I. Supplementary Material

A. Network Architecture

The architecture details for 2D nuclei segmentation are reported as follows: h and w: height and width of the input images, n_s: the dimension of the AdaIN parameters, N: the number of output channels, K: kernel size, S: stride size, P: padding size, FC: fully connected layer, IN: instance normalization, AdaIN: adaptive instance normalization, ReLU: rectified linear unit, LReLU: Leaky ReLU with a negative slope of 0.2.

TABLE I

Architecture of the unified content encoder $\mathcal{G}_{enc}$. The style representation learned from MLP for each domain is injected by adaptive instance normalization.

Part	Input → Output Shape	Layer Information
Down-sampling	(h,w,1)+(64+64) → (h,w,64)	CONV-(N64, K7x7, S1, P3), AdaIN, ReLU
	(h,w,64)+(128+128) → ($\frac{h}{2}$,$\frac{w}{2}$,128)	CONV-(N128, K3x3, S2, P1), AdaIN, ReLU
	($\frac{h}{2}$,$\frac{w}{2}$,128)+(256+256) → ($\frac{h}{4}$,$\frac{w}{4}$,256)	CONV-(N256, K3x3, S2, P1), AdaIN, ReLU
Bottleneck	($\frac{h}{4}$,$\frac{w}{4}$,256)+(256+256) → ($\frac{h}{4}$,$\frac{w}{4}$,256)	ResBlock: CONV-(N256, K3x3, S1, P1), AdaIN, ReLU
	($\frac{h}{4}$,$\frac{w}{4}$,256)+(256+256) → ($\frac{h}{4}$,$\frac{w}{4}$,256)	ResBlock: CONV-(N256, K3x3, S1, P1), AdaIN, ReLU
	($\frac{h}{4}$,$\frac{w}{4}$,256)+(256+256) → ($\frac{h}{4}$,$\frac{w}{4}$,256)	ResBlock: CONV-(N256, K3x3, S1, P1), AdaIN, ReLU
	($\frac{h}{4}$,$\frac{w}{4}$,256)+(256+256) → ($\frac{h}{4}$,$\frac{w}{4}$,256)	ResBlock: CONV-(N256, K3x3, S1, P1), AdaIN, ReLU

TABLE II

Architecture of MLP for encoder $\mathcal{G}_{enc}$ and decoder $\mathcal{G}_{dec}$. One single style representation for each domain is learned from the domain label d_i.

Part	Input → Output Shape	Layer Information
MLP	(2) → (256)	FC(2, 256), ReLU
	(256) → (256)	FC(256, 256), ReLU
	(256) → (n_s)	FC(256, n_s)

TABLE III

Architecture of the unified decoder $\mathcal{G}_{dec}$. The style representation learned from MLP for each domain is injected by adaptive instance normalization.

Part	Input → Output Shape	Layer Information
Bottleneck	($\frac{h}{4}$,$\frac{w}{4}$,256)+(256+256) → ($\frac{h}{4}$,$\frac{w}{4}$,256)	ResBlock: CONV-(N256, K3x3, S1, P1), AdaIN, ReLU
	($\frac{h}{4}$,$\frac{w}{4}$,256)+(256+256) → ($\frac{h}{4}$,$\frac{w}{4}$,256)	ResBlock: CONV-(N256, K3x3, S1, P1), AdaIN, ReLU
	($\frac{h}{4}$,$\frac{w}{4}$,256)+(256+256) → ($\frac{h}{4}$,$\frac{w}{4}$,256)	ResBlock: CONV-(N256, K3x3, S1, P1), AdaIN, ReLU
	($\frac{h}{4}$,$\frac{w}{4}$,256)+(256+256) → ($\frac{h}{4}$,$\frac{w}{4}$,256)	ResBlock: CONV-(N256, K3x3, S1, P1), AdaIN, ReLU
Up-sampling	($\frac{h}{4}$,$\frac{w}{4}$,256)+(128+128) → ($\frac{h}{2}$,$\frac{w}{2}$,128)	DECONV-(N128, K3x3, S2, P1), AdaIN, ReLU
	($\frac{h}{2}$,$\frac{w}{2}$,128)+(64+64) → (h,w,64)	DECONV-(N64, K3x3, S2, P1), AdaIN, ReLU
	(h,w,64)→(h,w,1)	CONV-(N1,K7x7,S1,P3), ReLU, Tanh

TABLE IV

Architecture of domain discriminator $\mathcal{D}$.

Part	Input → Output Shape	Layer Information
Shared body	(h,w,1) → ($\frac{h}{2}$,$\frac{w}{2}$,64)	CONV-(N64, K4x4, S2, P1), LReLU
	($\frac{h}{2}$,$\frac{w}{2}$,64) → ($\frac{h}{4}$,$\frac{w}{4}$,128)	CONV-(N128, K4x4, S2, P1), IN, LReLU
	($\frac{h}{4}$,$\frac{w}{4}$,128) → ($\frac{h}{8}$,$\frac{w}{8}$,256)	CONV-(N256, K4x4, S2, P1), IN, LReLU
	($\frac{h}{8}$,$\frac{w}{8}$,256) → ($\frac{h}{8}$,$\frac{w}{8}$,512)	CONV-(N512, K4x4, S1, P1), IN, LReLU
Output Branch D_1	($\frac{h}{8}$,$\frac{w}{8}$,512) → ($\frac{h}{8}$,$\frac{w}{8}$,1)	CONV-(N1, K4x4, S1, P1)
Output Branch D_2	($\frac{h}{8}$,$\frac{w}{8}$,512) → ($\frac{h}{8}$,$\frac{w}{8}$,1)	CONV-(N1, K4x4, S1, P1)

B. Adaptive Instance Normalization (AdaIN)

Instance Normalization is defined in the following:

$$IN(x) = \gamma(\frac{x - \mu(x)}{\sigma(x)}) + \beta \tag{1}$$

where γ and β are learnable parameters, and $\mu(x)$ and $\sigma(x)$ are the channel-wise mean and variance which are defined as:

$$\mu_{nc}(x) = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{nchw}, \sigma_{nc}(x) = \sqrt{\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{nchw} - \mu_{nc}(x))} \tag{2}$$

According to previous works [1]–[3], AdaIN receives a content input x. And a style representation y contains y_μ and y_σ. Unlike Instance Normalization (IN), AdaIN has no learnable affine parameters. Instead, it uses adaptively the input style representations as the affine parameters:

$$AdaIN(x, y_\mu, y_\sigma) = y_\mu(\frac{x - \mu(x)}{\sigma(x)}) + y_\sigma. \tag{3}$$

More details can be seen in the codes.

C. Implementation Details

1) Image Preprocessing: All the datasets are scaled using min-max normalization to [-1, 1] directly. 3D data are then spatially normalized to have the same spacing on the three axis. 2D patches or 3D sub-volumes are randomly cropped during training. For some 3D datasets, we resize the normalized images with the ratio of 0.5 to reduce their computational redundancy, as shown in Table V.

2) Mask Synthesis: To minimize the influence of mask synthesis on the evaluation of unsupervised methods, we utilize the simplest simulation model to describe the nucleus. More concretely, we use the ellipse and ellipsoid to represent nuclei in 2D and 3D, respectively. To generate such masks, three properties are typically required: ellipse/ellipsoid per image (P_n), the size of ellipse/ellipsoid (P_s), and the eccentricity of ellipse/ellipsoid (P_e). To imitate real unsupervised training, the range of nuclei number n in each image is roughly estimated. P_s are tuned manually by determining the range of each ellipse/ellipsoid's major axis a. For P_e, the eccentricity e is randomized in the range of [0.25, 0.75] for all the experiments to calculate the minor axis b by the equation:

$$b = \sqrt{1 - e^2}a \tag{4}$$

The generated ellipse/ellipsoid are then randomly rotated and placed in the masks without overlapping. Details of parameters setting can be found in Table V.

TABLE V
DETAILED PARAMETERS USED TO SYNTHESIZE MASKS FOR ALL EXPERIMENTS.

Dataset	Resize Ratio	Major axis of ellipse/ellipsoid a	Number of ellipse/ellipsoid n	Synthetic Patch Size
Fluo-N2DL-HeLa (semantic segmentation)	1.0	[13, 18]	[5, 40]	[256,256]
HaCaT	1.0	[20, 30]	[5, 15]	[256,256]
BBBC024	1.0	[30, 35]	[13, 15]	[128,512,768]
Scaffold-A549	0.5	[10, 15]	[50, 100]	[64,128,128]
Fluo-N2DL-HeLa (instance segmentation)	1.0	[13, 18]	[10, 40]	[256,256]
Fluo-N2DH-SIM+	1.0	[23, 35]	[5, 13]	[256,256]
Fluo-N2DH-GOWT1	1.0	[15, 23]	[4, 12]	[256,256]
Fluo-N3DH-CE	0.5	[10, 18]	[2, 60]	[256,256]
Fluo-N3DH-CHO	1.0	[15, 30]	[6, 16]	[64,128,128]
Fluo-N3DH-SIM+	0.5	[10, 15]	[2, 24]	[64,128,128]

D. Two-stage Pipeline with AD-GAN

We conduct further comparison of the proposed AD-GAN-INS and a pipeline with a state-of-the-art supervised medical image method TransUNet [4] trained on synthetic AD-GAN training data. As shown in Table VI, we conduct experiments under different setting.

First, we train the TransUNet with the synthetic paired data from CycleGAN which contains nuclei offset, nuclei deletion/addition, and inconsistent shape. Compared with CGU-Net which utilizes U-Net as segmentor, an improvement of 5.5% on DICE can be observed, showing the importance of using a robust model under erroneous data. However, compared with our method, there is still a large gap mainly because data synthesized by CycleGAN are significantly of lower quality than those by our method.

Second, we also show that training TransUNet with the synthetic paired data from AD-GAN can lead to a further improvement of 1.8% and 0.9% on segmentation and detection respectively, compared with baseline AD-GAN-INS. Using a two-stage pipeline with AD-GAN shall bring steady improvement but with extra computation cost and more training time, which can be seen as a trade-off between efficiency and accuracy.

Finally, a gap between the supervised TransUNet and the two-stage pipeline can be noticed. This might be partly caused by the simple simulation of nuclei in our pipeline, which may not reflect the diversity and complexity of nuclei shape. Nonetheless, considering the unsupervised nature of our proposed method which need no annotation, 91.1% or 89.3% can still be deemed pretty good. Future work may explore the possibility of generating data more realistically to minimize the gap between supervised methods and unsupervised methods.

TABLE VI
COMPARISON OF AD-GAN-INS AND TWO-STAGE AD-GAN PIPELINE ON DATASET HACAT.

Model	Training	DICE (segmentation)	F1-score (detection)
AD-GAN-INS	unsupervised	89.3	95.2
TransUNet	CycleGAN synthetic data	78.8	88.4
TransUNet	AD-GAN synthetic data	91.1	96.1
TransUNet	Annotated data	95.2	97.5

E. Interpolation of Domain Label

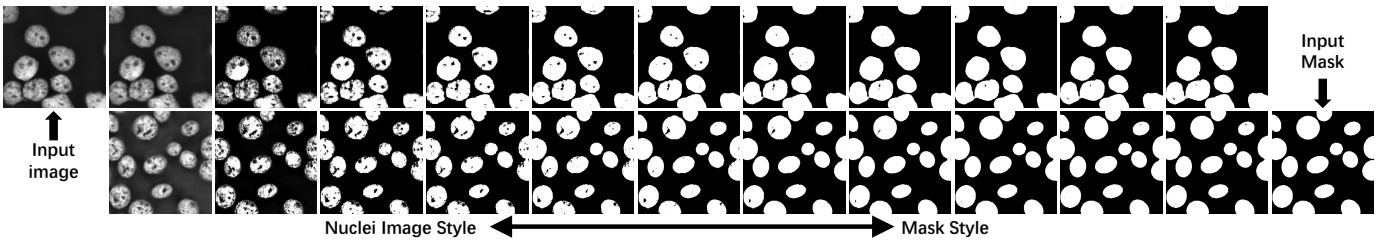

Fig. 1. Continuous change of style representation

According to UFDN [5], if a model can really learn the representation disentanglement, the generated images should show a continuous change when the input style representation is changed accordingly. In order to validate this, we generate continuous change of style representation by linearly interpolating the two domain labels between (0,1) and (1,0) with 10 intervals. We then input them to the trained AD-GAN (Fig. 1).

F. Cell Tracking Challenge

TABLE VII
COMPARISON OF AD-GAN-INS AND THE STATE-OF-THE-ART METHODS ON CELL SEGMENTATION BENCHMARK. CYCLEGAN FAILS TO ESTABLISH ONE-TO-ONE MAPPING IN SOME DATASETS AND HENCE WE LEAVE THEIR RESULTS AS "-".

	Method	Fluo-N2DH-GOWT1	Fluo-N2DL-HeLa	Fluo-N2DH-SIM+	Fluo-N3DH-CE	Fluo-N3DH-CHO	Fluo-N3DH-SIM+
$\mathrm{OP}_{\mathrm{csb}}$	First	0.952	0.954	0.905	0.83	0.926	0.949
	Second	0.948	0.953	0.897	0.816	0.913	0.885
	Third	0.948	0.951	0.897	0.811	0.913	0.866
	CycleGAN	-	0.654	0.632	-	0.644	-
	AD-GAN	0.804	0.894	0.824	0.729	0.852	0.733
SEG	First	0.938	0.923	0.832	0.729	0.917	0.906
	Second	0.933	0.919	0.825	0.705	0.914	0.786
	Third	0.931	0.917	0.822	0.688	0.903	0.759
	CycleGAN	-	0.486	0.457	-	0.506	-
	AD-GAN	0.787	0.850	0.720	0.624	0.823	0.613
DET	First	0.98	0.994	0.983	0.917	0.954	0.992
	Second	0.976	0.992	0.981	0.914	0.945	0.984
	Third	0.970	0.992	0.979	0.903	0.934	0.974
	CycleGAN	-	0.823	0.806	-	0.782	-
	AD-GAN	0.821	0.938	0.926	0.833	0.881	0.852

We conduct extra experiments of instance segmentation on Cell Tracking Challenge, and directly compare our method with the state-of-the-art methods on Cell Segmentation Benchmark[1]. Particularly, we evaluate our method on Fluo-N2DH-GOWT1, Fluo-N2DL-HeLa, Fluo-N2DH-SIM+, Fluo-N3DH-CE, Fluo-N3DH-CHO, and Fluo-N3DH-SIM+. We follow the official metrics to report the performance by SEG, DET, and $\mathrm{OP}_{\mathrm{csb}}$, where $\mathrm{OP}_{\mathrm{csb}} = 0.5 \times (\mathrm{SEG} + \mathrm{DET})$. We also report the results from CycleGAN, as a baseline of unsupervised nuclei segmentation. The training protocol is the same for AD-GAN-INS and CycleGAN, including mask synthesis and maximum training iteration.

Overall, CycleGAN fails to learn one-to-one mapping in the datasets Fluo-N2DH-GOWT1, Fluo-N3DH-CE, and Fluo-N3DH-SIM+. That is to say, the trained CycleGAN cannot generate masks depending on the input image, but generates masks randomly, which may be caused by the difficulty of building one-to-one mapping as well as lacking constraints in the training process of CycleGAN. In addition, due to the lossy transformation problem, CycleGAN achieves low performance in segmentation tasks (caused by nuclei offset, inconsistent shape) on the other three datasets. Meanwhile, the problem of nuclei addition/deletion also has an influence on detection results of CycleGAN, resulting in a low score of $\mathrm{OP}_{\mathrm{csb}}$.

In comparison, AD-GAN-INS achieves stable and reasonable results on all the six datasets. Nonetheless, it should be admitted that there is still a gap (0.113 ± 0.054) between our unsupervised method and the top ranked methods in the lead board of CTC. On one hand, we believe this is understandable, as the proposed AD-GAN-INS is unsupervised: without using any annotated data, the performance of around 80% (even on some hard datasets) appears already quite promising. By contrast, most of the top ranked methods in CTC are supervised methods (pretrained on ImageNet, and requiring heavy annotation). Though a few leading methods are image-processing based approaches, they rely on task-specific complicated image processing techniques (including sophisticated pre- and post-processing). They are hard to be adapted to the other tasks and difficult to be used by practitioners. On the other hand, we also argue that it may not be very fair to evaluate our method on CTC, which contains only two annotated videos in each dataset. As unsupervised methods may need more training samples to catch up with supervised methods, insufficient training samples actually limit our AD-GAN's performance. We believe it may be more suitable for our methods to be applied on larger un-annotated datasets where AD-GAN can become much powerful.

[1]The benchmark can be found in Cell Segmentation Benchmark

G. Scaffold-A549 dataset for Unsupervised Nuclei Segmentation

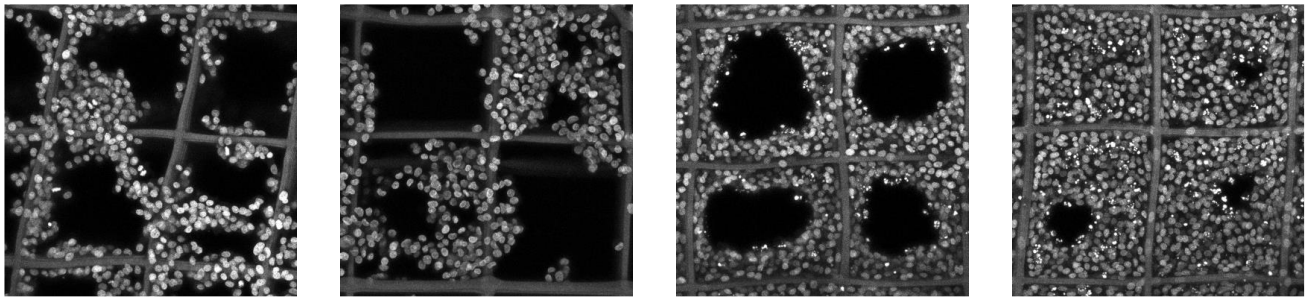

Fig. 2. Visualization of samples from the scaffold-A549 datasets.

As part of this work, we introduce a new 3D fluorescence image dataset that we term as scaffold-based A549 cell culture (Scaffold-A549) dataset[2], which is more challenging. Scaffold-A549 consists of 20 unlabelled training images and one fully annotated test image. The A549 human non-small cell lung cancer cells were seeded in fibrous scaffolds and the nuclei of A549 cells are stained with Hoechst 33342 (blue) for Confocal laser scanning microscope (CLSM) imaging. Images of 3D cell cultures were captured with a EC Plan-Neofluar 20X/0.5 air immersion objective, using a CLSM system (LSM-880, ZEISS, Germany), and the scanning depth of the cell cultured scaffolds is set about $60\mu m$ based on preliminary tests. Then, a spatial normalization and center crop operation is performed to all 21 collected CLSM with size of $1024 \times 1024 \times 128$ voxels and resolution of $0.4151\ \mu m$ per voxel. The density of nuclei varies among the images, and one image with the median density is adopted for manual annotation. To reduce the annotation ambiguity, this volume is labelled slice-by-slice from axial, sagittal and coronal views by three professional persons using 2D labelling software. A total of 9 annotations are merged using the average value, followed by a Gaussian filter for surface smoothing. Annotating the data in this way can ensure as accurate as possible the nuclear boundary in the annotation process, though the nucleus is elongated axially due to light diffraction.

[2]Scaffold-A549 is available at: https://github.com/Kaiseem/Scaffold-A549.

H. Visualization of BBBC024 Results

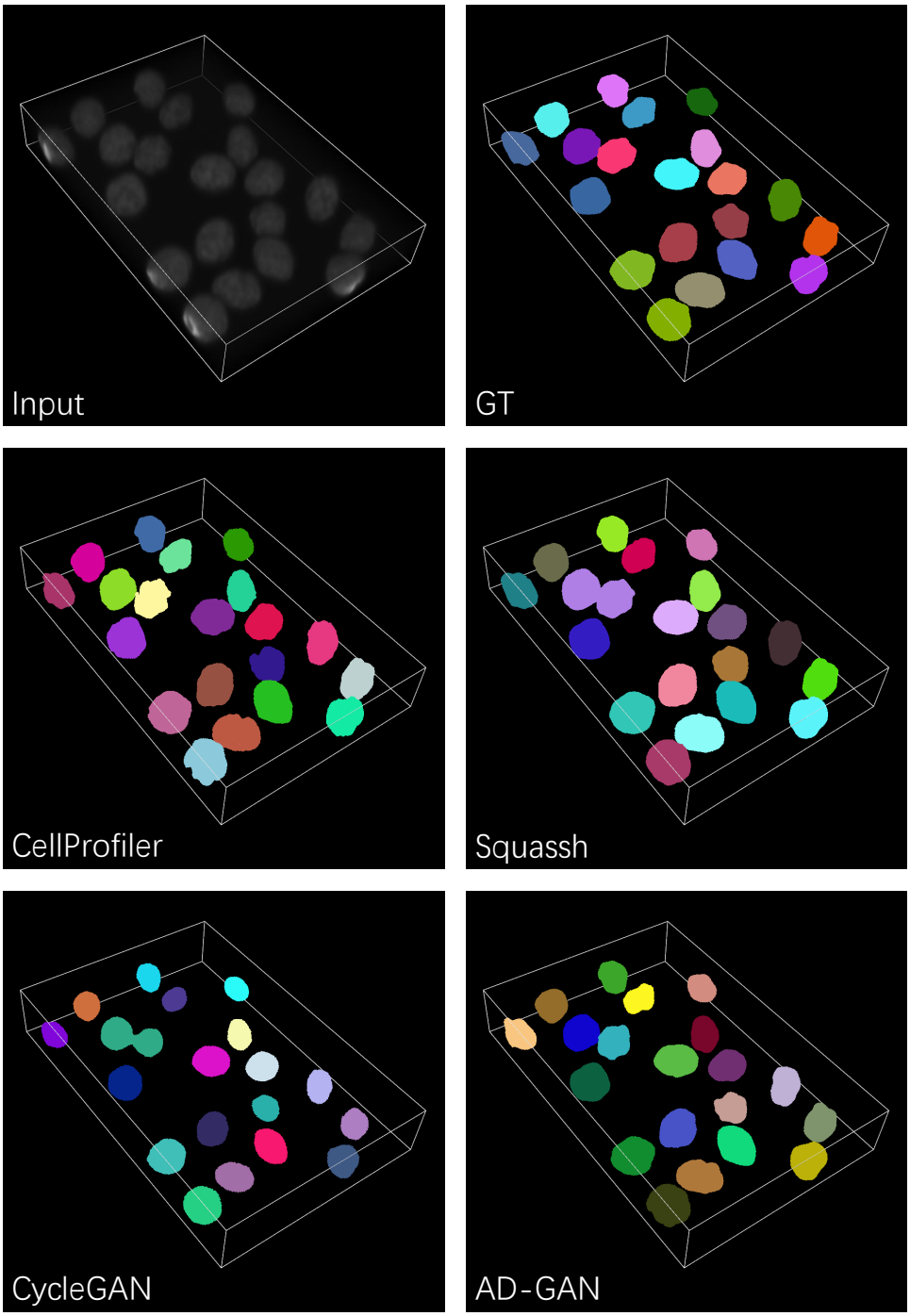

Fig. 3. Visualization of segmentation results on 3D data BBBC024.

I. Parameter Selection in Synthesizing Masks

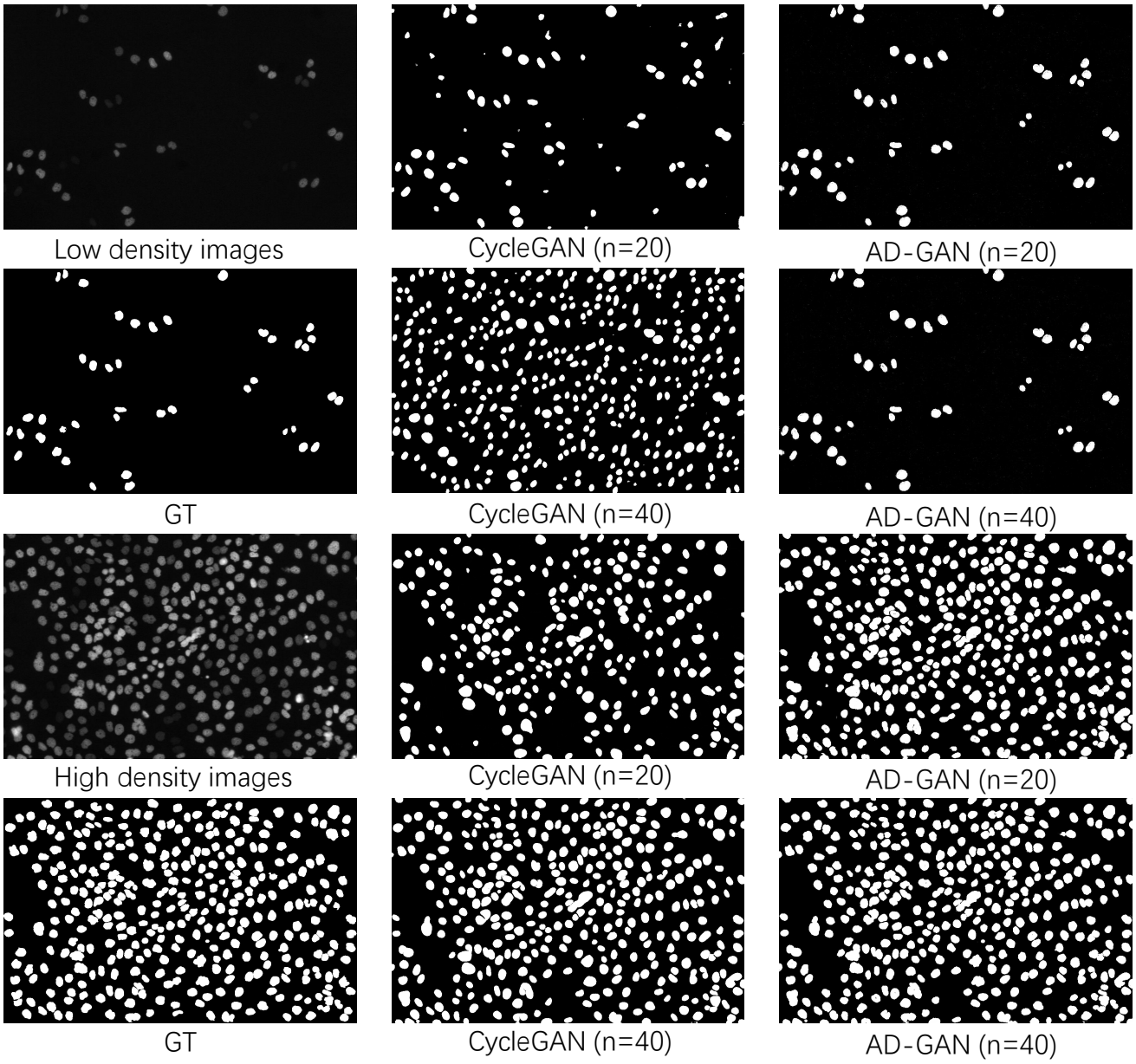

Fig. 4. Comparison results on image-to-mask translation with different nuclei number n in a synthetic mask.

J. Visualization of Generative Results

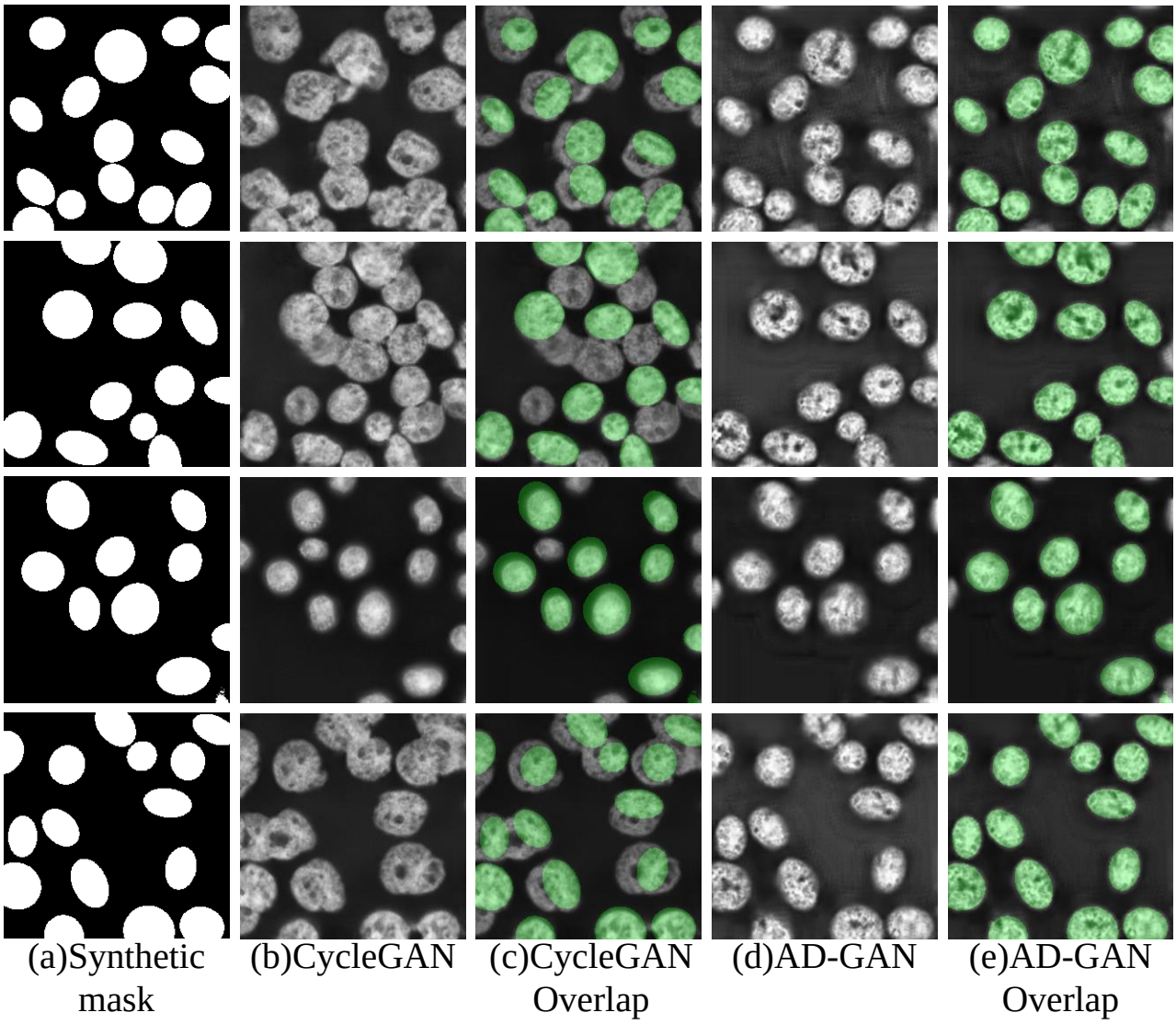

Fig. 5. Generative results comparison from mask to images on datasets HaCaT. (a): The randomly synthetic masks; (b) and (d): CycleGAN's and AD-GAN's generated images from (a); (c) and (e): overlapped images containing both synthetic mask and generated images. It is argued here that two-stage pipelines cannot guarantee good segmentation performance if the semantic object one-to-one mapping cannot be learned well at the first stage. More online generation results can be shown in the codes by changing the seed.

REFERENCES

[1] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, "Stargan v2: Diverse image synthesis for multiple domains," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.
[2] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Eur. Conf. Comput. Vis.*, 2018.
[3] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Int. Conf. Comput. Vis.*, 2017, pp. 1510–1519.
[4] J. Chen, Y. Lu, Q. Yu, X. Luo, E. Adeli, Y. Wang, L. Lu, A. L. Yuille, and Y. Zhou, "Transunet: Transformers make strong encoders for medical image segmentation," *arXiv preprint arXiv:2102.04306*, 2021.
[5] A. H. Liu, Y.-C. Liu, Y.-Y. Yeh, and Y.-C. F. Wang, "A unified feature disentangler for multi-domain image translation and manipulation," in *Adv. Neural Inform. Process. Syst.*, 2018.