Adipocyte Cell Imaging Challenge

SOFT MATTER LAB @ GU

Benjamin Midtvedt
Jesús Pineda
Saga Helgadottir
Daniel Midtvedt
Giovanni Volpe

1. HIGH LEVEL DESCRIPTION OF SUBMITTED SOLUTION

A high-level description of our solution is shown in Figure 1. Briefly, after being aligned, normalized and augmented (see details below), the stack of brightfield images are passed as input to our first-stage model (Model 1 in Figure 1), which is trained to output three virtually-stained fluorescence images (nuclei, lipids, and cytoplasm). Afterwards, the brightfield images and all three generated fluorescence images are passed as input to each of our second-stage models (Models 2A, 2B and 2C in Figure 1), which are trained specifically to output the fluorescence images for nuclei (Model 2A), lipids (Model 2B), and cytoplasm (Model 2C). These second-stage models are useful to correct details in the generated fluorescence images, but they increase the overall computational complexity of the inference. Thus, we only use these second-stage models for the generation of nuclei and cytoplasm fluorescence images for the 20X magnification case, where we see a significant improvement in the mean absolute error (MAE).

For all neural network models (Models 1, 2A, 2B, and 2C in Figure 1), we employ a patch-based generative adversarial network (Patch-GAN), because GANs have been successfully employed in the literature to perform virtual staining [1–3]. Specifically, we employ a Patch-GAN architecture with a U-Net as *generator* and a convolutional encoder as *discriminator*. For Model 1, the inputs of the generator are a stack of brightfield images of the same xy-field of view acquired at different z-positions and its output are virtually stained fluorescence images of nuclei, lipids, and cytoplasm. The inputs of the discriminator are the brightfield images and the corresponding fluorescence images, which can be either the ground-truth fluorescence images or those predicted by the generator, and aims to distinguish between these two cases.[1] We train three independent set of models, one for each magnification setting (60X, 40X, 20X).

To implement and train the Patch-GANs, we have used a specialized version of DeepTrack 2.0, a freeware comprehensive software for quantitative microscopy using deep learning that we have recently developed [4, 5]. All the code is provided in the folder `apido` in the GitHub repository, where the core DeepTrack 2.0 code is in the folder `deeptrack` and the specialized objects are `models.py` (the Patch-GAN model), `metrics.py` (the metrics and losses), `plotting.py`, and `util.py`.

The details of the data processing, training and analysis workflow are:

1. **Data pre-processing: Image alignment.** The input brightfield images are corrected to ensure alignment with the output fluorescence images. This is realized through an affine transformation whose parameters depend on the magnification and the site, but are the same for all wells. **Pixel-value normalization.** The pixel values are normalized to ensure that they are in $[-1, 1]$. See details in section 2 "Data processing".

2. **Model architecture definition.** The specific model architecture is determined by three parameters: the depth of the U-Net (generator), the depth of the convolutional

[1] For Models 2A, 2B, and 2C the inputs to the generator include also the virtually-stained images obtained by Model 1 and each discriminator operated on only one fluorescence image.

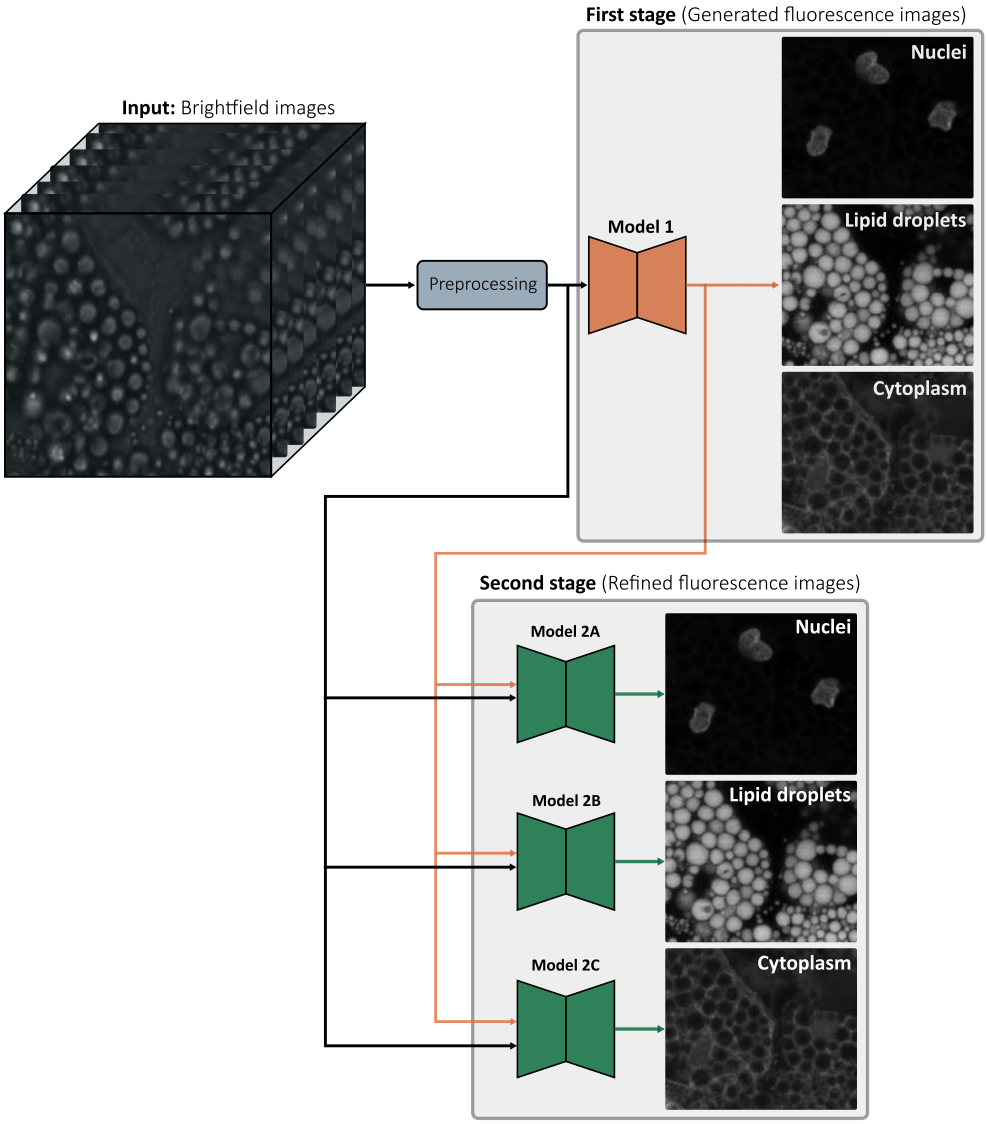

Fig. 1. High-level description of our solution. Model 1. After the stack of brightfield images is preprocessed to correct misalignements, Model 1 uses them to generate three virtually-stained fluorescence images for nuclei, lipids, and cytoplasm, respectively. **Models 2A, 2B, and 2C.** An optional second-stage analysis refines these virtually-stained images using Models 2A, 2B, and 2C for nuclei, lipids, and cytoplasm, respectively. Each of these three models takes as inputs the brightfield images and the three virtually-stained fluorescence images genertd by Model 1. All models are Patch-GANs with the same hyper-parameters (see Figure 3). We train three independent sets of models, one for each magnification setting (60X, 40X, 20X).

encoder (discriminator), and the MAE loss weight (which determines the relative weight of the U-Net MAE loss vs. the adversarial loss). We use the same hyper-parameters for all models. See details in section 3 "Model architecture".

3. **Training (Model 1).** The first-stage network (Model 1 in Figure 1) is trained using patches extracted from the brightfield images and the ground-truth fluorescence im-

ages, augmented in various ways. Apart from the number of epochs and the batch size, there is only one parameter for the training: the dimension of the patches, which should be large enough to capture the features of interest (but might be limited by the available computational resources). See details in section 4 "Model training". Furthermore, we provide Jupyter notebooks that reproduce the training:
`Train 60X Virtual Stainer.ipynb`,
`Train 40X Virtual Stainer.ipynb`,
and `Train 20X Virtual Stainer.ipynb`.
These notebooks can be easily adapted also for other virtual staining applications.

4. **Training (Models 2A, 2B, and 2C).** The second-stage networks (Models 2A, 2B and 2C in Figure 1) have the same architecture and training as Model 1, except that these networks take as input the previously generated fluorescence images in addition to the brightfield images. We see this as an optional second stage because of the necessary tradeoff between image quality and computational complexity of the inference.

5. **Testing with validation images.** The trained Patch-GANs are tested using validation images. The use of the trained models is exemplified in Jupyter notebooks:
`Stain 60X Data.ipynb`,
`Stain 40X Data.ipynb`,
and `Stain 20X Data.ipynb`.

6. **Testing with CellProfiler.** The morphological properties of the resulting virtually-stained fluorescence images are tested with CellProfiler. Interestingly, we observe that the best pixel-value MAE does not necessarily correlate with the best CellProfiler-feature MAE; therefore, a tradeoff between the two is necessary. See details in section 5 "Analysis of model performance/output".

2. DATA PROCESSING

The images are pre-processed before being used to train the network. This pre-processing includes three operations: image alignment; pixel-value normalization; and patch augmentation.

Image alignment. As a first step, the brightfield images and corresponding fluorescence images need to be aligned. In fact, we observed that the brightfield images and the fluorescence images are consistently slightly misaligned. To quantify this misalignment, we compute the cross-correlation of the brightfield images with the lipid droplet images via the Wiener-Khinchin theorem (`Calculate Offset.ipynb`). [2] Specifically, the images are divided into patches of 256×256 pixels, and their correlation is computed as the inverse Fourier transform of the product of their Fourier transforms:

$$C(x,y) = \mathcal{F}^{-1}(\mathcal{F}(I_{\mathrm{BF}})\overline{\mathcal{F}(I_{\mathrm{L}})}), \tag{1}$$

where I_{L} denotes the lipid image, I_{BF} denotes the brightfield image, and the bar denotes complex conjugation. An example of this correlation for site 06 of well D04 at 20X magnification is shown in Figure 2a. A Gaussian peak is fitted to the resulting correlation map to obtain the misalignment of the two channels, and the offset of this peak relative to $(0,0)$ defines a misalignment vector (δ_x, δ_y) (arrow in Figure 2a).

We found that, for each magnification, the misalignment vector varies between sites on the same well, but remains fairly consistent on corresponding site across wells.[3] Based on the circular arrangement of the sites within the wells, we assign an angle ϕ_i to each site such that the sites are uniformly distributed over the edge of a circle with unit radius, and set

[2] We have also checked that all brightfield images in each stack are aligned, and that the three fluorescence images of nuclei, lipids and cytoplasm are aligned. In both cases, this is verified to within $\approx$1 pixel.

[3] This misalignment is probably due to some optical effects related to the different paths in the brightfield and fluorescence configurations.

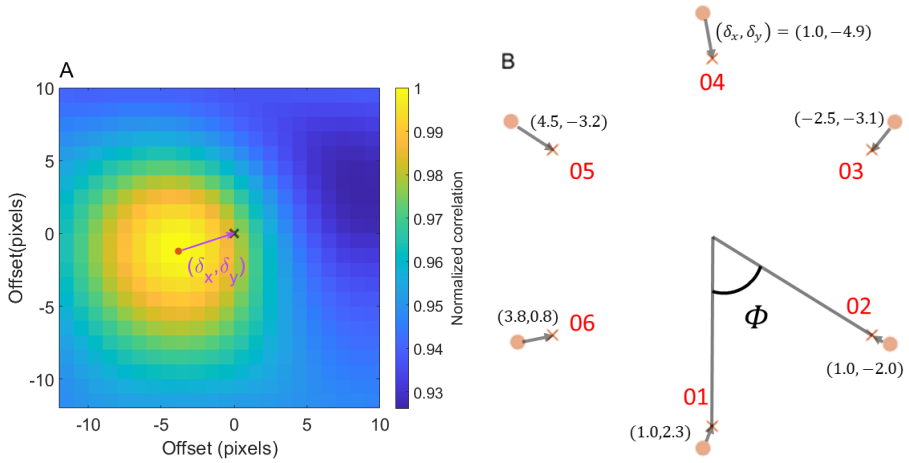

Fig. 2. Misalignment between brightfield and fluorescence images. A Correlation matrix calculated between the brightfield image ($z = 1$ and the lipid droplet image for magnification 20X, well D04, site 06). For two perfectly aligned images the correlation matrix is expected to be sharply peaked around $(0,0)$ (gray cross). Thus, the position of the peak in the correlation matrix (orange dot) relative to $(0,0)$ defines the misalignment vector (δ_x, δ_y) between the two images. **B** For each magnification, the misalignment depends on the position of the site within the well, but is consistent across wells. Here, we show the vectors for magnification 20X. Orange dots correspond to the average estimated misalignment at each site, and the crosses correspond to the position of the site within the well, according to our parameterization.

$\phi_1 = 0$ for site $i = 1$. Based on this parameterization, the misalignment is well-described by a linear function of the position of the site within the well, such that the misalignment vector is

$$\delta_{x,i} = A_x \sin(\phi_i) + B_x \cos(\phi_i) + C_x$$
$$\delta_{y,i} = A_y \sin(\phi_i) + B_y \cos(\phi_i) + C_y$$

where A_x, B_x, C_x, A_y, B_y, C_y are constants that depend on the magnification (see Table 1), and i is the site index (e.g., for magnification 20X, $i = 1,2,3,4,5,6$ and $\phi_i = \frac{i-1}{6}2\pi$). This allows us to predict and correct for the misalignment based on the site position and magnification. The correction vectors are shown in Figure 2b for the case of magnification 20X, while the correction parameters for all three magnifications are provided in Table 1.

Furthermore, dividing the images into 256×256 patches prior to computing the cross-correlation allows us to spatially resolve the misalignment within the images. We found that there is a constant gradient in the misalignment within each image, consistent with a slight difference in magnification between the brightfield images and the fluorescence images. This is particularly prominent for the samples imaged at 20X magnification, for which the difference in magnification is estimated to be about 0.12%. This is corrected by rescaling the images by a constant factor M (see Table 1).

We therefore define a site-dependent affine transformation that has to be applied to the brightfield images in order to align them with the corresponding fluorescence images. This transformation corrects both the misalignment and the difference in magnification. Finally, we determine the affine transformation iteratively, by:

1. estimating the affine transformation using the above procedure,
2. using `ndarray.affine_transformation` to apply the transformation to the brightfield images,

3. estimating the affine transformation again on the transformed images, until the affine transformation between the transformed brightfield images and the fluorescence images is close to unitary.

Based on the determined affine transformation (from the mismatch in magnification and image misalignment) the brightfield images are corrected prior to training/evaluation using the function `ndarray.affine_transformation`.

Pixel-value normalization. As a second step, the pixel-values of the images are rescaled as

$$\hat{x} = \tanh\left[\frac{x - \mathrm{mean}(x)}{\mathrm{std}(x)}\right], \tag{2}$$

where x is the pixel intensity of the original image and $\hat{x}$ is that of the rescaled image, while $\mathrm{mean}(x)$ and $\mathrm{std}(x)$ are the average and standard deviation of the pixel values of the intensity over the image. This choice of normalization has the advantage of not assuming that the staining procedure and illumination conditions are identical for all images, and it is therefore more robust than a global normalization over the entire dataset. Furthermore, by using statistical properties of the distribution of intensities rather than the minimum and maximum of the intensities for normalization, we preserve a local correspondence between the intensities of the different channels which aids the training procedure. Finally, the choice of $\tanh$ as a normalization function ensures that all values fall in the range $[-1, 1]$, while mitigating the effect of outliers in the intensity distribution. The normalization is reverted in the last layer of the generator, to produce images with the same range of values as the original image.

Patch augmentation. The data augmentation part is performed on-the-fly during training. The images are cropped into images of fixed sizes at random locations. The resulting image crops are rotated, mirrored, elastically deformed and sheared to further expand the dataset. Finally, patches are extracted from these crops to perform the training.

A. Data Processing Parameters

The parameters of the affine transformation applied to the brightfield are given in Table 1.

Magnification	M	A_x (px)	A_y (px)	B_x (px)	B_y (px)	C_x (px)	C_y (px)
60X	0.99975	2.31	-0.014	-0.032	-2.31	-0.84	0.81
40X	0.9996	2.49	-0.17	-0.030	-2.60	-0.96	0.62
20X	0.9988	3.95	-0.20	0.067	-4.09	-1.22	0.76

Table 1. Parameters for affine transformation to align the brigtfield images to the target fluorescence images.

3. MODEL ARCHITECTURE

The model architecture is shown in Figure 3. All models in Figure 1 consist of two neural networks working in tandem: a *generator* and a *discriminator*. The generator receives as input a stack of brightfield images of the same field of view acquired at different z-positions and generates virtually stained fluorescence images of nuclei, lipids, and cytoplasm. The discriminator, in turn, evaluates the quality of the three generated images and determines whether the generated stained images plausibly could have been drawn from fluorescently stained samples.

The generator is based on the U-Net-architecture, in which the input image is first downsampled to a smaller representation, then upsampled to its original size, with skip connections between the downsampling path and the upsampling path to retain local information. The

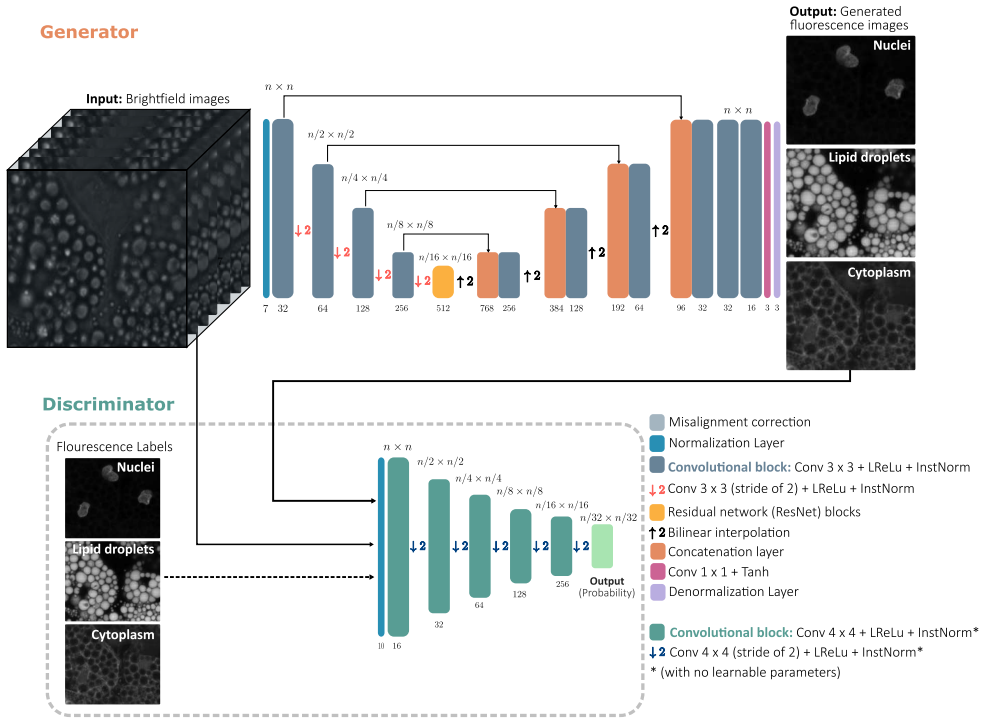

Fig. 3. Neural network architecture based on a Patch-GAN. The generator transforms an input stack of brightfield images into virtually-stained fluorescence images of nuclei, lipids, and cytoplasm, using a U-Net architecture with the most condensed layer being replaced by two residual network blocks [6]. The discriminator is designed similar to the PatchGan discriminator [7] and receives both the brightfield images and fluorescence images (either the ground-truth fluorescence images or those predicted by the generator). In the first layer of the generator, we normalize each input channel between -1 and 1 using Equation 2. The U-Net encoder consists of convolutional blocks followed by strided convolutions for downsampling. Each convolutional block contains two sequences of 3×3 convolutional layers. In the decoder, we use bilinear interpolations for upsampling followed by concatenations layers and convolutional blocks. Next, the hyperbolic tangent activation (tanh) is employed to transform the output to the range $[-1, 1]$. In the last layer of the U-Net, the network learns to denormalize the output images back to ground-truth pixel values by scaling and adding an offset to the output. Every layer in the generator, except the last two layers, is followed by an instance normalization ($\alpha = 2$) and a Leaky Relu activation layer. The inputs to the discriminator are newly normalized to make the predictions. The discriminator's convolutional blocks consist of 4×4 convolutional layers. We use 4×4 strided convolutions for downsampling. In all layers in the discriminator, we use instance normalization (with no learnable parameters) and Leaky Relu activation. Finally, the discriminator outputs an $n/32 \times n/32$ single channel tensor containing the predicted probability for each patch.
This figure shows the architecture for Model 1 in Figure 1; the architecture for Models 2A, 2B, and 2C is essentially the same, but the input of the generators including also the three virtually-stained fluorescence images, the output of each generator consisting of a single fluorescence image, and the input of the discriminator including only this fluorescence image and the brightfield images.

bottleneck of the U-Net architecture is composed of two residual networks (ResNet) blocks [6], each with 512 feature channels. We observed that using ResNet blocks helps to estimate a more descriptive data latent space and tackles vanishing gradient problems [2, 4, 8].

The discriminator follows a conditional PatchGan architecture [7], which receives the brightfield images and fluorescence images (either the ground-truth fluorescence images or the virtually stained images), divides them into overlapping patches, and classifies each patch as real or fake, rather than using a single descriptor. This splitting arises naturally as a consequence of the discriminator's convolutional architecture [9]. This network outputs a single-channel tensor in which each pixel represents the predicted probability for each patch.

In addition to these three models acting on the stack of brightfield images at the various magnifications, we have found that the results for the lowest magnification (20X) can be significantly improved by adding a second inference step, aimed at refining the prediction of the first model (Figure 1). This step consists of three independently trained Patch-GANs, one for each fluorescence image (here, we only used this step for the nuclei and cytoplasm images). The models take both the brightfield images as well as the outputs from the first inference step as input, and the output from each of the models is the corresponding refined virtually stained image.

A. Model Parameters

There are three fundamental hyper-parameters of the model architecture that can be tuned:[4]

1. **Depth of the U-Net generator.** This determines the number of downsampling and upsampling blocks.
2. **Depth of the convolutional encoder discriminator.** This determines the number of downsampling blocks
3. **MAE loss weight.** The loss function of the generator is a weighted sum of the adversarial loss (typical of GANs) and the MAE loss between the generated images and the target images. See details in section 4C "Loss Function".

We remark that the activation function used throughout the models is Leaky ReLu, which is defined as $\Phi(x) = \alpha \cdot x$, where $\alpha = 1$ for $x > 0$ and $0 < \alpha < 1$ for $x < 0$. In contrast to standard ReLu, this choice of activation retains a gradient in the backpropagation step even for layer outputs $x < 0$. We have used $\alpha = 0.2$ for $x < 0$.

B. Model Size

The total number of trainable parameters of the $60X$ and $40X$ models (which include only Model 1) is 11-million, of which 8.8 million parameters are related to the image generation part of the models. For the $20X$ model (which includes Models 1, 2A, and 2C), the corresponding values are 33-million parameters, of which 26.4-million parameters are related to the generator. The size of the model is 600 MB for 60x and 40x. For 20x, the model size is 1.8 GB. The exact values are reported in Table 2.

Magnification	Total parameters	Generator parameters	Size
60X	10 885 770	8 784 073	600 MB
40X	10 885 770	8 784 073	600 MB
20X	32 659 016	26 353 871	1.8 GB

Table 2. Number of trainable parameters and size of the trained models for the different magnifications

[4] Note that the number of convolutional layers per block and the number of channels in each convolutional layer are fixed in our model.

C. Model Output

The output of the model are the three virtually stained images, containing the nuclei (output channel 1), lipid droplets (output channel 2) and cytoplasm (output channel 3). The size of the output virtually-stained images is equal to that of the target fluorescence images.

4. MODEL TRAINING

The training of the model comprises a pipeline for loading, preprocessing and augmenting images on-the-fly, and a single training step for each of the models. For the 20X magnification, we proceed to train two additional models for the refinement of the generated nuclei and cytoplasm images, respectively.

Prior to training, we randomly split the available training dataset into two disjoint sets constituting the *training set* and the *validation set*, as shown in table 3.[5]

60x		40x		20x	
Well	Site	Well	Site	Well	Site
B02	2	B04	4	B04	1
C04	4	C03	8	D03	5
B04	3	C02	8	B04	2
C02	1	C04	1	C04	2
D02	3	C02	5	C03	5
B03	1	B03	1	D02	5
D04	12	B03	4	C02	1
B04	1	B03	7	C04	3
B03	10	D04	5		
B04	4	C04	2		
C02	4				
D02	9				
C04	9				
D04	1				
C02	6				

Table 3. The wells and sites of the images used exclusively for validation while training the models.

A. Training Scheme Overview

The training scheme consists of the following steps:

1. Define a pipeline for loading the training dataset image by image.
2. Define the set of augmentations to be performed on the dataset. We used flipping, rotations, and affine transformations.

[5] We remark that we decided early on not to employ more complex training procedures that can make more use of the available training dataset (e.g., n-fold cross-validation or final retraining on all data), because the available training data were sufficient and the computational resources were limited.

3. Normalize the images and correct the misalignment between brightfield and fluorescence images based on the site position as described in section 2 "Data processing".

4. Define a Python generator which performs the above steps on-the-fly during training.

5. Train the model for ~ 300 epochs, with 32 batches of 8 images per epoch.

We provide Jupyter notebooks that reproduce the training:
`Train 60X Virtual Stainer.ipynb`,
`Train 40X Virtual Stainer.ipynb`,
and `Train 20X Virtual Stainer.ipynb`.
These notebooks can be easily adapted also for other virtual staining applications.

B. Pre-Training

Not applicable.

C. Loss Function

The generator loss $\mathcal{L}_{\text{gen}}$ is a weighted sum of the adversarial loss (which arises naturally when using GANs) and the MAE loss between the generated images and the ground-truth fluorescence images, i.e.,

$$\mathcal{L}_{\text{gen}} = \beta \cdot \text{MAE}\left\{z_{\text{label}}, z_{\text{output}}\right\} + \left(1 - \text{D}(z_{\text{output}})\right)^2, \tag{3}$$

where β is the weighting factor or regularization parameter, and the discriminator loss is defined as

$$\mathcal{L}_{\text{disc}} = \text{D}\left(z_{\text{output}}\right)^2 + \left(1 - \text{D}(z_{\text{label}})\right)^2, \tag{4}$$

where $\text{D}(\cdot)$ denotes the discriminator network prediction, z_{label} refers to the ground-truth fluorescence images, and z_{output} are the generated images. Note that the generator loss function, $\mathcal{L}_{\text{gen}}$, aims to minimize the MAE between the generator output image and its label, based on the regularization parameter β. β is the coefficient to balance the loss between the MAE and the adversarial loss. The values of β for the various magnifications are given in Table 4.

Magnification	β
20X	0.001
40X	0.004
60X	0.001

Table 4. Weighting of the MAE in the model loss function. These values were determined experimentally to reduce the dynamic range of the MAE and balance the contribution of each element that compose the overall loss function.

Note that in Equations 3, and 4, we employ the least-squares adversarial loss function, which contrary to the conventional GAN loss, provides more stability during training and acts as a second barrier against the vanishing gradient problem (the first is the use of ResNet blocks in the U-Net generator) [10]. This loss function is implemented by optimizing the mean squared error (MSE) for the discriminator loss and the adversarial term in the generator, and using target values of 1s and 0s for experimental and generated images, respectively.

D. Training Parameters

The parameters defining the training stage are the chosen optimizer and the corresponding learning rate. These parameters are detailed in Table 5.

Magnification	Optimizer	Learning rate	β_1
60X	Adam	0.002	0.5
40X	Adam	0.002	0.5
20X	Adam	0.002	0.5
20X (second stage)	Adam	0.002	0.5

Table 5. Parameters for training the first and second inference step. In all cases, we use Adam optimizer with learning rate of 0.002 and $\beta_1 = 0.5$ (the exponential decay rate for the 1st moment estimates for the Adam optimizer).

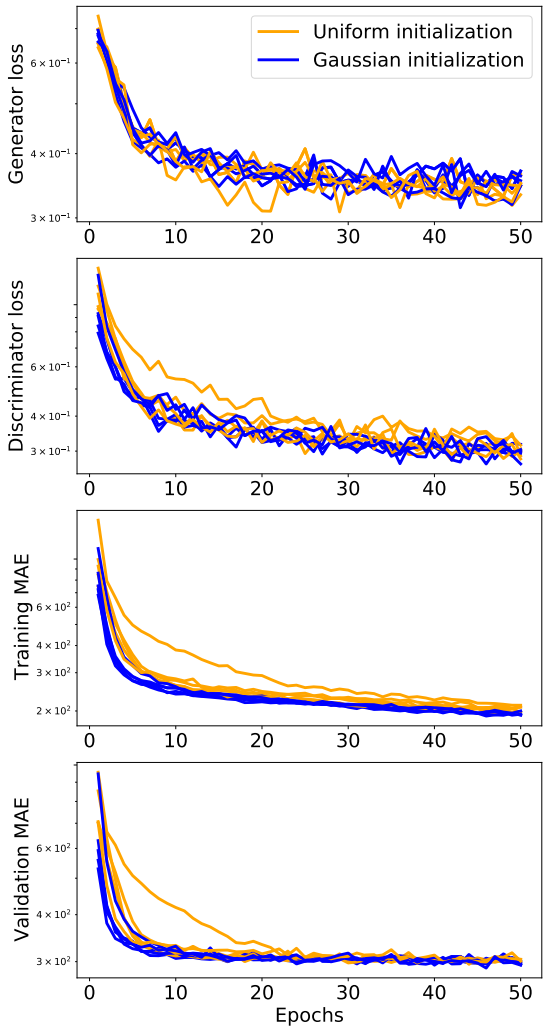

Fig. 4. Training consistency. The generator and discriminator losses, and the training and validation MAE for each epoch in the training of 10 models with 60X images, 5 models with uniform initialization of the network weights and 5 with Gaussian initialization. All models converge to the same result independent of the initialization, showcases robustness and consistency of our model.

E. Consistency in training results

To check the consistence of our training results, we trained 10 models for 60X magnification for 50 epochs, 5 with uniform initialization of the neural network weights (standard option

provided by Keras) and 5 with Gaussian initialization (mean 0 and standard deviation 0.02, which has been shown to be optimal for GAN architectures [9]). In Figure 4, we plot the generator and discriminator losses, and the training and validation MAE for each epoch. It can be seen that, independently of the initialization, all models converge to the same result. This makes us confident that the model we propose is robust and can produce consistent results across different training sessions.

F. Training time

For each magnification, the model was trained on roughly 100000 samples of 512 by 512 pixel crops, split into 400 epochs of size 8 batches. Each epoch takes 11 seconds on a RTX A100 GPU, for a total training time of just under 1.5 hours. The refinement models were trained similarly for 300 epochs, for a total training time of just under 1 hour. The final per magnification training time is shown in Table 6.

Magnification	Training time (h)
60X	1.5
40X	1.5
20X	3.5

Table 6. The training time of each the models for each magnification in hours. The 20x training is longer than the others since two refinement models were trained.

5. ANALYSIS OF MODEL PERFORMANCE/OUTPUT

MAE analysis

To evaluate the model output, we calculate the mean absolute error (MAE) between the predicted and ground-truth fluorescence images using both the pixel values and the features extracted by the CellProfiler. The MAE metrics (CellProfiler-feature MAE, pixel-value MAE, and average MAE) from our validation dataset (Table 3) are shown in Table 7.

Magnification	CellProfiler MAE	Pixel-value MAE	Average MAE
60X	0.208	0.327	0.268
40X	0.263	0.355	0.309
20X	0.238	0.370	0.304
Weighted	0.232	0.346	**0.289**

Table 7. CellProfiler-feature MAE, pixel-value MAE, and average MAE for all magnifications for the generated images from our validation dataset (Table 3). The last row reports the MAEs weighted by number of images for each magnification in the corresponding validation set.

Examples of virtually-stained images

Examples of generated fluorescence images for randomly selected brightfield images from our validation dataset (Table 3) for each magnification are shown in Figures 5, 6, and 7. The generated images for the rest of the validation dataset are in GitHub repository.

In general, the performance appears better for higher magnifications, which can be explained by the presence of more details in the brightfield images. It is also better for lipids and cytoplasm than for nuclei. This also can be understood considering that the nuclei are

essentially invisible in the brightfield images; therefore, we hypothesized that the network can only infer the nucleus position based on the spatial distribution of lipid droplets.[6] Furthermore, some artifacts are present in the virtually-stained images of the cytoplasm; these artifacts can be removed using a second-stage refinement stage (we have not done this because it increase the computational cost, but produces only a small improvement on the pixel-value and CellProfiler-feature MAE).

CellProfiler analysis

We calculate the CellProfiler parameters separately for the nuclei, lipid droplets and cytoplasm fluorescence images to determine how each of them are affected by various choices about the models and their training. This analysis led us to two main conclusions: (1) it is useful to correct the misalignment between the brightfield and fluorescence images (as described in section 2 "Data Processing"); (2) the there is a tradeoff between the MAE of the pixel values and that of the CellProfiler features.

Improvement of model performance when correcting for misalignement. In Figure 8, we see that a model that does not correct the misalignment between brightfield and fluorescence channels (described in section 2 "Data Processing") results in large jumps in the MAE for the corresponding values (compare blue lines for model without correction and orange lines for model with corrections), probably caused by a blurring of the lipid droplets. This leads to an increase in the average value of the MAE from the CellProfiler features of 50% (note that this analysis has been made on preliminary models, see symbols for models "before correction" and "after correction" in Figure 9). We therefore conclude that the correction of the misalignment is a useful preliminary step in the preparation of the training input and ground-truth images.

Tradeoff between pixel-value MAE and CellProfiler-feature MAE. Figure 9 shows that there is a tradeoff between the MAE of the pixel values and that of the CellProfiler features. In fact, this represented a surprising and unexpected observation as we naturally expected the optimal MAE for the CellProfiler to correspond to the optimal MAE for the

[6] In order for the network to determine this distribution, the models needs a broad scope, leading us to gradually increase the number of downsampling steps.

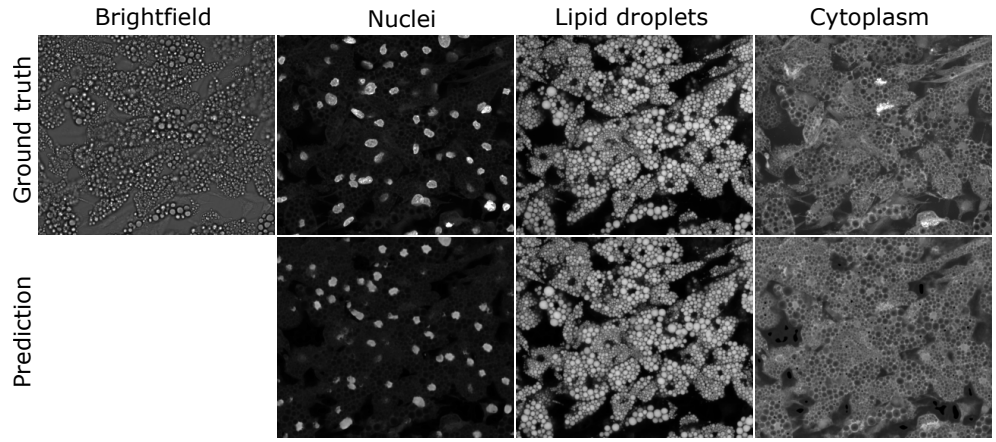

Fig. 5. Comparison of ground-truth fluorescently stained and virtually-strained images (60X). Top row, from left to right: Brightfield image (z=1) and ground-truth fluorescently stained nuclei, lipids, and cytoplasm. Bottom row, from left to right: Corresponding virtually-stained images. Note that the artifacts in the virtually-stained cytoplasm image can be eliminated using the second refinement stage. The randomly chosen image is for magnification 60X, well B04, site 01.

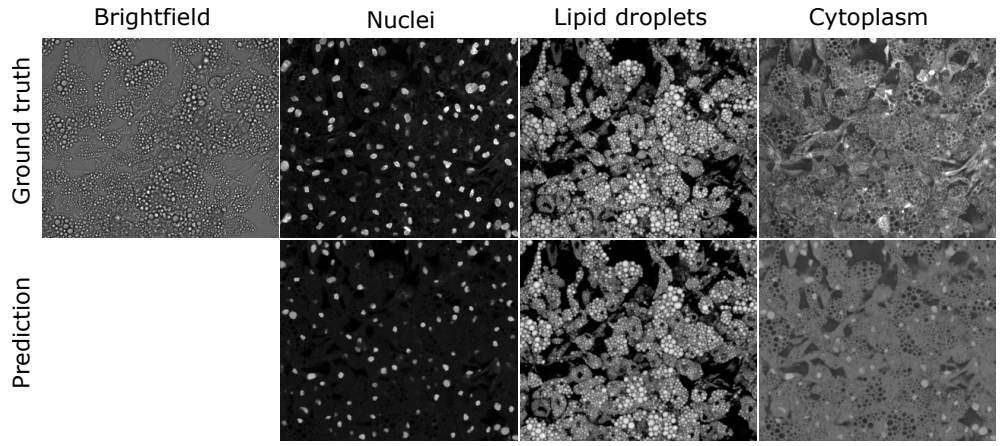

Fig. 6. Comparison of ground-truth fluorescently stained and virtually-strained images (40X). Top row, from left to right: Brightfield image (z=1) and ground-truth fluorescently stained nuclei, lipids, and cytoplasm. Bottom row, from left to right: Corresponding virtually-stained images. Note that the artifacts in the virtually-stained cytoplasm image can be eliminated using the second refinement stage. The randomly chosen image is for magnification 40X, well B03, site 01.

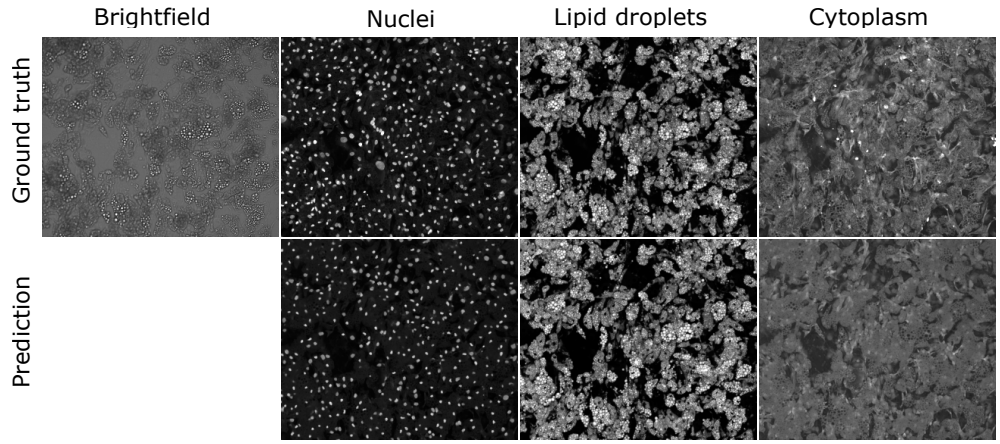

Fig. 7. Comparison of ground-truth fluorescently stained and virtually-stained images (20X). Top row, from left to right: Brightfield image (z=1) and ground-truth fluorescently stained nuclei, lipids, and cytoplasm. Bottom row, from left to right: Corresponding virtually-stained images. Note that the artifacts in the cytoplasm image have been removed by the second refinement stage. The randomly chosen image is for magnification 20X, well B04, site 02.

pixel values. Therefore, we carefully weighted the MAE loss (described in section 4.C "Loss Function") to find the optimal between these two metrics: the full symbols in Figure 9 represent the optimal models we employed for each of the magnifications.

6. MODEL EXECUTION END-TO-END

Detailed instructions to run the training and inference codes are given in the readme file on the GitHub repository.

Training. This is a step-by-step guide on how to train the model:

1. **Define input and output**: Set constants to determine the model parameters and input and output images.
2. **Load training data**: Define image loaders and image preprocessing and create the training image generation pipeline.

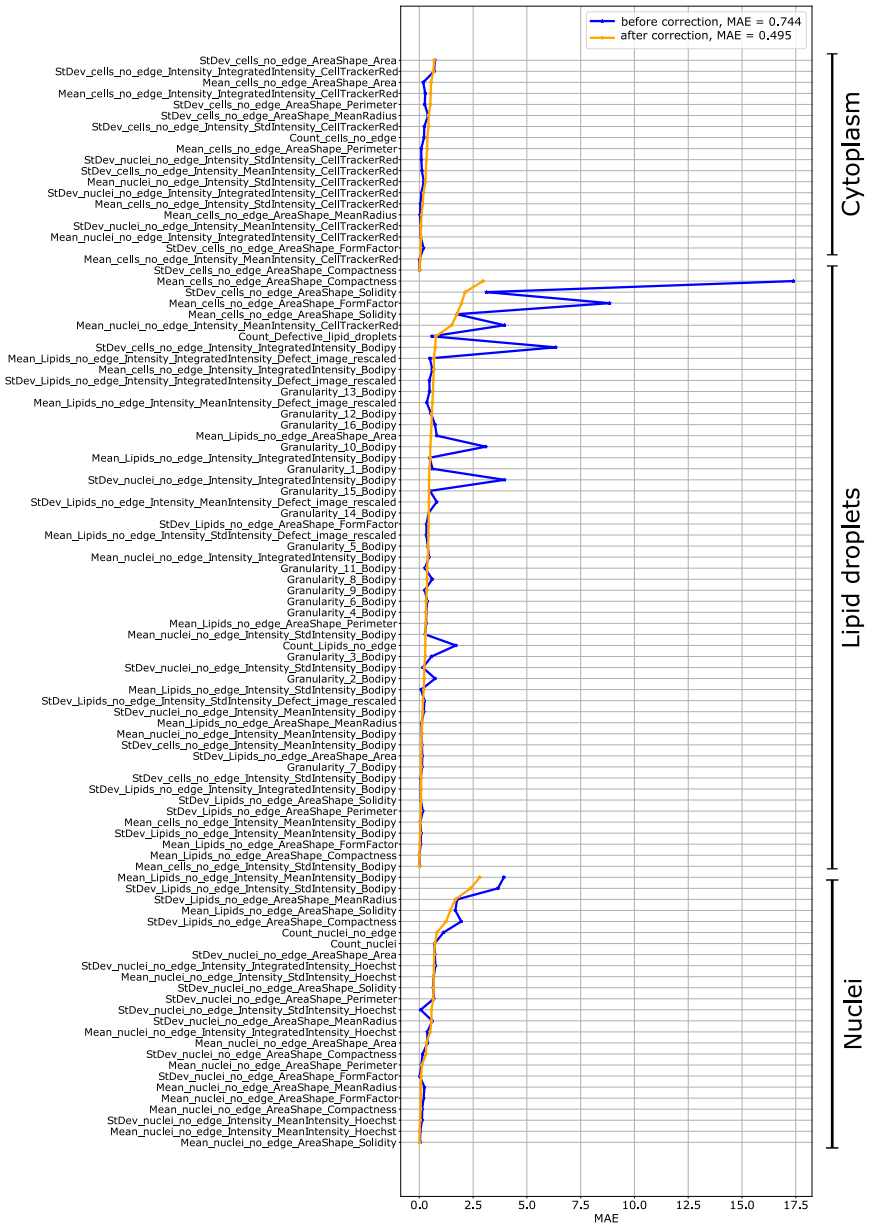

Fig. 8. MAE for CellProfiler features for models trained with images with and without correction of the misalignment between the brightfield and fluorescence images. MAE between generated and ground-truth fluorescence 20X images for features extracted by the CellProfiler (sorted top to bottom for cytoplasm, lipid droplets and nuclei) using a model trained without (blue) and with (orange) misalignment correction (see section 2 "Data Processing").

3. **Define image generator**: This will continuously generate images during the training of the model.
4. **Define model**: Define the generator and discriminator and the corresponding loss functions.
5. **Train model**: Execute the dataset pipeline and train the model.
6. **Visualize validation set**: Plot model predictions for images in the validation set.
7. **Save model**: Save the generator and discriminator separately.

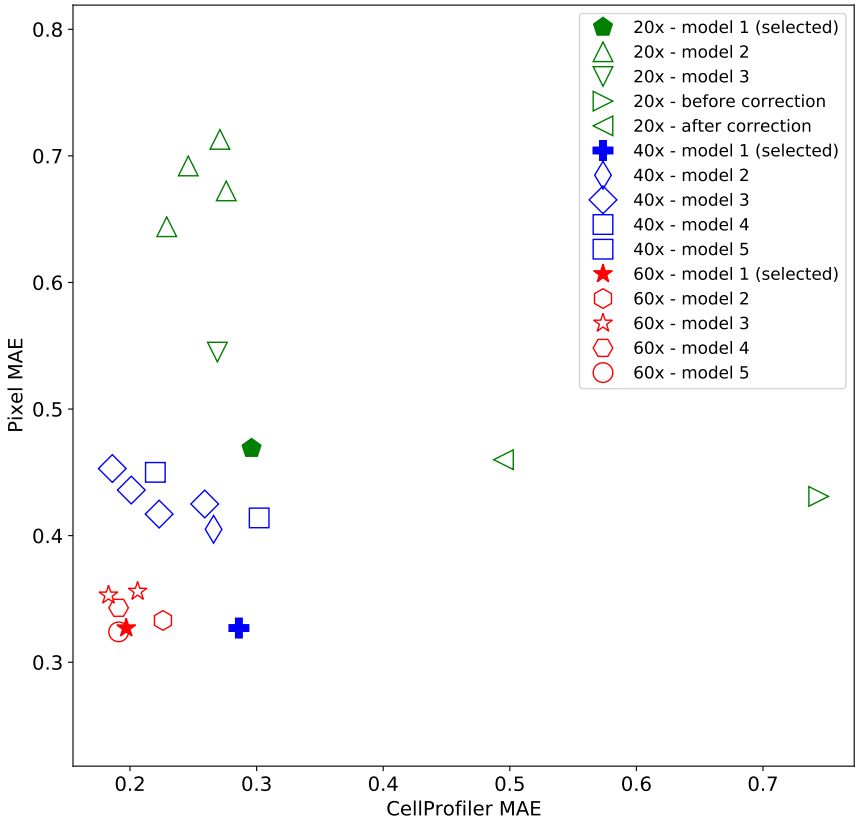

Fig. 9. The tradeoff between MAE for the pixel values and the CellProfiler features. MAEs for the pixel values plotted as a function of the MAEs for the CellProfiler features for various models for images at magnifications 20X (green), 40X (blue), and 60X (red). The full symbols represents the models that minimized the total MAE and computational complexity (a model using second stage refinement for cytoplasm for 60x images has a slightly lower total MAE but has a about 2x longer inference time). The same symbols represent the same model at different training epochs. The MAEs before and after correcting the misalignment between brightfield and fluorescence 20X images are also shown.

Furthermore, we provide Jupyter notebooks that reproduce the training and can be, therefore, used as a step-by-step guide to training the networks:
`Train 60X Virtual Stainer.ipynb`,
`Train 40X Virtual Stainer.ipynb`,
and `Train 20X Virtual Stainer.ipynb`.
These notebooks can be easily adapted also for other virtual staining applications.

Inference. This is a step-by-step guide on how to use the trained model for inference:

1. **Define input and output**: Set constants to determine the input and output images.
2. **Load pretrained model**.
3. **Load input data**: Define image loaders and image preprocessing and create the training image generation pipeline.
4. **Generate virtually-stained images**: Use the pretrained model to calculate virtually stained images from the brightfield images.
5. **Save generated images**.

Furthermore, the use of the trained models is exemplified in Jupyter notebooks:
`Stain 60X Data.ipynb`,
`Stain 40X Data.ipynb`,
and `Stain 20X Data.ipynb`.

7. DISCUSSION AND OUTLOOK

Since the signal in the brightfield images are due to local variations in refractive index within the sample, the underlying assumption when developing this model is that there is a correspondence between the refractive index of an object and the number of fluorescence photons emitted per unit volume in the respective fluorescence channel. This assumption is reasonable considering that the refractive index of a biological object is directly proportional to the concentration of macromolecules within the object [11]. Another assumption is that there is a clear difference in the morphological and optical properties of the three types of structures that are being virtually stained. The lipid droplets, consisting primarily of lipids at high concentration, have a larger refractive index than most other intracellular objects [12], and as such are clearly visible in the brightfield images. The nuclei and cytoplasm, on the other hand, are known to have very similar refractive index [13], and as a consequence there is very little information about the nuclei in the brighfield image. One may speculate that the network learns to localize the nuclei based on the spatial distribution of lipid droplets. Considering that the cell is typically at its thickest around the position of the nucleus, complementing the brightfield images with phase contrast images may give additional information that is helpful for increasing the robustness of the virtual nuclei staining.

REFERENCES

1. Y. Rivenson, H. Wang, Z. Wei, K. de Haan, Y. Zhang, Y. Wu, H. Günaydın, J. E. Zuckerman, T. Chong, A. E. Sisk *et al.*, "Virtual histological staining of unlabelled tissue-autofluorescence images via deep learning," Nat. Biomed. Eng. **3**, 466 (2019).
2. Y. N. Nygate, M. Levi, S. K. Mirsky, N. A. Turko, M. Rubin, I. Barnea, G. Dardikman-Yoffe, M. Haifler, A. Shalev, and N. T. Shaked, "Holographic virtual staining of individual biological cells," Proc. Natl. Acad. Sci. **117**, 9223–9231 (2020).
3. Y. Zhang, K. de Haan, Y. Rivenson, J. Li, A. Delis, and A. Ozcan, "Digital synthesis of histological stains using micro-structured and multiplexed virtual staining of label-free tissue," Light. Sci. & Appl. **9**, 1–13 (2020).

4. B. Midtvedt, S. Helgadottir, A. Argun, J. Pineda, D. Midtvedt, and G. Volpe, "Quantitative digital microscopy with deep learning," arXiv preprint arXiv:2010.08260 (2020).
5. B. Midtvedt, S. Helgadottir, A. Argun, J. Pineda, D. Midtvedt, and G. Volpe, "Deeptrack 2.0," https://github.com/softmatterlab/DeepTrack-2.0 (2020).
6. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition,* (2016), pp. 770–778.
7. P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition,* (2017), pp. 1125–1134.
8. Y. Lei, D. Li, H. Zhang, and X. Li, "Wavelet feature outdoor fingerprint localization based on resnet and deep convolution gan," Symmetry **12**, 1565 (2020).
9. D. Foster, *Generative deep learning: teaching machines to paint, write, compose, and play* (O'Reilly Media, 2019).
10. X. Zhu, Y. Liu, J. Li, T. Wan, and Z. Qin, "Emotion classification with data augmentation using generative adversarial networks," in *Pacific-Asia conference on knowledge discovery and data mining,* (Springer, 2018), pp. 349–360.
11. Y. Park, C. Depeursinge, and G. Popescu, "Quantitative phase imaging in biomedicine," Nat. Photonics **12**, 578–589 (2018).
12. I. Y. Yanina, E. N. Lazareva, and V. V. Tuchin, "Refractive index of adipose tissue and lipid droplet measured in wide spectral and temperature ranges," Appl. Opt. **57**, 4839–4848 (2018).
13. M. Schürmann, J. Scholze, P. Müller, J. Guck, and C. J. Chan, "Cell nuclei have lower refractive index and mass density than cytoplasm," J. Biophotonics **9**, 1068–1076 (2016).