Influence of Synthetic Label Image Object Properties on GAN Supported Segmentation Pipelines

Moritz Böhland, Tim Scherr, Andreas Bartschat, Ralf Mikut, Markus Reischl

Institute for Automation and Applied Informatics,
Karlsruhe Institute of Technology, Karlsruhe, Germany
E-Mail: moritz.boehland@kit.edu

1 Introduction

Artificial neural networks (ANNs) are a state-of-the-art method for image segmentation [1, 2, 3]. To train a neural network for a specific segmentation task, a dataset containing images and corresponding label images is required [4]. Usually, label images are annotated manually or semi-automatically. This is time consuming, expensive and prone to errors. A recent approach substitutes manual labeling by a Generative Adversarial Network (GAN) [5, 6, 7, 8, 9, 10, 11, 12, 13]. GANs are able to transform data from one domain into another. For example from the domain of horse images to the domain of zebra images, while preserving the general properties of the image [14]. This approach can be used to create a synthetic training dataset based on real-world images and synthetic label images (which have to be created manually or semi-automatically). The main advantage of this approach is, that during the training of the GAN no direct one-to-one relation between image and label is required. The GAN learns the mapping from the label domain to the image domain and vice versa. During inference, the trained model can be used to create a paired synthetic dataset from the synthetic label images [12, 9, 11]. For example, this can be achieved by the CycleGAN architecture [14].

Since the synthetic label images are part of the synthetic training dataset created by the GAN, they highly influence the subsequent segmentation results. This turns the creation of the synthetic label images into a central point of the method. General guidelines on how to create synthetic label images are already given in [9]. However, no specific approach for synthetic label image generation and no quantitative results regarding the quality of the synthetic label images are provided.

This article examines the influence of synthetic cell nuclei label image properties on the segmentation quality. Therefore, the segmentation results of a GAN and an U-Net trained on the generated synthetic training dataset are compared. Furthermore, insights on how to create high quality synthetic label images for the GAN are given. Analyzing the quantitative relation between the synthetic label image properties and the resulting segmentation quality leads to the identification of the relevant object properties. Focusing on these properties reduces the time necessary for synthetic label images creation.

In Section 2 GANs in general, the CycleGAN and frameworks to create synthetic datasets are discussed. A segmentation pipeline utilizing GANs to create a synthetic dataset as well as possibilities to quantify the influence of synthetic label images on the segmentation pipeline are described in Section 3. The experimental setup and the results are shown in Section 4 and Section 5. Section 6 gives a conclusion and an outlook.

2 State-of-the-Art

2.1 Generative Adversarial Networks

A new framework for neural networks called Generative Adversarial Networks was introduced in 2014 by Goodfellow [15]. In the introduced framework, two models are simultaneously trained. The generator model tries to generate samples that matches a given distribution. The discriminator model tries to distinguish between a sample coming from the generator and a sample coming

from the given distribution. They compete against each other and therefore the training goal with the value function V can be described as

$$\begin{aligned}\min_{\theta_G}\max_{\theta_D} V =& \mathbb{E}_{X\sim p_{data}}[\log D(x,\theta_D)] \\ &+\mathbb{E}_{Z\sim p_z}[\log(1-D(G(z,\theta_G),\theta_D))],\end{aligned} \tag{1}$$

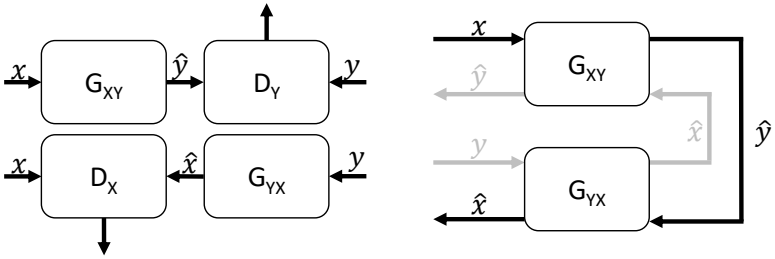
where z is a sample from a noise distribution p_z given to the generator $G(z,\theta_G)$ (with the network parameters θ_G) to generate samples. Respectively x is a sample from the real data distribution p_{data} and $D(x,\theta_D)$ is the discriminator. The output of $D(x,\theta_D)$ is the estimated probability of a sample being from p_{data}. The expected value is denoted by $\mathbb{E}$. When the discriminator is able to distinguish between a sample from the given distribution and the generated distribution, $D(x,\theta_D)$ and $1-D(G(z,\theta_G),\theta_D)$ will be close to 1 and therefore Equation 1 is maximized. When the generator is able to produce samples close to the given distribution and the discriminator is not able to distinguish between both samples, $D(x,\theta_D)$ will be close to 0 and $1-D(G(z,\theta_G),\theta_D)$ will also be close to 0. This leads to the training goals of minimizing Equation 1 for the generator and maximizing it for the discriminator [16].

With the trained generator, additional samples of p_{data} can be generated. This is also possible for highly complex distributions like the distribution of images containing a dog [17, 18]. A drawback is the limited influence on the generated samples, since the generator is only dependent on the random input z.

2.2 CycleGAN

Unlike the framework described in Subsection 2.1, CycleGAN does not create samples from a random input, but is able to perform image-to-image translation which is a generalization of style transfer [16]. In image-to-image translation, the mapping from a source domain of images (a highly complex distribution) to a target domain of images is performed. For example images of horses are transformed to images of zebras, while preserving the overall properties such as number of animals, pose and background. Furthermore, CycleGAN is able to learn the translation in an unpaired manner, therefore no paired training data

is needed (images from the source domain have no corresponding image in the target domain while training) [14].



(a) Generator - discriminator data flow (b) Cycle-consistency

Figure 1: CycleGAN with two generators (G) and discriminators (D) to transform images $x \in X$ to $\hat{y} \in Y$ and vice versa (a). The generators transform the images to the target domain, while the discriminators try to distinguish between generated images and real images from the target domain. To ensure cycle-consistency (generated images match original images), images $\hat{x}$ and $\hat{y}$ are created by passing the corresponding generators (b).

CycleGAN is able to perform unpaired image-to-image translation by extending the original GAN framework. It contains two generators and two discriminators (see Figure 1a). Generator G_{XY} is used to transform images $x \in X$ from the distribution $\mathscr{X} \sim p_x$ to an image $\hat{y}$ of the distribution $\mathscr{Y} \sim p_y$. The distribution $\mathscr{X} \sim p_x$ is approximated by images of the domain X and the distribution $\mathscr{Y} \sim p_y$ is approximated by the images of the domain Y (with images $y \in Y$). The discriminator D_Y is trained to distinguish between images y and images $\hat{y}$ created by G_{XY}. Generator G_{YX} is used to transform images y to the distribution $\mathscr{X} \sim p_x$. The discriminator D_X works accordingly to D_Y. The loss functions can be described equivalent to Equation 1. Therefore, the loss $\mathscr{L}_{GAN}(G_{XY}, D_Y, X, Y)$ for mapping $\mathscr{X} \sim p_x$ to $\mathscr{Y} \sim p_y$ can be defined by

$$\begin{aligned}\mathscr{L}_{GAN}(G_{XY}, D_Y, X, Y) = &\mathbb{E}_{\mathscr{Y} \sim p_y}[\log D_Y(y, \theta_Y)] \\ &+ \mathbb{E}_{\mathscr{X} \sim p_x}[\log(1 - D_Y(G_{XY}(x, \theta_{XY}), \theta_Y))].\end{aligned} \tag{2}$$

The loss for mapping $\mathscr{Y} \sim p_y$ to $\mathscr{X} \sim p_x$ is defined accordingly by $\mathscr{L}_{GAN}(G_{YX}, D_X, Y, X)$.

To further restrain the mapping and force the CycleGAN to preserve important properties of the input image, cycle-consistency is introduced. Otherwise, the input image is only considered as a random input variable by the CycleGAN. As seen in Figure 1b an image x is transformed to $\mathscr{Y} \sim p_y$ by G_{XY} creating image $\hat{y}$. Afterwards the image is transformed back to $\mathscr{X} \sim p_x$ by G_{YX} creating image $\hat{x}$. Ideally x and $\hat{x}$ should be the same. The same is done for the images from images y. Therefore the cycle-consistency loss

$$\begin{aligned} \mathscr{L}_{cyc}(G_{XY}, G_{YX}) = &\mathbb{E}_{\mathscr{X} \sim p_x}[\|G_{YX}(G_{XY}(x)) - x\|] \\ &+ \mathbb{E}_{\mathscr{Y} \sim p_y}[\|G_{XY}(G_{YX}(y)) - y\|] \end{aligned} \tag{3}$$

can be defined. With Equation 2 and Equation 3, the combined loss

$$\begin{aligned} \min_{\theta_{XY}, \theta_{YX}} \max_{\theta_X, \theta_Y} \mathscr{L}(G_{XY}, G_{YX}, D_X, D_Y) = &\mathscr{L}_{GAN}(G_{XY}, D_Y, X, Y) \\ &+ \mathscr{L}_{GAN}(G_{YX}, D_X, Y, X) \\ &+ \lambda \mathscr{L}_{cyc}(G_{XY}, G_{YX}) \end{aligned} \tag{4}$$

can be defined. Again, the discriminators try to maximize Equation 4 (by tuning their parameters θ_X and θ_Y), while the generators try to minimize it (by tuning their parameters θ_{XY} and θ_{YX}). The hyperparameter λ balances the importance of both losses. Further details can be found in [14]. During inference, the generators are able to create paired samples by creating the corresponding images from the target domain. Variations of CycleGAN and GANs able to perform unpaired image-to-image translation can be found in [9, 19, 20, 12, 21].

2.3 Synthetic Dataset Creation

Frameworks to create synthetic datasets are used for training and benchmarking [22, 23, 24, 25]. Datasets containing 3D+t cell images can be created by placing spheres in the images, distorting them to match the variability of the real cells and applying texture. Afterwards, the microscopy image acquisition is simulated by applying a point spread function and different noise sources. This can be done by Cytopacq which is freely available and has an online

benchmark dataset generator available[1] [26, 27]. A different approach does not create the cells by placing spheres, but by extracting a set of cell nuclei form existing images. The cells are placed in a new dataset using cell positions of repaired tracks computed from realistic images [28]. In general, methods simulating the microscopy acquisition have many parameters which need to be tuned to match a specific real-world microscopy image dataset.

Recent approaches try to substitute the simulation of the microscopy acquisition by using different types of GANs [5, 6, 7, 8, 9, 10, 11, 12, 13]. High quality results are achieved for biological data, while simplifying the process drastically. Datasets are created by training the GAN using existing label images. However, there are still open questions: how are the synthetic label images created, which label image properties influence the results as well as how and why they were chosen. Furthermore, the segmentation quality (when evaluating real-world datasets by training with the synthetic dataset) is often compared to different GANs, but not to segmentation networks trained with manually annotated label images.

3 Methods

3.1 Segmentation Pipeline

Segmentation pipelines based on ANNs like the U-Net [3] need paired training samples which is e.g. manually annotated. By the use of CycleGAN, paired training samples can be created without the need for manual annotations. A segmentation pipeline utilizing a CycleGAN for the creation of the training data is shown in Figure 2. However, any GAN able to perform unpaired image-to-image translation can be used.

In the first step of the segmentation pipeline, synthetic label images X need to be created. This can be done by visual inspection of the real-world microscopy images Y or by simulation of the underlying physical model: If the synthetic label images are created by visual inspection, properties describing the labels

[1] Benchmark dataset generator available at `https://cbia.fi.muni.cz/simulator`.

(e.g. the size of objects or the number of objects per image) need to be visually extracted from the microscopy images. These properties can then be used to create synthetic label images. On the other hand, synthetic label images can be created by simulation of a corresponding physical model. For example label images for the segmentation of porous membranes, can be acquired by the simulation of the phase separation of polymer solutions [29].

After creation of the synthetic label images, the CycleGAN needs to be trained with X and Y. The resulting network parameters are used for inference. Inference can be done in two different ways:

1. The CycleGAN is used to directly create the label images $\hat{X}$ from Y.
2. The CycleGAN is used to generate the corresponding synthetic microscopy images $\hat{Y}$ to X. The resulting paired synthetic label and microscopy
   images (X and $\hat{Y}$) are used as a training dataset for a segmentation network (e.g. a U-Net). After training on the synthetic training dataset, the segmentation network is used in inference mode to create the corresponding label images $\hat{X}$ to Y.

3.2 Quantification of the Synthetic Label Image Object Properties Influence

A crucial step in the method described in Subsection 3.1 is the creation of the synthetic label images. If they do not match the properties of the label images of the real microscopy images, CycleGAN learns a wrong mapping and therefore the results are biased. Essential properties for cells are e.g. size or shape. They have to be defined individually for different types of data. If the size of an object is too small in the label image, CycleGAN could learn to increase it. This would result in a synthetic dataset with big cells in the synthetic microscopy images and small cells in the label images. To gain insight into which properties need to be modelled accurately and which properties have no influence on the final segmentation results, the following procedure is introduced:

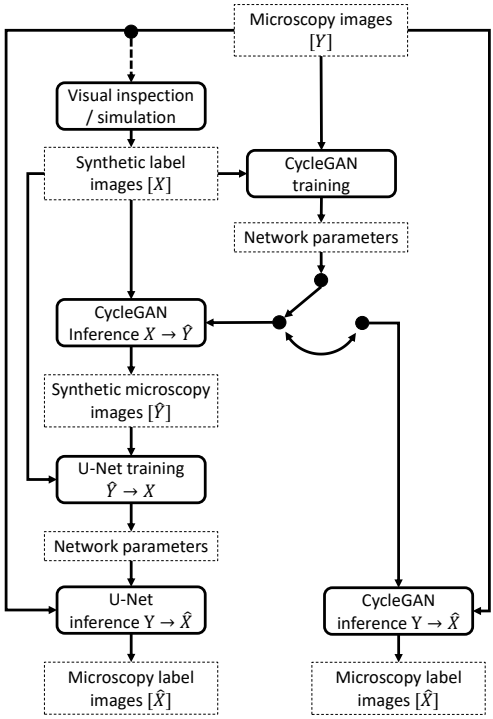


Figure 2: Segmentation pipeline utilizing a CycleGAN. The CycleGAN can directly be used to generate the desired microscopy label images. Furthermore, it can be used to generate a synthetic training dataset for a segmentation network.

1. Extract the real properties from the manually annotated microscopy images (ground truth),
2. create different synthetic label image datasets in which one property varies, while the other properties are fixed,
3. apply the segmentation pipeline from Subsection 3.1 to each of the datasets,
4. evaluate the segmentation results for the different datasets,
5. repeat steps 2-4 for each property.

Although optimization of the properties is a multivariate optimization problem, the proposed univariate optimization can be used to discuss effects. The univariate optimization is done to reduce the computational expense. Property extraction and reducing the amount of relevant properties can be challenging for complex scenarios (e.g. simulation of street scenes). In this study, a relatively easy dataset is used (see Subsection 4.1) to reduce the risk of errors in the property extraction step and have a straightforward insight into the resulting influence of the object properties. However, the pipeline can also be applied to more complex datasets.

4 Experimental Setup

4.1 Dataset

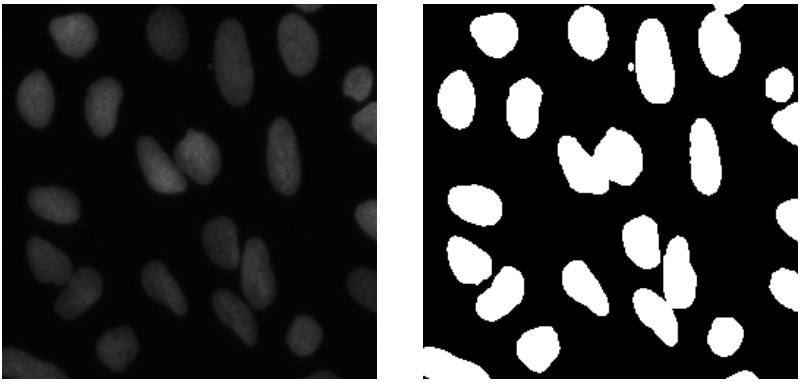
The influence of synthetic label image properties has been evaluated by applying the segmentation pipeline from Subsection 3.1. The BBBC039v1[2] dataset has been used for the experiments [23].

The dataset contains 200 images of U2OS cells acquired by fluorescence microscopy. The images are stored as 520×696 px 16-bit grayscale images. The cell nuclei in the corresponding label images are manually annotated. An exemplary crop with 256×256 px and the corresponding label image are shown in Figure 3. The different sizes and shapes of the cells are visible. It is notable, that ellipses are approximating the cells well, even though the borders of the real cells are distorted.

4.2 Synthetic Label Image Generation

To create the synthetic label images, label image properties have been extracted from the ground truth containing all 200 images. For the used cell dataset, the properties are cells per image (P_C), size of the cells (P_S) (diameter of a circle with the same area) and the eccentricity of ellipse that have the same

[2]The dataset is available at `https://data.broadinstitute.org/bbbc/BBBC039/`.



(a) Microscopy image (b) Corresponding ground truth label image

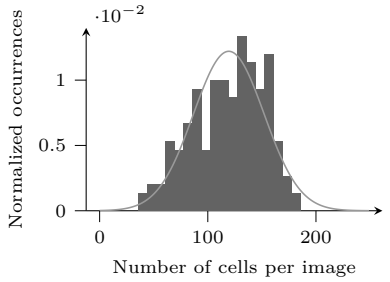
Figure 3: Exemplary 256×256 px crop from a BBBC039v1 cell microscopy image (a) and the corresponding label image (b).

second-moment as the cells (P_E). For each property, the mean and the standard deviation are calculated. To handle outliers, the distributions have been clipped between the 5% and 95% percentiles. Afterwards, each distribution is approximated by a normal distribution. Histograms of the properties and the approximations are shown in Figure 4.

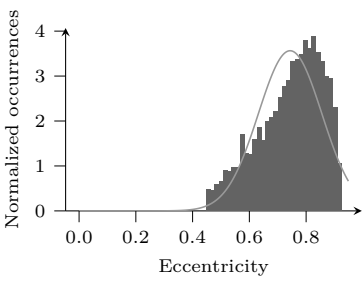
The resulting means and standard deviations of the fitted normal distributions are 120 ± 33 for P_C, 27 ± 6 for P_S and 0.74 ± 0.11 for P_E. Especially for P_S, using a normal distribution does not result in a perfect fit to the data. Nevertheless, it can be used to evaluate the influence of the different properties on the segmentation performance.

A synthetic dataset is created by drawing from the normal distributions. It has to be assured, that no invalid values (e.g. cell size < 0) are drawn. An example with different means for the cell eccentricity is shown in Figure 5.

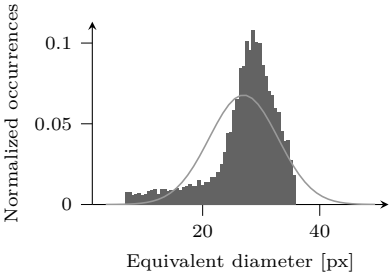
For each image in a dataset the number of cells per image and for each cell in the images the size and the eccentricity are individually drawn from the corresponding distributions. As introduced in Subsection 3.2, different datasets are created by varying the mean of the normal distributions while keeping the standard deviation constant (being the ground truth value). For each property, 8 datasets with the following means:



(a) Cells per image distribution



(b) Cell eccentricity distribution



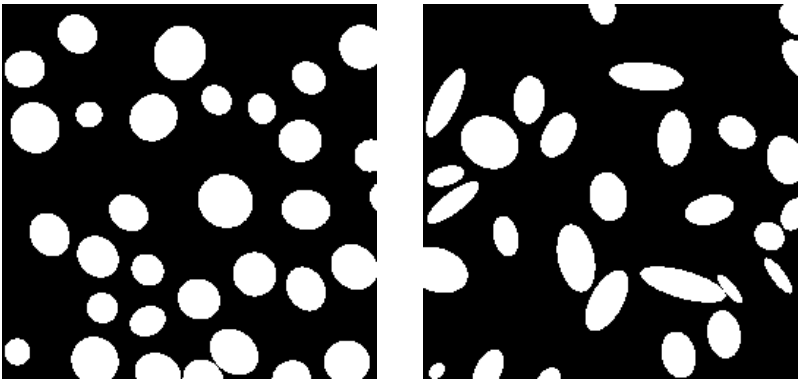
(c) Equivalent cell diameter distribution

Figure 4: Distributions of different image parameters extracted from the ground truth images. The distributions are clipped between the 5% and 95% percentiles and normalized to an area of 1. A normal distribution is fitted.

- $\mu_{P_C} = 40, 60, 80, ..., 180,$
- $\mu_{P_S} = 7, 12, 17, ..., 42$ and
- $\mu_{P_E} = 0.34, 0.44, 0.54, 0.64, 0.69, 0.74, 0.79, 0.84,$

have been created. The mean values were selected to have datasets containing both smaller and larger values than the true mean values.

Each of the datasets contains 560 label images with 520×696 px, which is equivalent to the BBBC039v1 dataset. The dataset is split as follows: 160 images are used for CycleGAN training (alongside 160 of the BBBC039v1 images). The remaining 400 label images are used for CycleGAN inference creating the training dataset for the U-Net.



(a) Ellipses with eccentricity of 0.44 ± 0.11 (b) Ellipses with eccentricity of 0.79 ± 0.11

Figure 5: Comparison of two 256×256 px areas extracted from synthetic datasets with different mean eccentricity. The left image (a) has a mean eccentricity of 0.44 and therefore the ellipses are more round than the ellipses on the right image (b) with a mean of 0.79. Both have a standard deviation of 0.11.

4.3 Neural Network Configurations, Training and Post-Processing

The PyTorch implementation[3] of [14] is used for the CycleGAN. The generators consist of a downsampling block, 9 ResNet blocks and an upsampling block. The discriminators each consist of a 70×70 px PatchGAN [30]. The CycleGAN is trained with random crops (256×256 px) for 200 epochs. For better results, each image of the BBBC039v1 dataset has been scaled to the minimum and maximum intensity in the dataset. The inference is done on the full image, resulting in output images of 520×696 px.

The applied segmentation pipeline consists of a modified U-Net with batch normalization and transposed convolutions. In the output layer the sigmoid activation function is used. The loss function is a weighted sum of Dice loss [31] and binary crossentropy loss. Training is terminated after 100 epochs or

[3]The CycleGAN PyTorch implementation is available at `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`.

when the loss on the validation data has not improved for 10 epochs. Post-processing is done by binarizing the network output with a threshold, applying the Euclidean distance transform and thresholding the Euclidean distance transform to create seeds. The seeds and the binarized network output are used to create an instance segmentation (touching cells can be distinguished) with a watershed algorithm. The dataset created by the CycleGAN is used (synthetic binary label images and synthetic microscopy images) for training. The 400 images are split in 80% training (320 images) and 20% validation (80 images).

Furthermore, a reference segmentation is acquired by training the U-Net with 60 of the images used to train the GAN and the manually annotated label images. This represents the best performance achievable by manual labeling with the applied U-Net.

The 40 images from the BBBC039v1 dataset (and the corresponding manually annotated label images) not used during CycleGAN training (respectively training of the reference segmentation) are used for evaluation. When using the CycleGAN for processing the 40 evaluation images instead of the U-Net (see Subsection 3.1), the same post-processing as for the U-Net is applied.

5 Results

The performance of the segmentation is measured by the mean average precision at different intersection over union thresholds (P_{IoU}) and the F-Score (true positive: predicted cell matches ground truth cell with $\mathrm{IoU} > 0.5$) [32, 33]. The results for P_C, P_S and P_E are shown in Figure 6. For each property the performance of the segmentation with the CycleGAN, the CycleGAN and U-Net (CG+U-Net) and the reference segmentation are compared.

For the mean number of cells, the U-Net performs well near the mean value of the dataset (120 cells per image). It performs even better if the number of cells is slightly higher, but falls back, if the number is way higher. When compared to Figure 4c it is notable, that the maximum of the histogram of the data is also higher, than the mean of the approximated normal distribution. When μ_{P_C}

is too low, the P_{IoU} decreases drastically. This is even more visible for the F-Score. While the U-Net performs better for values in the range of the real distribution, the CycleGAN performs better for very small values of μ_{P_C}. It is notable, that the U-Net trained with the CycleGAN is able to outperform the reference for $\mu_{P_C} = (140, 160)$. Although, it has to be further evaluated whether this is significant (see Section 6).

For the mean equivalent circle diameter, the performance of the P_{IoU} drops quickly as the mean moves away from the optimum. The difference between the U-Net segmentation and the CycleGAN segmentation is especially notable in areas around the optimum. The F-Score does not drop as quickly as the P_{IoU}. This is due to the fact, that the P_{IoU} depends heavily on the correct size estimation.

Also for the mean cell eccentricity, the F-Score is less dependent on the correct choice of the mean value. Only for very small values, it drops rapidly. This could also be due to the dependence on the correct size estimation. The maximum performance is achieved for $\mu_{P_E} = 0.74$. That corresponds to the mean of the normal distribution fitted to the real data.

The mean P_{IoU} of the best performing CG+U-Net segmentation for each of the properties is 0.64 while it is 0.46 for the CycleGAN segmentation. This encourages usage, of the more complex CG+U-Net segmentation. As shown in Figure 7, the visual quality of the synthetic images is high. Even for synthetic label images with $P_S = 12 \pm 6$, viable images are created by the CycleGAN. It is obvious, that a U-Net, trained with the dataset shown in the third row in Figure 7, will fail to produce good results, since all labels are too small. This underlines the results from Figure 6c.

6 Conclusions and Outlook

In this article, the influence of object properties of synthetic label images on a segmentation pipeline using a CycleGAN instead of manually annotated label images has been examined. As an example, the BBBC039v1 dataset with 200 images of U2OS cells acquired by fluorescence microscopy has been used.

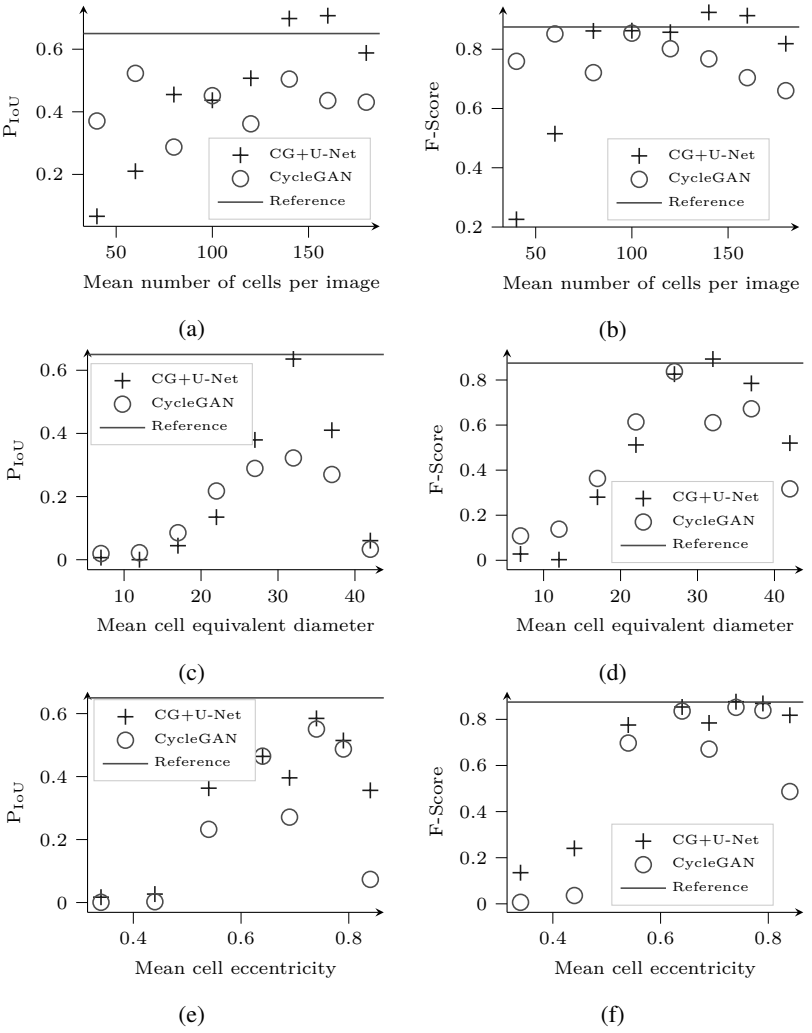


Figure 6: Resulting performance for variation of properties in the synthetic label images. Variation of the mean number of cells per image is shown in (a, b). Variation of the mean equivalent diameter of a circle is shown in (c, d) and variation of the mean cell eccentricity is shown in (e, f). The P_{IoU} is used as a performance measurement in (a, c, e) and the F-Score is used in (b, d, f).

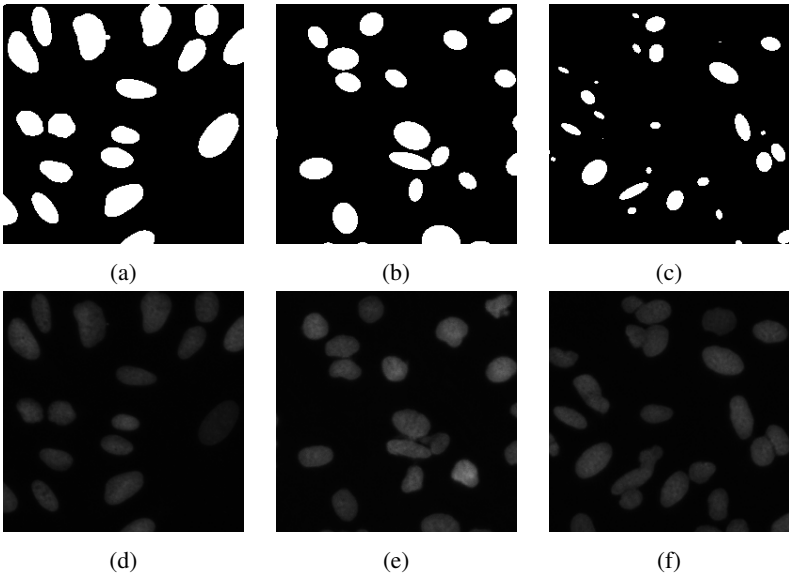


Figure 7: The left column shows a crop from a BBBC039v1 microscopy image with 256×256 px (d) and the corresponding label image (a). The middle column (b, e) shows a synthetic label image and the corresponding synthetic microscopy image from a synthetic dataset with normal distributions $P_C = 120 \pm 33$, $P_S = 27 \pm 6$ and $P_E = 0.74 \pm 0.11$. For the right column (c, f), the mean diameter P_S is changed to 12 ± 6.

The following conclusions can be drawn:

1. The quality of the segmentation results is highly dependent on the label image properties. Choosing the wrong parameters, results in unusable results.

2. A U-Net trained with a synthetic dataset generated by a GAN can perform as well as a U-Net trained with real-world manually annotated data. However, this has to be further examined regarding significance and application on other datasets.

3. The effort that has to be put into the synthetic image generation depends strongly on the result to be achieved. For counting, less effort has to be invested, than for generating knowledge about the exact shape.

4. It is worth the computational expense to train a U-Net for segmentation and not use the CycleGAN itself.

5. It is possible to achieve high quality results, without further investment into the GAN training data (e.g. apply noise on the synthetic label images). The GAN is able to reproduce the additional properties of the microscopy image (e.g. noise and illumination) through the properties of the label image itself.

The results show quantitatively, that the introduced pipeline is able to be used in scientific image analysis, without having to sacrifice performance. Therefore, the method enables scientists to reduce the time needed to evaluate datasets by reducing manual annotation. Choosing the right object size seems to be most important and time should be invested to estimate it right. However, it must be investigated individually whether it is worth to manually annotate the microscopy images or to use the GAN supported segmentation pipeline. Especially for 3D or 3D+t datasets, where manual annotation is extremely complex, the GAN supported segmentation pipeline could be useful [34, 35].

Besides the results for this real-world cell dataset, several open questions remain. Choosing the right distributions to create the synthetic dataset is challenging without having the manually annotated label images available. To solve that problem, e.g. a grid-search could be applied. This reduces the time needed by the scientist but strongly increases the computational load. The GAN might not be able to transfer label images to highly complex images (e.g. using the method for street scenes). Therefore the usability has to be explored quantitatively for different scientific areas. There are more advanced methods to train a U-Net than training with the binary label images such as training with cell borders [33]. It has to be examined whether the introduced pipeline benefits from that as much as training a U-Net with the manually annotated images.

References

[1] K. He et al. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

[2] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, 2015.

[3] 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*, pp. 234–241, Cham, 2015. Springer International Publishing.

[4] A. Gupta et al. Deep Learning in Image Cytometry: A Review. *Cytometry Part A*, 95(4): pp. 366–380, 2019.

[5] T. de Bel et al. 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*, volume 102, pp. 151–163, 2019.

[6] A. Osokin et al. GANs for Biological Image Synthesis. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2252–2261, 2017.

[7] L. Hou et al. Robust Histopathology Image Analysis: To Label or to Synthesize? In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[8] P. Welander, S. Karlsson, and A. Eklund. Generative Adversarial Networks for Image-to-Image Translation on Multi-Contrast MR Images - A Comparison of CycleGAN and UNIT, 2018. arXiv: 1806.07777.

[9] I. J. Stephan et al. UDCT: Unsupervised Data to Content Transformation with Histogram-Matching Cycle-Consistent Generative Adversarial Networks, 2019. bioRxiv: 563734.

[10] Z. Zhang, L. Yang, and Y. Zheng. Translating and Segmenting Multimodal Medical Volumes with Cycle- and Shape-Consistency Generative Adversarial Network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[11] Y. Huo et al. Adversarial Synthesis Learning Enables Segmentation without Target Modality Ground Truth. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pp. 1217–1220, 2018.

[12] C. Fu et al. Three Dimensional Fluorescence Microscopy Image Synthesis and Segmentation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2018.

[13] S. Han et al. Nuclei Counting in Microscopy Images with Three Dimensional Generative Adversarial Networks. In *Medical Imaging 2019: Image Processing*, volume 10949, p. 105, 2019.

[14] J.-Y. Zhu et al. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[15] I. Goodfellow et al. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. 2014.

[16] H. Huang, P. S. Yu, and C. Wang. An Introduction to Image Synthesis with Generative Adversarial Nets, 2018. arXiv: 1803.04469.

[17] H. Zhang et al. Self-Attention Generative Adversarial Networks, 2018. arXiv: 1805.08318.

[18] A. Brock, J. Donahue, and K. Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis, 2018. arXiv: 1809.11096.

[19] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised Image-to-Image Translation Networks. In *Advances in Neural Information Processing Systems 30*, pp. 700–708. 2017.

[20] Z. Yi et al. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[21] S. Benaim and L. Wolf. One-Sided Unsupervised Domain Mapping. In *Advances in Neural Information Processing Systems 30*, pp. 752–762. Curran Associates, Inc., 2017.

[22] V. Ulman et al. An Objective Comparison of Cell-Tracking Algorithms. *Nature Methods*, 14: pp. 1141–1152, 2017.

[23] V. Ljosa, K. L. Sokolnicki, and A. E. Carpenter. Annotated High-Throughput Microscopy Image Sets for Validation. *Nature Methods*, 9: p. 637, 2012.

[24] S. Sankaranarayanan et al. Learning from Synthetic Data: Addressing Domain Shift for Semantic Segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[25] J. Tremblay et al. Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.

[26] D. Wiesner et al. CytoPacq: A Web-Interface for Simulating Multi-Dimensional Cell Imaging. *Bioinformatics*, 2019.

[27] 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*, 75A(6): pp. 494–509, 2009.

[28] J. Stegmaier et al. Generating Semi-Synthetic Validation Benchmarks for Embryomics. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pp. 684–688, 2016.

[29] F. Wang et al. Progress Report on Phase Separation in Polymer Solutions. *Advanced Materials*, 31(26): p.1806733, 2019.

[30] P. Isola et al. Image-to-Image Translation with Conditional Adversarial Networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[31] F. Milletari, N. Navab, and S. Ahmadi. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 565–571, 2016.

[32] Kaggle. 2018 Data Science Bowl, 2018.

[33] T. Scherr et al. Best Practices in Deep Learning-Based Segmentation of Microscopy Images. In *Proc., 28. Workshop Computational Intelligence, Dortmund, 2018*, pp. 175 – 195, 2018.

[34] K. A. Philbrick et al. RIL-Contour: A Medical Imaging Dataset Annotation Tool for and with Deep Learning. *Journal of Digital Imaging*, 32(4): pp. 571–581, 2019.

[35] A. Y. Kobitski et al. An Ensemble-Averaged, Cell Density-Based Digital Model of Zebrafish Embryo Development Derived from Light-Sheet Microscopy Data with Single-Cell Resolution. *Scientific Reports*, 5: p. 8601, 2015.