Technical Report

Pattern Recognition and Image Processing Group
Institute of Visual Computing and Human-Centered Technology
TU Wien
Favoritenstrasse 9-11/193-03
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801 - 18661
Fax: +43 (1) 58801 - 18697
E-mail: e12133404@student.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-155 October 3, 2022

Learning-based Leaf Cell Instance Segmentation

Fabian Wolf
supervised by Jiří Hladůvka

Abstract

Due to climate change, variability of temperature and rainfall increases. Based on this, understanding changes in leaf anatomy and function under those conditions is desirable. We aim to advance the understanding of changes in leaf anatomy by investigating deep-learning based instance segmentation of leaf cells. Accordingly, state-of-the-art (cell) instance segmentation methods are discussed. Based on the discussion, we investigated 2D- and 3D-Cellpose with and without additional semantic information. In general, we found that Cellpose is not suited well for the task, but might be suited for segmenting roundish (sponge and palisade) cells. We conclude that further research is required and propose to investigate embedding-based approaches.

1 Introduction

Against the backdrop of climate change, increase in variability of temperature and rainfall seems inevitable. Globally, for many regions, this leads to extreme environmental conditions beyond their previous capacity to adapt [1]. Based on this, understanding changes in leaf anatomy under those conditions and the resulting changes in function is desirable. Changes in anatomy can be investigated by segmenting the individual leaf cells. Segmenting the individual leaf cells allows to localize where the changes in volume, diameter, length, etc. occur. Based on the location of the changes in the anatomy, the biological function might be altered. Based on this, models to simulate photosynthesis under these changes can be developed. This is part of the Watergate research project [1]. Furthermore, segmenting the individual leaf cells is not just limited to this task. For example, segmenting the individual leaf cells during growth periods can help to examine how anatomical structures are formed. Hence, investigating the segmentation of individual leaf cells is expedient.

Our goal is to segment the individual cells of 3D-leaf-scans. Cell segmentation, or more general instance segmentation, is done by pixel/voxel-wise classification. Instance segmentation classifies such that pixels/voxels of the same instance are assigned the same label. The assigned labels may be chosen arbitrarily the only constraint is that pixels/voxels of the same instance have the same label [2]. This creates an image/3D-scan like output which is referred to as mask.

3D-scans are hard to visualize as a whole. Accordingly, in this work, a 3D-scan is partially visualized by 2D-slices a long the x,y, or z-axis. An example is shown in Figure 1: here the cells are cluttered, vary in shape and size, and some are really small. Furthermore, the number of instances varies between leafs.

Dense labeling 3D-scans is a labor intensive task. Therefore, we only got one sparsely labeled Micro-CT scan. Our working group used the sparsely labeled scan to create a densely labeled scan for instance segmentation. This densely labeled scan for instance segmentation was obtained via a rule based approach. Scans were taken at the Paul Scherrer Institute at Swiss Light

[1] Based on Correspondence with Guillaume Theroux Rancourt, Ph.D., Institute for Botany Boku Vienna

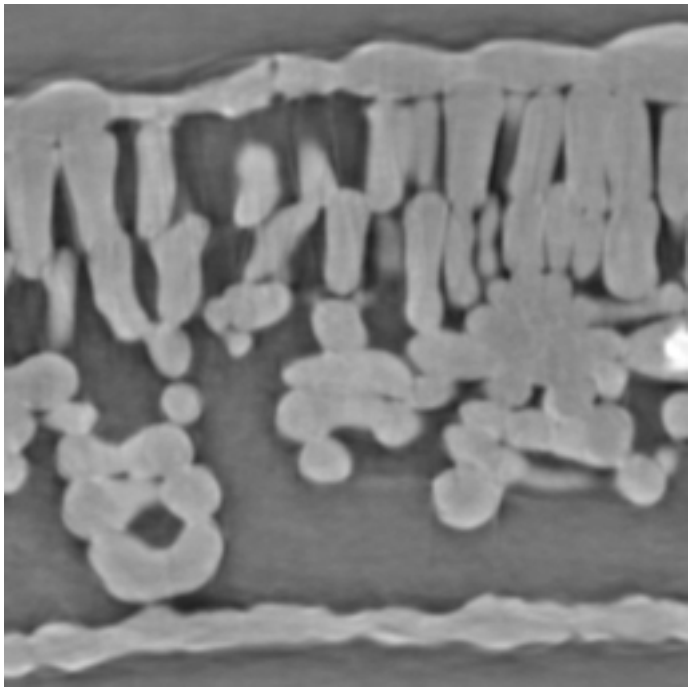

Figure 1: 2D-slice of a leaf scan

Source using acquisition protocols similar to those published in Théroux-Rancourt *et al.* [3]. Furthermore, our working group used the sparsely labeled scan to create a densely labeled scan for semantic segmentation. This densely labeled scan for semantic segmentation was obtained via a U-Net [4] based approach. These densely labeled scans are used as our training data. It is important to mention that these densely labeled scans are not a perfect ground truth.

We restrict the scope of this work to deep-learning based approaches and train them purely supervised on the data available to us. However, we do not exclude pre-trained models. Time and resources are limited. For this reason, we do not conduct a full architecture search, but focus on state-of-the-art architectures. We conduct a review on the state of the art of 3D-instance segmentation, especially cell instance segmentation. Based on the results of the review, the state-of-the-art architectures are discussed in regard to our task. Based on the discussion we select an architecture, i.e. Cellpose [5, 6]. We train this architecture and try different approaches based on the available data.

In Section 2, the current state of the art of 3D-cell instance segmentation is presented and discussed in regard to our task. Based on this discussion a state-of-the-art architecture is chosen. In Section 3, the architecture and overall approach is described. In Section 4, training and evaluation are described and the results of the experiments are presented. In Section 5, the results are discussed. Finally, in Section 6, we draw a concise conclusion and give an outlook for future work.

2 Related Work

Bouget *et al.* [7] and Felfeliyan *et al.* [8] use Mask R-CNN based approaches. Bouget *et al.* [7] segment lymph nodes from 3D-scans directly using 3D-convolution. Felfeliyan *et al.* [8] use instance segmentation for objective assessment of osteoarthritis based on magnetic resonance imaging. Instead of predicting the instances directly from the 3D-scan, Felfeliyan *et al.* [8] split the 3D-scan into 2D-slices, segment the 2D-slices, and reassemble the 3D-scan from the segmented slices. Mask R-CNN based approaches predict bounding boxes and segment inside the bounding boxes to predict the individual instances. This way more bounding boxes are predicted than instances ex-

ist. Therefore, non-maximum suppression is used to remove excess bounding boxes. In general, we found several Mask R-CNN based approaches for 2D and 3D [9, 10, 11].

Similar approaches exist which use another shape representation instead of the bounding box. Luo *et al.* [12] propose to predict bounding spheres, determined by center and radius, instead of bounding boxes. Schmidt *et al.* [13] propose Stardist, i.e. a U-Net [4] with a ResNet [14] backbone. Stardist predicts a star-shaped, convex polygon for each pixel, here the pixel serves as the center of the star-shaped, convex polygon. The star-convex polygon is represented by a predefined number n of distances along evenly spaced predefined radial directions. For non-max suppression, for each polygon an object probability is predicted. The object probability is defined as the distance between the center and the closest boundary. Upschulte *et al.* [15] propose to predict contours instead of bounding boxes. The contours are given by a set of n points in a Fourier descriptor along each dimensional axis.

Another approach is based on learning an embedding. The embedding approach embeds each pixel such that pixels/voxels belonging to the same instance are in close proximity to each other in the embedding space. This way instances can be constructed by clustering, e.g. mean-shift. The definition of proximity differs between papers [16, 17, 18, 19, 20]. For example, Liu and Furukawa [19] define distance by an affinity score. The affinity score is the likelihood of two neighboring pixels/voxels belonging to the same instance. Payer *et al.* [20] use a cosine embedding and define proximity via the cosine distance of two embedding vectors. If the cosine distance between two embedding vectors is below a predefined threshold, the corresponding pixels/voxels are predicted to belong to the same instance.

Another approach is based on semantic segmentation and watershed algorithm to get an instance segmentation. Eschweiler *et al.* [21] and Lin *et al.* [22] segment into the classes cell centroids (utilized for seeding watershed), cell membranes (cell shape info), and background. Caicedo *et al.* [23] use a U-Net to segment into the classes object interior, object boundary, and background.

Another approach is Cellpose [5, 6]. Cellpose represents instances by their gradient flow. The gradient flow yields a gradient for each pixel/voxel of an instance. The gradient points from the pixel/voxel to the center of the instance.

Although some of these presented approaches are 2D, in general two methods exists to apply these approaches to 3D-instance segmentation. One method splits the 3D-scan into 2D-slices, segments the 2D-slices, and reassembles the 3D-scan from the segmented slices. The other method inputs the 3D-scan directly and returns the 3D-segmentation [7, 8]. As stated in Section 1, the cells, vary in shape a lot. Hence, approaches using a fixed representation like bounding boxes or spheres do not suit the task. This leaves the contour based approach to be explored, since it is able to represent complex and varying shapes. However, in general, Mask R-CNN based approaches do not work so well for cluttered instances [24]. As stated in Section 1, cluttered instances are a challenge for our task. For the embedding-based approaches the authors did not describe how they obtained the gradients for training from the clustering algorithm and neither Tensorflow nor PyTorch had a pre-implemented module for this. Reverse engineering their approach is not possible with our limited time. For these reasons Mask R-CNN and embedding-based approaches are not investigated in our work.

Since the approaches are rarely tested on the same datasets, performance is complicated to compare. In the following, we will compare the approaches evaluated on datasets which contain cluttered instances, since this ensures at least some similarity to our data. The Mask R-CNN based approaches, including the contour based approach and Stardist were evaluated on datasets with cluttered instances. The contour based approach achieved slightly better F_1-Score , but also applied many auxiliaries [15]. Because the differences in F_1-Score between the approaches is small (i.e. 0.01-0.02), it is arguable that a vanilla contour based approach might achieve the same F_1-Score as the other approaches. Depending on the datasets the segmentation based approach, Cellpose, and Stardist beat each other in achieving higher mean average precision (mAP) [22]. As a conclusion, no approach obviously outperforms the other approaches. In general, Cellpose is tested on plant (including leaf) cells. Consequently, Cellpose was tested on data that is the most similar to our data. For these reasons we investigate Cellpose for our task. However, we recognize that semantic segmentation and embedding-based approaches are also worth investigating, but are not investigated due to limited time and resources.

3 Methodology

Cellpose represents instances by their gradient flow. The gradient flow yields a gradient for each pixel/voxel of an instance. The authors describe the gradient map as heat diffusion from center of the instance to borders of the instance. However, they simply compute the gradient $\nabla_v = c - v$ for a pixel/voxel by subtracting the position vector of the pixel/voxel v from the position vector of the center c [6]. Thus, the gradient of a pixel/voxel can be conceptualized as arrow pointing from the pixel/voxel to the center of the corresponding instance. Based on this, the instances can be reconstructed by grouping all pixels/voxels to an instance whose gradients flow roughly to the same center [6]. Eschweiler *et al.* [5] find that the instances can also be reconstructed by simply following the *tanh* of the gradient flow. Therefore, Eschweiler *et al.* [5] simplify the problem by letting the model only predict the *tanh* of the gradient flow. Accordingly, Cellpose is able to express a variety of shapes. However, this does only consider pixels/voxels belonging to an instance. Therefore, the model also needs to predict whether a pixel/voxel belongs to an instance or not. This is done via a binary foreground segmentation mask. The resulting output for Cellpose consist of three or four masks for 2D or 3D respectively. These masks are foreground segmentation mask, gradient flow mask in x-direction, gradient flow mask in y-direction, and gradient flow mask in z-direction (for 3D). To foster understanding, we illustrate the output for the 2D-case in Figure 2. For the 3D-case, the output comprises an extra gradient flow mask in z-direction.

The Cellpose architecture is based on U-Net [4]. ResNet [14] serves as the backbone for the U-Net architecture of Cellpose. The backbone network is the network used in the contracting path of U-Net. In the case of ResNet, this basically means that the convolutional blocks of U-Net are implemented as residual blocks. Using the 3D-scan as input directly leads to a massive increase in parameters such that the GPU runs out of memory when training the model. To ensure a reasonable batch size, the input shape is restricted to $64 \times 64 \times 64$ cuboids. Using 2D-slices as input the slices can be inputted in whole. U-Net is composed of a contracting and an expansive path. The contracting path consists of convolutional blocks each followed by a max pooling layer. The expansive path consists of up-convolutions each followed by a convolutional block. The paths are connected through a convolutional block. The output layer is a convolution with c kernels of shape 1×1 followed

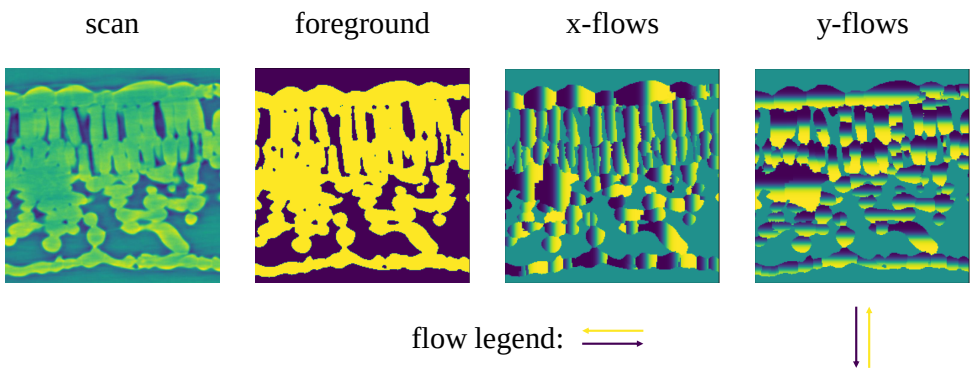

Figure 2: Output of Cellpose

by an output activation. For Cellpose $c = 3$ or $c = 4$ for 2D or 3D respectively. The activation function is the *sigmoid* activation for the binary foreground segmentation mask and the *tanh* activation for the gradient flow masks in x-, y-, and z-direction

A max pooling layer has a kernel shape of 2×2 and a stride of 2, leading to downsampling. In each downsampling step, the spatial dimensions are halved while the number of kernels is doubled. An up-convolution consists of an upsampling layer followed by a convolution with a kernel shape of 2×2. The upsampling layer has a kernel shape of 2×2. In each upsampling step, the spatial dimensions are doubled, while the number of kernels is halved by the convolution. Consequently, for each convolutional block in the contracting path exists a convolutional block in the expansive path with corresponding number of kernels. The number of base kernels K is the number of kernels in the first convolutional block [4].

The corresponding paths are connected via a skip-connection. If padding is not applied for convolutions, the spatial dimensions are reduced with each convolution. This leads to smaller feature maps in the expansive path, compared to the corresponding feature maps in the contracting path. Thus, the

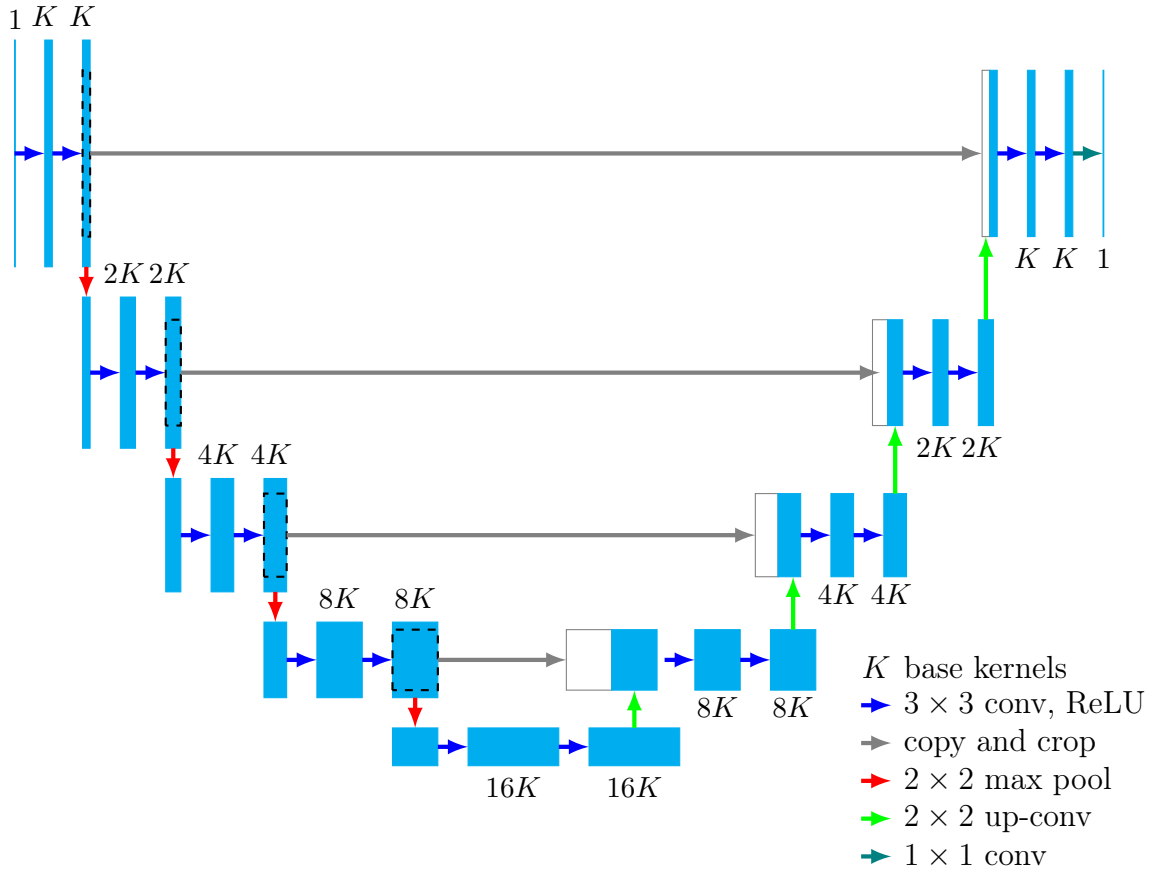

Figure 3: Architecture of U-Net (based on [4])

skip-connection crops the output of a convolutional block to fit the input of the corresponding convolutional block in the expansive path. The whole architecture of U-Net [4] is illustrated in Figure 3. The skip connections are usually implemented as concatenation, Cellpose implements them as summation in order to reduce the number of parameters [6].

ResNets address the degradation of accuracy of deep neural networks. In general, a deeper network should at least achieve the same results as its shallower counterpart, because an equivalent deeper network can be constructed by considering the shallower counterpart and adding identity mappings on top. Empirically, however, the deeper network and its shallower counterpart

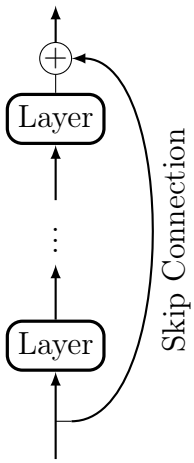

Figure 4: Residual Block

do not always converge in the same way. This indicates that not all networks are similarly easy to optimize [14].

ResNets address the degradation problem by letting stacked layers learn the residual function $\mathcal{F}(X) = \mathcal{H}(X) - X$ instead of the desired function $\mathcal{H}(X)$. Thus, the original desired function is transformed into $\mathcal{F}(X) + X$. This is based on the hypothesis that stacked layers can learn any function. Assuming this hypothesis is true, we can derive that stacked layers can learn the residual functions $\mathcal{H}(X) - X$. Both approaches, with or without residual function, should be able to learn the desired functions. The difficulty of learning may vary [14]. He *et al.* [14] empirically show that, by using residual networks, deeper neural networks can be learned.

$\mathcal{F}(X) + X$ is implemented by skip connections. Skip connections skip one or more layers, by adding the input of the skip connection to the output of the skipped layers. Adding can be implemented as element-wise sum or concatenation. A stack of skipped layers is called residual block, see Figure 4. ResNets are comprised of residual blocks [14].

4 Experiments

4.1 Experimental Setup

The experiments were conducted using Google Colab and Vienna Scientific Cluster. Google Colab provided 12GB RAM, 78 GB storage, and a GPU. The provided GPU varied depending on availability. The following GPUs may have been provided 12GB NVIDIA Tesla K80, 16 GB NVIDIA Tesla T4, or 16 GB NVIDIA Tesla P100 [25]. Vienna Scientific Cluster provided 264 GB RAM, and 48 GB NVIDIA A40 GPU [2]. The method was implemented in Python 3.7.12 [26] using Tensorflow 2.7.0 [27] and the repository of Stringer *et al.* [6] [3]. As explained in Section 1, only one densely labeled 3D-scan of shape $256 \times 256 \times 256$ was available. Thus, we cropped all possible cuboids of shape $64 \times 64 \times 64$, i.e. $(256 - 64) * (256 - 64) * (256 - 64)$, and randomly sampled them for training. From those we split off 1000 cuboids for validation and 7 slices for testing. For testing we split off slices to be able to compare the two methods for 3D-segmentation, i.e. segmenting 2D-slices and reassembling the 3D-scan or segmenting the 3D-scan directly. For pre-processing we normalized the input. Cellpose was trained on the training dataset and its performance was monitored on the validation dataset. Training was conducted by optimizing a custom loss function $\mathcal{L}$. This loss is constituted of a flow loss $\mathcal{L}_{flow}$ for each axis and a foreground segmentation loss $\mathcal{L}_{fg}$. The flow loss is the mean squared error between the true and predicted flow along the given axis for all pixels/voxels which are part of an instance. This is done, because the flow is only well defined for pixels/voxels which are part of an instance. The foreground segmentation loss is the mean absolute error between the true and predicted foreground segmentation loss. The total loss is shown in Equation (1). The flow losses and the foreground segmentation loss are weighted by the hyperparameters α and β respectively [4].

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{fg} + \frac{\beta}{3} \cdot (\mathcal{L}_{flowx} + \mathcal{L}_{flowy} + \mathcal{L}_{flowz}) \tag{1}$$

The loss was optimized using the Adam optimizer [28] with an initial learning rate of 0.001 and a batch size of 64. The learning rate was reduced by a factor of 0.1 each time the validation loss did not improve for 10 epochs. Cellpose was trained until convergence. Training was completed if the validation loss

[2] https://vsc.ac.at

[3] https://github.com/MouseLand/cellpose

[4] Following Eschweiler *et al.* [5], we chose $\alpha = 1$ and $\beta = 1$

did not improve for 25 epochs. We used the maximal possible batch size given the available GPU memory.

4.2 Metrics

To evaluate different versions of Cellpose, we used mAP [5]. mAP is the area under the recall precision curve which arises from plotting recall against precision over different Intersection over Union (IoU) thresholds $t_{iou} \in \{0.1, 0.2, ..., 0.9\}$. Increasing t_{iou} results in fewer detections, usually this increases precision and decreases recall. Given the number of true positives TP, false positives FP, and false negatives FN, precision $Prec$ and recall Rec are defined as follows:

$$Prec = \frac{TP}{TP + FP} \tag{2}$$

$$Rec = \frac{TP}{TP + FN} \tag{3}$$

As we can clearly see, $Rec \in [0; 1]$ and $Prec \in [0; 1]$, therefore follows $mAP \in [0; 1]$. An instance is counted as TP if IoU of predicted and true instance is greater t_{iou}. If several predicted instances score sufficient IoU in regard to a true instance, only the one with the highest IoU is counted as TP. A predicted instance is counted as FP if no true instance with sufficient IoU exists. A true instance is counted as FN if no predicted instance with sufficient IoU exists. IoU is used to measure similarity of sets. Considering the pixels/voxels of an instance as sets, the intersection of the predicted set $\hat{y}$ and the ground truth set y can be interpreted as the overlapping region of both instances. IoU is defined as :

$$IoU = \frac{|y \cap \hat{y}|}{|y \cup \hat{y}|} \tag{4}$$

4.3 Results

We evaluated 3D-Cellpose trained on our data with and without semantic information and 2D-Cellpose as trained by Stringer *et al.* [6]. Jain *et al.* [29] have shown that semantic information can enhance performance. As stated in Section 1, a semantic segmentation of our scan was available. Accordingly,

[5] The term mAP and AP are not used uniformly, some authors would use our definition for AP

Model	mAP
3D-Cellpose	0.0135
3D-Cellpose-Sem.	0.0298
2D-Cellpose	**0.1652**

Table 1: Test results

we investigated the impact of incorporating semantic information via the semantic segmentation mask. We incorporated the semantic segmentation by simply concatenating the input with the corresponding semantic segmentation. The resulting model is called 3D-Cellpose-Sem. The test results achieved by 3D-Cellpose, 3D-Cellpose-Sem, and 2D-Cellpose are displayed in Table 1. To illustrate the results, Figure 5 and 6 display the test results for the best performing model, i.e. 2D-Cellpose.

5 Discussion

We investigated 2D- and 3D-Cellpose with and without additional semantic information to segment the individual cells of 3D-leaf-scans. In general, we found that Cellpose is not suited well for the task. As shown in Table 1, 3D-Cellpose-Sem achieved slightly higher mAP than 3D-Cellpose, meaning semantic information could have contributed to performance. However, this statement must be questioned, since the improvement is minimal and 3D-Cellpose and 3D-Cellpose-Sem achieved especially low mAP in general. The low mAP means they did not learn to segment the leaf cells well. This could have several reasons. For difficult datasets many instance segmentation methods do not perform well [15, 5]. Considering that the scan is 3D, the cells are cluttered, vary in shape, vary in size, and some are really small, the task might be a hard task in general. As explained in Section 3, processing 3D-input, e.g. via 3D-convolution, requires more parameters than processing 2D-input. A model with more parameters might be harder to optimize in general [14]. To reduce the number of parameters, 3D-Cellpose and 3D-Cellpose-Sem used $64 \times 64 \times 64$ cuboids as input. As a consequence, the whole 3D-scan was processed using a tiling strategy [4]. Tiling without overlap creates hard borders. Flows could be susceptible to this, because flows

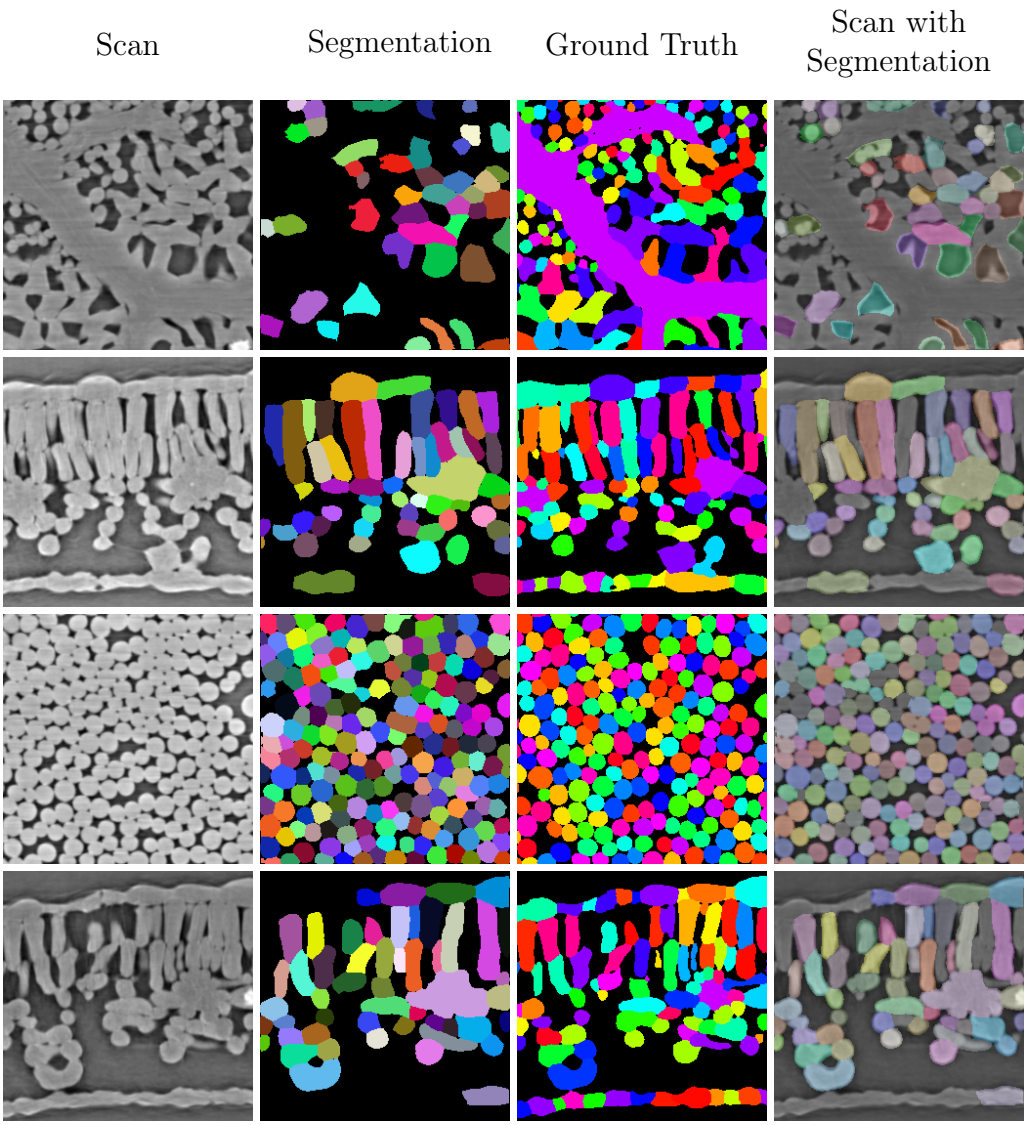

Figure 5: Test results

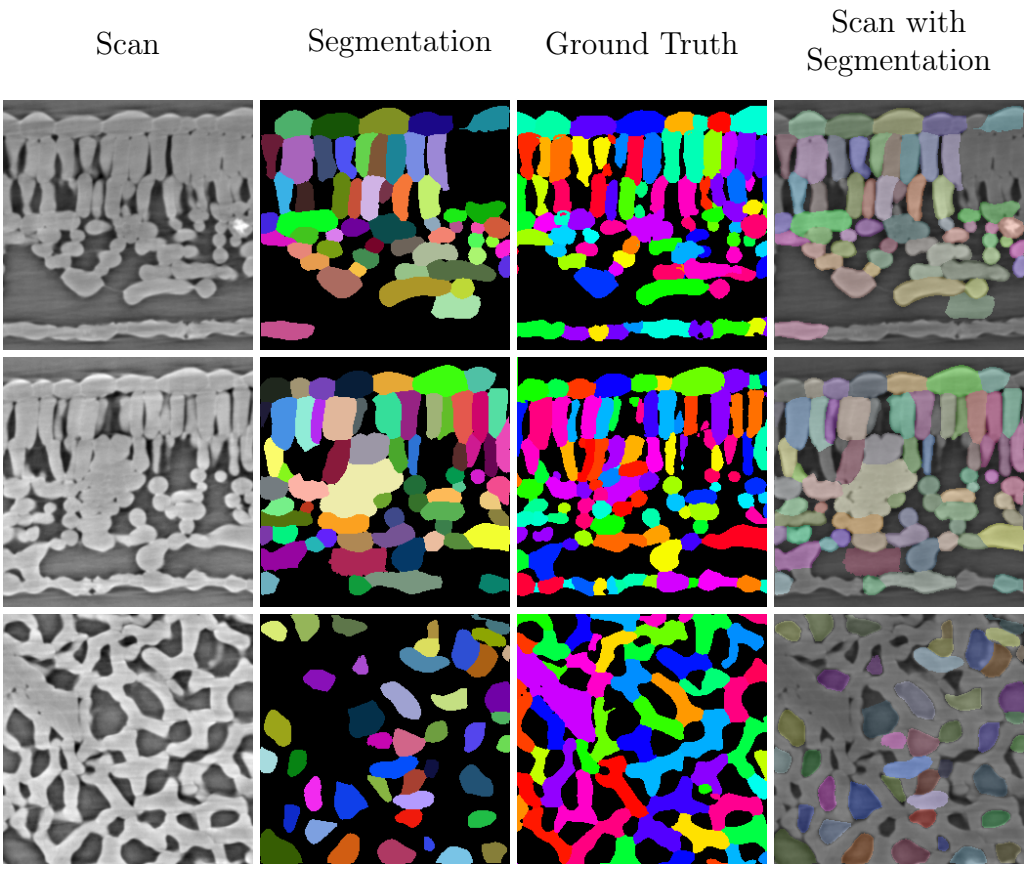

Figure 6: Test results (continued)

need continuity. Continuity is intercepted by hard borders. This might be improved by utilizing an overlapping tile strategy [4]. However, this would also increase the number of parameters or decrease the output shape. As explained in Section 1, only one scan was available for training and the dense label was created artificially and thus is not a perfect ground truth. Quantity and quality of training data can significantly impact performance and few training data can be easily overfitted [30]. As explained in Section 4, we cropped all possible cuboids of shape $64 \times 64 \times 64$, and randomly sampled them for training. From those we split off 1000 cuboids for validation and 7 slices for testing. Accordingly, the 3D-Cellpose models have seen all parts constituting the test slices. This is not best practice, but was the best way to ensure comparability and training with our limited training data and the general ranking of the models was not distorted by this, since 3D-Cellpose and 3D-Cellpose-Sem performed worse than 2D-Cellpose anyway. Furthermore, the labeled slices were not yet available and planned when we trained the model. Consequently, we expected 3D-Cellpose and 3D-Cellpose-Sem to overfit the training and hence the testing data. However, as shown in Table 1, they did not overfit the testing data. Thus, we hypothesise, that the low mAP is not (solely) due to the quantity and quality of training data.

Note that despite the mAP being low in general, as we can see in Figure 5, for some cells, especially the ones with only roundish shapes, 2D-Cellpose produces promising results. However, for scans with a context of very long stretched cells, as shown in the first and last example, the 2D-Cellpose is not suited. Furthermore, we can see that 2D-Cellpose is not suited to predict instances of epidermic cells, since almost all of those are neglected.

Finally, as explained in Section 2, to conduct 3D-instance segmentation using 2D-models, the 3D-scan is split into 2D-slices, the 2D-slices are segmented, and reassembled to the 3D-instance segmentation mask. This does not ensure that corresponding instances in subsequent slices have the same label. However, since 2D-Cellpose predicts the flows this is not a problem. For 3D-instance segmentation 2D-Cellpose needs to predict the flows along the x-, y-, and z-axis. This is implemented by processing slices along different axis. The 3D-instance segmentation mask is reconstructed from the flows along the x-, y-, and z-axis. This is implemented in the repository of Stringer *et al.* [6].

6 Conclusion and Future Work

To conclude, we implemented and evaluated 2D- and 3D-Cellpose with and without additional semantic information to segment the individual cells of 3D-leaf-scans. We found that Cellpose is not suited well for the task in general. However, as discussed in Section 5, for cells with roundish shapes, 2D-Cellpose produces promising results. Accordingly, 2D-Cellpose might be suitable to segment sponge cells and palisade cells from slices along the horizontal z-axis.

For future work, we propose to investigate semantic segmentation and embedding-based approaches. Furthermore, we propose to investigate fine-tuning 2D-Cellpose on target task data. For both proposes, data in higher quantity and quality is recommended. Finally, we propose to investigate an evaluation method that makes the comparison of 2D- and 3D-based models more fair.

References

[1] H.-O. Pörtner, D. C. Roberts, H. Adams, C. Adler, P. Aldunce, E. Ali, R. A. Begum, R. Betts, R. B. Kerr, R. Biesbroek, *et al.*, "Climate change 2022: Impacts, adaptation and vulnerability," *IPCC Sixth Assessment Report*, 2022.

[2] A. M. Hafiz and G. M. Bhat, "A survey on instance segmentation: State of the art," *International journal of multimedia information retrieval*, vol. 9, no. 3, pp. 171–189, 2020.

[3] G. Théroux-Rancourt, M. R. Jenkins, C. R. Brodersen, A. McElrone, E. J. Forrestel, and J. M. Earles, "Digitally deconstructing leaves in 3d using x-ray microcomputed tomography and machine learning," *Applications in plant sciences*, vol. 8, no. 7, 2020.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer Assisted Interventions (MICCAI)*, 2015, pp. 234–241.

[5] D. Eschweiler, R. S. Smith, and J. Stegmaier, "Robust 3d cell segmentation: Extending the view of cellpose," *arXiv:2105.00794*, 2021.

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

[7] D. Bouget, A. Pedersen, J. Vanel, H. O. Leira, and T. Langø, "Mediastinal lymph nodes segmentation using 3d convolutional neural network ensembles and anatomical priors guiding," *arXiv:2102.06515*, 2022.

[8] B. Felfeliyan, A. R. Hareendranathan, G. Kuntze, J. L. Jaremko, and J. L. Ronsky, "Improved-mask R-CNN: Towards an accurate generic msk mri instance segmentation platform (data from the osteoarthritis initiative)," *Computerized medical imaging and graphics*, vol. 97, 2022.

[9] J. Hou, A. Dai, and M. Nießner, "3d-sis: 3d semantic instance segmentation of RGB-D scans," in *Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4416–4425.

[10] W. Wang, R. Feng, J. Chen, Y. Lu, T. Chen, H. Yu, D. Z. Chen, and J. Wu, "Nodule-plus R-CNN and deep self-paced active learning for 3d instance segmentation of pulmonary nodules," *IEEE Access*, vol. 7, pp. 128 796–128 805, 2019.

[11] S. Wang, Y. Zhu, S. Lee, D. C. Elton, T. C. Shen, Y. Tang, Y. Peng, Z. Lu, and R. M. Summers, "Global-local attention network with multi-

task uncertainty loss for abnormal lymph node detection in MR images," *Medical Image Analysis*, vol. 77, 2022.

[12] X. Luo, T. Song, G. Wang, J. Chen, Y. Chen, K. Li, D. N. Metaxas, and S. Zhang, "SCPM-Net: An anchor-free 3d lung nodule detection network using sphere representation and center points matching," *Medical Image Analysis*, vol. 75, 2022.

[13] U. Schmidt, M. Weigert, C. Broaddus, and E. W. Myers, "Cell detection with star-convex polygons," in *Medical Image Computing and Computer Assisted Interventions (MICCAI)*, 2018.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[15] E. Upschulte, S. Harmeling, K. Amunts, and T. Dickscheid, "Contour proposal networks for biomedical instance segmentation," *Medical Image Analysis*, vol. 77, 2022.

[16] W. Wang, R. Yu, Q. Huang, and U. Neumann, "Sgpn: Similarity group proposal network for 3d point cloud instance segmentation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2569–2578.

[17] X. Wang, S. Liu, X. Shen, C. Shen, and J. Jia, "Associatively segmenting instances and semantics in point clouds," *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4091–4100, 2019.

[18] P. Hirsch, L. Mais, and D. Kainmueller, "Patchperpix for instance segmentation," in *European Conference on Computer Vision (ECCV)*, 2020.

[19] C. Liu and Y. Furukawa, "Masc: Multi-scale affinity with sparse convolution for 3d instance segmentation," *arXiv:1902.04478*, 2019.

[20] C. Payer, D. Štern, M. Feiner, H. Bischof, and M. Urschler, "Segmenting and tracking cell instances with cosine embeddings and recurrent hourglass networks," *Medical Image Analysis*, vol. 57, pp. 106–119, 2019.

[21] D. Eschweiler, T. V. Spina, R. C. Choudhury, E. Meyerowitz, A. Cunha, and J. Stegmaier, "Cnn-based preprocessing to optimize watershed-based cell segmentation in 3d confocal microscopy images," in *International Symposium on Biomedical Imaging (ISBI)*, IEEE, 2019, pp. 223–227.

[22] Z. Lin, D. Wei, M. D. Petkova, Y. Wu, Z. Ahmed, S. Zou, N. Wendt, J. Boulanger-Weill, X. Wang, N. Dhanyasi, *et al.*, "Nucmm dataset: 3d neuronal nuclei instance segmentation at sub-cubic millimeter scale," in

International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2021, pp. 164–174.

[23] J. C. Caicedo, J. Roth, A. Goodman, T. Becker, K. W. Karhohs, M. Broisin, C. Molnar, C. McQuin, S. Singh, F. J. Theis, *et al.*, "Evaluation of deep learning strategies for nucleus segmentation in fluorescence images," *Cytometry Part A*, vol. 95, no. 9, pp. 952–965, 2019.

[24] Z. Liang, M. Yang, H. Li, and C. Wang, "3d instance embedding learning with a structure-aware loss function for point cloud segmentation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4915–4922, 2020.

[25] colab-billing@google.com. "Making the most of your colab subscription." (2018), [Online]. Available: https://colab.research.google.com/notebooks/pro.ipynb#scrollTo=QMMqmdiYMkvi (visited on 01/26/2021).

[26] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. CreateSpace, 2009.

[27] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, M. Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[29] J. Jain, A. Singh, N. Orlov, Z. Huang, J. Li, S. Walton, and H. Shi, "Semask: Semantically masked transformers for semantic segmentation," *arXiv:2112.12782*, 2021.

[30] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics: Conference Series*, 2019.