images, while *Standard + weighted tiling* increases the inference time by 64. Memory usage is the same for both methods because only the generator is needed for inference.

The application to 3D data has shown that the processing routines for *SATI* must be adapted in addition to the neural networks of CycleGAN. However, the adaptations required for *SATI* are available in the published source code. Therefore, there is no difference in the complexity of programming for future projects whether *SATI* is applied to 2D or 3D data. In addition, no further adjustments were necessary for the real-world dataset used. Only the scaling factor λ_{stitch} was increased to 20 to further improve the transition between patches.

4.6 Discussion

The state-of-the-art method *Standard + weighted tiling* reduces the error by 46% and errors are visually less prominent. Nevertheless, *Standard + weighted tiling* is a post-processing method applied during inference and it is not possible to prevent all errors. In contrast, *SATI* changes training and post-processing and is therefore able to reduce the error by another 85% compared to *Standard + weighted tiling*. The high-quality results of *SATI* enable the synthesis of 3D microscopy images or large-scale 2D data such as whole slide images or aerial hyperspectral images.

A GAN trained with *SATI*, learns the transition between adjacent patches. Therefore, the learning task is more complex compared to a standard GAN. The additional complexity is reduced with the added *domain encoding*. The results on the real-world dataset show that the GAN is able to learn domain transfer for complex training tasks. As for standard GANs, the network size can be increased for more complex tasks.

Integration of *SATI* increases memory usage during training. The VRAM utilization increases from 19.9 GB to 22.8 GB for the CycleGAN trained on the *tiling strategy benchmark dataset*. If VRAM is limited, the batch size or the crop size need to be decreased.

Training for a fixed number of epochs increases the training time from approximately 24 to 27:30 hours on the *tiling strategy benchmark dataset*. It cannot be analyzed whether convergence with *SATI* takes longer than without *SATI*, because it is still an open question how to stop GAN training. Currently, GANs are trained for a fixed number of epochs.

The relevant information for the next patch must be present in the overlapping area to allow *SATI* to work properly. As a result, *SATI* is not applicable for datasets where the object properties relevant for the prediction of an object in a patch are not present in the overlapping areas of adjacent patches, but in patches far away. However, the spatial extend of objects in many biological, medical, or material science datasets is small.

Finally, *SATI* is designed to be integrated into different GAN architectures by adding the adversarial loss $\mathcal{L}_{GAN}^{\mathcal{Z}\mathcal{Y}}$ and the stitching loss $\mathcal{L}_{stitch}$. This ensures a fast and easy integration into existing and new projects, and scientists can stick to their preferred GAN architecture. A possible workflow to integrate *SATI* into new projects is as follows: (i) Adapt the preferred GAN architecture to the new problem. (ii) Evaluate whether the GAN is able to learn the task. (iii) Add *SATI* to remove errors at patch borders existing due to patch-based inference.

For the first time, a benchmark dataset for a quantitative comparison of patch-based tiling strategies is introduced. The results on the benchmark dataset show that new methods need to be developed to further increase image quality if patch-based inference is used. The shortcomings of existing methods are eliminated with *SATI*. In contrast to existing methods, *SATI* learns the correct prediction of the next patch during GAN training and is:

- integrable into different GANs to enable easy integration into new and existing projects which utilize GAN architectures and workflows manually adapted to a distinct task.
- computationally lightweight and requires low additional VRAM and computation.
- able to recover from previously bad predicted patches during inference. Being able to recover allows for the prediction of arbitrary large images with an unlimited number of individual patches.
- applicable for 2D and 3D data.

With these features, *SATI* can be used to improve the quality of large-scale image synthesis. Future research fields for patch-based inference include the influence of different tiling strategies on background pixels. For example, a pixel in a 3D image synthesized with *Standard + weighted tiling* is created by weighing up to eight individual patches. This can possibly change the learned background noise. Furthermore, *SATI* needs to be incorporated into different GAN architectures and can be expanded towards 3D+time data.

Conclusion and Outlook 5

Supervised neural networks used for segmentation require labeled training data. Manual annotation is time-consuming, expensive, and error-prone. GANs that perform unsupervised image-to-image translation with a set of synthetic label images can be used instead. However, the large number of challenges prevents the broad application of the methodology. Consequently, this thesis deals with the identification and elimination of these challenges and the use of GANs that perform unsupervised image-to-image translation in segmentation workflows to replace the manual annotation of data.

GANs trained with synthetic label images, in which the object properties do not correspond to the object properties in the target images, generate poor results. However, the influence of the object properties is unclear. Therefore, the synthesis of label images is examined in Chapter 2. The influence of synthetic label image properties on a downstream segmentation task is theoretically assessed and a new pipeline for quantitative evaluation is introduced. Two new benchmark datasets are developed and used to acquire quantitative insight using the proposed pipeline. In addition, GANs are currently trained without the use of validation metrics. This leads to more training and suboptimal inference results. For this reason, Chapter 3 presents a new procedure that allows the evaluation of any unsupervised image-to-image GAN validation metric for its usefulness for the synthesis of segmentation data. Two real-world benchmark datasets are used to compare metrics for GAN validation during training and to collect quantitative insight. Chapter 4 focuses on the inference of GANs performing unpaired image-to-image translation. A theoretical study of the emergence of errors in patch-based inference is used to develop a new benchmark dataset for automated error quantification. A new method to perform patch-based inference called *SATI* is introduced and compared to existing methods on the benchmark dataset. In addition, an ablation study for *SATI* is carried out.

The major contributions of this work can be described as follows:

- **A new label image property influence workflow:** For the first time, the workflow allows to quantitatively evaluate the influence of the properties of synthetic label images on the subsequent downstream segmentation performance.
- **Two new label image property benchmark datasets:** The benchmark datasets are essential to quantitatively evaluate the influence of the synthetic label images on the subsequent segmentation. The Contour Complexity Benchmark Dataset (CoCoBeDa) is used to evaluate the influence of the complexity of the object contours in the synthetic label images. The Shape And Spatial Property Benchmark Dataset (SSPBeDa) is used to evaluate the influence of spatial properties and shape properties in the synthetic label images.
- **Recommendations for the creation of datasets with roundish objects:** The results on the benchmark datasets show that a low contour complexity is enough to achieve high-quality segmentation results for roundish objects present in many biological or biomedical datasets. Among other findings, it is shown that a large hyperparameter study can be omitted when synthetic labels are created. Instead, the results of the downstream segmentation can be used to adapt the synthetic label images accordingly. This is still a manual process.
- **A new GAN validation metric evaluation workflow:** In the past, GANs were trained for a fixed number of epochs. There is no workflow to quantitatively examine metrics for their suitability as validation metrics during GAN training. A new workflow is developed to quantitatively evaluate validation metrics for GANs. Validation metrics are assessed based on their ability to determine when to stop GAN training or select the best training epoch, specifically when the GAN is being used to synthesize data for training downstream segmentation models.
- **Quantitative validation metric results:** The real-world datasets BBBC039v1 and Lizard are extended with synthetic label images. For the first time, the datasets are utilized to derive quantitative results for the Fréchet Inception Distance and Kernel Inception Distance metrics, assessing their effectiveness as GAN validation metrics. The results demonstrate that metrics, which are used to assess the similarity between two unpaired sets of images, can be used for epoch selection after GAN training if the GAN is used to synthesize data to train downstream segmentation models. Early stopping is not possible due to the unstable training of GANs.

- **A semiautomatic procedure for epoch selection:** Metrics like the Fréchet Inception Distance or the Kernel Inception Distance fail as an epoch selection criterion if object properties, such as the shape, differ between synthetic label images and target images. Therefore, a new semiautomatic procedure is introduced to be able to use the image metrics to select the best GAN epoch even in the case of flawed object properties in the synthetic label images.

- **A new patch-based inference benchmark dataset:** Patch-based inference with GANs leads to errors in the synthesized target images. For the first time, a benchmark dataset for patch-based inference is introduced, which enables automatic quantification of patch-based errors and, therefore, the evaluation of different patch-based inference methods.

- **A new method for patch-based inference:** A new method called *SATI* is proposed to reduce the errors introduced by patch-based inference of GANs if large-scale 2D or 3D images are processed. With *SATI*, the correct continuation of objects in adjacent patches is learned during the GAN training. The method can be integrated into arbitrary GANs.

- **Quantitative comparison of patch-based inference methods:** The benchmark dataset for patch-based inference is used to quantitatively compare existing methods for the first time. In addition, results for *SATI* integrated into CycleGAN are generated. Using *SATI* results in significantly fewer errors compared to existing methods. An ablation study quantifies the benefits of the individual components integrated in *SATI*.

These main contributions further increase the synthesis quality of GANs used to synthesize training data for downstream segmentation models, while the computational expense is reduced. The increased quality of the synthetic training data results in better downstream segmentation performance. Furthermore, the results can be used to reduce the manual work required during label image synthesis. With these improvements, it is possible to avoid manual annotation more often by using GANs.

However, GANs are currently used mainly in two large areas: The research and improvement of the methodology itself and data synthesis for downstream segmentation tasks where manual labeling is difficult or impossible, such as for 3D images. It can be assumed that GANs will not completely replace manual labeling in the future, but they will be a valuable addition to it. It must be decided on a case-by-case basis which workflows achieve the desired results and are target-oriented.

There are still various reasons why GANs are often not used in practice to omit manual labeling. They require a lot of computation, training is unstable, and the hyperparameters are difficult to set. The high speed of GPU development will overcome the first point, though lightweight model development could be useful to enable e.g.: fast retraining to adapt to new imaging settings. The other disadvantages can possibly be remedied by further research into the architectures or the training process itself.

Although it is still astonishing that GANs can learn from unpaired data in an unsupervised manner, the results achievable for complex data are often inferior to manual labeling. A thorough investigation of the reasons has the potential to enable future improvements of the downstream segmentation quality. A first important step is to investigate whether the domain gap between synthetic data and real data is responsible for the reduced segmentation quality and how the domain gap can be further reduced.

A future area of application for GANs is the creation of benchmark datasets for tracking. The manual effort to generate fully annotated 3D+t data is enormous. However, without these data, a meaningful quantitative comparison of tracking methods is difficult. Here, GANs can support the development and evaluation of tracking algorithms by providing high-quality fully annotated data.

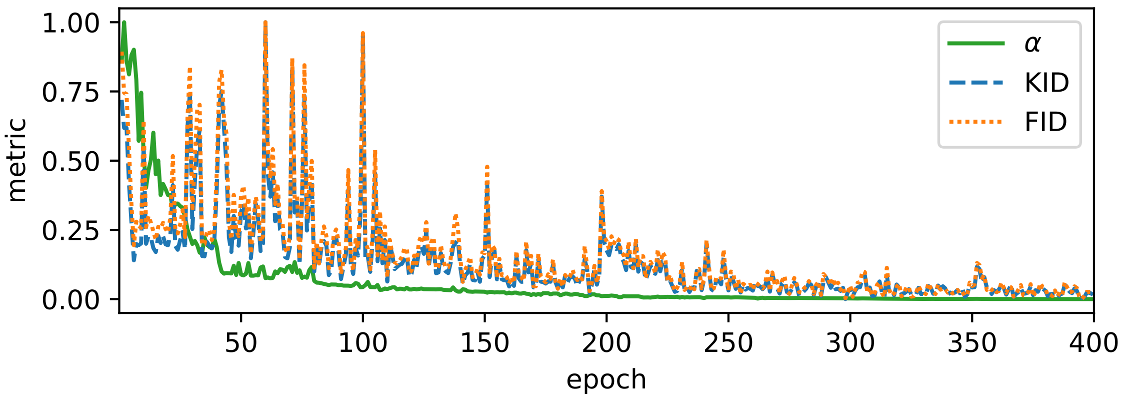
Unfortunately, there are no universally valid benchmark datasets or workflows for GANs which are used to generate synthetic data for segmentation. Therefore, as a researcher, it is almost impossible to decide in advance which GAN architecture will work well to solve a particular problem. As of now, several major architectures and many adaptions to perform unsupervised image-to-image translation exist. As a result, many different methods are currently in use. Although a wide variety of applicable methods is good, further narrowing down to a few methods that work best would make it easier to get started and enable the use of out-of-the-box methods.

Appendix A

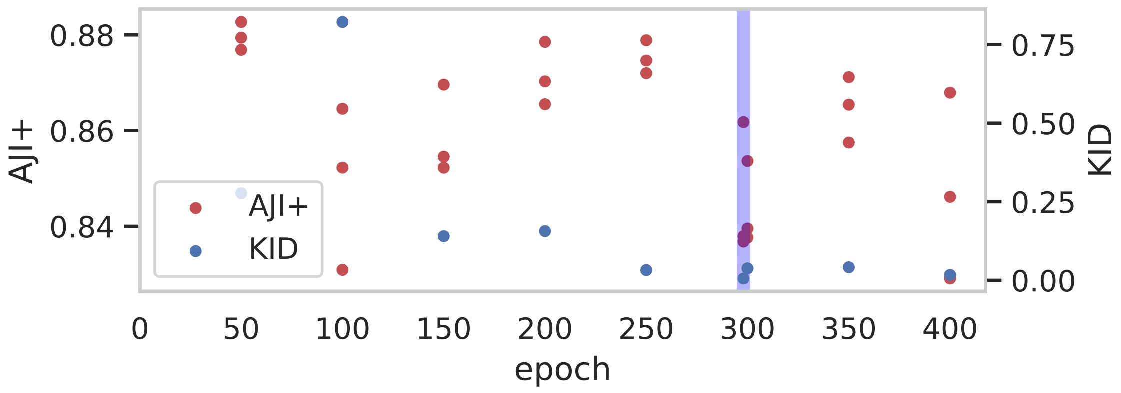
A.1 GAN Training Stopping and Epoch Selection Results

This section contains additional material to the experiments carried out in Chapter 3. The results for the second and third GAN trained on the BBBC039v1 dataset are shown in Fig. A.1 and Fig. A.2. The GANs behave similarly to the GAN examined in Section 3.5.2. The GANs alter the shape of the nuclei in the synthetic target images towards the end of the training. Fig. A.1b shows a severely worse KID for epoch 100 than for epoch 50, resulting in worse synthetic targets (see Fig. A.1c) and a worse AJI^{+}.

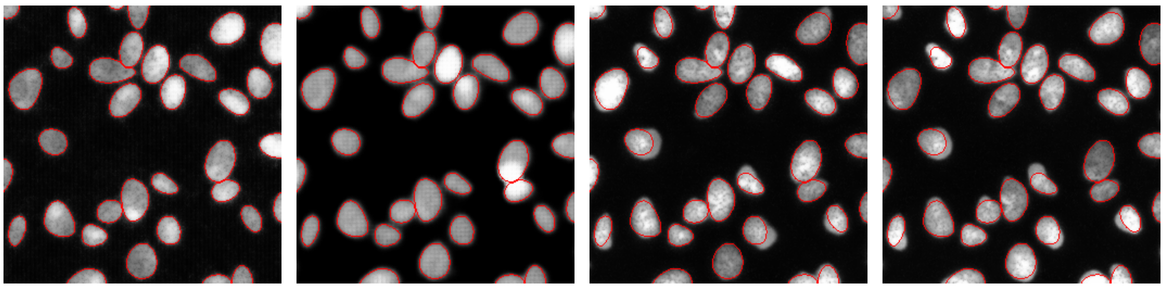
The results for the second and third GAN trained on the Lizard dataset are shown in Fig. A.3 and Fig. A.4. Again, the results are equivalent to the results shown in Section 3.5.3.



a) Metrics acquired during GAN training

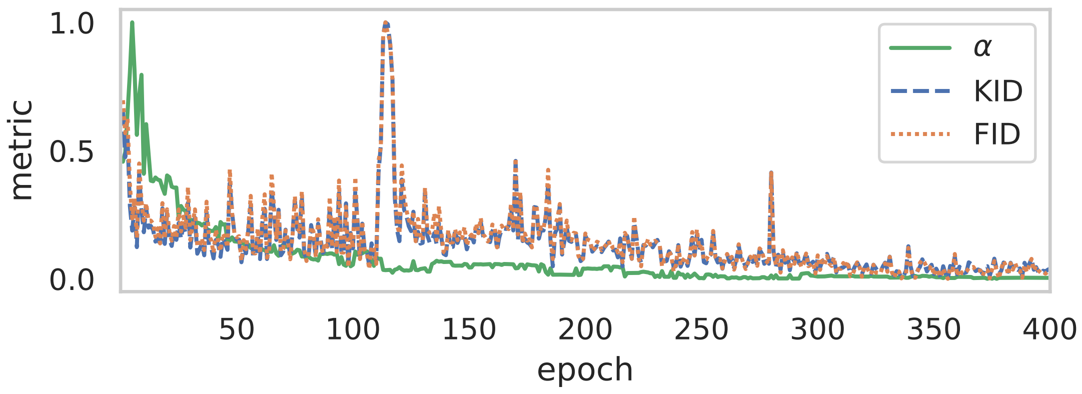


b) KID for key epochs and corresponding AJI$^+$ scores

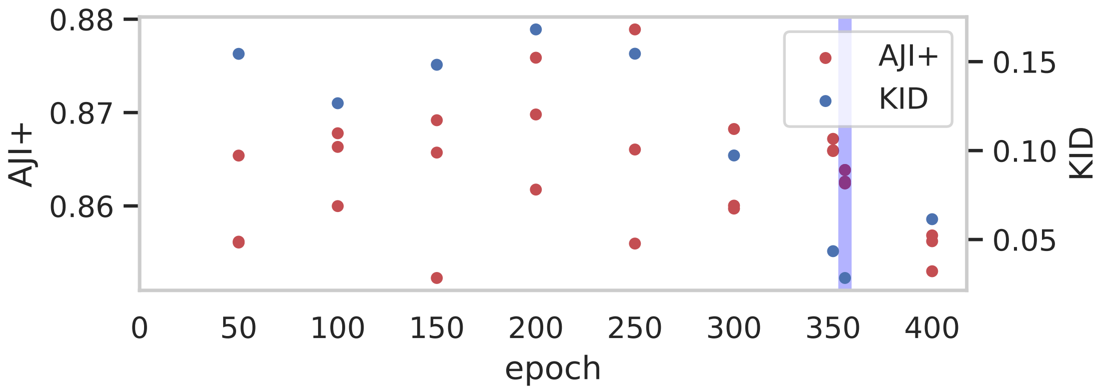


c) Crops of the synthetic images for epochs 50, 100, 298 and 400

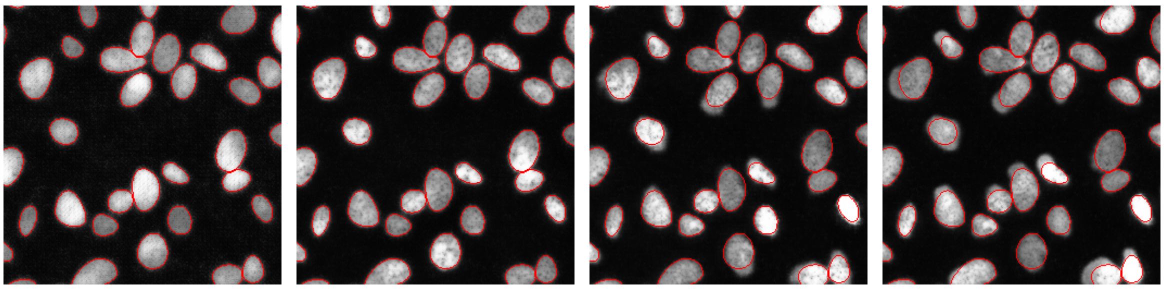
Figure A.1 | (a) shows the metrics acquired during GAN training for all epochs. Each of the metrics is normalized to the range [0, 1]. Strong peaks occur for KID and FID and they are highly correlated. The α-metric shows a steady decline. The AJI$^+$ scores of the U-Nets trained for the nine key epochs and the corresponding KID scores are shown in (b). The epoch with the lowest KID is epoch 298 and is marked in purple. The images in (c) show example crops synthesized from the same label image for different epochs. The red overlay shows the contours of the objects in the synthetic label image. The images show, that the GAN starts to alter the shapes for late epochs.



a) Metrics acquired during GAN training

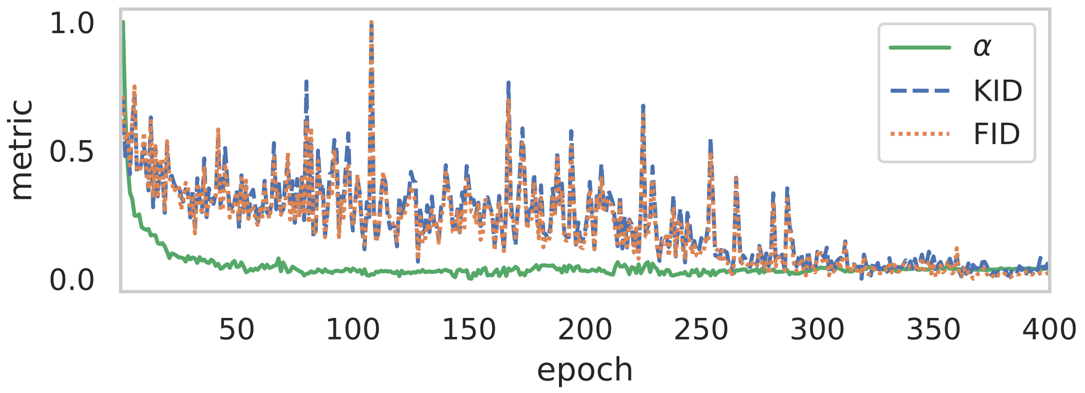


b) KID for key epochs and corresponding AJI$^+$ scores

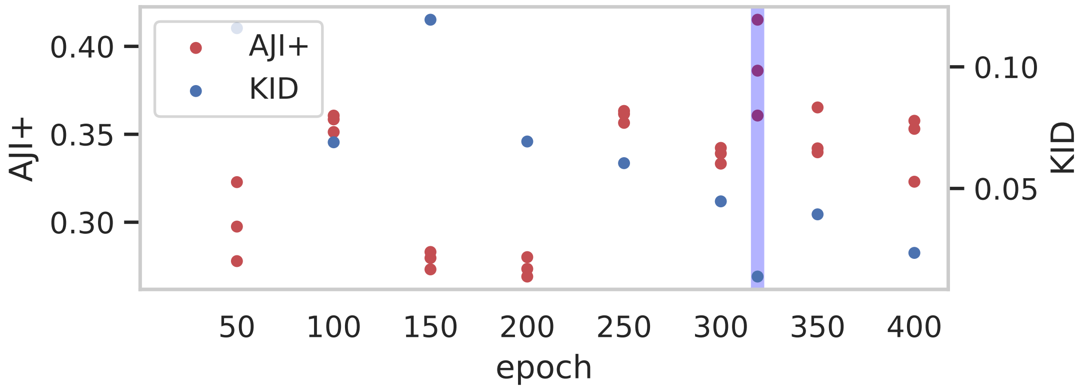


c) Crops of the synthetic images for epochs 50, 100, 356 and 400

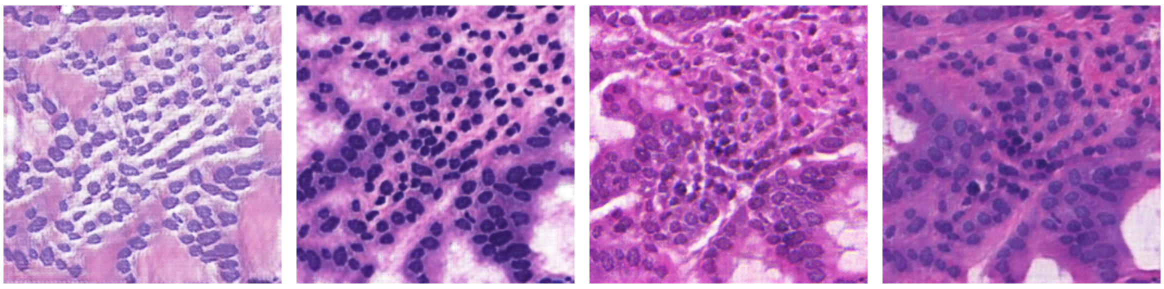
Figure A.2 | The metrics in (a) are acquired during GAN training and are normalized to the range [0, 1] for visualization. The KID for the nine key epochs and the corresponding AJI$^+$ scores of the U-Nets are shown in (b). Epoch 356 has the lowest KID and is marked in purple. Targets synthesized from the same label image for different epochs are shown in (c). The contours of the objects in the synthetic label image are depicted with a red overlay. The crops show, that the GAN alters the shape properties in late epochs.



a) Metrics acquired during GAN training

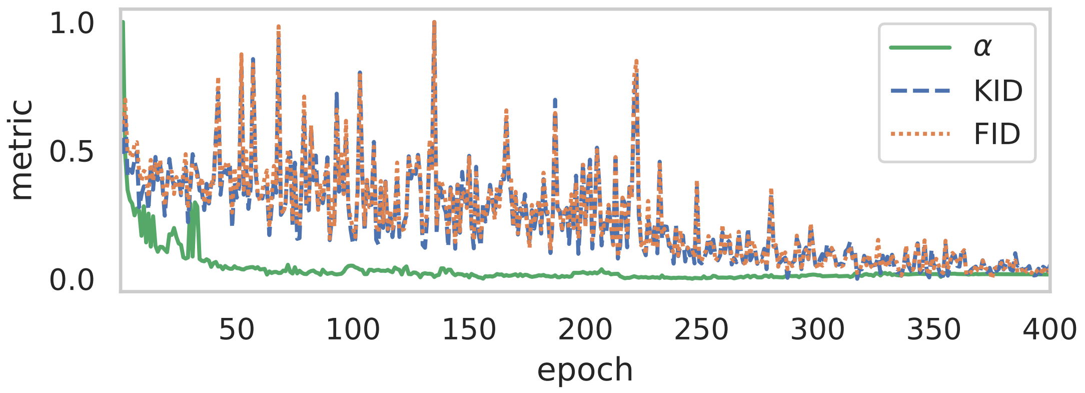


b) KID for key epochs and corresponding AJI$^+$ scores

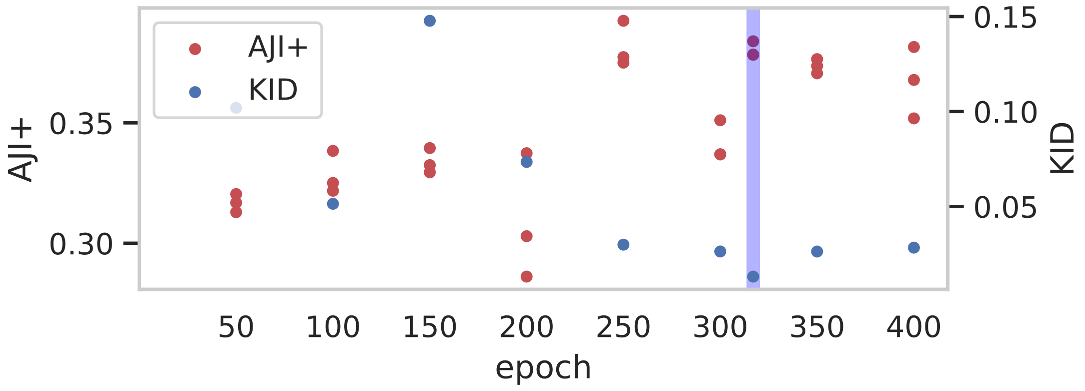


c) Crops of the synthetic images for epochs 50, 100, 319 and 400

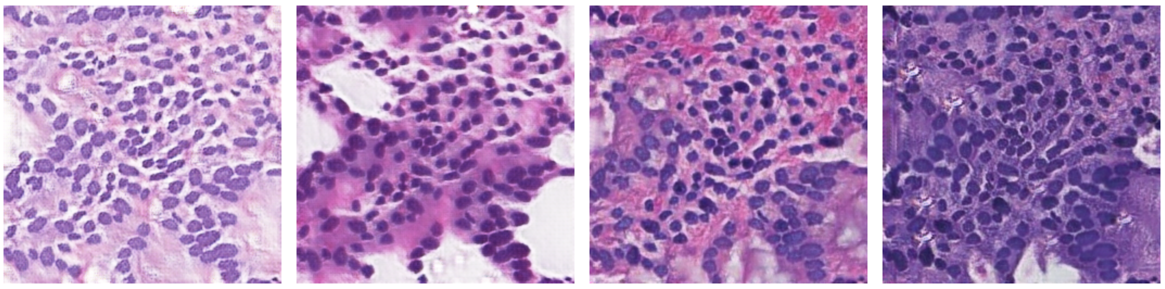
Figure A.3 | The metrics obtained during GAN training across all epochs are presented in (a). Each metric is normalized to a range of 0 to 1. The KID and FID metrics demonstrate a high correlation and have pronounced peak values. In contrast, the α-metric exhibits a more gradual decreasing trend. (b) shows the KID scores for nine key epochs and the corresponding AJI$^+$ values for the U-Nets. Epoch 319, highlighted in purple, has the minimum KID value. (c) shows crops of images synthesized for the same label image at different epochs. Overlaying the contours of the objects in the label images on the synthetic targets is omitted for better visual assessment.



a) Metrics acquired during GAN training



b) KID for key epochs and corresponding AJI^{+} scores



c) Crops of the synthetic images for epochs 50, 100, 317 and 400

Figure A.4 | (a) presents the metrics acquired during the GAN training across all epochs. The metrics are each normalized to a range of 0 to 1. The KID and FID metrics show a high correlation and pronounced peak values. In contrast, the α-metric displays a more gradual downward trend over the epochs. In (b), the KID scores for nine key epochs during training are shown, as well as the corresponding AJI^{+} values. Epoch 319, highlighted in purple, has the minimum KID value. (c) displays crops of images synthesized for the same label image at different epochs during training. To enable a better visual assessment, the overlaying of the contours of objects present in the label images is omitted.

A.2 3D Label Image Synthesis

This section describes the label images synthesis procedure for the experiments on the real-world dataset presented in Section 4.5.2. The synthetic label image domain $\mathcal{X}$ is created by random placement of ellipsoids. In Chapter 2, ellipsoids have been shown to be a good representation of cell nuclei. The sizes in pixel of the three major axes of the ellipsoids are each distributed by $\mathcal{N}(12, 2^2)$. The values were estimated by measuring single nuclei in the microscopy images and visually comparing the synthetic label images with the real-world target images. Additionally, the real-world images are densely packed. To acquire densely packed synthetic label images, the position of an ellipsoid is randomly chosen. If the overlap with existing objects is less than 5%, the ellipsoid is placed. Otherwise, 5000 different positions are checked. If no position is found, a new ellipsoid is created by drawing from the distributions of the major axes. This ensures filling the small gaps remaining, where larger ellipsoids do not fit. If the program is unable to place 30 different ellipsoids in a row, the volume is considered densely packed, and the placement of the ellipsoids is ended. Four synthetic label images with a size of 256 px $\times$ 256 px $\times$ 256 px are created. The foreground is set to a value of 130 and the background to 10. Afterwards, Gaussian noise with a mean of zero and $\sigma = 3.33$ is applied. In the last step, the values are rounded to integers, and the range is clipped to [0, 255] to be able to visualize the images as standard 8-bit grayscale images. Clipping results in an increase in the extreme values, which is not relevant for GAN training, as long as the entropy in the images is still high enough to enable variation during synthesis of different areas in an image.

Abbreviations B

Abbreviation	Description
AJI$^+$	Advanced version of the aggregated Jaccard index
CCD	Charge-coupled device
CMOS	Complementary metal-oxide semiconductor
CNN	Convolutional neural network
CoCoBeDa	Contour complexity benchmark dataset
CT	Computed tomography
DSLR	Digital single-lens reflex camera
EFD	Elliptic fourier shape descriptor
FID	Fréchet inception distance
FSD	Fourier shape descriptor
GAN	Generative adversarial network
GMM	Gaussian mixture model
GPU	Graphics processing unit
HSI	Hyperspectral imaging
H&E	Haematoxylin and eosin
IS	Inception score
KaIDA	Karlsruhe image data annotation
KID	Kernel inception distance
KL	Kullback–Leibler
LPIPS	Learned perceptual image patch similarity

MMD	Maximum mean discrepancy
MRI	Magnetic resonance imaging
MSE	Mean squared error
NN	Neural network
PCA	Principal components analysis
PDE	Partial differential equation
PDF	Probability density function
PSNR	Peak signal-to-noise ratio
SATI	Stitching aware training and inference
SGD	Stochastic gradient descent
SSIM	Structural similarity index measure
SSM	Statistical shape model
SSPBeDa	Shape and spatial property benchmark dataset
SVM	Support vector machine
VRAM	Video random-access memory
WW	Weight watcher
w/o	Without

List of Figures

List of Tables

References

[1] K. D. Toennies, *Guide to Medical Image Analysis: Methods and Algorithms*, 1st ed. London, UK: Springer London, 2012. DOI: 10.1007/978-1-4471-2751-2.

[2] C. A. Casacio, L. S. Madsen, A. Terrasson, *et al.*, “Quantum-enhanced nonlinear microscopy”, *Nature*, vol. 594, no. 7862, pp. 201–206, 2021. DOI: 10.1038/s41586-021-03528-w.

[3] O. Russakovsky, J. Deng, H. Su, *et al.*, “ImageNet Large Scale Visual Recognition Challenge”, *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

[4] G. Litjens, T. Kooi, B. E. Bejnordi, *et al.*, “A survey on deep learning in medical image analysis”, *Medical Image Analysis*, vol. 42, pp. 60–88, 2017. DOI: 10.1016/j.media.2017.07.005.

[5] S. R. Spurgeon, C. Ophus, L. Jones, *et al.*, “Towards data-driven next-generation transmission electron microscopy”, *Nature Materials*, vol. 20, no. 3, pp. 274–279, 2021. DOI: 10.1038/s41563-020-00833-z.

[6] K. Muhammad, A. Ullah, J. Lloret, J. D. Ser, and V. H. C. de Albuquerque, “Deep Learning for Safe Autonomous Driving: Current Challenges and Future Directions”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4316–4336, 2021. DOI: 10.1109/TITS.2020.3032227.

[7] G. Cheng, X. Xie, J. Han, L. Guo, and G.-S. Xia, “Remote Sensing Image Scene Classification Meets Deep Learning: Challenges, Methods, Benchmarks, and Opportunities”, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 3735–3756, 2020. DOI: 10.1109/JSTARS.2020.3005403.

[8] J. A. Richards, *Remote Sensing Digital Image Analysis*. Cham, Switzerland: Springer Cham, 2022. DOI: 10.1007/978-3-030-82327-6.

[9] Q. Yuan, H. Shen, T. Li, *et al.*, "Deep learning in environmental remote sensing: Achievements and challenges", *Remote Sensing of Environment*, vol. 241, p. 111 716, 2020. DOI: 10.1016/j.rse.2020.111716.

[10] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving", *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020. DOI: 10.1002/rob.21918.

[11] H.-P. Chan, R. K. Samala, L. M. Hadjiiski, and C. Zhou, "Deep Learning in Medical Image Analysis", in *Deep Learning in Medical Image Analysis : Challenges and Applications*, G. Lee and H. Fujita, Eds., Cham, Switzerland: Springer International Publishing, 2020, pp. 3–21. DOI: 10.1007/978-3-030-33128-3_1.

[12] M. K. Ferguson, A. Ronay, Y.-T. T. Lee, and K. H. Law, "Detection and Segmentation of Manufacturing Defects with Convolutional Neural Networks and Transfer Learning", *Smart and sustainable manufacturing systems*, vol. 2, no. 1, 2018. DOI: 10.1520/SSMS20180033.

[13] R. Xu and C. Li, "A Review of High-Throughput Field Phenotyping Systems: Focusing on Ground Robots", *Plant Phenomics*, vol. 2022, 2022. DOI: 10.34133/2022/9760269.

[14] C. Lei, H. Kobayashi, Y. Wu, *et al.*, "High-throughput imaging flow cytometry by optofluidic time-stretch microscopy", *Nature Protocols*, vol. 13, no. 7, pp. 1603–1631, 2018. DOI: 10.1038/s41596-018-0008-7.

[15] C. Belthangady and L. A. Royer, "Applications, promises, and pitfalls of deep learning for fluorescence image reconstruction", *Nature Methods*, vol. 16, no. 12, pp. 1215–1225, 2019. DOI: 10.1038/s41592-019-0458-z.

[16] P. M. Bhatt, R. K. Malhan, P. Rajendran, *et al.*, "Image-Based Surface Defect Detection Using Deep Learning: A Review", *Journal of Computing and Information Science in Engineering*, vol. 21, no. 4, p. 040 801, 2021. DOI: 10.1115/1.4049535.

[17] J. Zhang, C. Li, M. M. Rahaman, *et al.*, "A comprehensive review of image analysis methods for microorganism counting: From classical image processing to deep learning approaches", *Artificial Intelligence Review*, vol. 55, no. 4, pp. 2875–2944, 2022. DOI: 10.1007/s10462-021-10082-4.

[18] F. Renard, S. Guedria, N. D. Palma, and N. Vuillerme, "Variability and reproducibility in deep learning for medical image segmentation", *Scientific Reports*, vol. 10, no. 1, p. 13 724, Aug. 2020. DOI: 10.1038/s41598-020-69920-0.

[19] L. Zhang, R. Tanno, M.-C. Xu, *et al.*, “Disentangling Human Error from Ground Truth in Segmentation of Medical Images”, in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, vol. 33, Curran Associates, Inc., 2020, pp. 15 750–15 762.

[20] A. Das, Y. Xian, Y. He, Z. Akata, and B. Schiele, “Urban Scene Semantic Segmentation With Low-Cost Coarse Annotation”, in *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Waikoloa, Hawaii: IEEE, 2023, pp. 5967–5976. DOI: 10.1109/WACV56688.2023.00592.

[21] S. Müller, C. Sauter, R. Shunmugasundaram, *et al.*, “Deep learning-based segmentation of lithium-ion battery microstructures enhanced by artificially generated electrodes”, *Nature Communications*, vol. 12, no. 1, p. 6205, 2021. DOI: 10.1038/s41467-021-26480-9.

[22] K. W. Dunn, C. Fu, D. J. Ho, *et al.*, “DeepSynth: Three-Dimensional Nuclear Segmentation of Biological Images Using Neural Networks Trained with Synthetic Data”, *Scientific Reports*, vol. 9, no. 1, p. 18 295, 2019. DOI: 10.1038/s41598-019-54244-5.

[23] James B. Campbell, Randolph H. Wynne, and Valerie A. Thomas, *Introduction to Remote Sensing*, 6th ed. New York, USA: The Guilford Press, 2023, ISBN: 978-1-60918-177-2.

[24] M. Aminzadeh and T. R. Kurfess, “Online quality inspection using Bayesian classification in powder-bed additive manufacturing from high-resolution visual camera images”, *Journal of Intelligent Manufacturing*, vol. 30, no. 6, pp. 2505–2523, 2019. DOI: 10.1007/s10845-018-1412-0.

[25] Y. Wang, N. K. Kanagaraj, C. Pylatiuk, R. Mikut, R. Peravali, and M. Reischl, “High-Throughput Data Acquisition Platform for Multi-Larvae Touch-Response Behavior Screening of Zebrafish”, *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 858–865, 2022. DOI: 10.1109/LRA.2021.3134281.

[26] N. J. Pelc, “Recent and Future Directions in CT Imaging”, *Annals of Biomedical Engineering*, vol. 42, no. 2, pp. 260–268, 2014. DOI: 10.1007/s10439-014-0974-z.

[27] Dominik Weishaupt, Victor D. Köchli, and Borut Marincek, *How Does MRI Work?*, 2nd ed. Heidelberg, Germany: Springer Berlin, Heidelberg, 2006. DOI: 10.1007/978-3-540-37845-7.

[28] D. Onder, S. Zengin, and S. Sarioglu, “A Review on Color Normalization and Color Deconvolution Methods in Histopathology”, *Applied Immunohistochemistry & Molecular Morphology*, vol. 22, no. 10, p. 713, 2014. DOI: 10.1097/PAI.0000000000000003.

[29] M. J. Sanderson, I. Smith, I. Parker, and M. D. Bootman, “Fluorescence Microscopy”, *Cold Spring Harbor protocols*, vol. 2014, no. 10, pdb.top071795, 2014. DOI: 10.1101/pdb.top071795.

[30] Jyotismita Chaki and Nilanjan Dey, *A Beginner’s Guide to Image Preprocessing Techniques*, 1st ed. Boca Raton, Florida, USA: CRC Press, 2019, ISBN: 978-0-367-57080-4.

[31] T. Peng, K. Thorn, T. Schroeder, *et al.*, “A BaSiC tool for background and shading correction of optical microscopy images”, *Nature Communications*, vol. 8, no. 1, p. 14 836, 2017. DOI: 10.1038/ncomms14836.

[32] A. Vahadane, T. Peng, A. Sethi, *et al.*, “Structure-Preserving Color Normalization and Sparse Stain Separation for Histological Images”, *IEEE Transactions on Medical Imaging*, vol. 35, no. 8, pp. 1962–1971, 2016. DOI: 10.1109/TMI.2016.2529665.

[34] M. Maška, V. Ulman, P. Delgado-Rodriguez, *et al.*, “The Cell Tracking Challenge: 10 years of objective benchmarking”, *Nature Methods*, vol. 20, no. 7, pp. 1010–1020, 2023. DOI: 10.1038/s41592-023-01879-y.

[35] L. Zhang, Z. Shi, M.-M. Cheng, *et al.*, “Nonlinear Regression via Deep Negative Correlation Learning”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 3, pp. 982–998, 2021. DOI: 10.1109/TPAMI.2019.2943860.

[36] G. Guo, Y. Fu, C. R. Dyer, and T. S. Huang, “Image-Based Human Age Estimation by Manifold Learning and Locally Adjusted Robust Regression”, *IEEE Transactions on Image Processing*, vol. 17, no. 7, pp. 1178–1188, 2008. DOI: 10.1109/TIP.2008.924280.

[37] D. Tome, C. Russell, and L. Agapito, “Lifting From the Deep: Convolutional 3D Pose Estimation From a Single Image”, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, Hawaii: IEEE, 2017, pp. 5689–5698. DOI: 10.1109/CVPR.2017.603.

[38] C. Zheng, W. Wu, C. Chen, *et al.*, “Deep Learning-based Human Pose Estimation: A Survey”, *ACM Computing Surveys*, vol. 56, no. 1, 11:1–11:37, 2023. DOI: 10.1145/3603618.

[40] A. Latif, A. Rasheed, U. Sajid, *et al.*, “Content-Based Image Retrieval and Feature Extraction: A Comprehensive Review”, *Mathematical Problems in Engineering*, vol. 2019, 2019. DOI: 10.1155/2019/9658350.

[41] W. K. Mutlag, S. K. Ali, Z. M. Aydam, and B. H. Taher, “Feature Extraction Methods: A Review”, *Journal of Physics: Conference Series*, vol. 1591, no. 1, p. 012 028, 2020. DOI: 10.1088/1742-6596/1591/1/012028.

[42] D. p. Tian, “A Review on Image Feature Extraction and Representation Techniques”, *International Journal of Multimedia and Ubiquitous Engineering*, vol. 8, no. 4, pp. 385–396, 2013.

[43] J. M. Patel and N. C. Gamit, “A review on feature extraction techniques in Content Based Image Retrieval”, in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Chennai, India: IEEE, 2016, pp. 2259–2263. DOI: 10.1109/WiSPNET.2016.7566544.

[44] S. Menard, *Applied Logistic Regression Analysis*, 2nd ed. Thousand Oaks, CA, USA: SAGE, 2002, ISBN: 978-0-7619-2208-7.

[45] J. M. Keller, M. R. Gray, and J. A. Givens, “A fuzzy K-nearest neighbor algorithm”, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 4, pp. 580–585, 1985. DOI: 10.1109/TSMC.1985.6313426.

[46] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Schölkopf, “Support vector machines”, *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998. DOI: 10.1109/5254.708428.

[47] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System”, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 785–794. DOI: 10.1145/2939672.2939785.

[48] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, ISBN: 978-0-262-03561-3.

[51] I. Bello, W. Fedus, X. Du, *et al.*, “Revisiting ResNets: Improved Training and Scaling Strategies”, *arXiv*, 2021, preprint. DOI: 10.48550/arXiv.2103.07579.

[52] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition”, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, Nevada, USA: IEEE, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[53] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, “Benchmark Analysis of Representative Deep Neural Network Architectures”, *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018. DOI: 10.1109/ACCESS.2018.2877890.

[56] P. Ma, C. Li, M. M. Rahaman, *et al.*, “A state-of-the-art survey of object detection techniques in microorganism image analysis: From classical methods to deep learning approaches”, *Artificial Intelligence Review*, vol. 56, no. 2, pp. 1627–1698, 2023. DOI: 10.1007/s10462-022-10209-1.

[57] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, “Contour Detection and Hierarchical Image Segmentation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011. DOI: 10.1109/TPAMI.2010.161.

[58] F. Kulwa, C. Li, X. Zhao, *et al.*, “A State-of-the-Art Survey for Microorganism Image Segmentation Methods and Future Potential”, *IEEE Access*, vol. 7, pp. 100 243–100 269, 2019. DOI: 10.1109/ACCESS.2019.2930111.

[59] E. Meijering, “Cell Segmentation: 50 Years Down the Road [Life Sciences]”, *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 140–145, 2012. DOI: 10.1109/MSP.2012.2204190.

[60] J. Kuruvilla, D. Sukumaran, A. Sankar, and S. P. Joy, “A review on image processing and image segmentation”, in *2016 International Conference on Data Mining and Advanced Computing (SAPIENCE)*, Ernakulam, India: IEEE, 2016, pp. 198–203. DOI: 10.1109/SAPIENCE.2016.7684170.

[61] H. Seo, M. Badiei Khuzani, V. Vasudevan, *et al.*, “Machine learning techniques for biomedical image segmentation: An overview of technical aspects and introduction to state-of-art applications”, *Medical Physics*, vol. 47, no. 5, e148–e167, 2020. DOI: 10.1002/mp.13649.

[62] N. O’Mahony, S. Campbell, A. Carvalho, *et al.*, “Deep Learning vs. Traditional Computer Vision”, in *Advances in Computer Vision*, Las Vegas, Nevada, USA: Springer, Cham, 2020, pp. 128–144. DOI: 10.1007/978-3-030-17795-9_10.

[63] L. Vincent, “Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms”, *IEEE Transactions on Image Processing*, vol. 2, no. 2, pp. 176–201, 1993. DOI: 10.1109/83.217222.

[64] B. P. Marsh, N. Chada, R. R. Sanganna Gari, K. P. Sigdel, and G. M. King, “The Hessian Blob Algorithm: Precise Particle Detection in Atomic Force Microscopy Imagery”, *Scientific Reports*, vol. 8, no. 1, p. 978, 2018. DOI: 10.1038/s41598-018-19379-x.

[65] S. Graham, Q. D. Vu, S. E. A. Raza, *et al.*, “HoVer-Net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images”, *Medical Image Analysis*, vol. 58, p. 101 563, 2019. DOI: 10.1016/j.media.2019.101563.

[66] J. Vuchkova, T. S. Maybury, and C. S. Farah, “Testing the Educational Potential of 3D Visualization Software in Oral Radiographic Interpretation”, *Journal of Dental Education*, vol. 75, no. 11, pp. 1417–1425, 2011. DOI: 10.1002/j.0022-0337.2011.75.11.tb05198.x.

[67] M. Eid, C. N. De Cecco, J. W. Nance, *et al.*, “Cinematic Rendering in CT: A Novel, Lifelike 3D Visualization Technique”, *American Journal of Roentgenology*, vol. 209, no. 2, pp. 370–379, 2017. DOI: 10.2214/AJR.17.17850.

[68] A. J. Pretorius, I. A. Khan, and R. J. Errington, “A Survey of Visualization for Live Cell Imaging”, *Computer Graphics Forum*, vol. 36, no. 1, pp. 46–63, 2017. DOI: 10.1111/cgf.12784.

[69] B. Schott, M. Traub, C. Schlagenhauf, *et al.*, “EmbryoMiner: A new framework for interactive knowledge discovery in large-scale cell tracking data of developing embryos”, *PLOS Computational Biology*, vol. 14, no. 4, e1006128, 2018. DOI: 10.1371/journal.pcbi.1006128.

[70] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors”, *Nature*, vol. 323, pp. 533–536, 1986. DOI: 10.1038/323533a0.

[71] L. Bottou, “Stochastic Gradient Descent Tricks”, in *Neural Networks: Tricks of the Trade: Second Edition*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds., Berlin, Heidelberg, Germany: Springer, Berlin, Heidelberg, 2012, pp. 421–436. DOI: 10.1007/978-3-642-35289-8_25.

[72] S. Jadon, “A survey of loss functions for semantic segmentation”, in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, IEEE, 2020, pp. 1–7. DOI: 10.1109/CIBCB48159.2020.9277638.

[73] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, “A Comprehensive Survey of Loss Functions in Machine Learning”, *Annals of Data Science*, vol. 9, no. 2, pp. 187–212, 2022. DOI: 10.1007/s40745-020-00253-5.

[74] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization”, *arXiv*, 2017, preprint. DOI: 10.48550/arXiv.1412.6980.

[75] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, “Domain Adaptation via Transfer Component Analysis”, *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011. DOI: 10.1109/TNN.2010.2091281.

[76] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, “Generative Adversarial Nets”, in *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, vol. 27, Montreal, Canada: Curran Associates, Inc., 2014, pp. 2672–2680.

[77] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition”, *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[78] M. Mirza and S. Osindero, “Conditional Generative Adversarial Nets”, *arXiv*, 2014, preprint. DOI: 10.48550/arXiv.1411.1784.

[79] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-To-Image Translation With Conditional Adversarial Networks”, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, Hawaii: IEEE, 2017, pp. 5967–5976. DOI: 10.1109/CVPR.2017.632.

[80] J.-Y. Zhu, R. Zhang, D. Pathak, *et al.*, “Toward Multimodal Image-to-Image Translation”, in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, vol. 30, Long Beach, CA, USA: Curran Associates, Inc., 2017.

[81] A. Bansal, Y. Sheikh, and D. Ramanan, “PixelNN: Example-based Image Synthesis”, *arXiv*, 2018, preprint. DOI: 10.48550/arXiv.1708.05349.

[82] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”, in *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy: IEEE, 2017, pp. 2242–2251. DOI: 10.1109/ICCV.2017.244.

[83] M.-Y. Liu, T. Breuel, and J. Kautz, “Unsupervised Image-to-Image Translation Networks”, *arXiv*, 2018, preprint. DOI: 10.48550/arXiv.1703.00848.

[84] Y. Pang, J. Lin, T. Qin, and Z. Chen, “Image-to-Image Translation: Methods and Applications”, *IEEE Transactions on Multimedia*, vol. 24, pp. 3859–3881, 2022. DOI: 10.1109/TMM.2021.3109419.

[85] H. Huang, P. S. Yu, and C. Wang, “An Introduction to Image Synthesis with Generative Adversarial Nets”, *arXiv*, 2018, preprint. DOI: 10.48550/arXiv.1803.04469.

[87] K. Yao, J. Sun, K. Huang, *et al.*, “Analyzing Cell-Scaffold Interaction through Unsupervised 3D Nuclei Segmentation”, *International journal of bioprinting*, vol. 8, no. 1, p. 495, 2022. DOI: 10.18063/ijb.v8i1.495.

[88] J. Wang, N. Tabassum, T. T. Toma, Y. Wang, A. Gahlmann, and S. T. Acton, “3D GAN image synthesis and dataset quality assessment for bacterial biofilm”, *Bioinformatics*, vol. 38, no. 19, pp. 4598–4604, 2022. DOI: 10.1093/bioinformatics/btac529.

[90] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, “Least Squares Generative Adversarial Networks”, *arXiv*, 2017, preprint. DOI: 10.48550/arXiv.1611.04076.

[91] Y. Taigman, A. Polyak, and L. Wolf, “Unsupervised Cross-Domain Image Generation”, *arXiv*, 2022, preprint. DOI: 10.48550/arXiv.1611.02200.

[92] A. Alotaibi, “Deep Generative Adversarial Networks for Image-to-Image Translation: A Review”, *Symmetry*, vol. 12, no. 10, p. 1705, 2020. DOI: 10.3390/sym12101705.

[93] A. Brock, J. Donahue, and K. Simonyan, “Large Scale GAN Training for High Fidelity Natural Image Synthesis”, *arXiv*, 2019, preprint. DOI: 10.48550/arXiv.1809.11096.

[94] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and Improving the Image Quality of StyleGAN”, *arXiv*, 2020, preprint. DOI: 10.48550/arXiv.1912.04958.

[95] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, “Synthetic data augmentation using GAN for improved liver lesion classification”, in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, Washington, DC, USA: IEEE, 2018, pp. 289–293. DOI: 10.1109/ISBI.2018.8363576.

[96] A. Radford, L. Metz, and S. Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, *arXiv*, 2016, preprint. DOI: 10.48550/arXiv.1511.06434.

[97] H.-J. Kong, J. Y. Kim, H.-M. Moon, *et al.*, “Automation of generative adversarial network-based synthetic data-augmentation for maximizing the diagnostic performance with paranasal imaging”, *Scientific Reports*, vol. 12, no. 1, p. 18 118, 2022. DOI: 10.1038/s41598-022-22222-z.

[98] A. Odena, C. Olah, and J. Shlens, “Conditional Image Synthesis with Auxiliary Classifier GANs”, in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, Sydney, NSW, Australia: PMLR, 2017, pp. 2642–2651.

[99] L. Zhu, Y. Chen, P. Ghamisi, and J. A. Benediktsson, “Generative Adversarial Networks for Hyperspectral Image Classification”, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 9, pp. 5046–5063, 2018. DOI: 10.1109/TGRS.2018.2805286.

[100] A. Qin, Z. Tan, R. Wang, *et al.*, “Distance Constraint-Based Generative Adversarial Networks for Hyperspectral Image Classification”, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–16, 2023. DOI: 10.1109/TGRS.2023.3274778.

[101] Á. G. Dieste, F. Argüello, and D. B. Heras, “ResBaGAN: A Residual Balancing GAN with Data Augmentation for Forest Mapping”, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 16, pp. 6428–6447, 2023. DOI: 10.1109/JSTARS.2023.3281892.

[102] J. J. Bird, C. M. Barnes, L. J. Manso, A. Ekárt, and D. R. Faria, “Fruit quality and defect image classification with conditional GAN data augmentation”, *Scientia Horticulturae*, vol. 293, p. 110 684, 2022. DOI: 10.1016/j.scienta.2021.110684.

[103] S. Niu, B. Li, X. Wang, and H. Lin, “Defect Image Sample Generation With GAN for Improving Defect Recognition”, *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1611–1622, 2020. DOI: 10.1109/TASE.2020.2967415.

[104] D. Eschweiler, T. Klose, F. N. Müller-Fouarge, M. Kopaczka, and J. Stegmaier, “Towards Annotation-Free Segmentation of Fluorescently Labeled Cell Membranes in Confocal Microscopy Images”, in *Simulation and Synthesis in Medical Imaging. SASHIMI 2019*, Shenzhen, China: Springer Cham, 2019, pp. 81–89. DOI: 10.1007/978-3-030-32778-1_9.

[105] D. Eschweiler, M. Rethwisch, M. Jarchow, S. Koppers, and J. Stegmaier, “3D fluorescence microscopy data synthesis for segmentation and benchmarking”, *PLOS ONE*, vol. 16, no. 12, e0260509, 2021. DOI: 10.1371/journal.pone.0260509.

[106] C. Fu, S. Lee, D. Joon Ho, *et al.*, “Three Dimensional Fluorescence Microscopy Image Synthesis and Segmentation”, in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Salt Lake City, UT, USA: IEEE Computer Society, 2018, pp. 2302–23 028. DOI: 10.1109/CVPRW.2018.00298.

[107] S. J. Ihle, A. M. Reichmuth, S. Girardin, *et al.*, “Unsupervised data to content transformation with histogram-matching cycle-consistent generative adversarial networks”, *Nature Machine Intelligence*, vol. 1, no. 10, pp. 461–470, 2019. DOI: 10.1038/s42256-019-0096-2.

[108] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation”, in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Munich, Germany: Springer International Publishing, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[109] D. Neven, B. D. Brabandere, M. Proesmans, and L. V. Gool, “Instance Segmentation by Jointly Optimizing Spatial Embeddings and Clustering Bandwidth”, in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019, pp. 8829–8837. DOI: 10.1109/CVPR.2019.00904.

[110] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN”, in *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy: IEEE, 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[111] C. Stringer, T. Wang, M. Michaelos, and M. Pachitariu, “Cellpose: A generalist algorithm for cellular segmentation”, *Nature Methods*, vol. 18, no. 1, pp. 100–106, 2021. DOI: 10.1038/s41592-020-01018-x.

[112] M. Weigert, U. Schmidt, R. Haase, K. Sugawara, and G. Myers, “Star-convex Polyhedra for 3D Object Detection and Segmentation in Microscopy”, in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Snowmass Village, CO, USA: IEEE, 2020, pp. 3655–3662. DOI: 10.1109/WACV45572.2020.9093435.

[114] T. Vo and N. Khan, “Edge-preserving Image Synthesis for Unsupervised Domain Adaptation in Medical Image Segmentation”, in *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, Glasgow, UK: IEEE, 2022, pp. 3753–3757. DOI: 10.1109/EMBC48229.2022.9871402.

[115] A. Chartsias, T. Joyce, R. Dharmakumar, and S. A. Tsaftaris, "Adversarial Image Synthesis for Unpaired Multi-modal Cardiac Data", in *Simulation and Synthesis in Medical Imaging. SASHIMI 2017*, vol. 10557, Québec City, Canada: Springer, Cham, 2017, pp. 3–13. DOI: 10.1007/978-3-319-68127-6_1.

[116] Y. Huo, Z. Xu, S. Bao, A. Assad, R. G. Abramson, and B. A. Landman, "Adversarial synthesis learning enables segmentation without target modality ground truth", in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, Washington, DC, USA: IEEE, 2018, pp. 1217–1220. DOI: 10.1109/ISBI.2018.8363790.

[117] X. Chen, C. Lian, L. Wang, *et al.*, "Anatomy-Regularized Representation Learning for Cross-Modality Medical Image Segmentation", *IEEE Transactions on Medical Imaging*, vol. 40, no. 1, pp. 274–285, 2021. DOI: 10.1109/TMI.2020.3025133.

[118] T. d. Bel, M. Hermsen, J. Kers, J. Laak, and G. Litjens, "Stain-Transforming Cycle-Consistent Generative Adversarial Networks for Improved Segmentation of Renal Histopathology", in *Proceedings of The 2nd International Conference on Medical Imaging with Deep Learning*, vol. 102, London, UK: PMLR, 2019, pp. 151–163.

[119] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric", in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA: IEEE, 2018, pp. 586–595. DOI: 10.1109/CVPR.2018.00068.

[120] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity", *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. DOI: 10.1109/TIP.2003.819861.

[121] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", *arXiv*, 2015, preprint. DOI: 10.48550/arXiv.1409.1556.

[122] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved Techniques for Training GANs", *arXiv*, 2016, preprint. DOI: 10.48550/arXiv.1606.03498.

[123] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium", in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, vol. 30, Long Beach, CA, USA: Curran Associates, Inc., 2017.

[124] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision", *arXiv*, 2015, preprint. DOI: 10.48550/arXiv.1512.00567.

[125] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying MMD GANs", *arXiv*, 2021, preprint. DOI: 10.48550/arXiv.1801.01401.

[126] L. Maier-Hein, A. Reinke, P. Godau, *et al.*, "Metrics reloaded: Recommendations for image analysis validation", *arXiv*, 2023, preprint. DOI: 10.48550/arXiv.2206.01653.

[127] N. Kumar, R. Verma, S. Sharma, S. Bhargava, A. Vahadane, and A. Sethi, "A Dataset and a Technique for Generalized Nuclear Segmentation for Computational Pathology", *IEEE Transactions on Medical Imaging*, vol. 36, no. 7, pp. 1550–1560, 2017. DOI: 10.1109/TMI.2017.2677499.

[128] C. H. Martin, T. (Peng, and M. W. Mahoney, "Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data", *Nature Communications*, vol. 12, no. 1, p. 4122, 2021. DOI: 10.1038/s41467-021-24025-8.

[129] A. Lehmussola, P. Ruusuvuori, J. Selinummi, H. Huttunen, and O. Yli-Harja, "Computational Framework for Simulating Fluorescence Microscope Images With Cell Populations", *IEEE Transactions on Medical Imaging*, vol. 26, no. 7, pp. 1010–1016, 2007. DOI: 10.1109/TMI.2007.896925.

[130] D. Svoboda, M. Kozubek, and S. Stejskal, "Generation of digital phantoms of cell nuclei and simulation of image formation in 3D image cytometry", *Cytometry Part A*, vol. 75A, no. 6, pp. 494–509, 2009. DOI: 10.1002/cyto.a.20714.

[131] T. J. Rudge, P. J. Steiner, A. Phillips, and J. Haseloff, "Computational Modeling of Synthetic Microbial Biofilms", *ACS Synthetic Biology*, vol. 1, no. 8, pp. 345–352, 2012. DOI: 10.1021/sb300031n.

[132] L. Benoit, D. Rousseau, É. Belin, D. Demilly, and F. Chapeau-Blondeau, "Simulation of image acquisition in machine vision dedicated to seedling elongation to validate image processing root segmentation algorithms", *Computers and Electronics in Agriculture*, vol. 104, pp. 84–92, 2014. DOI: 10.1016/j.compag.2014.04.001.

[133] P. Malm, A. Brun, and E. Bengtsson, "Simulation of bright-field microscopy images depicting pap-smear specimen", *Cytometry Part A*, vol. 87, no. 3, pp. 212–226, 2015. DOI: 10.1002/cyto.a.22624.

[134] J. Stegmaier, J. Arz, B. Schott, *et al.*, "Generating semi-synthetic validation benchmarks for embryomics", in *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, Prague, Czech Republic: IEEE, 2016, pp. 684–688. DOI: 10.1109/ISBI.2016.7493359.

[135] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes", in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 3234–3243. DOI: 10.1109/CVPR.2016.352.

[136] R. Barth, J. IJsselmuiden, J. Hemming, and E. J. V. Henten, "Data synthesis methods for semantic segmentation in agriculture: A Capsicum annuum dataset", *Computers and Electronics in Agriculture*, vol. 144, pp. 284–296, 2018. DOI: 10.1016/j.compag.2017.12.001.

[137] M. Scalbert, F. Couzinie-Devy, and R. Fezzani, "Generic Isolated Cell Image Generator", *Cytometry Part A*, vol. 95, no. 11, pp. 1198–1206, 2019. DOI: 10.1002/cyto.a.23899.

[138] R. Sagues-Tanco, L. Benages-Pardo, G. López-Nicolás, and S. Llorente, "Fast Synthetic Dataset for Kitchen Object Segmentation in Deep Learning", *IEEE Access*, vol. 8, pp. 220 496–220 506, 2020. DOI: 10.1109/ACCESS.2020.3043256.

[139] A. Chen, L. Wu, S. Han, P. Salama, K. W. Dunn, and E. J. Delp, "Three Dimensional Synthetic Non-Ellipsoidal Nuclei Volume Generation Using Bézier Curves", in *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, Online: IEEE, 2021, pp. 961–965. DOI: 10.1109/ISBI48211.2021.9434149.

[140] Q. Liu, I. M. Gaeta, M. Zhao, *et al.*, "ASIST: Annotation-free Synthetic Instance Segmentation and Tracking by Adversarial Simulations", *arXiv*, 2021, preprint. DOI: 10.48550/arXiv.2101.00567.

[141] A. Gilbert, M. Marciniak, C. Rodero, P. Lamata, E. Samset, and K. Mcleod, "Generating Synthetic Labeled Data From Existing Anatomical Models: An Example With Echocardiography Segmentation", *IEEE Transactions on Medical Imaging*, vol. 40, no. 10, pp. 2783–2794, 2021. DOI: 10.1109/TMI.2021.3051806.

[142] E. Colleoni, D. Psychogyios, B. Van Amsterdam, F. Vasconcelos, and D. Stoyanov, "SSIS-Seg: Simulation-Supervised Image Synthesis for Surgical Instrument Segmentation", *IEEE Transactions on Medical Imaging*, vol. 41, no. 11, pp. 3074–3086, 2022. DOI: 10.1109/TMI.2022.3178549.

[143] D. Leitner, S. Klepsch, G. Bodner, and A. Schnepf, “A dynamic root system growth model based on L-Systems”, *Plant and Soil*, vol. 332, no. 1, pp. 177–192, 2010. DOI: 10.1007/s11104-010-0284-7.

[144] F. Ambellan, H. Lamecker, C. von Tycowicz, and S. Zachow, “Statistical Shape Models: Understanding and Mastering Variation in Anatomy”, in *Biomedical Visualisation : Volume 3*, P. M. Rea, Ed., Cham, Switzerland: Springer International Publishing, 2019, pp. 67–84. DOI: 10.1007/978-3-030-19385-0_5.

[145] C. T. Zahn and R. Z. Roskies, “Fourier Descriptors for Plane Closed Curves”, *IEEE Transactions on Computers*, vol. C-21, no. 3, pp. 269–281, 1972. DOI: 10.1109/TC.1972.5008949.

[146] F. P. Kuhl and C. R. Giardina, “Elliptic Fourier features of a closed contour”, *Computer Graphics and Image Processing*, vol. 18, no. 3, pp. 236–258, 1982. DOI: 10.1016/0146-664X(82)90034-X.

[147] D. Svoboda and V. Ulman, “Generation of Synthetic Image Datasets for Time-Lapse Fluorescence Microscopy”, in *Proceedings of the 9th International Conference on Image Analysis and Recognition - Volume Part II*, Aveiro, Portugal: Springer, 2012, pp. 473–482. DOI: 10.1007/978-3-642-31298-4_56.

[148] G. A. Fontanelli, M. Selvaggio, M. Ferro, F. Ficuciello, M. Vendittelli, and B. Siciliano, “A V-REP Simulator for the da Vinci Research Kit Robotic Platform”, in *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*, Twente, Netherlands: IEEE, 2018, pp. 1056–1061. DOI: 10.1109/BIOROB.2018.8487187.

[149] M. Cordts, M. Omran, S. Ramos, *et al.*, “The Cityscapes Dataset for Semantic Urban Scene Understanding”, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, Nevada, USA: IEEE, 2016, pp. 3213–3223. DOI: 10.1109/CVPR.2016.350.

[150] V. Ljosa, K. L. Sokolnicki, and A. E. Carpenter, “Annotated high-throughput microscopy image sets for validation”, *Nature Methods*, vol. 9, no. 7, pp. 637–637, 2012. DOI: 10.1038/nmeth.2083.

[151] F. Ji, X. Zhang, and J. Zhao, “α-EGAN: α-Energy distance GAN with an early stopping rule”, *Computer Vision and Image Understanding*, vol. 234, p. 103 748, 2023. DOI: 10.1016/j.cviu.2023.103748.

[152] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild”, in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile: IEEE, 2015, pp. 3730–3738. DOI: 10.1109/ICCV.2015.425.

[154] M. P. Schilling, S. Schmelzer, L. Klinger, and M. Reischl, “KaIDA: A modular tool for assisting image annotation in deep learning:” *Journal of Integrative Bioinformatics*, vol. 19, no. 4, 2022. DOI: 10.1515/jib-2022-0018.

[155] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, “Albumentations: Fast and Flexible Image Augmentations”, *Information*, vol. 11, no. 2, p. 125, 2020, Number: 2 Publisher: Multidisciplinary Digital Publishing Institute. DOI: 10.3390/info11020125.

[156] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, “SciPy 1.0: Fundamental algorithms for scientific computing in Python”, *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

List of Own Publications

This list shows all my journal and conference articles and preprints produced and published while working on this thesis. The items are ordered by their appearance in this thesis.

[33] T. Scherr, K. Streule, A. Bartschat, *et al.*, “BeadNet: Deep learning-based bead detection and counting in low-resolution microscopy images”, *Bioinformatics*, vol. 36, no. 17, pp. 4668–4670, 2020. DOI: 10.1093/bioinformatics/btaa594.

[39] S. Bäuerle, M. Böhland, J. Barth, M. Reischl, A. Steimer, and R. Mikut, “CAD-to-real: Enabling deep neural networks for 3D pose estimation of electronic control units: A transferable and automated approach for industrial use cases”, *at - Automatisierungstechnik*, vol. 69, no. 10, pp. 880–891, 2021. DOI: 10.1515/auto-2021-0020.

[49] M. Böhland, W. Doneit, L. Gröll, R. Mikut, and M. Reischl, “Automated design process for hybrid regression modeling with a one-class SVM”, *at - Automatisierungstechnik*, vol. 67, no. 10, pp. 843–852, 2019. DOI: 10.1515/auto-2019-0013.

[50] M. Böhland, L. Tharun, T. Scherr, *et al.*, “Machine learning methods for automated classification of tumors with papillary thyroid carcinoma-like nuclei: A quantitative analysis”, *PLOS ONE*, vol. 16, no. 9, e0257635, 2021. DOI: 10.1371/journal.pone.0257635.

[54] S. Graham, Q. D. Vu, M. Jahanifar, *et al.*, “CoNIC Challenge: Pushing the frontiers of nuclear detection, segmentation, classification and counting”, *Medical Image Analysis*, vol. 92, p. 103 047, 2024. DOI: 10.1016/j.media.2023.103047.

[55] M. Böhland, O. Neumann, M. P. Schilling, *et al.*, “Ciscnet - a Single-Branch Cell Nucleus Instance Segmentation and Classification Network”, in *2022 IEEE International Symposium on Biomedical Imaging Challenges (ISBIC)*, Kolkata, India: IEEE, 2022, pp. 1–5. DOI: 10.1109/ISBIC56247.2022.9854734.

[86] M. Böhland, T. Scherr, A. Bartschat, R. Mikut, and M. Reischl, “Influence of Synthetic Label Image Object Properties on GAN Supported Segmentation Pipelines”, in *Proceedings - 29. Workshop Computational Intelligence*, Dortmund, Germany: KIT Scientific Publishing, 2019, pp. 289–305. DOI: 10.5445/IR/1000100253.

[89] R. Bruch, F. Keller, M. Böhland, *et al.*, “Synthesis of large scale 3D microscopic images of 3D cell cultures for training and benchmarking”, *PLOS ONE*, vol. 18, no. 3, e0283828, 2023. DOI: 10.1371/journal.pone.0283828.

[113] T. Scherr, K. Löffler, M. Böhland, and R. Mikut, “Cell segmentation and tracking using CNN-based distance predictions and a graph-based matching strategy”, *PLOS ONE*, vol. 15, no. 12, e0243219, 2020. DOI: 10.1371/journal.pone.0243219.

[153] M. Böhland, R. Bruch, K. Löffler, and M. Reischl, “Unsupervised GAN epoch selection for biomedical data synthesis”, in *Current Directions in Biomedical Engineering*, vol. 9, Berlin, Germany: De Gruyter, 2023, pp. 467–470. DOI: 10.1515/cdbme-2023-1117.

[157] M. Böhland, R. Bruch, S. Bäuerle, L. Rettenberger, and M. Reischl, “Improving Generative Adversarial Networks for Patch-Based Unpaired Image-to-Image Translation”, *IEEE Access*, vol. 11, pp. 127 895–127 906, 2023. DOI: 10.1109/ACCESS.2023.3331819.

[158] L. Rettenberger, M. Schilling, S. Elser, M. Böhland, and M. Reischl, “Self-Supervised Learning for Annotation Efficient Biomedical Image Segmentation”, *IEEE Transactions on Biomedical Engineering*, vol. 70, no. 9, pp. 2519–2528, 2023. DOI: 10.1109/TBME.2023.3252889.