April 1, 2024

Localizing Cells in Phase-Contrast Microscopy Images using Sparse and Noisy Center-Point Annotations

Benjamin Eckhardt
benjamin.eckhardt@stud.uni-goettingen.de
Georg-August-University of Göttingen,
University of Tartu

Abstract

WIP

Question: Is the section structure good?

Contents

1 Introduction

In the field of computer vision the localization of dense homogenous objects is an important task with many applications. Classical image analysis approaches often suffer from susceptibility to small fluctuations in the data. Machine learning and in particular deep learning algorithms can learn robust labeling functions, but often require large amounts of high-quality annotations. Producing annotations of sufficient quality takes a lot of work and may therefore be a major bottleneck.

In this work we focus on predicting cell counts from microscopy images. Estimating the number of living cells is the central step in many biological workflows. The currently used method, Countess 3 FL by ThermoFisher, we wish to improve upon, requires a trained human to expend several minutes per sample, preparing it with costly consumables, and applying it to a dedicated expensive device. The method yields unacceptable inaccuracies. The preparation of the samples makes them unusable for further experiments. Especially in high-throughput situations where this analysis process has to be repeated over very many times, these stack up to significant costs, time investments, and material expenses.

Improving upon this via a passive and automated image analysis can drastically reduce the costs, by saving a lot of human hours and reducing the required amount of costly consumables. Furthermore, the less dedicated hardware is needed, the easier a method can be continuously improved with new data and new software.

Labeling only cell centres minimizes the annotation effort, while providing just enough information to localize individual cells. The images are obtained via phase-conrast microscopy with relatively low resolution. It was impossible even for a trained expert to discern cells in the parts of the images with high confluency. Therefore, while being cheap to obtain, the point labels are sparse and noisy.

We apply several pre-existing zero-shot methods like Cellpose, Stardist and MicroSAM and show that they fail to provide the desired generalization on our dataset. We fine-tune and retrain well-known general computer vision models like MaskRCNN and YOLOX on a suitable task automatically derived from our dataset, and show that these lead to little promising results as well.

We address these issues by developing our own method based on predicting Gaussian heatmaps derived from the point annotations. We show that this method provides easily interpretable results which are on par with human annotations. However, it falls short to extrapolate into the systematically annotation free regions of high confluence. In our discussion and outlook sections, we consider further approaches to deal with this and other shortcomings. As a baseline, we include a cell count estimate based on foreground-background segmentation and an assumed constant cell density in the foreground. We show that this method has severe shortcomings.

Briefly we report on other failed experiments and explorations, such as clustering based approaches, classical computer-vision filters and Fourier space related transformations.

WIP: Write something about the actual application. Eg. what research at Sanger our software will be used for.

2 Related Work

WIP

3 Background

Many approaches have been developed to address the recognition of objects in images. This chapter is dedicated to the research application in human genome studies which motivates our work and to

the previously developed methods and concepts of data collection and deep learning image analysis on which we build our work.

3.1 Histopathology and Research Application

WIP Talk about what Sanger Institute wants to find out using our software

3.2 Previous Method

3.3 Phase Contrast Microscopy

WIP: Talk about phase-contrast microscopy. The properties of the images can be understood thereby.

3.4 Deep Learning Computer Vision

WIP Very short intro and further pointers for Deep Learning, point out that we have only supervised approaches.

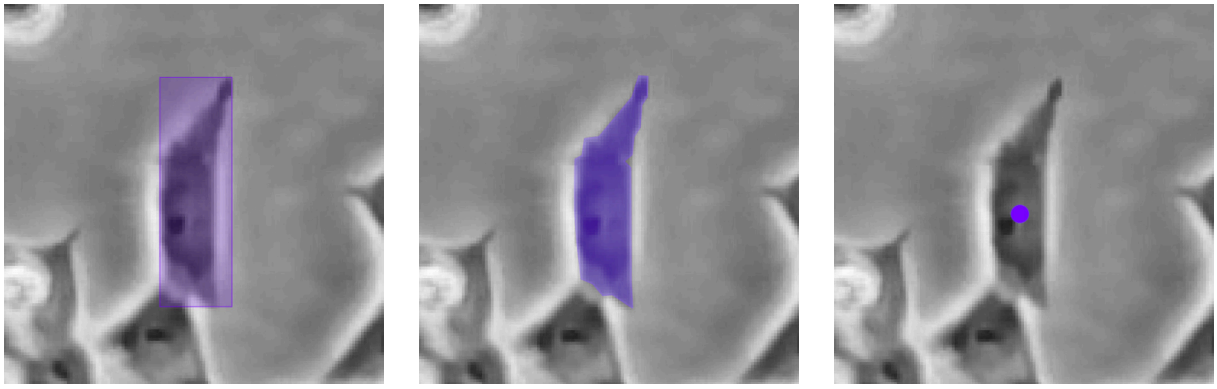

Figure 1: Different computer vision tasks. Object detection (left) typically requires two clicks to determine the minimal fully enclosing bounding box of an object. Instance segmentation (middle) labels each pixel as part of an instance or not. It can be done with brushes or polygons and requires many highly precise clicks. Keypoint detection (right) requires only one click per point of interest. All annotations typically require an additional class label - in our case there is only one class "Cell".

Amongst others, three major supervised learning objectives in computer vision (CV) and in particular deep learning (DL) in CV are Object Detection, Instance Segmentation and Keypoint Detection, as illustrated in Figure 1. While these are all different possible framings for localizing cells, they greatly differ in the type and amount of information they need as training targets. From the cheapest (keypoint detection) to the most expensive labels (instance segmentation) the amount of work and domain insight necessary for their creation differs greatly. Therefore, it is crucial to choose the right framing given the application's demands and annotation availabilities.

WIP label noise and sparse labeling

3.4.1 Object Detection

The goal in Object Detection, as seen in Figure 1, left, is to predict for every *object* in an image it's position, and commonly also its class, and spatial extension. The maximum number of detectable objects is usually determined by the architecture of a model, namely by the dimensions of the feature volumes from which the predictions are drawn.

As an evaluation metric commonly Average Precision (AP) is used. AP is given by the average of the precision over the recall, given by different confidence thresholds. Mean Average Precision (mAP) generalizes to multi-class problems by taking the mean of the AP over the class.

3.4.2 Instance Segmentation

The semantic segmentation task, as seen in Figure 1, middle, is to attribute each pixel in the target the class of the object it belongs to - the whole prediction is called a class map. Instance segmentation furthermore discerns between different objects. Whereas two visually touching or overlapping objects would be unified into one big region of similar predictions in semantic segmentation, in instance segmentation these class maps would be separate for each object, aka instance of a target class.

Class maps are usually expressed during training as an independent spatial probability distribution for each class, whereas the final output can be obtained through an argmax-projection. In instance segmentation the class can be inferred in a later step from one-dimensional spatial probability maps of each instance to save computational costs. In binary semantic segmentation tasks, eg. foreground-background segmentation, the output may be formed via binary thresholding of a probability map of one of the classes.

For evaluation of instance segmentations usually metrics like mean avarage precision on the basis of the intersection of predicted and ground-truth masks are used. However in this work we study segmentation methods only quantitatively, especially since we train using only point labels. For evaluation we therefore only note our visual impression.

3.4.3 Keypoint Detection

WIP

4 Problem Formulation

WIP

5 Dataset Description

Question: Is this a suitable place for it? (eg merge with Problem Formulation?)

WIP

6 Methods

6.1 Data Collection

WIP Introduce annotation with labelstudio.

WIP Explain microscope and decisions leading to it.

6.2 Object Detection with MaskRCNN

As an object detection model we chose MaskRCNN [1] in the implementation by *torchvision*[1]. We adapted the *TorchVision Object Detection Finetuning Tutorial*[2] to our setting.

Since this model was build for the common task of object detection, it requires information that goes beyond the one available in our single-class key-point detection task. We find a simple translation between the two tasks, by assuming a constant symmetrical spatial extension for all cells and using only one object class besides background. Concretely, each annotated center point became the center of a square target bounding box, which's side length was set constant to 50 pixels. MaskRCNN additionally predicts an instance segmentation on a separate output branch. As target instance masks we use the area covered by the target bounding boxes.

[1] We used stable release version 0.15 available at https://pytorch.org/vision/0.15/.

[2] https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html

We derive the model architecture and training settings from the *Torchvision Object Detection Finetuning Tutorial*[2]. These consisted in using a ResNet-50 [2] backbone with Feature Pyramid Pooling [3], a hidden dimension of the mask predictor of 256 and weights pretrained on COCO [4]. Stochastic Gradient Descent (SGD) is used as an optimizer.

Intended as a test-run we aim to overfit the model to the whole single image that was available to us at the moment. However, because the great number of masks to predict overflowed the working memory we implemented a custom tiling logic, which makes each of the 12 image tiles appear as an independent image to the model, as seen in Figure 3. We use no data augmentation and only color-channel-wide normalization as further preprocessing.

This model implementation would occasionally yield invalid empty masks, which would break other code. As a crude solution to this problem we chose to train the model only on masks fully contained within each image tile.

6.3 Zero-Shot Instance Segmentation

A secondary of our work was to generate segmentation masks in order to evaluate morphological properties of the studied cells. However, due to the lack of instance segmentation ground truth, we consider only zero-shot segmentation models. We evaluate Cellpose [5], [6], Stardist [7], SAM [8], and MicroSAM [9], established or novel promising methods for cellular segmentation. For Cellpose we tried to fine-tune the model with artificial ground truth segmentation masks, which's obtaining we describe in Section TODO. Similarly, we use artificially derived multiple-point- and box-prompts for the inference of the promptable SAM-based models.

Lacking ground truth, we do not evaluate the models quantitatively according to common segmentation metrics, but remain qualitative.

6.3.1 Cellpose

We use the official Python implementation of Cellpose [5], a generalist cellular segmentation algorithm and framework[3]. First we simply run inference of the pretrained Cellpose model, which ships with the package, according to the provided API. The Cellpose framework includes a size model to predict the average cell diameter. We use this size model's diameter prediction as a value for the respective required parameter of the segmentation model. We additionally test slightly and greatly smaller and bigger values derived from this.

Lastly, we aim to use the fine-tuning facilities of Cellpose 2 [6] to adapt the model to our dataset and improve the predictions. As we do not have real ground-truth segmentation masks, we used segmentations generated by running a watershed with the annotated center points as sources together with a foreground-background mask, obtained as described in Section 6.5.

6.3.2 SAM and MicroSAM

We use the work on MicroSAM by Archit et al. [9] for their API and for access to the original pretrained SAM models [8] and the models specifically fine-tuned on microscopy image datasets.[4] We survey different ways to make use of SAM's flexible promptability to find the best way to generate high quality segmentation masks using only the available center point annotations.

QUESTION: shall I invest time to include a graphic depicting each prompt type clearly?

[3]We used release version 2.0.5 of the official Python implementation available at https://anaconda.org/conda-forge/cellpose.

[4]We used release version 0.3.0 of the official Python implementation available at https://anaconda.org/conda-forge/micro_sam.

Automatic Instance Segmentation. The automatic instance segmentation capability of SAM as implemented by Kirillov et al. is simply wrapping the point prompt mechanism using individual single point prompts on a regular grid. Only promising instance mask candidates are selected. [8]

Single Point Prompts. Point prompts allow any number of positive and negative points to hint an instance. Given the available center point annotations, we prompt the model with each of them independently as a single positive point to generate a segmentation mask for each annotated cell.

Multiple Point Prompts. We observed general confusion of the SAM models regarding the scale of the objects which the point prompts pointed at. Therefore, we aimed to improve the results through hinting at the cells' sizes by including additional 4 negative points arranged in an X-shape around the positive annotation point in a constant radius according to an average cell size estimate.

Box Prompts. We make use of SAM's box prompt mechanism to generate segmentation masks, by converting each annotated center-point to a square of constant size, according to an average cell size estimate. This conversion is identical to the one employed for the artificial MaskRCNN ground-truth boxes as described in Section 6.2. We prompt the model with each box independently to generate a segmentation mask for each annotated cell.

6.3.3 Stardist

WIP

6.4 Regression based Counting with CellNet

WIP

(cellnet is my own heatmap based approach very similar to regression based counting by Xie et al. [10])

6.5 Other Failed Explorations

WIP (maybe merge with corresponding results)

6.5.1 Foreground-Background Segmentation

MicroSAM using Foreground Point Prompts. A rough baseline estimate for the number of cells on an image may be derived from measuring the area covered by cell material and assuming a constant density of cells in the cell material. For this purpose we aim to generate foreground-background segmentation masks using SAM. We aggregate all annotated cell center points as a single prompt for one big object. To improve results, we furthermore add a few manual opposite points in the background areas. We test different polarities of this set of points, eg. prompting positively for the background areas or for the cell material areas.

7 Experiments and Results

WIP general set of explanations what hypothesis we were after and how did we approach testing it?

WIP Explain microscope settings and decisions

7.1 Computational Environment

Neural network training was carried out on the University of Tartu High Performance Computing cluster Rocket (HPC) equipped with Nvidia Tesla V100 and A100 GPUs with varying amount of vRAM of which only one was used per experiment at a time. We kept no record of the specific node each experiment was run on, despite the great differences in different hardware components which may have affected training.

WIP BRIEFLY EXPLAIN MY SOFTWARE SETUP WITH VERSION CONTROL AND SYNCHING FOR REPRODUCIBLE COMPUTER SCIENCE EXPERIMENTS

7.2 MaskRCNN

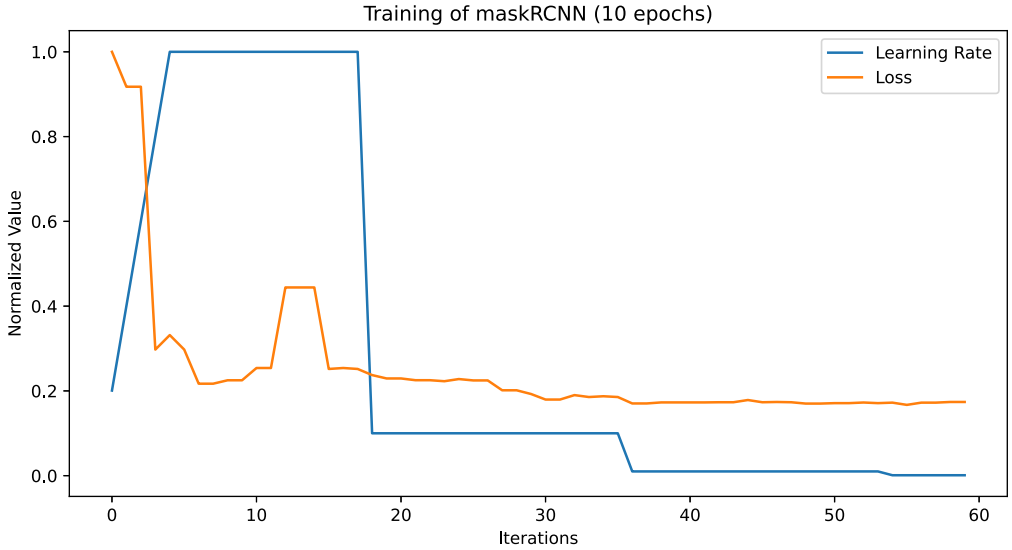

Figure 2: MaskRCNN training graph. The model trains well. Though a finer training schedule might improve results, the missing convergence is more likely due to inadequate training data.

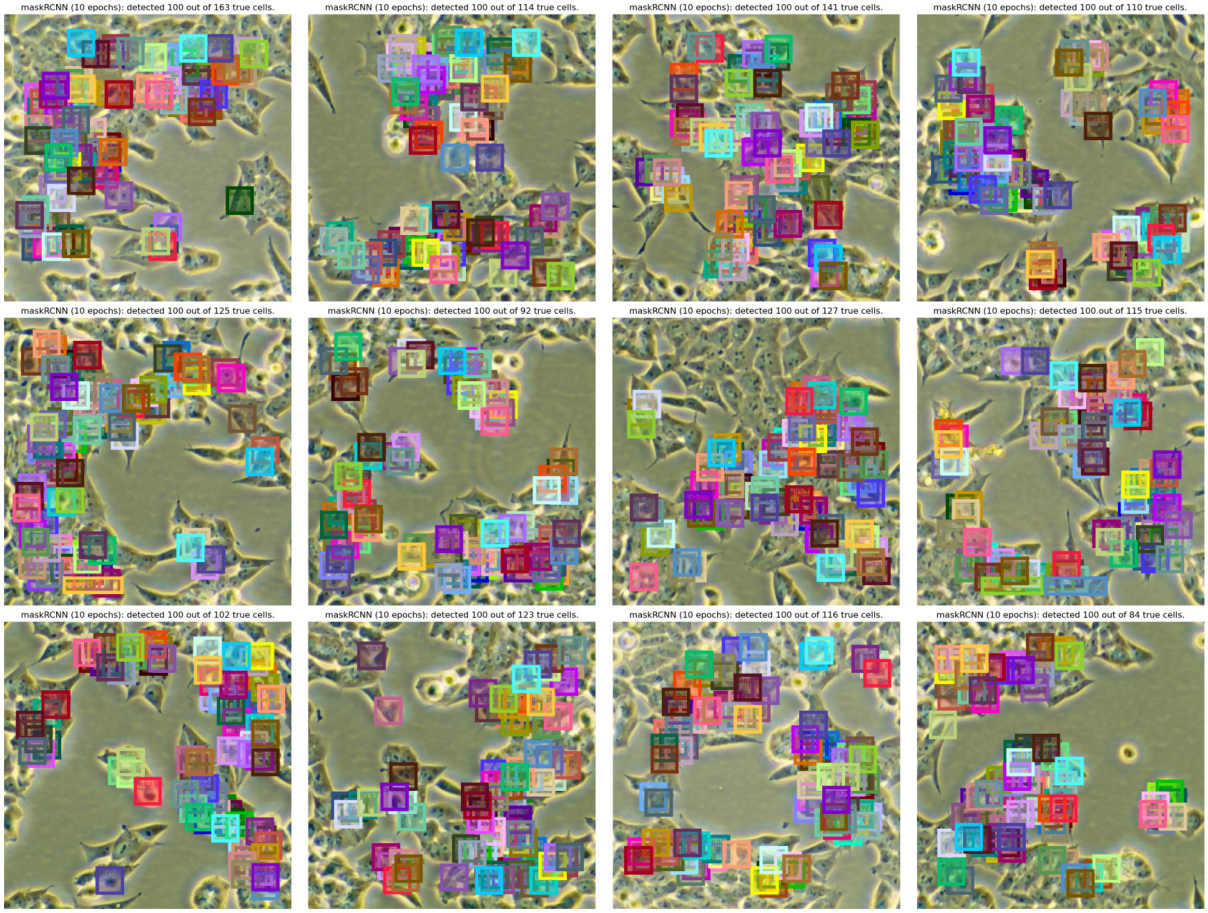

Figure 3: MaskRCNN predictions on the tiled image. Each cell center is modeled with a constant sized bounding box and an equivalent instance mask. Only 100 objects which lie fully within each tile are predicted. The brightness of the tiles differ, because each tile is normalized independently. Many redundant predictions cluster at certain locations, while predictions are missing completely in other regions.

We trained the model with an SGD momentum of 0.9, a weight decay of 0.0005 and step-wise reducing learning rate of 0.005, decaying every 3 epochs with γ of 0.1 and with a linear warm-up of 5 epochs. We trained 10 epochs long.

The model trains well with the proposed settings according to the training graphs in Figure 2. Despite raising the number of object predictions in the model's API, the maximum of 100 objects remained fixed for unknown reasons. We included only masks lying fully within each tile to avoid invalid empty predictions. We report no understanding of the mechanisms in the model API that lead to this behavior.

We observe that the model systematically avoids predictions in certain image regions, while others are characterized by a vastly overestimated number of cells, as can be seen by the irregular placement of bounding boxes in Figure 3. We think this is due to the fact that MaskRCNN is designed to let us choose the most promising instances according to a probability threshold or a fixed number of objects. However, we already maxed out the number of predictions as possible through the API and therefore couldn't anymore make a probability based selection. All model predictions have a confidence of $37 - 39\%$. This model implementation thus seems unusable for such densely populated scenes as a 10x magnification microscopy slice. This could have been improved by using smaller tiles.

The model learns well to predict the constantly square bounding boxes and instance masks. However, we assume that the loss terms which refer to these reduce the training signal for the only actually relevant information - the object positions. Because of the great memory requirements of handling several hundred independent masks on the whole image, we needed to separate the image into inde-

pendent tiles. We therefore concluded that it would be more promising to abandon object detection models at this point.

Due to the constant object extension and the too low cap in the number of predictions, the results were not yet suitable for quantification according to an object detection metric. We decided to not iprove the shortcomings, but move on to other approaches, which seemed more suitable and promising.

7.3 Cellpose

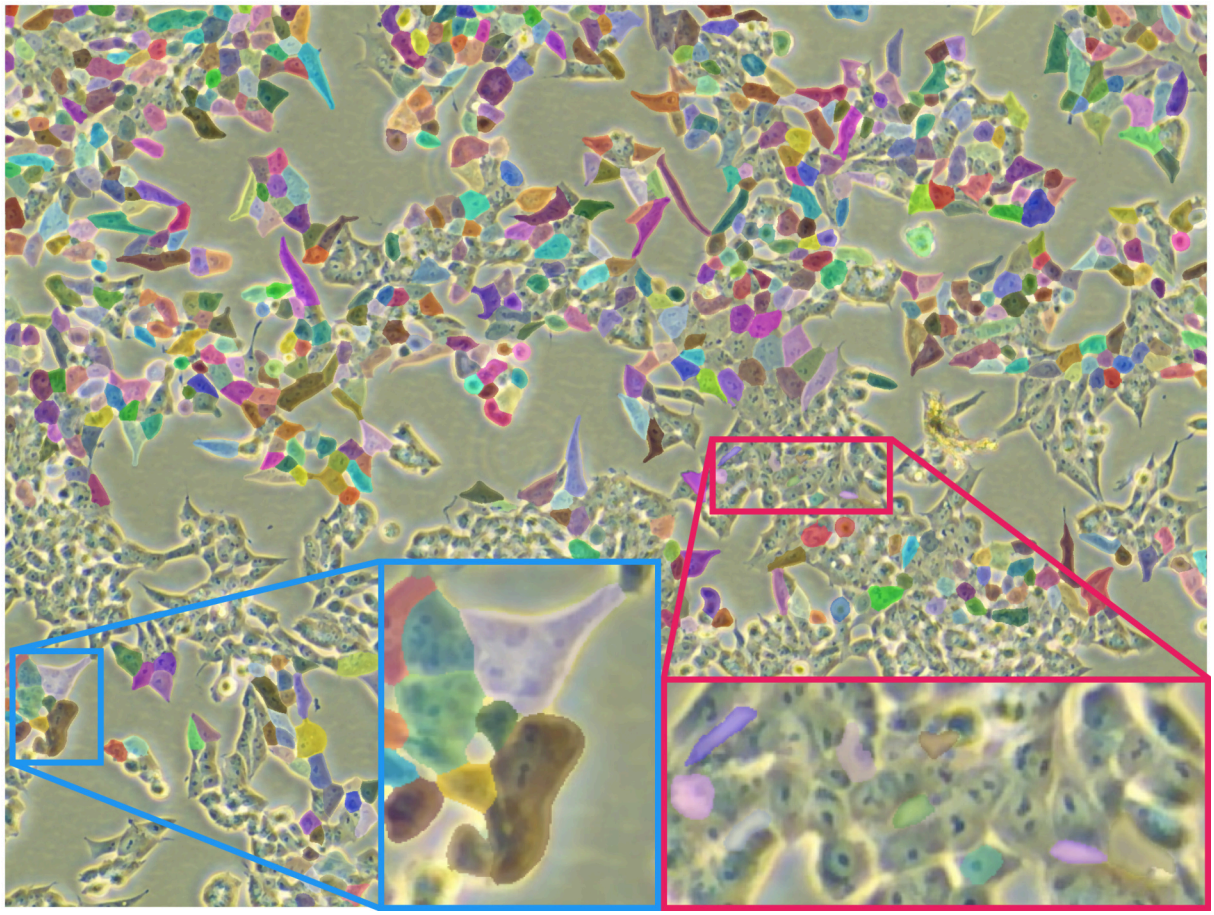

Figure 4: Predictions by pretrained Cellpose. Many cells are accurately segmented by the pretrained Cellpose model. But most cells are missed completely. Some neighbouring cells get merged and are segmented as one (blue). Whereas some cell-shaped holes within the cell material get falsely segmented as cells (red).

The average cell diameter was predicted by the pretrained Cellpose size model as between 38.5 and 39.3 pixels for all 3 annotated images. This matches with a human sight estimate. Due to a lack of representative amounts of annotations with spatial extension, the relative insignificance of the exact value of this parameter to the qualitative nature of the experiments based on it, and the different focus of this work, we chose not to quantitatively reevaluate this model.

The pretrained Cellpose segmentation model's predictions are displayed in Figure 4. It fails to detect a great proportion of cells in the image. A few detections are false positives of background holes between cells. Some clusters of neighbouring cells get predicted as a single instance. For the instances that have been identified correctly, however, the quality of the masks is generally very high. The masks mostly follow the visible outlines well and have smooth edges. The predictions are very sparse, but of high quality with very little artefacts.

The model has detected a total number of 2971 cells on three images using the automatedly estimated diameter, which have been annotated with a total number of 5671 cells. Cellpose thus exhibits a very

low recall of $< 52\%$ on the dataset, even not accounting for the significant ground-truth sparsity and false positive detections.

We prompted the segmentation model with different diameters besides the one automatically predicted by the size model, with ±2px and $\cdot\, 3^{\pm 1}$ relative to that diameter. However, we report no improvement in the results. The algorithm is not very robust to small changes in the diameters - a change from 39px to 40px or 36px drastically reduces the amount of segmented cells. Very small diameters lead to increased false positive detections of impurities in the sample and of background enclosures (even large ones). Larger diameters tend to merge increasingly more cells, with segmentation boundaries adhering less and less to the boundaries visible in the image.

We failed to retrain Cellpose on our custom generated fake ground-truth segmentations for unknown technical reasons. Despite the simplicity of the training API, the training just froze at the beginning. Also, changing the model into training mode suddenly nulls the model's predictions. Because of these difficulties we decided to abandon Cellpose. We discuss the possibility of retraining Cellpose in Section 8.

7.4 SAM and MicroSAM

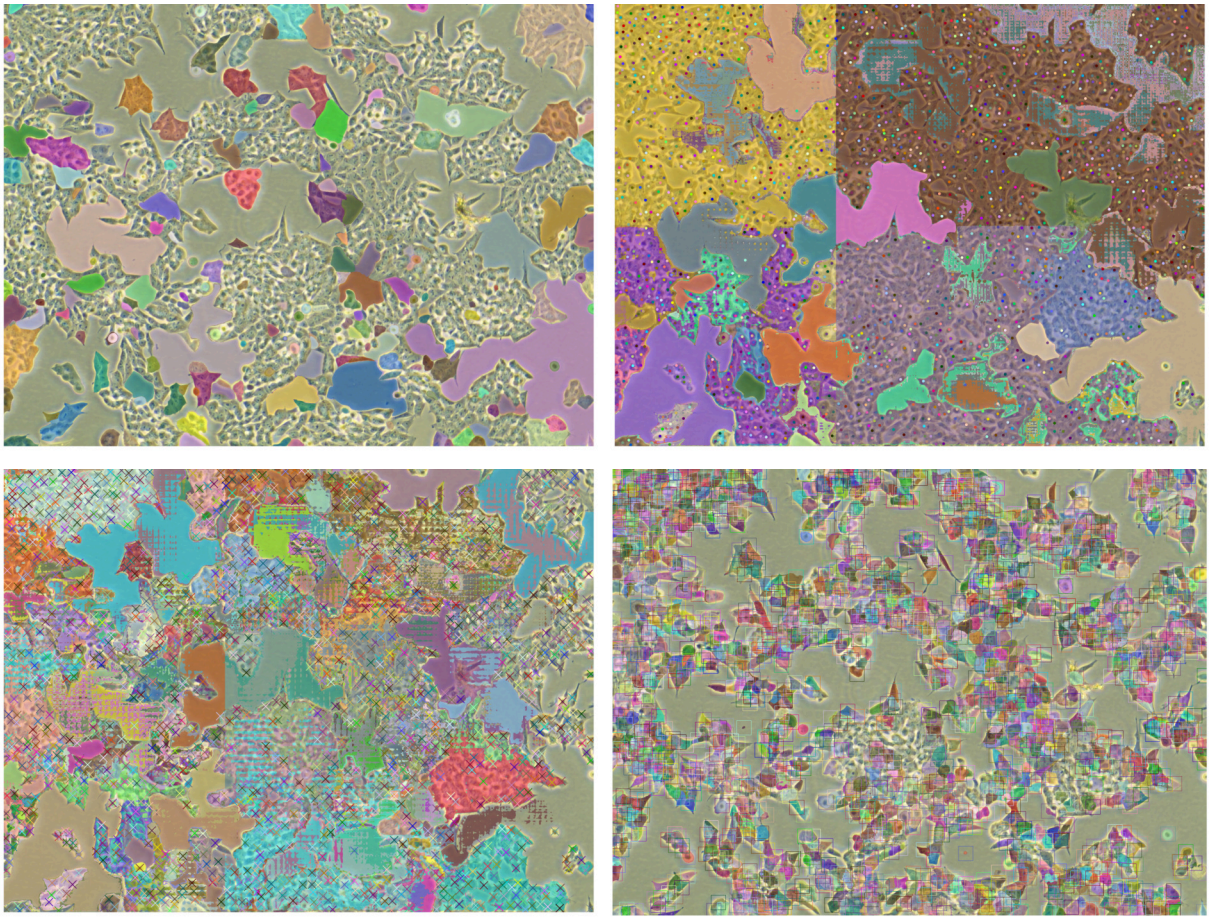

Figure 5: Results from different methods to prompt SAM (using the ViT-H backbone [11]). Automatic instance segmentation based on a regular point grid (upper left); independent point prompts at each annotated center point (upper right); independent multi-point prompts including 4 negative points in X-configuration with 12 pixel radius (lower left); and independent box prompts with 25 pixel radius (lower right).

WIP

7.4.1 Stardist

WIP

7.5 CellNet

WIP

7.6 Other Failed Explorations

WIP

QUESTION: There will be many figures here. Instead of letting each one consume a lot of space, I'd combine them into a gridded view, despite being partly completely separated methods?

7.6.1 Stardist

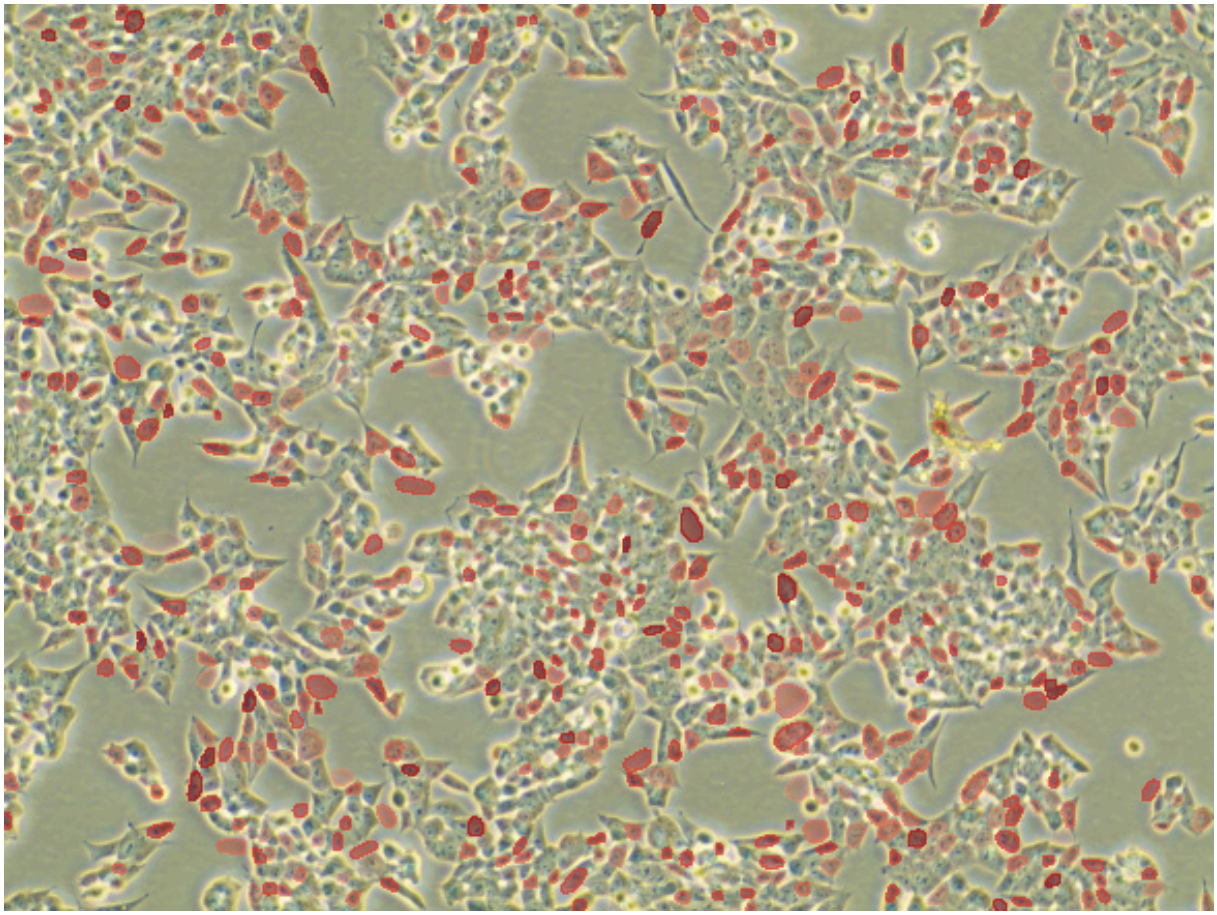

Figure 6: 0-shot instance segmentation predictions of Stardist [7].

8 Discussion

9 Conclusion and Outlook

Bibliography

[1] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN", in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[2] 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.

[3] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[4] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context", in *Computer Vision--ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 2014, pp. 740–755.

[5] 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.

[6] M. Pachitariu and C. Stringer, "Cellpose 2.0: How to Train Your Own Model", *Nature methods*, vol. 19, no. 12, pp. 1634–1641, 2022.

[7] U. Schmidt, M. Weigert, C. Broaddus, and G. Myers, "Cell Detection with Star-Convex Polygons", in *Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II*, 2018, pp. 265–273. doi: 10.1007/978-3-030-00934-2_30.

[8] A. Kirillov *et al.*, "Segment Anything", *arXiv preprint arXiv:2304.02643*, 2023.

[9] A. Archit *et al.*, "Segment Anything for Microscopy", *bioRxiv*, pp. 2023–8, 2023.

[10] W. Xie, J. A. Noble, and A. Zisserman, "Microscopy cell counting and detection with fully convolutional regression networks", *Computer methods in biomechanics and biomedical engineering: Imaging & Visualization*, vol. 6, no. 3, pp. 283–292, 2018.

[11] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale", *arXiv preprint arXiv:2010.11929*, 2020.

10 Appendix

WIP

10.1 Comparison of SAM and MicroSAM Backbones

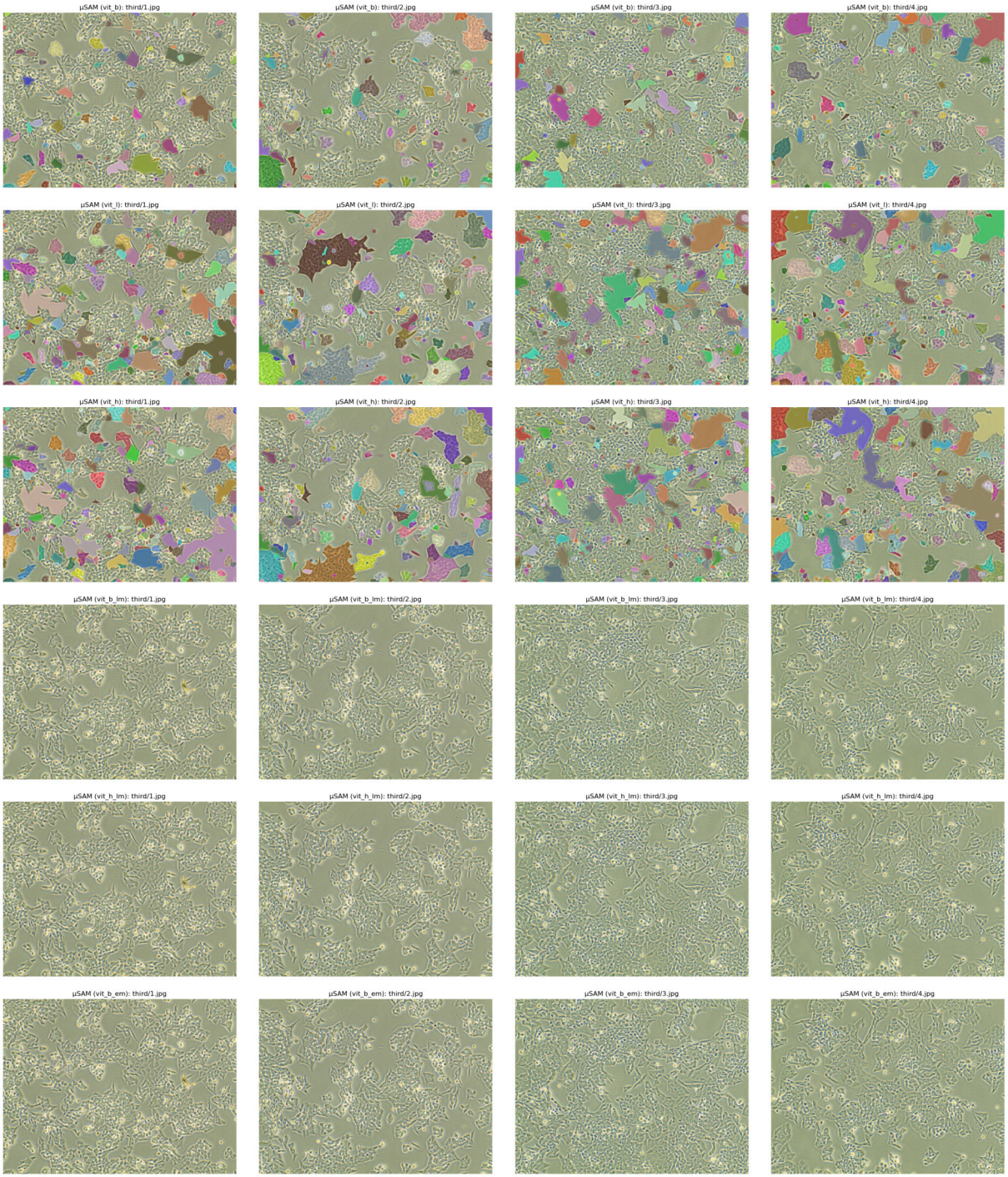

Figure 7: 0-shot instance segmentation results using different pretrained SAM and MicroSAM backbones. The columns show four of the collected images. The rows show the predictions using different backbones. The upper three correspond to the pretrained models by Kirillov et al. [8] sorted by size ascending downwards. The lower three correspond to the models finetuned on microscopy images by Archit et al. [9]. We include the two differently sized models pretrained on light microscopy (LM) and one pretrained on electron microscopy (EM).

WIP