Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Improving generative adversarial networks for patch-based unpaired image-to-image translation

MORITZ BÖHLAND[1], ROMAN BRUCH[1], SIMON BÄUERLE[1], LUCA RETTENBERGER[1],and MARKUS REISCHL[1]

[1]Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Eggenstein- Leopoldshafen, Germany

Corresponding author: Moritz Böhland (e-mail: moritz.boehland@kit.edu).

ABSTRACT Deep learning models for image segmentation achieve high-quality results, but need large amounts of training data. Training data is primarily annotated manually, which is time-consuming and often not feasible for large-scale 2D and 3D images. Manual annotation can be reduced using synthetic training data generated by generative adversarial networks that perform unpaired image-to-image translation. As of now, large images need to be processed patch-wise during inference, resulting in local artifacts in border regions after merging the individual patches. To reduce these artifacts, we propose a new method that integrates overlapping patches into the training process. We incorporated our method into CycleGAN and tested it on our new 2D tiling strategy benchmark dataset. The results show that the artifacts are reduced by 85% compared to state-of-the-art weighted tiling. Additionally, we demonstrate transferability to real-world 3D biological image data, receiving a high-quality synthetic dataset. Increasing the quality of synthetic training datasets can reduce manual annotation, increase the quality of model output, and can help develop and evaluate deep learning models.

INDEX TERMS GAN, unpaired image-to-image translation, 3D image synthesis, stitching, CycleGAN, tiling, large-scale

I. INTRODUCTION

Supervised deep learning models provide high-quality results for image segmentation tasks. They are often used for instance segmentation of biological, biomedical and material science data [1]–[10]. Training supervised deep learning models requires large amounts of training data, mostly annotated manually. Manual annotation is a tedious and time-consuming task. Great effort is being made to reduce manual annotation, resulting in semi-supervised methods, sparse annotations, and assisted labeling [11]–[14].

In contrast to these methods, synthetic training data can be used. In the past, synthetic training data was created mainly by physical simulation [15], [16]. A framework to create realistic synthetic bright-field microscopy images has been developed to omit manual labeling [16]. The framework was part of the development of a new generation of a cervical cancer screening system. An easy-to-use, modern, and modular web interface was developed to simulate various fluorescence microscopy systems in [15]. It reduces the installation and configuration barrier of existing tools. The downside of physical simulation is the expert domain knowledge needed to create high-quality results. Additionally, quality is reduced by unknown physical processes and approximations. On the other hand, simulation provides explainability and interpretability if needed.

Synthetic training data can also be created with a small amount of domain knowledge and neural networks (NNs) that perform unpaired (unsupervised) image-to-image translation. The neural networks learn to transform images x from source domain $\mathcal{X}$ to images y in the target domain $\mathcal{Y}$. The transformation is often learned in both directions. When synthetic label images are used for one domain and the real-world images are used for the other domain, the NN learns to transfer between both domains. After training, paired synthetic training data can be synthesized from the synthetic label images.

Unpaired image-to-image translation can be performed with Energy-Based Models (EBMs). An EBM parametrized by neural networks, trained by Markov Chain Monte Carlo (MCMC) sampling-based maximum likelihood estimation

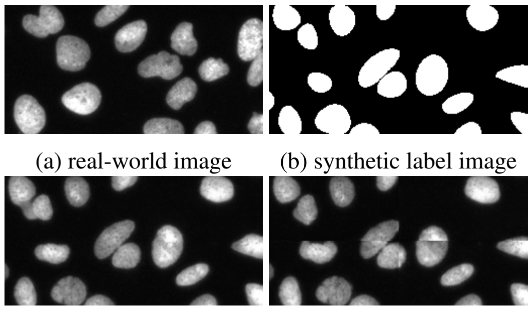


FIGURE 1: GANs trained with real-world images (a) and synthetic label images (b) are able to predict high quality images (c) from (b). If the prediction is performed patch-based, errors occur at the patch borders. This is shown in (d) with four patches used for the prediction. The real-world image is a crop from the BBBC039v1 dataset [33].

has been developed [17]. The problems of instability and the lack of diversity have been solved with a coarse-to-fine image generation, increasing image resolution by expanding the energy function throughout training. Later, an EBM with a multidimensional latent space and a pretrained autoencoder was introduced to further increase the quality of image translation [18].

In addition to EBMs, generative adversarial networks (GANs) like CycleGAN, UNIT, DRIT++ or others can perform unpaired image-to-image translation [19]–[26]. Many GANs for unpaired image-to-image translation consist of one or more generators, transforming data between the domains and one or more discriminators evaluating the authenticity of the generated images. Furthermore, the cycle-consistency constraint introduced in CycleGAN enforces that an image translated from one domain to another and then back should closely resemble the original image, guiding the generators to learn meaningful mappings while reducing the need for paired training data. Cycle-consistency can also be enforced implicitly by a shared latent space used in UNIT. For a thorough description of the different architectures or an introduction to GANs, refer to [27].

While EBMs have mostly been applied to perform unpaired image-to-image tasks like the translation between cats and dogs or oranges and apples, GANs have already been used to create synthetic 2D, and 3D training data from unpaired synthetic label images and real-world images [8], [28]–[32].

When researchers decide to use GANs to synthesize training data, they must deal with the large amount of VRAM required. When the available VRAM is too small for a large-scale 2D or 3D image, and the resolution cannot be reduced, training and inference must be performed patch-wise. For inference, different tiling strategies can be applied. A naive tiling strategy creates patches without overlap, and each patch is processed individually by the GAN. While the mapping for individual patches is correct, errors at patch boundaries in the final image occur. Objects present in multiple patches often inherit a sharp transition in texture, lightning condition, and color pattern. These errors especially appear when there is no direct one-to-one mapping between images in the source domain and the target domain, but a one-to-many mapping. The one-to-many mapping exists due to low entropy in the input image domain and high entropy in the output image domain [34]. An example for the errors introduced when predicting microscopy images of cell nuclei, is shown in Fig. 1. While the prediction without tiling yields virtually no errors, the patch-based prediction with tiling yields errors at the patch borders. Another well-known example is the edges-to-shoes setting, where a GAN is trained to create pictures of shoes solely from the edges of the shoe [19]. Low entropy edges of a shoe can match multiple drawings of a shoe, e.g., different colors, laces, or soles. When processing the edge image patch-wise, there is no guarantee that the GAN infers the same color, laces, and sole from the low entropy input domain to the high entropy output domain for all patches.

Advanced tiling strategies have been developed to reduce these errors without adding more domain knowledge to the synthetic label image domain. Bel et al. [35] use a CycleGAN to adapt histopathological image staining between centers. They introduced a tiling strategy to reduce the tiling artifacts of simple tiling. They made several adaptations to simple tiling: (i) Large overlapping tiles are processed. This increases the similarity of adjacent patches' mean and standard deviation during inference. Therefore, when using standard instance normalization output is also more similar. (ii) Overlapping patches are cropped after being processed by the GAN to reduce border effects introduced by padding and the difference in the receptive field for border pixels compared to pixels in the middle of the patch. (iii) The cropped patches still overlap, and the overlapping patches are stitched together with a weight map to ensure a smooth transition from one patch to the next. While the advanced tiling strategy has been proven to produce high-quality outputs and no changes in GAN training are needed, two drawbacks occur: (i) The large overlap used increases single-dimensional execution time nearly by a factor of four, while inference time scales exponentially with the number of input dimensions. For 3D data, this results in an increase of inference time by 64 compared to naive tiling. (ii) a one-to-one mapping and a reasonable output for an object present in two adjacent patches can still not be ensured. This can still lead to errors in the final image.

II. TILING STRATEGY BENCHMARK DATASET

On a single patch level, methods able to adjust the output for one-to-many mappings exist [23], [25]. These multimodal GANs map an image x to many different correct versions of an output image $\hat{y}$. This is, e.g., done by injecting random noise into the generator or drawing a random style code from a style encoding feature space. Although one can adjust the single patch output for multimodal GANs, consistency

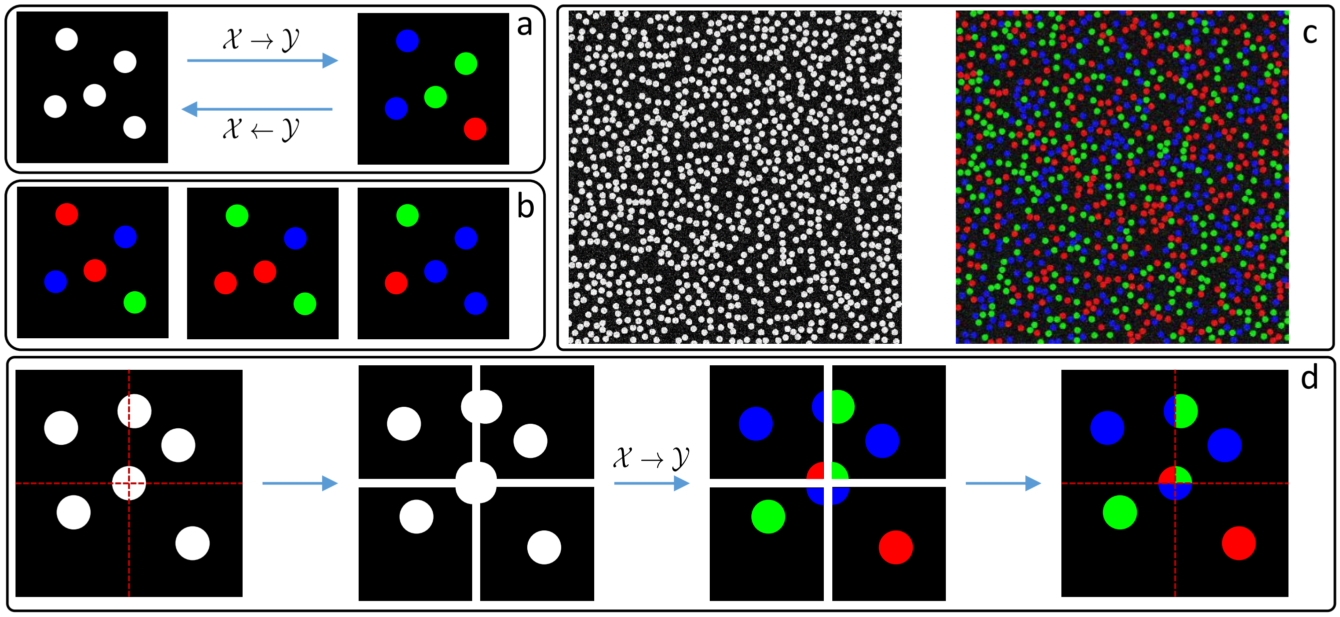

$\mathcal{X} \rightarrow \mathcal{Y}$
$\mathcal{X} \leftarrow \mathcal{Y}$
a
b
c
d
$\mathcal{X} \rightarrow \mathcal{Y}$

FIGURE 2: a: *Tiling strategy benchmark dataset* with problem setting transferring images between domain $\mathcal{X}$ and $\mathcal{Y}$. b: Visualization of the one-to-many translation. The image from domain $\mathcal{X}$ in (a) can lead to all images from domain $\mathcal{Y}$ in (b). c: An example image for each domain of the final *tiling strategy benchmark dataset*. The dataset consists of 512 unpaired images in each domain. In each image of size 2048 px $\times$ 2048 px 1000 circles with diameter 40 px are present. d: Simple tiling into four patches, piecewise inference, and stitching. The individual patch predictions are correct, while errors at the patch borders occur in the final image. The errors at the patch borders in the final image can be used to evaluate different tiling strategies.

across patches can not be guaranteed when processing an image with a tiling strategy. A straightforward strategy to use multimodal GANs to process large-scale images patch-wise would be to create multiple patches until the next patch matches the previously created patches. For real-world images, this is not feasible because oftentimes, there are no automated measures to decide whether the next patch does match the previous patches or not.

Neither existing multimodal GANs nor existing tiling strategies are able to omit the errors introduced during patch-based inference completely. This shows the need for improvement. Because GANs are complex architectures and do not work out-of-the-box for different problem settings, the adoption of new architectures is slow. Therefore, instead of developing a new GAN architecture, we developed a new tiling strategy, which can be directly incorporated into GAN training to further decrease the errors introduced during patch-based inference.

Our tiling strategy enables the GAN to incorporate adjacent patch information into the prediction of the next patch. The tiling strategy allows the GAN to produce arbitrary-sized high-quality images while inference time is reduced compared to existing tiling strategies. Our contributions are as follows: (i) We introduce a *tiling strategy benchmark dataset* to quantitatively compare tiling strategies for GANs, (ii) we show and quantify errors of advanced tiling strategies, (iii) we introduce our new *Stitching Aware Training and Inference (SATI)* to reduce tiling errors and give quantitative results and (iv) we apply our method to a real-world 3D biological dataset.

The *tiling strategy benchmark dataset* created to compare tiling strategies is introduced in Section II. Afterwards, we present our method and show how we incorporated it into the CycleGAN architecture in Section III. In Section IV, quantitative results on the benchmark dataset are shown. Furthermore, we applied our method to a real-world 3D microscopy dataset and present qualitative results. Finally, we discuss our work in Section V and summarize our findings together with an outlook for future work in Section VI.

For real-world images, errors occurring during patch-based inference are manifold and vary depending on the images in both domains. Therefore, visual assessment and error quantification are often not possible. To enable both, the *tiling strategy benchmark dataset* is introduced.

We used a coloring task for the *tiling strategy benchmark dataset*. The task is exemplarily shown in Fig. 2a. Each white circle in domain $\mathcal{X}$ is colored in red, blue, or green in domain $\mathcal{Y}$. Since no color information is present in $\mathcal{X}$, transformation $\mathcal{X} \rightarrow \mathcal{Y}$ is a one-to-many mapping. Different mappings are shown in Fig. 2b. There is no dependency between different circles in domain $\mathcal{Y}$. This aligns with many image-to-image translation tasks. For example, the styles of two cars are not dependent when transferring labels to photos of street scenes.

To create a more diverse dataset and assure that many

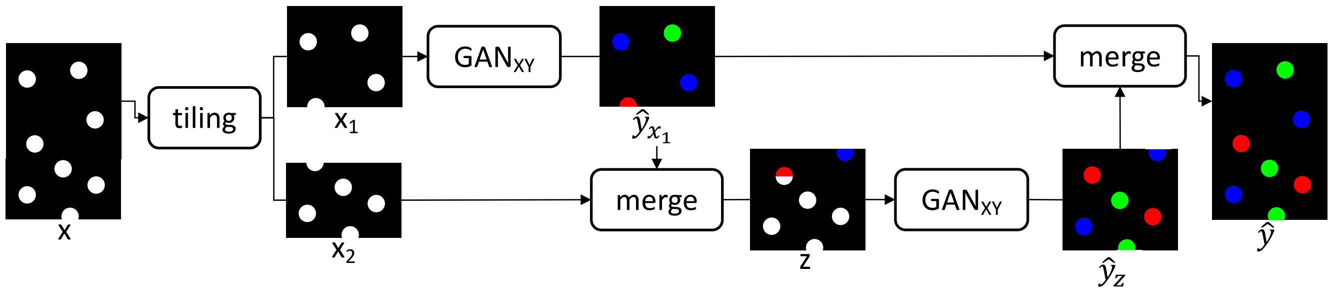


FIGURE 3: Stitching aware inference workflow. An image x too large to be processed by GAN$_{XY}$ without tiling is predicted patch-wise. The image z is created to provide the GAN with context of the previous prediction $\hat{y}_{x_1}$. This enables the GAN to predict the correct color (red) for the circle on the top left of x_2. Finally, $\hat{y}_{x_1}$ and $\hat{y}_z$ are merged to $\hat{y}$.

circles are at patch borders regardless of patch size, we scaled the problem to images with size 2048 px × 2048 px and placed 1000 circles with diameter 40 px on each image. Afterwards, grayscale Gaussian noise is added to domain $\mathcal{X}$, and channel-wise Gaussian noise is added to domain $\mathcal{Y}$. Images in domain $\mathcal{X}$ are afterwards converted to RGB color space, to match the generator input dimensions. All images are encoded with 8-bit for each color channel. A total of 512 unpaired images are created for each domain. Exemplary images for each domain are shown in Fig. 2c.

The simple output domain allows easy visual analysis and computational quantification of the errors introduced during patch-based inference. A circle is predicted correctly, if it consists exclusively of one of the colors red, green or blue. An erroneous circle is present, when multiple colors are in the circle. Because Gaussian noise is present in the images and the images are encoded with 8-bit per color channel, we cannot use simple thresholding to detect the presence of a color. Instead, we check whether a connected area of more than 30 px^2 with color values above the threshold of 60 exists. This is done for each color channel individually. We choose these values to ensure no areas are selected due to the Gaussian noise while being able to identify small mistakes.

III. METHOD

Our new method integrates information from previous predictions into the training and inference process for data with a one-to-many mapping. With this information, GANs are able to infer accurate results for consecutive patches during patch-based prediction. We call this method *Stitching Aware Training and Inference (SATI)*. In this section, we present *SATI* together with the adaptations we made. First, we introduce overlap sampling, domain encoding, and loss ramping. Finally, the inference stitching strategy optimized for our approach and the pixel overlap weighting is introduced. An implementation to create the benchmark dataset and conduct the experiments is available at https://github.com/MoritzBoe/patch_based_image_translation.git.

A. STITCHING AWARE TRAINING AND INFERENCE

When training a standard unpaired image-to-image translation GAN on a one-to-many dataset for $\mathcal{X} \rightarrow \mathcal{Y}$, the GAN will reduce the problem to a one-to-one mapping. After training, an image x will be matched to an image $\hat{y}$. A different image $\hat{y}$ can only be acquired when retraining with modified network initialization or hyperparameters. Based on the premise that the GAN can learn the mapping, the output for a single image will always be a correct prediction from the target domain. When an image is processed patch-wise with a tiling strategy, each patch is still a correct prediction from the target domain. However, errors arise when an object is visible in two or more patches (see Fig. 2d).

We solve this problem by adding information about adjacent patches when single patches are processed during inference (see Fig. 3). Adding all adjacent patches to the input vastly increases input size and is not feasible. Instead, we process overlapping patches with areas already predicted from domain $\mathcal{Y}$ and new areas from domain $\mathcal{X}$. In the example in Fig. 3, two patches are needed to process the entire input image x. Patches are created by tiling. The patch x_1 is processed by the GAN and $\hat{y}_{x_1}$ is synthesized. Subsequently, the bottom part of $\hat{y}_{x_1}$ merges to the top of x_2. A new image z consisting of both domains is created. We call this new domain $\mathcal{Z}$. The same network that transforms images from $\mathcal{X}$ to $\mathcal{Y}$ is used to transform $\mathcal{Z}$ to $\mathcal{Y}$, and $\hat{y}_z$ is created from z. Adding the already predicted areas from $\hat{y}_{x_1}$ to x_2 enables the GAN to continue the prediction of the circle on the bottom left in $\hat{y}_{x_1}$, which is on the top left of x_2, in the correct color (red). Finally, $\hat{y}_z$ and $\hat{y}_{x_1}$ merge into $\hat{y}$.

In contrast to standard GAN inference, our inference workflow adds images consisting of both domains $\mathcal{X}$ and $\mathcal{Y}$ to the process. Therefore, standard GAN training has to be adapted to handle the domain transfer $\mathcal{Z} \rightarrow \mathcal{Y}$. This transfer has two constraints: (i) Areas in $\mathcal{Z}$ which are already from $\mathcal{Y}$ need to stay constant, and (ii) areas in $\mathcal{Z}$ which are from $\mathcal{X}$ need to be transferred to $\mathcal{Y}$ with respect to the areas from $\mathcal{Y}$ present in $\mathcal{Z}$.

To meet both constraints, we added the procedure depicted in Fig. 4 to the training. The GAN transfers an image x to $\hat{y}_x$. Afterwards, a merged image z is created, where border

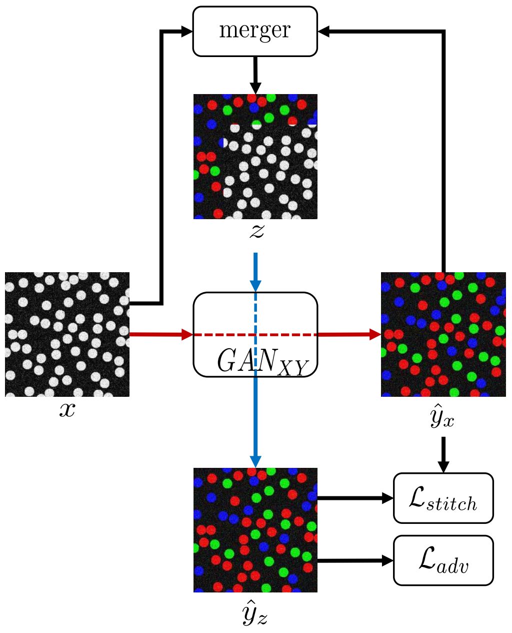


FIGURE 4: Training procedure added to the standard training. After transferring an image x to $\hat{y}_x$, both are merged to z and transferred again to $\hat{y}_z$. Afterwards, the adversarial loss and the stitching loss are calculated. The red and blue line illustrates the information flow through the GAN. The patch size used for x during training can be adapted to the available VRAM.

regions of x are replaced with parts of $\hat{y}_x$. The merged image z is processed again by the GAN to create $\hat{y}_z$. A GAN able to perform the translation of an image x to $\hat{y}_x$ and an image z to $\hat{y}_z$, can perform the inference workflow depicted in Figure 3.

Two new loss functions $\mathcal{L}_{stitch}$ and $\mathcal{L}_{adv}^{\mathcal{ZY}}$ are introduced to enable the transfer from z to $\hat{y}_z$ and to meet the required constraints. The first loss function $\mathcal{L}_{stitch}$ ensures, that the pixels from $\hat{y}_x$ present in z stay constant after transfer to $\hat{y}_z$. We use the mean squared error as a loss function. To enable the transfer of pixels from domain $\mathcal{X}$ present in z to the image $\hat{y}_z$ and therefore to domain $\mathcal{Y}$, these areas are excluded from $\mathcal{L}_{stitch}$. Subsequently, the stitching loss $\mathcal{L}_{stitch}$ is defined by:

$$\mathcal{L}_{stitch} = \mathbb{E}_{z\sim p_{data}(z)}[||\text{GAN}_{XY}(z)(M) - z(M)||_2], \quad (1)$$

where M corresponds to the indices of all pixels from $\hat{y}_x$ in z and $\text{GAN}_{XY}(z)$ to $\hat{y}_z$.

The $\mathcal{L}_{adv}^{\mathcal{ZY}}$ is used to ensure the overall quality of $\hat{y}_z$. A discriminator D_Y trained to differentiate between real images from domain $\mathcal{Y}$ and synthetic images ($\hat{y}_x$ and $\hat{y}_z$) is needed. Most GANs like CycleGAN, UNIT or DRIT++ have a discriminator D_Y. Otherwise, D_Y can be added to the architecture. $\mathcal{L}_{adv}^{\mathcal{ZY}}$ can be defined as follows:

$$\mathcal{L}_{adv}^{\mathcal{ZY}} = \mathbb{E}_{z\sim p_{data}(z)}[||1 - D_Y(\text{GAN}_{XY}(z))||_2]. \quad (2)$$

Errors like the ones shown in Fig. 2c will be detected by the discriminator, and therefore, the generator is trained to omit these errors.

In addition to Fig. 4, the pseudocode for a training step using *SATI* is depicted in Alg. 1. A random image from each domain is required for a training step. The steps to calculate the standard CycleGAN losses for the generators $\mathcal{L}_{\text{G}}^{\text{CycleGAN}}$ and the discriminators $\mathcal{L}_{\text{D}}^{\text{CycleGAN}}$ are not shown for simplicity. The CycleGAN generator that transfers images from $\mathcal{X}$ to $\mathcal{Y}$ is expressed by Gen_{XY} and the discriminator trained to differ between real and generated images from $\mathcal{Y}$ by D_Y. Finally, λ_{stitch} is a scaling factor to vary the influence of $\mathcal{L}_{\text{stitch}}$ on the training of the generator.

Algorithm 1 Pseudocode for a training step using *SATI* integrated into CycleGAN. The steps needed to calculate $\mathcal{L}_{stitch}$ and $\mathcal{L}_{adv}^{\mathcal{ZY}}$ are shown. Calculations of standard CycleGAN losses are not included for simplicity. Comments are marked with #.

```
Require: $x, y$
    $\hat{y}_x \leftarrow \text{Gen}_{XY}(x)$
    # Generator training:
    $z, M \leftarrow \text{merger}(x, \hat{y}_x)$ # $M \leftarrow$ indices of px from $\hat{y}_x$ in $z$
    $\hat{y}_z \leftarrow \text{Gen}_{XY}(z)$
    $\mathcal{L}_{\text{adv}}^{\mathcal{ZY}} \leftarrow ||1 - \text{D}_Y(\hat{y}_z)||_2$
    $\mathcal{L}_{\text{stitch}} \leftarrow ||\hat{y}_z(M) - z(M)||_2$
    $\mathcal{L}_{\text{SATI}} \leftarrow \mathcal{L}_{\text{adv}}^{\mathcal{ZY}} + \lambda_{\text{stitch}}\mathcal{L}_{\text{stitch}}$
    $\text{optimizer}_{\text{Gen}}(\mathcal{L}_{\text{G}}^{\text{CycleGAN}} + \mathcal{L}_{\text{SATI}})$
    # Discriminator training:
    $\mathcal{L}_{\text{D}}^{SATI} \leftarrow ||0 - \text{D}_Y(\hat{y}_z)||_2$
    $\text{optimizer}_{\text{D}}(\mathcal{L}_{\text{D}}^{\text{SATI}} + \mathcal{L}_{\text{D}}^{\text{CycleGAN}})$
```

We made several adaptations to the stitching aware training and inference to increase performance. The adaptations are shown in the following paragraphs.

B. OVERLAP SAMPLING

For 2D images, inference starts with a patch from domain X (see Fig. 3). All remaining patches are from domain $\mathcal{Z}$. When processing an image row by row, patches in the first row have only one adjacent patch already predicted, shown in Fig. 5 z_2. The first patch in each following row is depicted in Fig. 5 z_3. All patches not present in the first row or column have two adjacent patches and are shown in Fig. 5 z_1. Starting from the bottom right results in overlaps on the bottom side and the right side and therefore an equally complex training task. Starting in the middle results in more overlap combinations and should be avoided if not needed.

Image statistics for the mean and variance differ severely for the three cases. We use instance normalization without running mean and variance for our experiments. This combination will result in erroneous predictions when evaluating overlap combinations not used during training. Therefore, all overlap combinations are added to the training workflow to enable high-quality output for all three cases. While z_1 is utilized the most during inference, high-quality outputs for z_2 and z_3 are desired to omit propagation of errors from

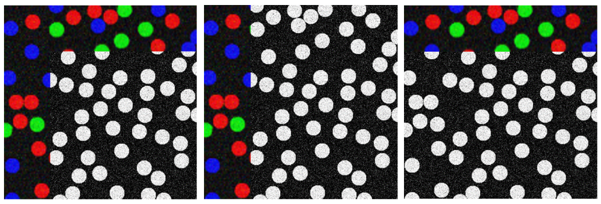

(a) Overlap z_1 (b) Overlap z_2 (c) Overlap z_3

FIGURE 5: Different overlaps from domain $\mathcal{Z}$ created by the merger to train the GAN on all overlaps needed during inference. For 2D images, overlaps z_2 (b) and z_3 (c) are used for the first row and column of patches during inference, while z_1 (a) is used for all other patches.

the image borders to the center of the image. Therefore, the merger (see Fig. 4) creates each of these cases with the same probability.

C. DOMAIN ENCODING

Convolutional neural networks work locally, and the receptive field is limited, especially in the early layers. When *SATI* is used, the GAN has to learn which parts of an image z are from domain $\mathcal{X}$ and which parts are from domain $\mathcal{Y}$ during training. This can be a challenging task for datasets where domain $\mathcal{X}$ and $\mathcal{Y}$ share significant local similarities. For our *tiling strategy benchmark dataset*, large local similarities are present in background areas, where no circles are placed.

Instead of adding an additional layer to the input patch which encodes the position in the image, we encode the origin domain directly into the image. We transfer the images from domain $\mathcal{X}$ into the range $[-1, 0]$ and the images from domain $\mathcal{Y}$ into range $[0, 1]$. As a result, the GAN can identify the domain according to the range of values and change or keep pixel values accordingly.

D. LOSS RAMPING

Unpaired image-to-image transfer is a challenging task and it is common for synthetic images to yield low quality for the first epochs. It is not useful to force the GAN to keep these low-quality parts of an image z in $\hat{y}_z$. Therefore, we increase the scaling of $\mathcal{L}_{stitch}$ from zero to the final scaling factor λ_{stitch} throughout the training.

E. STITCHING STRATEGY

As shown in Fig. 3, the final image consists of overlapping patches. The overlapping area can be selected from one of the patches. In Figure 3, the overlapping area from the bottom patch is selected, while the overlapping area from the top patch is dismissed. Preliminary tests showed, that using the complete overlapping area from one of the patches introduces errors for objects barely starting or ending in the adjacent patch. We prevent these errors by using the middle of the overlapping areas as the transition between patches in the final image.

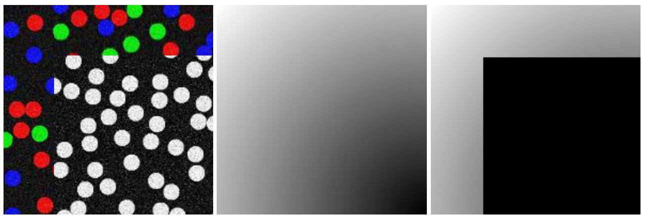


FIGURE 6: Pixels for $\mathcal{L}_{stitch}$ are weighted with respect to two superpixels. The pixel with the biggest distance to areas from domain $\mathcal{X}$ is weighted with one (top left, white). The pixel with the biggest distance to areas from domain $\mathcal{Y}$ is weighted with zero (bottom right, black). The two superpixels are used for linear weighting with the euclidean distance for all other pixels (middle image). Subsequently, pixels from domain $\mathcal{X}$ are set to zero (right image).

F. PIXEL OVERLAP WEIGHTING

With the stitching strategy, we cut patches in the middle of the overlapping area to create the final image. By doing so, we can allow the GAN to slightly change pixels at the border between domain X and Y when transforming an image z to $\hat{y}_z$. This can be beneficial if an object just starts at the end of a patch and the majority of the object is in the next patch. Having more information about an object in the next patch will allow the GAN to change the complete object accordingly. We enable this by weighting the pixels for $\mathcal{L}_{stitch}$ according to their location.

An example is shown in Fig. 6. The final image on the right shows the utilized weight map. All pixels from domain $\mathcal{X}$ are weighted with zero. The farther away a pixel in the area from domain $\mathcal{Y}$ is from a pixel in domain $\mathcal{X}$, the more the weight is increased. The weights are scaled linear between zero and one. Therefore, the relative size of the overlap is included in the weighting. For bigger overlapping regions, the GAN is given more freedom to change pixels near the transition from both domains.

IV. EXPERIMENTS

We incorporated the stitching aware training into CycleGAN[1], since CycleGAN and its variations are often used for biomedical data synthesis and in material science [8], [28], [29], [31], [32], [36]. For the generator architecture, we used the ResNet-Generator with instance normalization, 96 initial generator feature maps, and nine ResNet blocks in the feature space. For the discriminator architecture, we used the PatchGAN-Discriminator with instance normalization. We used the mean squared error (MSE) 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 ($\mathcal{L}_{adv}$). For the stitching loss $\mathcal{L}_{stitch}$, we also used the MSE and apply our pixel overlap weighting afterwards. We set the scaling for the stitching loss to λ_{stitch}=10. The other scaling

[1]For our work, we adapted the original implementation from https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix (accessed: 24.07.2022).

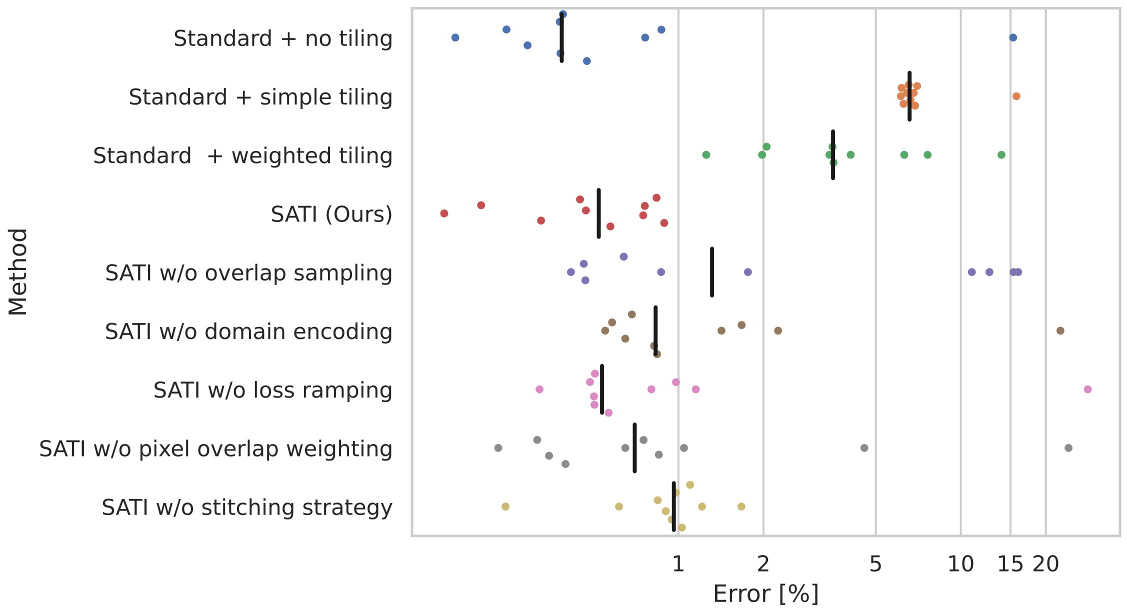


FIGURE 7: Our method (*SATI*) is benchmarked against *Standard + no tiling*, *Standard + simple tiling* and *Standard + weighted tiling*. For benchmarking, we used models trained with standard CycleGAN (Standard + [tiling strategy]). Furthermore, the plot shows an ablation study, where we deactivated different adaptations (*SATI w/o [adaptation]*). The error rate is displayed in percent on a logarithmic scale. The black lines indicate the median for each method. *Standard + no tiling* is the performance achievable when the whole image can be transferred to the GPU during inference. While this is possible for the *tiling strategy benchmark dataset*, this is not possible for large-scale 2D and 3D data.

factors are set according to the original implementation of CycleGAN [19] with λ_{cycle}=10, λ_{idt}=5 and $\mathcal{L}_{adv}$ is not scaled. The overall loss is defined by:

$$\mathcal{L} = \mathcal{L}_{adv} + \lambda_{cycle}\mathcal{L}_{cycle} + \lambda_{idt}\mathcal{L}_{idt} \\ + \mathcal{L}_{adv}^{\mathcal{ZY}} + \lambda_{stitch}\mathcal{L}_{stitch}, \tag{3}$$

where the first row represents the standard CycleGAN loss and the second row is the additional loss added with *SATI*.

A. TILING STRATEGY BENCHMARK DATASET

In our experiments on the *tiling strategy benchmark dataset*, we compared *SATI* to a simple tiling strategy (simple tiling), to the advanced tiling strategy (weighted tiling) used in [35], [37] and to the results processing the whole image at once (no tiling). Furthermore, we conducted an ablation study to quantify the usage of the different adaptations we made to the core of *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. We chose this size to enable benchmarking against the maximum achievable performance, which is done by processing the whole 2048 px × 2048 px image during inference at once (no tiling). Evaluations are performed on 50 images not present in the training data. Using 50 images results in 50.000 objects being present in the test data. Training a network takes around 27 hours on an NVIDIA GeForce RTX 3090 GPU. Therefore, we limited the number of runs for each method to ten. The ablation study, together with the benchmark methods, results in 60 trained

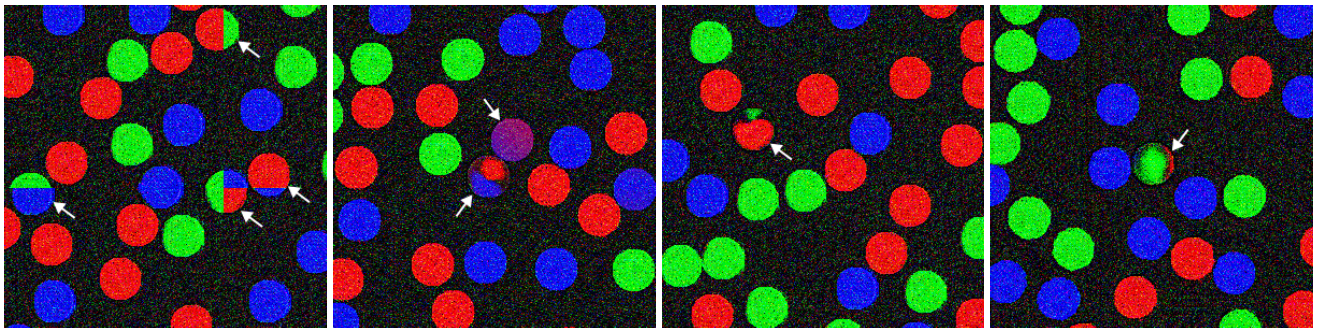
TABLE 1: Results of the tiling strategies and the ablation study on the *tiling strategy benchmark dataset*. The best case, *Standard + no tiling*, and our method (*SATI*) are highlighted. The values correspond to the error in percent.

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 (Ours)	**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 pixel overlap weighting	3.32	0.70	7.40	0.23
SATI w/o stitching strategy	0.95	0.96	0.37	0.24

networks and 67 days of training. No additional networks have to be trained for the *SATI w/o stitching strategy* results because the changes only affect the inference. The same ten trained standard CycleGAN networks have been used for all *Standard + [tiling strategy]* results.

B. BENCHMARK METHOD COMPARISON

The results for the comparison to the benchmark methods are shown in Fig. 7 and Tab. 1. The training of GANs is unstable and sometimes they do not converge, or mode collapse can occur [27]. To reduce the influence of these training runs, we opted to evaluate our experiments regarding the median instead of the mean. The benchmark methods *Standard + simple tiling*, *Standard + weighted tiling* and *Standard + no*



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

FIGURE 8: Exemplary errors present after application of different inference strategies compared to errors using our method *SATI*. Erroneous circles are marked with an arrow. For (c) and (d) few or no tiling-related errors are present. Therefore, we show examples of the remaining errors. In contrast to the tiling-related errors of (a) and (b), the errors in (c) and (d) can further be reduced with GANs better adapted to the task. The bottom left marked circle in (b) shows an error present, because the GAN made a faulty translation from $\mathcal{X}$ to $\mathcal{Y}$. This error is comparable to the errors in (c) and (d). The top right marked circle in (b) shows an error, where a circle was predicted in blue in one tile and red in the other one, resulting in a purple circle. Due to the overlap, this type of error can only occur in *Standard + weighted tiling*. For each method, a complete image synthesized from the same image in domain $\mathcal{X}$ can be seen in the supplementary material.

tiling only differ in the procedure used for inference, while the same trained networks are used. Processing complete images at once using *Standard + no tiling*, reduces the median error to 0.4%. This represents the best case scenario not applicable for large-scale 2D and 3D images. Using *Standard + simple tiling* results in a bad overall performance with a median error of 6.6%. Using *Standard + weighted tiling* reduces the median error to 3.5%. Using *SATI* results in a median error of 0.5%. This is a reduction of 92% compared to *Standard + simple tiling* and a reduction of 85% compared to *Standard + weighted tiling*.

Exemplary images of errors occurring in the final images are shown in Fig. 8. For *Standard + simple tiling* the patch borders are clearly visible. An example circle present in four patches is predicted in all three colors, dependent on the patch. Using *Standard + weighted tiling* highly improves the results, and due to the weighted overlap, it is not possible anymore to identify individual patches. Instead of a sharp transition between colors, colors are overlapped. The overlapping results, e.g., in a purple circle being a mixture of red and blue. The only errors left using *Standard + no tiling* are errors not related to tiling. These errors are present in all inference strategies. They could possibly be reduced by longer training, more training data, adaptation of GAN parameters, or usage of a different GAN architecture. Also, for our method, errors look similar to the remaining errors in *Standard + no tiling*.

1) Ablation study

For the ablation study, we deactivated the single adaptations we made to our base method and evaluated the performance. The results are shown in the bottom part of Fig. 7. Disabling any of the adaptations reduces the performance.

The least impact on performance has *SATI w/o loss ramping*, where the median error is only increased by 2.7%. However, one run collapsed, and the mapping was not learned correctly, which is a common problem for unpaired image-to-image translation. Because of the limited sample size, we can only assume that disabling loss ramping increases the chance of the GAN collapsing.

Disabling pixel overlap weighting increases the median error by 34%. Pixel overlap weighting allows the GAN to adapt already predicted pixels in border areas to new parts of the images not predicted yet. For our *tiling strategy benchmark dataset*, this enables the GAN to change the color of objects which just started at the border between both domains.

The median error increases by 84.5% when using *SATI w/o stitching strategy*. Example images show that the GAN changes the color of circles with a tiny part in the overlapping area. This could be because a small part results in a small weight compared to the overall loss. Color changes can result in errors in the final image. Using the stitching strategy omits this problem without needing a specialized loss.

Disabling domain encoding results in a decreased performance. The median error increases by 59.0%. Therefore, domain encoding eases the learning task even for a clear difference between circles in domains $\mathcal{X}$ and $\mathcal{Y}$ and no evaluation of background.

Finally, disabling overlap sampling and only learning to transfer images where the top and left regions are from domain $\mathcal{Y}$ results in the biggest drop in performance. Four models collapsed, resulting in an increase of the median error by 151.9%. An example image is shown in Fig. 9. The Figure shows that the GAN cannot produce good results when only the top or left region of a patch is from domain $\mathcal{Y}$. The input distribution differs a lot whether the top and left are from domain $\mathcal{Y}$ or only the top or the left. The GAN fails to achieve high-quality output in combination with instance

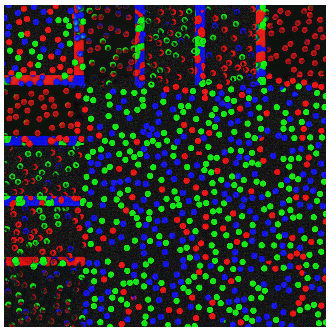


FIGURE 9: *SATI* with disabled overlap sampling. The GAN only learned to predict images with the top and left regions from domain $\mathcal{Y}$. It cannot produce good results for images with either top or left region from domain $\mathcal{Y}$.

normalization layers. It is interesting to see that, nevertheless, the GAN can recover for regions where the top and left are from domain $\mathcal{Y}$. Recovering from erroneous previous predictions is a key requirement for patch-wise inference of large-scale images. It can be concluded that a bad prediction will not reduce the quality of future patches.

The contours of some circles are distorted on all images in Fig. 8. This occurs due to the GAN making imperfect translations between both domains. Therefore, the contours of the circles can also be distorted when no tiling is applied (Fig. 8 (c), bottom left). In contrast, the contour information is available in both domains, and therefore, the GAN is able to infer contours throughout different patches. This can be seen in Fig. 8 (a). To reduce contour distortions, a spatial constrain can be added to CycleGAN [38].

C. REAL-WORLD DATASET

Additionally to the fully synthetic *tiling strategy benchmark dataset*, we expanded *SATI* for 3D data and applied it to a real-world dataset of KP-4 cells, where nuclei are stained with Draq5. The goal of this evaluation is to prove the following hypothesis: 1) *SATI* can be expanded to 3D, 2) *SATI* can be applied to complex real-world data and the GAN is still able to learn the mapping between both domains. The dataset consists of four images recorded with a Leica SP8 confocal microscope (Leica Microsystems, Wetzlar, Germany), has a voxel size of 568 nm×568 nm×1000 nm and a resolution of 8-bit. The images are cropped to remove areas without cells. The crops range from 380 px to 550 px in the XY-plane and 140 px to 190 px in the Z-direction. Afterwards, the crops are downscaled by the factor of 2 in the XY-plane. Thus, the Z-resolution is matched, and more objects are present in a single volume during training, easing the learning task. A crop of an XY-slice can be seen in Fig. 10 (a). Elaborate methods to create 3D nuclei for the synthetic label images exist [32], [37]. Both need a set of available annotations, which are hard to acquire for 3D data. On the other hand, it has been shown, that ellipsoids are a good estimate for 3D cell nuclei [29]. Therefore, we created four synthetic label images by randomly placing ellipsoids. Each image has a size of 256 px × 256 px × 256 px. The background is set to a value of 10 and the foreground is set to 130. Afterwards, Gaussian noise with $\mu = 0$ and $\sigma = 3.33$ is applied. Finally, we rounded the result to integers and clipped the result to the range $[0, 255]$. An exemplary crop of an XY-slice can be seen in Fig. 10 (b).

We trained the network with *SATI* for a total of 1120 epochs with a batch size of 12 and 256 random crops of size 64 px × 64 px × 64 px per epoch. We set the scaling for the stitching loss to λ_{stitch}=20 and the overlap to 16 px. The other scaling factors are the same as for the *tiling strategy benchmark dataset*. Furthermore, we use the MSE for all loss functions and start with a learning rate of 0.0002 for the Adam optimizer. The training took 31 hours on an NVIDIA A100.

A crop of an XY slice of the final generated image can be seen in Fig. 10 (e). For inference, we used patches of size 64 px × 64 px × 64 px with an overlap of 16 px. Therefore, 25 individual patches are shown in the crop. The generated crop shows that the GAN can match patches to their predecessors and no sharp transition inside a nucleus is visible at patch borders. In contrast, the crops shown in (c, d) were created with the same trained network, but *SATI* was not applied during inference. The borders of different patches are clearly visible in (c). The visual appearance of *Standard + weighted tiling* in (d) is comparable to (e). However, it must be denoted that a pixel in (d) is the weighted sum of up to eight individual predictions for 3D data. The results on the real-world dataset show, that *SATI* can be expanded to 3D and the GAN is still able to learn the mapping between both domains.

Although, the GAN is trained on the real-world data, a domain GAP regarding the brightness between the real-world data and the generated crops (c), (d) and (e) still exists. This is due to the spatial differences in the real-world images. The brightness of confocal microscopy images is lower towards the edges and for deep Z-slices. Standard GANs do not have information about spatial location during training and inference. If spatial consistency is needed, spatial information can be added to training and inference while still using *SATI* [32], [37].

V. DISCUSSION

Applying existing tiling strategies to the *tiling strategy benchmark dataset* shows a need for improvement. Weighted tiling improves the quality and errors are visually less prominent. The transition between patches is not learned, but improved in the post-processing. The advantage of this is that the training process remains unchanged. Due to the weighting, individual features are suppressed and erroneous objects could be smoothed or result in a mixture of object types. In contrast to this, *SATI* allows the GAN to learn what a meaningful transition between adjacent patches looks like. This allows our method to prevent errors that occur directly in the prediction of adjacent patches and cannot be corrected by the other tiling strategies.

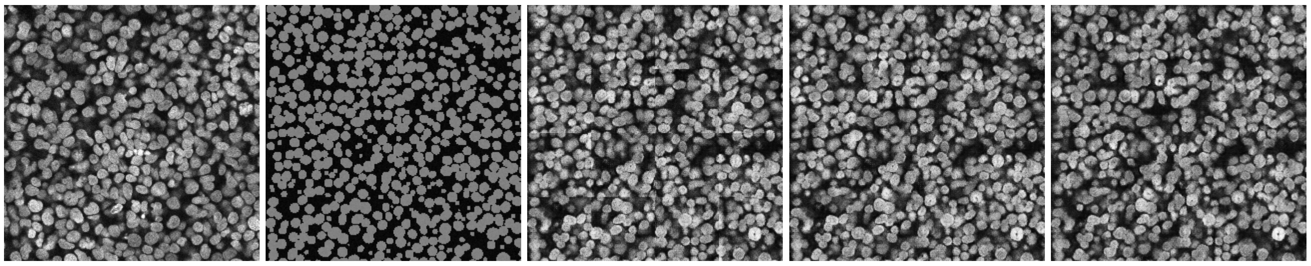

(a) Real-world crop
(b) Synthetic label image
(c) *Std. + simple tiling*
(d) *Std. + weighted tiling*
(e) *SATI* (ours)

FIGURE 10: Each image shows a XY-slice of the corresponding 3D volume cropped to 256 px $\times$ 256 px. A crop of the real-world image is shown in (a). The images c-e are generated with the image shown in (b). The generation process for *SATI* is performed with 3D volumes of 64 px in each dimension and an overlap of 16 px. The final image in (e) consists of 25 individual patches. The brightness of (c-e) is higher than (a). This is because the brightness of the real-world images decreases at the borders of an XY-slice and for deep Z-slices. A standard CycleGAN cannot reproduce this behavior. Several approaches, compatible with *SATI*, can be used to remove the decrease from real-world images or enable CycleGAN to reproduce the behavior [32], [37], [39]. An example for *Standard + simple tiling* is shown in (c). The patch borders are clearly visible compared to (d) and (e).

SATI produces high-quality results on the *tiling strategy benchmark dataset* comparable to the best case results without tiling. The GAN learns the desired behavior with the training and inference strategies introduced. The additional complexity of the learning task is significantly reduced by our domain encoding adaptation, which is shown by the decrease in performance when domain encoding is disabled. We do not think that the remaining increase in complexity of the learning task is a problem for real-world datasets. Still, it must be evaluated individually for different learning tasks.

For our 3D real-world dataset of KP-4 cells, *SATI* was able to synthesize large-scale images without a visually notable transition between patches. Also, *Standard + weighted tiling* yielded visually appealing images. Therefore, *Standard + weighted tiling* can still be a valid option for grayscale data, while weighting up to eight individual patches for 3D data can potentially change the noise in the image.

SATI is designed to work for objects with a limited spatial extent and no relations between distant objects. This is the case for many biological, medical, or material science datasets. However, there are datasets where *SATI* is of limited use. For example, creating a high-quality image of a blue car with red or green exterior mirrors can still result in a red left mirror and a green right mirror. The spatial distance of the mirrors is too large to be present in the overlapping areas. Therefore, the GAN has no information whether the first mirror was red or green when predicting the second one.

The inference time of *Standard + weighted tiling* increases by the factor of 4 for each dimension compared to *Standard + no tiling*. This is due to the large overlap [35]. The overlap is needed to guarantee a small change in image statistics between patches. *SATI* does not need a big overlap because it explicitly learns the transfer between patches. The overlap is only related to the spatial extent of information needed to predict the next patch correctly. In our experiments, the inference time is increased by a factor of 1.25 in each dimension compared to simple tiling. For 3D images, this results in an increase by the factor of 64 for *Standard + weighted tiling* and 1.95 for *SATI* compared to *Standard + simple tiling*. Memory usage is the same, for all methods during inference.

Using *SATI* adds additional predictions and loss functions to the training procedure. This increases training time. Standard training finished after approximately 24h. A training run for *SATI* took around 27:30h. Although this is an increase by 15%, we did not evaluate the influence on the training needed for convergence by adding *SATI* to the training procedure, as it is still an open question how to determine when to stop training GANs. Therefore, we advise users incorporating *SATI* to use the same number of epochs they used without *SATI*. This led to good results for all our experiments.

The complexity of the training task for the GAN is increased when *SATI* is used. This could potentially result in the need for increased network size. However, that was not the case for our experiments. Furthermore, using *SATI* during training results in increased VRAM utilization. The memory utilization on the benchmark dataset increased to 22.8GB compared to 19.9GB when training without *SATI*.

SATI results in high-quality images when used with the CycleGAN architecture. We aimed to design *SATI* to be integrable into different GAN architectures. This is necessary to ease usage and enable researchers to stick to their preferred architectures. A possible routine for working on new projects could be as follows: (i) Adapt a standard GAN architecture towards a new problem setting. (ii) Evaluate whether the patch quality is sufficient. (iii) Incorporate *SATI* to bridge the gap between patch-wise prediction and large-scale image prediction. This will result in a minimal additional workload.

VI. CONCLUSION AND OUTLOOK

Deep learning models for image segmentation require labeled training data. Labeling large-scale 2D and 3D data is a challenging task, time-consuming, and the interobserver variability is high. Researchers try to reduce manual labeling by using GANs performing unpaired image-to-image translation to create synthetic training data. Using GANs trained for unpaired image-to-image translation to predict large-scale 2D and 3D images requires patch-wise inference due to VRAM limitations. As of now, the final images are typically created using a simple tiling strategy or weighted tiling.

Our experiments show that GANs suffer from tiling-related errors for one-to-many transformation tasks. These errors are most prominent when using *Standard + simple tiling*. Advanced methods like *Standard + weighted tiling* cannot completely remove these errors. With *SATI*, GANs produce high-quality output when inference is performed patch-wise. We achieved an error of 0.5% compared to 3.5% using *Standard + weighted tiling* on the *tiling strategy benchmark dataset*. Therefore, we reduced the error by 85% compared to the state-of-the-art.

The ablation study shows that the individual adaptations made to *SATI* further increase the final output quality, and the GAN can recover from single erroneous predictions throughout the patch-wise inference. This allows the prediction of arbitrarily large images.

The results using *SATI* on a real-world dataset prove that our method can create high-quality synthetic 3D images with complex content.

SATI can be incorporated into different GAN architectures to create large-scale 2D and 3D images. We hope that this will lead to better synthetic datasets for real-world problems. While better synthetic data is always desirable, the implications on downstream tasks using the synthesized data e.g., for training of segmentation networks is up to future research. It is highly dependent on the data, the learning task and the downstream method used, whether large-scale 2D and 3D images are needed.

Possible applications of *SATI* range from 3D microscopy to large-scale 2D data like whole slide images or aerial hyperspectral images. In the future, we want to apply *SATI* to a variety of real-world datasets and examine the influence of different tiling strategies not only on foreground objects, but also on image properties such as background noise.

The performance of SATI incorporated in different GAN architectures, especially for multimodal architectures like DRIT or MUNIT using content, and style or attribute feature spaces, needs to be evaluated in future work. The main limitation of *SATI* is the increased complexity of the learning task which could lead to longer training or the need for larger networks. Future research needs to focus on reducing this increase. In the future, we will adapt *SATI* to handle 3D+time data.

ACKNOWLEDGMENT

We thank Mario Vitacolonna from the Institute of Molecular and Cell Biology, Mannheim University of Applied Sciences, Germany for providing the microscopy images used for our experiments on a real-world dataset.

REFERENCES

[1] M. Weigert, U. Schmidt, R. Haase, K. Sugawara, and G. Myers, "Star-convex Polyhedra for 3D Object Detection and Segmentation in Microscopy," in *Proc. IEEE/CVF Winter Conf. Appl. Comp. Vis. (WACV)*, 2020, pp. 3666–3673.

[2] 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, p. e0243219, 2020.

[3] M. Böhland, L. Tharun, T. Scherr, R. Mikut, V. Hagenmeyer, L. D. R. Thompson, S. Perner, and M. Reischl, "Machine learning methods for automated classification of tumors with papillary thyroid carcinoma-like nuclei: A quantitative analysis," *PLOS ONE*, vol. 16, no. 9, p. e0257635, 2021.

[4] S. Graham, Q. D. Vu, S. E. A. Raza, A. Azam, Y. W. Tsang, J. T. Kwak, and N. Rajpoot, "HoVer-Net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images," *Med. Image Anal.*, vol. 58, p. 101563, 2019.

[5] M. Böhland, O. Neumann, M. P. Schilling, M. Reischl, R. Mikut, K. Löffler, and T. Scherr, "Ciscnet - a Single-Branch Cell Nucleus Instance Segmentation and Classification Network," in *2022 IEEE Int. Symp. Biomed. Imag. Challenges (ISBIC)*, 2022, pp. 1–5.

[6] C. Stringer, T. Wang, M. Michaelos, and M. Pachitariu, "Cellpose: a generalist algorithm for cellular segmentation," *Nat. Met.*, vol. 18, no. 1, pp. 100–106, 2021.

[7] J. Fioresi, D. J. Colvin, R. Frota, R. Gupta, M. Li, H. P. Seigneur, S. Vyas, S. Oliveira, M. Shah, and K. O. Davis, "Automated Defect Detection and Localization in Photovoltaic Cells Using Semantic Segmentation of Electroluminescence Images," *IEEE J. Photovolt.*, vol. 12, no. 1, pp. 53–61, 2022.

[8] S. Müller, C. Sauter, R. Shunmugasundaram, N. Wenzler, V. De Andrade, F. De Carlo, E. Konukoglu, and V. Wood, "Deep learning-based segmentation of lithium-ion battery microstructures enhanced by artificially generated electrodes," *Nat. Commun.*, vol. 12, no. 1, p. 6205, Oct. 2021.

[9] J. Fan, J. Yang, Y. Wang, S. Yang, D. Ai, Y. Huang, H. Song, A. Hao, and Y. Wang, "Multichannel Fully Convolutional Network for Coronary Artery Segmentation in X-Ray Angiograms," *IEEE Access*, vol. 6, pp. 44 635–44 643, 2018.

[10] R. Ranjbarzadeh, A. Bagherian Kasgari, S. Jafarzadeh Ghoushchi, S. Anari, M. Naseri, and M. Bendechache, "Brain tumor segmentation based on deep learning and an attention mechanism using MRI multi-modalities brain images," *Scientific Reports*, vol. 11, no. 1, p. 10930, 2021.

[11] K. Han, L. Liu, Y. Song, Y. Liu, C. Qiu, Y. Tang, Q. Teng, and Z. Liu, "An Effective Semi-Supervised Approach for Liver CT Image Segmentation," *IEEE J. Biomed. Health Inform.*, vol. 26, no. 8, 2022.

[12] Ç. Özgün, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation," in *Med. Imag. Computing Comp.-Assited Intervention - MICCAI 2016*, 2016, pp. 424–432.

[13] R. Bruch, R. Rudolf, R. Mikut, and M. Reischl, "Evaluation of semi-supervised learning using sparse labeling to segment cell nuclei," *Current Directions in Biomedical Engineering*, vol. 6, no. 3, pp. 398–401, 2020.

[14] M. P. Schilling, L. Rettenberger, F. Münke, H. Cui, A. A. Popova, P. A. Levkin, R. Mikut, and M. Reischl, "Label Assistant: A Workflow for Assisted Data Annotation in Image Segmentation Tasks," in *Proc. - 31. Workshop Comput. Intell.*, 2021.

[15] D. Wiesner, D. Svoboda, M. Maška, and M. Kozubek, "CytoPacq: a web-interface for simulating multi-dimensional cell imaging," *Bioinformatics*, vol. 35, no. 21, pp. 4531–4533, 2019.

[16] 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.

[17] Y. Zhao, J. Xie, and P. Li, "Learning Energy-Based Generative Models via Coarse-to-Fine Expanding and Sampling," in *International Conference on Learning Representations*, 2020.

[18] G. Han, J. Min, and S. W. Han, "EM-LAST: Effective Multidimensional Latent Space Transport for an Unpaired Image-to-Image Translation With an Energy-Based Model," *IEEE Access*, vol. 10, 2022.

[19] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," in *Proc. IEEE Int. Conf. Comp. Vis. (ICCV)*, 2017, pp. 2223–2232.

[20] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised Image-to-Image Translation Networks," *arXiv:1703.00848*, 2018.

[21] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive Learning for Unpaired Image-to-Image Translation," in *Comp. Vis. - ECCV 2020*, Cham, 2020, pp. 319–345.
[22] Z. Yi, H. Zhang, P. Tan, and M. Gong, "DualGAN: Unsupervised Dual Learning for Image-To-Image Translation," in *Proc. IEEE Int. Conf. Comp. Vis. (ICCV)*, 2017, pp. 2849–2857.
[23] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proc. Eur. Conf. Comp. Vis. (ECCV)*, 2018, pp. 172–189.
[24] M.-Y. Liu, X. Huang, A. Mallya, T. Karras, T. Aila, J. Lehtinen, and J. Kautz, "Few-Shot Unsupervised Image-to-Image Translation," in *Proc. IEEE/CVF Int. Conf. Comp. Vis. (ICCV)*, 2019, pp. 10 551–10 560.
[25] H.-Y. Lee, H.-Y. Tseng, Q. Mao, J.-B. Huang, Y.-D. Lu, M. Singh, and M.-H. Yang, "DRIT++: Diverse Image-to-Image Translation via Disentangled Representations," *Int. J. of Comp. Vis.*, vol. 128, no. 10, pp. 2402–2417, 2020.
[26] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation," in *Proc. IEEE Conf. Comp. Vis. Pat. Recognit. (CVPR)*, 2018, pp. 8789–8797.
[27] Y. Pang, J. Lin, T. Qin, and Z. Chen, "Image-to-Image Translation: Methods and Applications," *IEEE Transactions on Multimedia*, vol. 24, 2022.
[28] M. Böhland, T. Scherr, A. Bartschat, R. Mikut, and M. Reischl, "Influence of Synthetic Label Image Object Properties on GAN Supported Segmentation Pipelines," in *Proc. - 29. Workshop Comput. Intell.*, 2019, pp. 289–305.
[29] K. W. Dunn, C. Fu, D. J. Ho, S. Lee, S. Han, P. Salama, and E. J. Delp, "DeepSynth: Three-Dimensional Nuclear Segmentation of Biological Images Using Neural Networks Trained with Synthetic Data," *Sci. Rep.*, vol. 9, no. 1, p. 18295, 2019.
[30] K. Yao, J. Sun, K. Huang, L. Jing, H. Liu, D. Huang, and C. Jude, "Analyzing Cell-Scaffold Interaction through Unsupervised 3D Nuclei Segmentation," *Int. J. of Bioprinting*, vol. 8, no. 1, 2022.
[31] 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*, 2022.
[32] R. Bruch, F. Keller, M. Böhland, M. Vitacolonna, L. Klinger, R. Rudolf, and M. Reischl, "Synthesis of large scale 3D microscopic images of 3D cell cultures for training and benchmarking," *PLOS ONE*, vol. 18, no. 3, 2023.
[33] 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.
[34] Z. Shen, S. K. Zhou, Y. Chen, B. Georgescu, X. Liu, and T. Huang, "One-to-one Mapping for Unpaired Image-to-image Translation," in *Proc. IEEE/CVF Winter Conf. Appl. Comp. Vis. (WACV)*, 2020, pp. 1170–1179.
[35] 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 *Proc. 2nd Int. Conf. Mde. Imag. Deep Learn.*, 2019, pp. 151–163.
[36] S. J. Ihle, A. M. Reichmuth, S. Girardin, H. Han, F. Stauffer, A. Bonnin, M. Stampanoni, K. Pattisapu, J. Vörös, and C. Forró, "Unsupervised data to content transformation with histogram-matching cycle-consistent generative adversarial networks," *Nature Machine Intelligence*, vol. 1, no. 10, pp. 461–470, 2019.
[37] 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, pp. 1–21, 2021.
[38] C. Fu, S. Lee, D. Joon Ho, S. Han, P. Salama, K. W. Dunn, and E. J. Delp, "Three Dimensional Fluorescence Microscopy Image Synthesis and Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2221–2229.
[39] T. Peng, K. Thorn, T. Schroeder, L. Wang, F. J. Theis, C. Marr, and N. Navab, "A BaSiC tool for background and shading correction of optical microscopy images," *Nature Communications*, vol. 8, no. 1, 2017.

MORITZ BÖHLAND received the bachelor's and master's degrees in mechanical engineering from the Karlsruhe Institute of Technology, Karlsruhe, Germany, in 2014 and 2017. He is currently a Doctoral Researcher with the Research Group "Machine Learning for High-Throughput and Mechatronics", Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Eggenstein-Leopoldshafen, Germany. His research interests include image processing, machine learning and data analysis.

ROMAN BRUCH received the bachelor's and master's degrees in mechanical engineering from the Karlsruhe Institute of Technology, Karlsruhe, Germany, in 2017 and 2019, respectively. He is currently a Doctoral Researcher with the Research Group "Machine Learning for High-Throughput and Mechatronics", Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Eggenstein-Leopoldshafen, Germany. His research interests include image processing, data analytics and machine learning in the biological field.

SIMON BÄUERLE received the bachelor's and master's degrees in mechanical engineering from the Karlsruhe Institute of Technology, Karlsruhe, Germany, in 2015 and 2018. He is currently pursuing the Ph.D. degree with a joint research project of the Institute for Automation and Applied Informatics at the Karlsruhe Institute of Technology in Karlsruhe, Germany and the Robert Bosch GmbH in Reutlingen, Germany. His research interests include machine learning, image processing and industrial AI.

LUCA RETTENBERGER holds a Bachelor's and Master's degree in Computer Science from the University of Applied Sciences Ravensburg-Weingarten. Currently, he is pursuing a Ph.D. at the Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Eggenstein-Leopoldshafen, Germany. Within his research pursuits, Luca specializes in the domains of image processing, machine learning for material science, representation learning, and self-supervised learning.

MARKUS REISCHL received the Dipl.-Ing. and Ph.D. degrees in mechanical engineering from the University of Karlsruhe, Karlsruhe, Germany, in 2001 and 2006, respectively. Since 2020, he has been an Adjunct Professor with the Faculty of Mechanical Engineering and heading the research group "Machine Learning for High-Throughput and Mechatronics" with the Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology. His research interests include man–machine interfaces, image processing, machine learning, and data analytics.