Spheroid Segmentation using Multiscale Deep Adversarial Networks

Sajith Kecheril Sadanandan
Department of IT
Uppsala University
SciLifeLab
Uppsala, Sweden
sajith.ks@it.uu.se

Johan Karlsson
Discovery Sciences
Innovative Medicines
AstraZeneca
Gothenburg, Sweden
johan.karlsson1@astrazeneca.com

Carolina Wählby
Department of IT
Uppsala University
SciLifeLab
Uppsala, Sweden
carolina.wahlby@it.uu.se

Abstract

In this work, we segment spheroids with different sizes, shapes, and illumination conditions from bright-field microscopy images. To segment the spheroids we create a novel multiscale deep adversarial network with different deep feature extraction layers at different scales. We show that linearly increasing the adversarial loss contribution results in a stable segmentation algorithm for our dataset. We qualitatively and quantitatively compare the performance of our deep adversarial network with two other networks without adversarial losses. We show that our deep adversarial network performs better than the other two networks at segmenting the spheroids from our 2D bright-field microscopy images.

1. Introduction

Deep neural networks outperform traditional methods in several image analysis applications. A special type of the deep neural network known as Fully Convolutional Neural Networks (FCNNs) are typically used for image segmentation [8]. Recently, a special type of FCNN known as U-net [12] was successfully applied to segment cells from bright-field images. The FCNNs for segmentation consist of a contracting convolution path for a coarse feature extraction followed by an expanding convolution path for fine grained segmentation results. He et al. [6] showed that short skip connections or residual connections improved performance on image classification problems. The U-net utilized long skip connections for better segmentation performance. Drozdzal et al. [2] showed that both long and short skip connections were required for better segmentation performance. Szegedy et al. [14] showed that residual connections along with many parallel convolution paths improved the learning of deep networks.

Spheroids are a type of cell cultures growing outside an organism in a controlled environment. In this environment they grow as 3D structures and don't adhere to culture substrates, such as polystyrene in 2D monolayer cultures. The spheroids are used for high throughput drug screening experiments and they typically vary in sizes, shapes and textures depending on the biology and experimental environments [3]. To extract features from different sizes or scales multiscale FCNNs [4, 13] have been used. The multiscale FCNNs extract features at many scales of observations and combine the results. In this work, we also use multiscale FCNNs with long and short skip connections along with multiple convolution paths.

Deep generative adversarial networks [5] (GANs) are a type of deep neural networks used to artificially create data similar to the training dataset. A GAN usually consists of a generator network to create the new data and an adversarial network to discriminate the ground truth (real data) from the the artificially generated data (fake data). The generator and the adversarial networks are usually created as deep neural networks and they are trained simultaneously. The performance of both improves over iterations. The prediction from the adversarial network becomes $50 : 50$ when the generator network starts creating images similar to the ground truth images. Recently, Luc et al. [9] used GAN for semantic segmentation of natural scenes. They used a segmentation network as the generator and a classifier as the adversarial network.

In this work, we explore the benefits of combining FCNNs with GANs. There are two main contributions in this work: (1) we create a novel multiscale deep adversarial network with two different types of deep feature extraction at different scales, (2) we linearly increase the adversarial loss to the segmentation network to stabilize them. We quantitatively compare the segmentation performance of the proposed deep adversarial network with a baseline network without adversarial loss and a modified U-net [12] network. In Section 2, we describe the datasets we use, the design and training of the adversarial and baseline networks, how

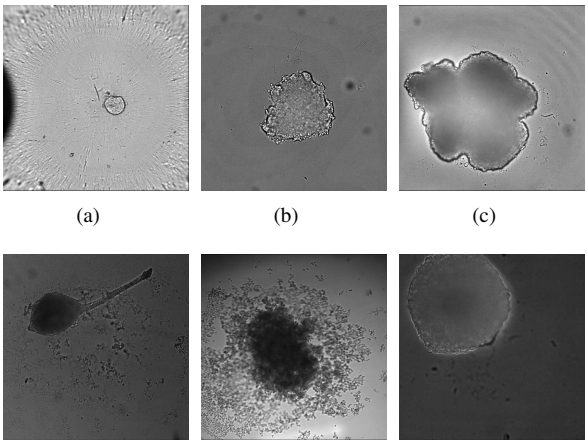


Figure 1. Input spheroid dataset. The cell clusters are formed in different sizes and shapes. (a-c) show the images from dataset 1 and (d-f) show images from dataset 2. The images are contrast enhanced for visualization.

we convert the probability maps from the networks into segmentation masks. In Section 3, we qualitatively and quantitatively evaluate the performance of the proposed network and compare our network with the baseline network and the U-net network.

2. Materials and methods

2.1. Input images

The input dataset consists of 2D bright-field images of spheroids representing a range of cell types, treatments, and experimental conditions. The sizes and shapes of the spheroids vary depending on how the cells respond to different treatments, but may also vary with varying cell types and culturing conditions [3], as shown in Fig. 1. The input images were acquired using different cameras and the images are of sizes 1080×1080 and 2160×2160 pixels in vertical and horizontal directions. The input images are categorized as easy or hard based on the difficulty in segmenting the dataset using conventional methods available in microscopy image analysis tools such as Columbus™ and MetaXpress®. The easy and hard datasets are hereafter referred to as dataset 1 and dataset 2, respectively. Example images from the datasets 1 and 2 are shown in Fig. 1. Dataset 1 consists of 2295 images and dataset 2 consists of 781 images.

2.2. Ground truth generation and data augmentation

We use two different approaches to create the ground truths for training. For dataset 1, we create the ground truths semi-automatically using conventional image analysis tools. After the images are segmented the results are manually curated to remove improper segmentations. For dataset 2, we manually annotate the input images to create the ground truths. We divide the datasets 1 and 2 to six approximately equal partitions. We set one of these partitions to be the test set. We use the rest of the five partitions for five fold cross validation. We perform all the hyper parameter optimizations, such as the values of λ (explained in Section 2.4), on the validation sets only.

The input images of sizes 1080×1080 and 2160×2160 are down-sampled to 480×480. After the resizing, we randomly select the fold and images from dataset 1 and dataset 2 at a ratio $70 : 30$. This ratio allows more samples from dataset 1, since dataset 1 contains more images than dataset 2. After selecting the images we randomly flip and rotate the images. We make both vertical and horizontal flips. The random rotations are made at $90, 180$ and 270 degrees. After the random spatial transformations we create a weighted label corresponding to each of the input images. We create the weighted labels considering the number of pixels in both foreground and background regions. The weighted labels force the network to learn on regions with higher weights than on the regions with lower weights. We calculate the weights as follows

$$W_{fg} = \frac{W_{bg} \times N_{bg}}{N_{fg}} \tag{1}$$

where W_{fg} is the weight of the foreground pixels, W_{bg} is the weight of the background pixels (here we set it to 1), N_{bg} is the number of background pixels and N_{fg} is the number of foreground pixels. The weighted labels give equal contribution to both foreground and background regions in the loss function. After calculating the dynamic weight labels we multiply the foreground weights with a constant value of 3 to force the network to learn more from the foreground regions. The training data for each fold consists of 6000 images after the data augmentation.

2.3. Deep neural network architectures

The deep neural network architecture we use in this work is inspired from a few recent deep neural network architectures such as U-net [12], long short skip connection networks [2], inception-residual network [14] and multiscale networks [4, 13].

2.3.1 Segmentation network

The architecture of the segmentation network is shown in Fig. 2 as Segmentation net. We use dilated convolutions [15] with kernels of size 3×3, dilations of 2×2 and strides 2×2 to reduce the spatial resolution of the input 480×480 images so that the network fits in the GPU memory. The dilated convolutions give larger receptive fields so that higher-

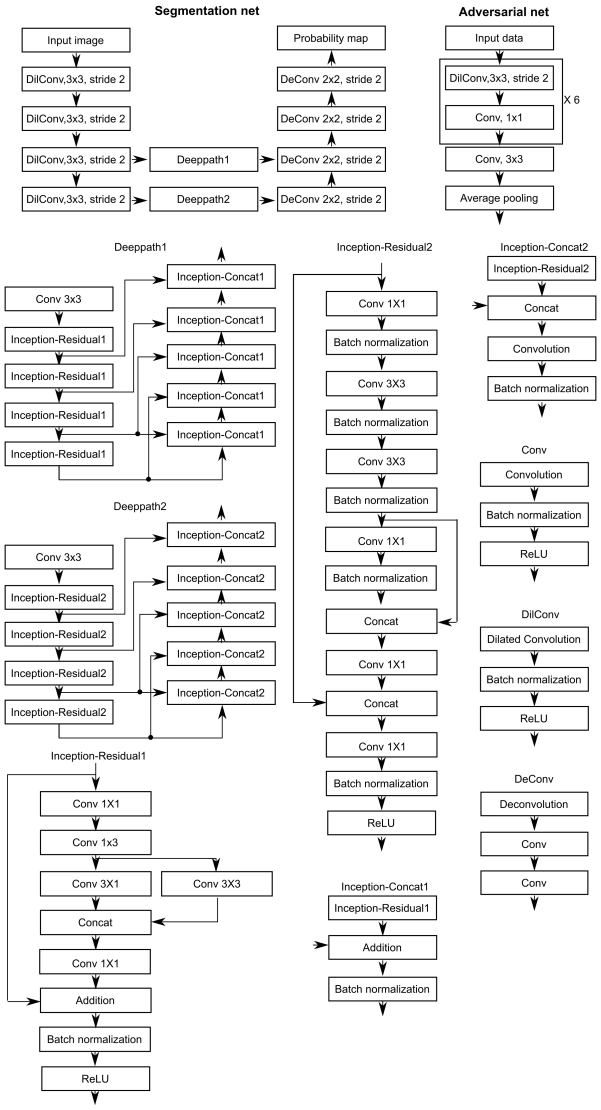


Figure 2. The architecture of the segmentation and adversarial network. The building blocks for each layer in the segmentation and adversarial networks is as illustrated in the second row.

level features are preserved while down-sampling the image. The first dilated convolution layer creates 60 feature maps and is doubled with every down-sampling. The input images contain spheroids of large variations in sizes as shown in Fig.1. Therefore we extract the deeper features at two scales such as $1/8$ and $1/16$ of the input image size. To prevent the extraction of redundant features we create two different deep paths for the two scales.

The two separate paths are shown in Fig. 2 as Deeppath 1 and 2. Each of the deep paths contains several Inception-Residual1 (IR1) and Inception-Residual2 (IR2) blocks. Deeppath1 contains IR1 blocks followed by a convolution layer. The IR1 block is a modified version of the Inception-Residual network [14]. In the IR1 block, we use

Parameter	Segmentation net	Adversarial net
learning rate	10^{-4}	10^{-3}
beta1	0.5	0.5
beta2	0.99	0.99
batch size	1	1

Table 1. Optimization parameters for segmentation and adversarial networks.

1×3 and 3×1 convolutions to reduce the memory utilization for the deep path with feature maps of size $1/8$ of the input image. In this block we element-wise add the residual connection to the output convolutions followed by batch normalization and Rectified Linear Unit (ReLU) [10] activation at the end. In Deeppath1, we use nine IR1 blocks with long and short skip connections among the different blocks.

Deeppath2 consists of nine IR2 blocks. In the IR2 blocks, we replace 1×3 and 3×1 convolutions with 3×3 convolutions since the feature maps size is $1/16$ of the input image size, which is smaller than those in Deeppath1. In the IR2 blocks, we concatenate the feature maps followed by 1×1 convolutions to reduce the number of feature maps. We batch normalize the final convolution results and apply ReLU activation. In DeepPath2, all the IR2 blocks are connected by long and short skip connections as shown in Fig. 2. After the feature extraction using Deeppath1 and Deeppath2, the outputs are deconvolved with 2×2 kernels of stride 2×2 so that the size of the feature maps are doubled after every deconvolution step (DeConv layers). The final probability maps are of the same size as the input images.

2.3.2 Adversarial network

The adversarial network performs a two class classification. It discriminates whether the inputs are real or fake. The real inputs are the ground truths and the fake inputs are the outputs from the segmentation network. We design the adversarial network as a fully convolutional network. The network consists of dilated convolutions with kernel of size 3×3, dilations of 2×2 and strides of 2×2 followed by regular convolution with 1×1 kernel. We repeat the combination of dilated convolutions and regular convolutions six times. The first layer has 8 feature maps and we double the feature map size with every dilated convolutions. The number of feature maps is reduced to two by a final 3×3 convolution and the output is pooled using average pooling.

2.4. Deep neural network training

After creating the segmentation and adversarial networks we create the loss functions for the training. For the segmentation network we create a modified version of the spa-

tial cross entropy loss that takes spatially weighted labels along with the labels as inputs. For the adversarial network we use binary cross entropy loss. We set the adversarial loss function [9], $L_{\mathbf{a}}$, as follows

$$L_{\mathbf{a}} = \sum_{n=1}^{N} L_{bce}(\mathbf{a}(I_n, T_n), 1) + L_{bce}(\mathbf{a}(I_n, \mathbf{s}(I_n)), 0) \quad (2)$$

where N is the total number of images in the dataset, L_{bce} the binary cross entropy loss, $\mathbf{a}$ the adversarial network, I_n the n^{th} input image, T_n the n^{th} ground truth image, $\mathbf{s}$ the segmentation network. For the fake input we multiply the output probability map from the segmentation network with the input image and pass to the adversarial network. Similarly, we obtain the real input by multiplying the one-hot ground truth image with the input image. We design the segmentation network loss [9], $L_{\mathbf{s}}$, as

$$L_{\mathbf{s}} = \sum_{n=1}^{N} L_{wsce}(\mathbf{s}(I_n), T_n, W_n) + \lambda L_{bce}(\mathbf{a}(I_n, \mathbf{s}(I_n)), 1) \quad (3)$$

where L_{wsce} is the weighted spatial cross entropy loss, W_n is the n^{th} weighted label image, λ the contribution of the adversarial loss to the segmentation network. Here we set λ as a linearly increasing value from 8.3×10^{-6} to 10^{-1}, with every iteration. Setting constant values for the λ resulted in unstable models in our case.

We use Adam optimization [7] for both the segmentation and the adversarial networks. The optimization parameters for the segmentation and the adversarial networks are shown in Table 1. All the parameters have the same meaning as mentioned in [7]. We train the network for two epochs and we use a batch size of 1. We train the segmentation and the adversarial networks alternatively. We repeat the whole process for all the five folds.

2.5. Segmentation and post processing

After creating the probability maps using the segmentation network we create the segmentation masks. We store the output probability maps as eight bit images. We threshold the probability maps at a fixed value of 120 for images in the range from 0 to 255. After thresholding we fill the holes and remove objects smaller that 64 pixels. After removing the small objects we label the images for connected components and select the largest connected component as the final output segment.

2.6. Deep neural network implementation

We used the open source framework torch [1] to implement our networks. We trained the different network models on a workstation with six core Intel(R) Core(TM) i7-5930K CPU running at 3.50GHz and 32Gb RAM and a Nvidia Titan X Pascal GPU with 12Gb GPU memory on Ubuntu 14.04 operating system. Training each network model took nearly one hour. The network could create probability maps at approximately 5 images per second.

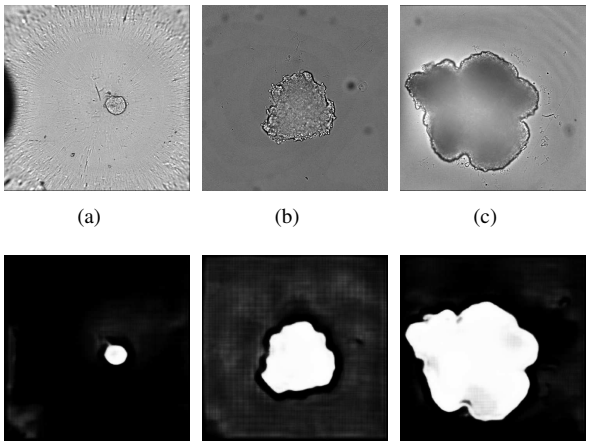


Figure 3. Output probability maps from our proposed network for dataset 1. (a-c) The input spheroid images and the corresponding probability maps in (d-f)

3. Results and discussions

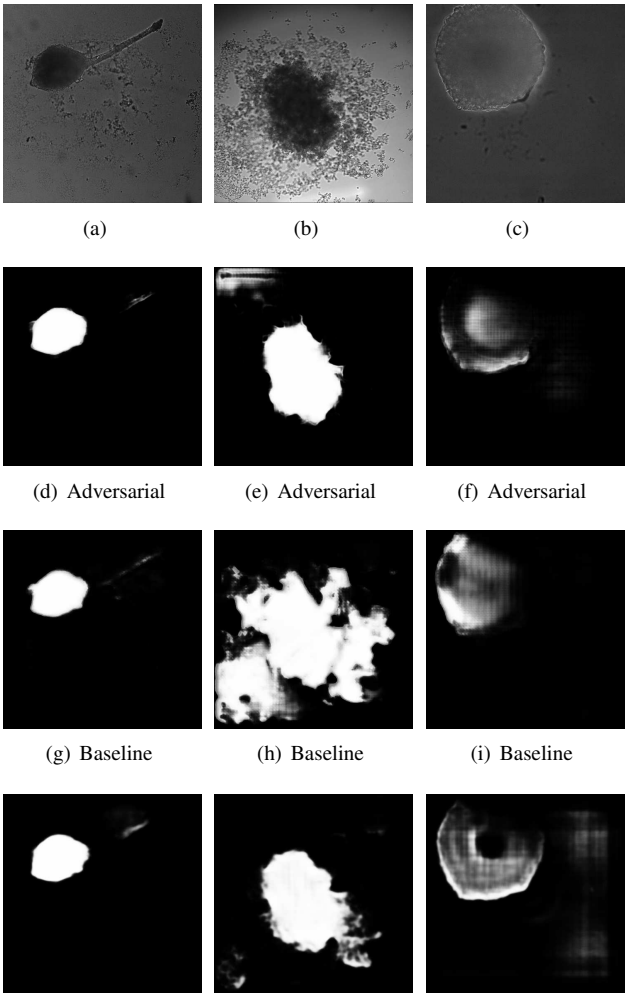
We create three different networks for spheroid segmentation and compare their results. The first network is a deep adversarial network with a segmentation network that generates probability maps and an adversarial network that evaluates the performance of the segmentation network with respect to the ground truth. The second network is a baseline segmentation network that has the same architecture as the segmentation network in the deep adversarial net without the adversarial loss. The third network is a U-net* segmentation network. The U-net* is a modified version of the original U-net [12]. We use zero padded convolutions so that the inputs and the outputs are of the same size, instead of non-padded convolutions in the original U-net. We use the same post processing steps to evaluate the performance of all the networks.

3.1. Segmentation evaluation

We create five network models corresponding to the five folds of cross validation. After training the networks we test the performance on the unseen sixth partition of the dataset. We create the probability maps using the trained networks and postprocess the probability maps to get segmentation masks, as mentioned in Section 2.5. After creating the segmentation masks we qualitatively and quantitatively evaluate the segmentation results. The output probability maps

Model	maF±std	maP±std	maR±std	maF±std	maP±std	maR±std
Adver	**77.09±06.89**	**82.16±06.00**	**73.07±07.40**	**64.21±04.79**	**66.33±04.31**	**66.85±05.11**
Base	68.79 ± 19.72	73.02 ± 20.15	65.67 ± 19.47	50.55 ± 20.47	51.84 ± 20.13	55.56 ± 25.99
U-net*	49.24 ± 19.38	52.80 ± 20.14	46.53 ± 18.77	36.53 ± 24.03	38.24 ± 24.72	40.99 ± 28.29

Table 2. Segmentation performance of different networks on dataset 1 and 2.



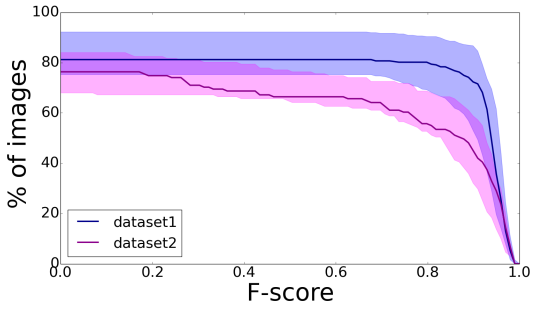
(a) (b) (c)

(d) Adversarial (e) Adversarial (f) Adversarial

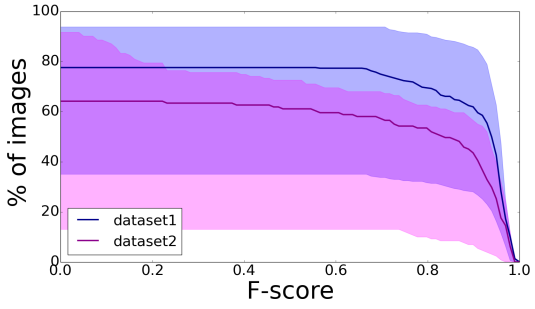
(g) Baseline (h) Baseline (i) Baseline

(j) U-net* (k) U-net* (l) U-net*

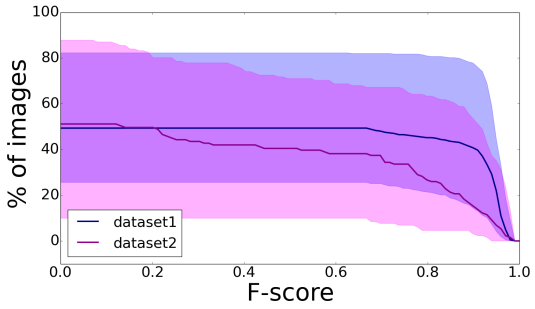
Figure 4. Output probability maps for dataset 2. (a-c) The input spheroid images, (d-f) the corresponding probability maps from adversarial network, (g-i) the results from the baseline network, and (j-l) the results from the U-net*. The third column shows an image in which all the networks failed to properly segment the spheroid.



(a) Adversarial network



(b) Baseline network



(c) U-net* network

Figure 5. Interval plot showing the range of percentage of images above particular F-score value v/s F-score for the test datasets 1 (blue) and 2 (magenta) for five folds. The solid line plots in blue and magenta show the median F-score values for datasets 1 and 2, respectively.

for a few examples from dataset 1 are as shown in Fig. 3. The probability maps for images from dataset 2 for different networks are shown in Fig. 4. The outputs correspond to the results from the best performing network in each of the architectures. Fig. 4 shows two examples, in the first and second columns, where our method performs satisfactorily while the third column shows a limitation of our method. We believe the poor performance on the third example is due to the small dataset size we trained the networks on as

this particular type of spheroid appearance is represented by only a few images in the dataset.

In addition to the qualitative comparison we also compare the performance quantitatively. We compare the segmentation result for each of the network models with the corresponding ground truth images. We use semi-automatic ground truths for dataset 1 and manually annotated ground truths for dataset 2. After finding the F-score [11] for every image we set different thresholds for the F-score from 0 to 1 at 100 regular intervals and find the percentage of images above that particular F-score. We plot the intervals corresponding to the maximum and minimum values of outputs from the five models for each of the architectures. The resulting interval plots are as shown in Fig. 5. We can see that the proposed method with adversarial learning (first row) is more stable over all the five folds than the other two methods for both dataset 1 (blue) and 2 (magenta). The solid lines show the median values for dataset 1 and 2.

We find the average F-score for the five folds. After finding the average F-scores we take the mean of the average F-scores to get the mean average F-score (maF). Similarly, we find mean average Precision (maP) and mean average Recall (maR). We repeat this for all the models for datasets 1 and 2. The results are shown in Table 2. The results from the adversarial models are better than both the baseline and the U-net* models.

4. Conclusion and future work

In this work, we designed deep adversarial networks for segmentation of bright-field spheroid images. We qualitatively and quantitatively evaluated the segmentation results. We quantitatively compared our proposed method with two other deep neural network architectures. We showed that our method was better at segmentation than the two other compared methods. In the future we plan to improve the segmentation results by combining different deep network architectures and use more data for training. We are also planning to combine the proposed approach with other segmentation methods.

Acknowledgements

This work was supported by the Swedish research council, grant 2012-4968, ERC Consolidator grant 682810, and Swedish strategic program, eSSENCE, to Carolina Wählby. We would like to thank Radek Polanski at AstraZeneca for doing the biological experiments and collecting the image data for us. We would also like to thank Jonatan Wählby for manually annotating dataset 2 for the experiments.

References

[1] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011. 4

[2] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal. The importance of skip connections in biomedical image segmentation. In *International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*, pages 179–187. Springer, 2016. 1, 2

[3] E. Fennema, N. Rivron, J. Rouwkema, C. van Blitterswijk, and J. de Boer. Spheroid culture as a tool for creating 3d complex tissues. *Trends in biotechnology*, 31(2):108–115, 2013. 1, 2

[4] W. J. Godinez, I. Hossain, S. E. Lazic, J. W. Davies, and X. Zhang. A multi-scale convolutional neural network for phenotyping high-content cellular images. *Bioinformatics*, 2017. 1, 2

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 1

[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*, pages 770–778, 2016. 1

[7] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[8] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 1

[9] P. Luc, C. Couprie, S. Chintala, and J. Verbeek. Semantic segmentation using adversarial networks. *arXiv preprint arXiv:1611.08408*, 2016. 1, 4

[10] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 3

[11] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011. 6

[12] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015. 1, 2, 4

[13] Y. Song, L. Zhang, S. Chen, D. Ni, B. Lei, and T. Wang. Accurate segmentation of cervical cytoplasm and nuclei based on multiscale convolutional network and graph partitioning. *IEEE Transactions on Biomedical Engineering*, 62(10):2421–2433, 2015. 1, 2

[14] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016. 1, 2, 3

[15] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 2