Usability of deep learning pipelines for 3D nuclei identification with Stardist and Cellpose

Giona Kleinberg, Sophia Wang, Ester Comellas, James R. Monaghan, Sandra J. Shefelbine

PII:	S2667-2901(22)00042-0
DOI:	https://doi.org/10.1016/j.cdev.2022.203806
Reference:	CDEV 203806
To appear in:	*Cells and Development*
Received date:	13 February 2022
Revised date:	21 July 2022
Accepted date:	22 August 2022

Usability of Deep Learning Pipelines for 3D Nuclei Identification with Stardist and Cellpose

Running Head: Cell Segmentation Pipelines

Giona Kleinberg[a], Sophia Wang[b], Ester Comellas[c,d], James R. Monaghan[e,f], and Sandra J. Shefelbine[a,d, *]

[a]Department of Bioengineering, Northeastern University, Boston, USA
kleinberg.g@northeastern.edu

s.shefelbine@northeastern.edu

[b]Department of Electrical and Computer Engineering, Northeastern University, Boston, USA

wang.soph@northeastern.edu

[c]Serra Húnter Fellow, Department of Physics, Laboratori de Càlcul Numeric (LaCàN), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain

ester.comellas@upc.edu

[d]Department of Mechanical and Industrial Engineering, Northeastern University, Boston, USA

[e]Department of Biology, Northeastern University, Boston, USA

j.monaghan@northeastern.edu

[f]Institute for Chemical Imaging of Living Systems, Northeastern University, Boston, USA

Grant Information:

National Science Foundation #1727518, European Commission MSCA-GF-COMPLIMB-841047

*Author for correspondence: s.shefelbine@northeastern.edu

334 SN, 360 Huntington Avenue, Boston, MA 02115

Abstract

Segmentation of 3D images to identify cells and their molecular outputs can be difficult and tedious. Machine learning algorithms provide a promising alternative to manual analysis as emerging 3D image processing technology can save considerable time. For those unfamiliar with machine learning or 3D image analysis, the rapid advancement of the field can make navigating the newest software options confusing. In this paper, two open-source machine learning algorithms, Cellpose and Stardist, are compared in their application on a 3D light sheet dataset counting fluorescently stained proliferative cell nuclei. The effects of image tiling and background subtraction are shown through image analysis pipelines for both algorithms. Based on our analysis, the relative ease of use of Cellpose and the absence of need to train a model leaves it a strong option for 3D cell segmentation despite relatively longer processing times. When Cellpose's pretrained model yields results that are not of sufficient quality, or the analysis of a large dataset is required, Stardist may be more appropriate. Despite the time it takes to train the model, Stardist can create a model specialized to the users' dataset that can be iteratively improved until predictions are satisfactory with far lower processing time relative to other methods.

Keywords: 3D Machine Learning, Microscopy, Cell Segmentation

1. Introduction

Quantifying data in microscopy images remains a challenge in biology research and biomedical applications. To analyze objects of interest such as cells, membranes, and nuclei, quantitative data such as size, location, and spatial distribution of the objects must be collected.[1] Many fields of biological study require image acquisition and quantification of data from these images.

Methods of quantifying data in 2D images are well established and feasible to do manually on smaller datasets.[2–5] Complications are introduced, however, when analyzing 3D image stacks (Figure 1).[4,6,7] Image stacks, sometimes containing hundreds of image slices generated by 3D microscopy methods, dramatically increase the dataset's size.

As modern microscopy methods grow in efficiency and complexity, the need for automated methods of processing the datasets also increases.[7] There are currently many emerging methods for handling the large influx of 3D microscopy image datasets.[9–14] Among these methods, advances in deep learning have led to algorithms that significantly reduce the time required to identify and segment distinct objects within 3D images.[4,6,15,16] Instance segmentation is the process of locating and delineating objects. Most cellular objects, membranes, and nuclei have relatively simple shapes, making them excellent targets for automated instance segmentation using deep learning.[17] Due to the benefits of deep learning approaches such as increased prediction quality and greatly increased efficiency, knowledge on how to use such tools should be widely distributed.[8,18] This is especially important for those with minimal knowledge of machine learning and coding who may be overwhelmed when applying the newest deep learning algorithms.

Cellpose is a relatively new user-friendly tool utilizing deep learning for 3D cell segmentation.[18] Cellpose uses a pre-trained model based on a diverse dataset containing over 70,000 fluorescently labeled cells. The model is consistently retrained on user data to improve model versatility and prediction performance. Cellpose can be easily installed and used via the custom GUI or the command line. Cellpose requires no manual model training to use, making it an ideal choice for those without experience in deep learning or those who are aiming to generate predictions quickly. Additionally, Cellpose excels at segmenting convoluted object shapes due to the large and varied training set of the pre-trained model.[19]

Another prominent deep learning algorithm for segmenting cell nuclei is Stardist.[19,20] The open-source Stardist algorithm can train a neural network on a user's data using star-convex polygons (a more versatile type of bounding box compared to a simple rectangle used to find each cell’s shape) to identify cells by finding objects that match a general shape. However, it is not designed specifically to handle more convoluted object shapes.[19,21] The algorithm was first created to handle blurry, crowded, or otherwise abnormal 2D images, for which other algorithms struggled to create accurate predictions. Later extended to 3D image stacks, the algorithm specializes in predicting dense groups of cells and nuclei in images with large amounts of background noise. We used an application of the Stardist algorithm in a Google Colab notebook from the ZeroCostDl4Mic toolbox.[7] This application is convenient as it can run in a notebook within Google Colab. Model training and data prediction can be performed using a Google Colab runtime, accessing files stored in Google Drive, which significantly increases the usability of the algorithm for unfamiliar users.

The objective of this paper is to provide a comparison of Stardist and Cellpose regarding prediction quality, the time investment in learning, and overall usability. The application of each algorithm is conveyed through a demonstration on a 3D light sheet fluorescence microscopy dataset of 24 axolotl salamander humeri. Proliferative EdU-stained cells were counted in both experimental and control groups.

2. Material and methods

2.1. Image Acquisition

The images used here are part of a larger study in which we analyzed the effect of local mechanical stimuli on joint shape in regenerating axolotl salamander (*Ambystoma mexicanum*) limbs.[23] Animal forelimbs were bilaterally amputated proximal to the elbow joint. GSK1016790A (GSK101) was reconstituted in DMSO and injected intraperitoneally at 50µg/kg at 21 days post amputation (dpa, n=6). GSK101 is a TRPV4 agonist, a channel involved in the mechanotransduction of stimuli.[27] Control animals (n=6) were injected with DMSO. Injections were repeated at 48-hour intervals. At 31 dpa, animals were injected intraperitoneally with 5-Ethynyl-2'-deoxyuridine (EdU) and L-Azidohomoalanine (AHA). EdU can be used as a measure of cell proliferation since it is incorporated in newly synthesized DNA. AHA was used for the segmentation of the humerus rudiment since it labels extracellular proteins. Limbs were collected 18 hours after EdU and AHA injection then fixed and stained as described in Duerr et al. 2020.[28] A Zeiss light sheet Z.1 microscope was used to obtain the images. The dataset was chosen for testing due to the large size (each image was approximately 0.75 GB, in-plane about 1,300 pixels by 1,500 pixels with approximately 200 slices) and the number of objects in each image (100-500 cells). The resolution of each image was 0.915 microns x 0.915 microns x 4.945 microns per voxel. Although Stardist and Cellpose can accommodate anisotropic data (such as this dataset), some datasets may require isotropic preprocessing in order to obtain satisfactory predictions. Another consideration, AHA staining resulted in image backgrounds of varying intensities, further challenging cell segmentation. A low signal-to-background ratio was observed between the objects of interest and background, which negatively impacted prediction quality and helped highlight each algorithm's ability to handle less than ideal data.

2.2. Creating Training Sets for Stardist

The training set consisted of 15 training images with dimensions of 256 pixels by 256 pixels by 16 slices, (Figure 2A) in which each cell in the image was manually identified to create labeled images (Figure 2B). The size of the training images was selected to be small enough to manually identify all cells in the image while remaining large enough to contain an average of at least ten cells. Training images were cropped out of the primary dataset such that they provided an accurate representation of the whole dataset. We ensured that some of our training images included boundaries of the limb as pilot data indicated boundaries of the limb could pose problems during processing due to intense AHA staining. We also included images in the training set from both the experimental and control groups. In the training images, cells were manually outlined using the freehand selection tool in Fiji[26] to trace the perimeter of each cell appearing within a slice. The selections were saved as regions of interest (ROIs) to the ROI Manager function and were then turned into an ROI Map using the LOCI plugin (Figure 2C). ROI Maps of each slice were then joined together using the concatenate ImageJ function to create the corresponding labeled data for each training image in the training set (Figure 2D). Across slices, the ROI for each cell was kept the same color in every ROI Map. With each cell having a unique color, Stardist recognizes cells across all the slices they appear in using that unique color of the cell across the concatenated ROI Maps.

2.3. Training Model

All training images and their corresponding labeled images making up the training set were then uploaded to Google Drive. Data augmentation was used to create a larger training set by reflecting, rotating, and distorting each pair of image stacks using the Stardist ZeroCostDL4 Google Colab Notebook.[7] The augmented training set was then used to train a model using 320 training iterations referred to as epochs, and a patch height of eight slices corresponding to the training set size in the same notebook. Patch height represents the depth of each comparison during prediction and was set to 8 slices in order to fit within the training images' height of 16 slices. It is possible to start training with a pre-trained model, but this feature was not used.

2.4. Preprocessing in Fiji

Before running Stardist or Cellpose, each image in the primary dataset (Figure 3A) was cropped using Fiji to the region of desired cell counting, which in our images was the humerus rudiment (Figure 3B). Background subtraction was then applied with a rolling ball radius of 50. A value of 50 was used based on performance on pilot data and represented the radius of the curvature of the paraboloid used to subtract the background on each pixel based on neighboring local pixels. We also examined the effects of this rolling ball radius on results. To avoid memory issues during Stardist processing, an ImageJ macro was created to crop each image stack into 16 or 25 equal-sized tiles using 4 or 5 divisions respectively in order to ensure the x and y dimensions of the image stack were each under 300 pixels (the limit that could successfully be processed in the Stardist Colab Notebook, Figure 3C). Fewer divisions were favored to minimize the number of image edges during processing as long as the x and y dimensions were under 300 pixels. By minimizing divisions, error was limited to a negligible amount for our data. If this error is not negligible, overlapping the tiles and then reconstructing them is a more complex yet effective way to further reduce error. To ensure an accurate comparison, tiling was done before processing images in Cellpose as well, even though the local installation of Cellpose was able to run the full image stack at once.

2.5. Processing in Stardist

Each tile was then processed from a Google Drive folder using the trained Stardist model via the Google Colab notebook. Stardist processed each humerus (consisting of multiple tiles) in approximately 3-4 minutes. Prediction outputs from Stardist in the form of ROI Maps (Figure 3D) were then downloaded for post-processing in Fiji. Processing was completed using a Google

Colab runtime which has a performance independent of a user's computer specifications since it runs remotely.

2.6. Processing in Cellpose

After preprocessing in Fiji using background subtraction and tiling, each image was run through Cellpose's algorithm (version 0.6) as well. Tiling is not necessary for Cellpose but was used in order to compare the algorithm accurately with Stardist. Cellpose's estimated diameter function was used, except for images with little to no cells for which we supplied an estimated radius based on the average within the dataset. The nuclei channel was used, but Cellpose can identify cytoplasm as well. Like Stardist, Cellpose generated an ROI map for each image processed. Unlike Stardist, however, Cellpose also generated an NPY file (NumPy array file that can be used for other methods of image quantification in python). Since Cellpose operates locally (instead of in a Google Colab session like Stardist), the runtime is heavily dependent on the computer's processing power. Note that a preliminary 3D Cellpose Colab Notebook has recently been developed and can be used instead of a local installation.[30] The estimated average runtime for an entire image stack (without tiling) was approximately 10 hours (on a 1.99 GHz processer with 16 GB ram). Varying tiling size had a negligible effect on the runtime.

2.7. Post-Processing in Fiji and MATLAB

Both algorithms output ROI maps during processing so post-processing was consistent between both methods. Since predictions were created for the entire image stack, including the area around the humerus, an ImageJ macro was created to mask the ROI Map predictions with a mask of the humerus. This was done to isolate the predictions within the humeri by excluding objects found outside the area of interest. Since the prediction image stacks were still tiled, the corresponding mask of each humerus was cropped and tiled using identical parameters from the original crop and tiling to mask each tile individually. Next, the 3D Objects Counter function[28] was run on each masked tile without counting objects that did not fall within the area of interest (the humerus) due to the masking. The cell counts of all tiles corresponding to each original image were then summed and filtered for outlier volumes that indicate a false positive prediction. Using MATLAB, humeri proximo-distal axes were aligned, and humeri were cut on the proximal end to a normalized length to ensure cells were counted within an equivalent length across all humeri. The total number of proliferating cells was assessed for each humerus, and each cell was plotted in 3D using MATLAB with a sphere representing the size of the cell (Figure 3E). More details on the pipeline are available online.[29]

3. Results

3.1. Training Image Size

Whereas Cellpose uses a pre-trained database of thousands of images, Stardist requires self-training a model on your data. Through multiple attempts at training the model, we found the optimal dimensions for each pair of source and target images in the training set to be x and y dimensions of 128 pixels by 128 pixels per slice with a stack size of 16 slices. “Optimal” was determined by the size and number of objects in the image. We found the most success for our data when each image in the training set contained around ten objects in each slice with around 20 objects throughout the image stack. This was ideal for balancing many objects to train the model in each stack while keeping the number of objects manageable for manual segmentation. Based on the size of each object and their spatial distribution within a dataset, it is recommended that the dimensions of each image stack in the training set be adjusted to maintain a similar number of objects. The number of image pairs (training stack and labeled stack) required for a successful training set was found to be 15 pairs for our dataset. This value is especially variable as datasets with harder-to-predict objects (usually due to blurry images or closely-packed objects) will need a larger training set. In comparison, datasets with easy-to-predict objects (usually far apart objects with a high signal-to-background ratio) will need a smaller training set. These dimensions found and then used to create our training set represent the optimal values for our specific images and should not be used for every dataset. Instead, the trends and methodology described to find these dimensions should be utilized to find the optimal training image and training set sizes for each specific dataset.

3.2. Processing Image Size

We ran Cellpose locally and used Google Colab to run Stardist. Cellpose was able to handle the 1 GB image sizes when run on a standard desktop processor albeit 10 hours to process a single image were required. Due to the large dimensions of the dataset, Google Colab ran out of memory when processing full image stacks. Google Colab offers higher memory runtimes; however, for larger images, the higher memory is still insufficient to process a full image stack at once. This was solved by cropping each image stack into tiles and processing the tiles individually. We found that for our data, image stacks with around 100 slices required x and y dimensions of less than 230 pixels to be processed successfully without incurring memory errors (Appendix, Figure A.1). Image stacks with more slices required a smaller number of pixels in the x and y dimensions. For our example dataset, larger image stacks with slightly over 200 slices were successfully processed with x and y dimensions as high as 200 pixels. These optimal dimensions were found through incrementally increasing the image dimensions until Stardist was no longer able to process them. This process should be replicated when training on a new dataset in order to find the largest images that can be processed without incurring an error. We also found that if cropping into tiles was required, cropping in the x and y dimensions (in the image plane) was preferable to cropping through the image stack (z dimension). Cropping images in the dataset creates more image edges which, when processed, result in more object edge cases where predictions are less reliable. Cropping through the image stack in the z dimension creates a larger edge area as the entire 2D slices around the crop location end up as edges that introduce error to the predictions. To avoid this, cropping to avoid memory errors should be limited to the x and y (in-plane) dimensions, if possible. Another possible alternative could involve deriving a model based on image parameters to find an ideal processing image size. The Stardist algorithm can also be run locally without using the ZeroCostDl4Mic Google Colab notebook to avoid incurring memory errors.

3.3. Epoch Number

One important parameter tested through the application of Stardist was the number of epochs used during model training. An epoch is a hyperparameter representing an iteration of the learning algorithm through the entire training set. Another related parameter of a model is its loss. The loss of a model represents the difference between a model's expected prediction and actual prediction during model validation. A slight loss is usually correlated with higher prediction quality. The value of each epoch on the model's resultant loss diminishes exponentially as the epoch number increases until eventually asymptotically approaching the minimum loss that can be reached with a particular training set. Based on this knowledge, we determined the smallest epoch number that still reached this minimum loss to avoid redundant training and overfitting. We found that a training set of our size required an epoch number of 320. Additional epochs (training iterations) beyond 320 resulted in little benefit (Appendix, Figure A.2). Similar to training and processing image dimensions, this number is specific to our dataset and can be found for a new dataset through incrementally increasing or decreasing the epoch number when training a model. When optimizing the number of epochs, performance on the validation set can be used as an indication of possible overfitting. Epoch optimizations have no effect on the quality of the predictions. However, if similar models are being trained for frequent or long term-use, the time it saves on each iteration of training can save considerable time overall.

3.4. Background Subtraction

We analyzed the effect of background subtraction by changing the rolling ball radius used in background subtraction. The recommended rolling ball radius is the radius of the largest object to be found in an image. The average diameter of axolotl nuclei is 30±10 µm, or about 25 pixels. Four radii were used to track the improvement in predictions: a minimal value of 5 pixels, about half the diameter, 10 pixels, the diameter, 25 pixels, and twice the diameter of the nucleus, 50 pixels. We determined smoothing in (another available parameter when using the background subtract tool in Fiji) did not have an effect on the cell count but provided clearer images as shown in Figure 4. Background subtraction enhanced the contrast between objects and the background (Figure 4). This typically increased the cell count as objects became easier to identify and segment. All rolling ball radii tested resulted in more segmented cells when compared with no background subtraction on the same image (Figure 4). Based on our testing, it is recommended that the rolling ball radius used is slightly greater than the average dimension of the largest object in the image. When processing our dataset, a rolling ball radius of 25 was the practical minimum required for successful segmentations without too many false positives. However, a rolling ball radius of 50 (following the theoretical guideline that the rolling ball radius should be approximately the size of the largest object in the image) was used for processing to ensure no cells were subtracted entirely out of the cell count. Since inspecting every dataset image for the largest object is not always feasible, allowing for a margin of error by assigning a high rolling ball radius is recommended. It is much easier to locate false positives in the cell count after post-processing as opposed to having too small a rolling ball radius that results in objects of interest getting subtracted. The trends in rolling ball radius were similar for both Stardist and Cellpose. The results from Cellpose are shown in Figure 4.

It is very challenging to know the ground truth for the exact dimensions or quantity of segmented cells in an image because all prediction methods, including manual segmentation, are susceptible to biases. However, by comparing predictions, we can determine how specific parameters quantitatively affect the cell count. As long as all images in a dataset are processed with the same parameters, quantitative comparisons between groups can be made (even if we do not know the ground truth).

3.5. Effect of Tiling on Background Subtract

To directly compare the results of Stardist and Cellpose, we analyzed a single image in Stardist with tiling and in Cellpose with and without tiling. In Cellpose, tiling resulted in an increase in cell count in images. This is most likely because each tile had a more consistent background compared to the background of the entire image, making it easier for the algorithm to identify objects. Due to the varying background intensity of uncropped images and varying staining intensity of nuclei throughout such images, background subtraction makes less drastic changes to pixel intensity. It is possible that when an image is cropped into tiles and background subtract is run on each tile, the localized area of each tile has a smaller range of intensities and can therefore show higher contrast once background subtraction is run.

3.6. Overlapped Cells and 3D Objects Counter

We found that Stardist performed consistently well on patches of closely packed cells that appeared overlapped in 2D slices of the image stack (Figure 5). The model used the expected shape and size profile from the algorithm’s training to consistently segment the clustered cells into distinct objects. Another important finding is that despite overlapped cells (like those shown in Figure 5) being recognized as independent objects by Stardist and Cellpose, the Fiji 3D Objects Counter function counts all the cells together as one object. This is due to how 3D Objects Counter identifies individual objects through searching for clusters of voxels in binary images. Since 3D Objects counter looks at a binary version (each voxel is either part of an object or the background) of the ROI Maps output by Stardist and Cellpose, it cannot recognize the unique color of each cell in an overlap and therefore labels the overlap as one large object. In order to solve this problem and count each cell as its own object correctly, the overlapped cells need to be split (by placing a minimal layer of background intensity in between the cells) using a watershed or similar Fiji function that can split the objects. This division allows 3D Objects Counter to view each overlapped cell as its own object to mitigate the undercounting of cells.

3.7. Direct Comparison with Manual Segmentation

To compare the pipelines with each other and with manual counting, a subsection of the data was segmented by each method (Figure 6). A 256-pixel by 256-pixel subsection of 16 slices was cropped from the initial dataset. The subsection was processed using the Stardist ZeroCostDL4 Google Colab notebook and Cellpose finding cell counts of 24 cells and 20 cells respectively. The count obtained from the manual segmentation was 30 cells. Each cell resides within approximately 3-5 slices and the manual segmentation for an image subsection of this size took 3 hours which further highlights the need for algorithms when attempting to segment large datasets. These results are specific to the dataset tested. However the tips and trends found are generalizable to other datasets.

4. Discussion

With most images being processed in less than five minutes, Stardist’s quick prediction time significantly increases processing efficiency compared to Cellpose, which generated predictions for a single image sometimes requiring over 10 hours. Cellpose runtime is dependent on hardware limitations and can therefore be improved with higher processing power although it is unlikely to match Stardist’s prediction time even on high-end hardware. A quick processing time is very favorable when working with larger datasets or determining the effects of the various preprocessing parameters. However, this advantage of Stardist over Cellpose is only valid once a model has been trained in Stardist, which itself requires a significant amount of time. Since the amount of training required to train a model is independent of the dataset size, there is a critical dataset size where Stardist's quick processing time will still save time despite a large initial amount of time spent training. Once a model is trained sufficiently, it no longer requires more training and, as a result, Stardist is better suited towards projects with large datasets or an expected continuous influx of similar images. Datasets with few images are likely to cost more time in training than they save in processing with Stardist.

The critical dataset size where Stardist becomes more efficient is variable depending on the actual elements of the dataset. Far less training is needed when images contain objects with high signal-to-background ratio or when objects are spaced far apart (Figure 7). Consequently, images with closely clustered cells or a high-intensity background require a larger training set to produce an accurate model. Since a good training set should ideally have components representative of all the images, a dataset with highly varied and inconsistent backgrounds, object shapes, and intensities will also require a larger training set. There are also many useful tools in development that can be used to reduce the time spent creating a training set manually such as APEER, Ilastik, Segmentor, and ImJoy.[14,22–25] By reducing the time needed to train an initial model using such tools, Stardist's advantages can become more accessible for small datasets.

Another advantage of Stardist is the ability to improve prediction quality. Once a model has been trained, further training sets can be made and used in conjunction with the model to train it further until an acceptable accuracy is achieved. Even when analyzing a small dataset, if Cellpose's predictions are not accurate enough, Stardist can be a valuable alternative. By training a model from scratch, Stardist's predictions can be tailored to a specific dataset and improved with further training leading to more accuracy. At the time of this research, Cellpose did not offer the ability to further train their model however this option has since become available.[34]

These findings are evidenced by our results from creating and comparing the most usable pipelines found on our dataset. We are therefore unable to determine a superior method for all types of datasets found in biological research. However, based on the results, informative trends can be observed. Since no model training is required, Cellpose should be used unless the dataset is large enough to make Stardist the more efficient tool even when accounting for training time. Stardist should be used when Cellpose predictions are not accurate enough to be used. This is likely to occur when datasets contain low contrast between the background and nuclei, contain highly clustered cells, or otherwise contain consistent irregularities that a model should be trained specifically to handle. Fitting these trends, Stardist would be the better option for the test dataset used here as well as similar datasets due to the high noise-to-background ratio and large image size and quantity.

The most efficient model for a project can be best determined primarily by the size of the dataset. Due to the large initial training time, Stardist is less efficient on small datasets when the time to train a model is greater than the time to process each image in Cellpose. On large datasets such as those with over a hundred images, the quick processing time of Stardist leaves it as the more efficient model despite the training time as processing the images in Cellpose would be slower than training a model and processing the images in Stardist.

There are still many opportunities for improvements to current deep learning software. Further development of tools that simplify manual object identification would significantly reduce the meticulous and lengthy process of creating a training set. This would make the dataset tailored predictions produced by Stardist far more accessible and efficient to use for the Google Colab application tested. As algorithms increase in complexity, the time to produce predictions of both methods will likely decrease while the accuracy of the predictions will increase. For larger datasets, manual methods of object isolation and segmentation are unfeasible and require such tools. As more development is done on these methods in 3D, they will be able to increase processing efficiency on datasets of all sizes and may surpass the accuracy provided by manual methods.

Acknowledgements

The authors would like to thank Timothy J Duerr and Johanna E Farkas for their help obtaining the images used in this study. Microscopy images were obtained from the Harvard University Center for Biological Imaging and the Northeastern University Chemical Imaging of Living Systems core. This work was completed using the Discovery cluster, supported by Northeastern University's Research Computing team. We acknowledge animal support from the Ambystoma Genetic Stock Center funded by NIH grant P40-OD019794.

Funding Statement

This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 841047 and NSF CMMI #1727518.

References

1. Noller CM, Boulina M, McNamara G, Szeto A, McCabe PM, Mendez AJ. A Practical Approach to Quantitative Processing and Analysis of Small Biological Structures by Fluorescent Imaging. *J Biomol Tech JBT*. 2016;27(3):90-97. doi:10.7171/jbt.16-2703-001

2. Al-Kofahi Y, Zaltsman A, Graves R, Marshall W, Rusu M. A deep learning-based algorithm for 2-D cell segmentation in microscopy images. *BMC Bioinformatics*. 2018;19(1):365. doi:10.1186/s12859-018-2375-z

3. Green JM, Appel H, Rehrig EM, et al. PhenoPhyte: a flexible affordable method to quantify 2D phenotypes from imagery. *Plant Methods*. 2012;8(1):45. doi:10.1186/1746-4811-8-45

4. Arganda-Carreras I, Kaynig V, Rueden C, et al. Trainable Weka Segmentation: a machine learning tool for microscopy pixel classification. *Bioinformatics*. 2017;33(15):2424-2426. doi:10.1093/bioinformatics/btx180

5. Carpenter AE, Jones TR, Lamprecht MR, et al. CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biol*. 2006;7(10):R100. doi:10.1186/gb-2006-7-10-r100

6. Nunez-Iglesias J, Kennedy R, Parag T, Shi J, Chklovskii DB. Machine Learning of Hierarchical Clustering to Segment 2D and 3D Images. *PLOS ONE*. 2013;8(8):e71715. doi:10.1371/journal.pone.0071715

7. von Chamier L, Laine RF, Jukkala J, et al. Democratising deep learning for microscopy with ZeroCostDL4Mic. *Nat Commun*. 2021;12(1):2276. doi:10.1038/s41467-021-22518-0

8. Jacquemet G. Deep learning to analyse microscopy images. *The Biochemist*. 2021;(bio_2021_167). doi:10.1042/bio_2021_167

9. Ayankoso S. The Surge in Deep Learning for Computer Vision: A Concise Review of Convolutional Neural Networks. Published online December 18, 2018. doi:10.6084/m9.figshare.11987520

10. Xie W, Noble JA, Zisserman A. Microscopy cell counting and detection with fully convolutional regression networks. *Comput Methods Biomech Biomed Eng Imaging Vis*. 2018;6(3):283-292. doi:10.1080/21681163.2016.1149104

11. Grishagin IV. Automatic cell counting with ImageJ. *Anal Biochem*. 2015;473:63-65. doi:10.1016/j.ab.2014.12.007

12. Takko H, Pajanoja C, Kurtzeborn K, Hsin J, Kuure S, Kerosuo L. ShapeMetrics: A userfriendly pipeline for 3D cell segmentation and spatial tissue analysis. *Dev Biol*. 2020;462(1):7-19. doi:10.1016/j.ydbio.2020.02.003

13. Falk T, Mai D, Bensch R, et al. U-Net: deep learning for cell counting, detection, and morphometry. *Nat Methods*. 2019;16(1):67-70. doi:10.1038/s41592-018-0261-2

14. Ouyang W, Mueller F, Hjelmare M, Lundberg E, Zimmer C. ImJoy: an open-source computational platform for the deep learning era. *Nat Methods*. 2019;16(12):1199-1200. doi:10.1038/s41592-019-0627-0

15. Bakas S, Reyes M, Jakab A, et al. Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge. Published online November 5, 2018. Accessed August 15, 2021. https://arxiv.org/abs/1811.02629v3

16. Jain V, Seung HS, Turaga SC. Machines that learn to segment images: a crucial technology for connectomics. *Curr Opin Neurobiol*. 2010;20(5):653-666. doi:10.1016/j.conb.2010.07.004

17. Prakash M, Buchholz TO, Lalit M, Tomancak P, Jug F, Krull A. Leveraging Self-supervised Denoising for Image Segmentation. In: *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*. ; 2020:428-432. doi:10.1109/ISBI45749.2020.9098559

18. Kromp F, Fischer L, Bozsaky E, et al. Evaluation of Deep Learning Architectures for Complex Immunofluorescence Nuclear Image Segmentation. *IEEE Trans Med Imaging*. 2021;40(7):1934-1949. doi:10.1109/TMI.2021.3069558

19. Stringer C, Michaelos M, Pachitariu M. Cellpose: a generalist algorithm for cellular segmentation. *bioRxiv*. Published online February 3, 2020:2020.02.02.931238. doi:10.1101/2020.02.02.931238

20. Schmidt U, Weigert M, Broaddus C, Myers G. Cell Detection with Star-convex Polygons. *ArXiv180603535 Cs*. 2018;11071:265-273. doi:10.1007/978-3-030-00934-2_30

21. Weigert M, Schmidt U, Haase R, Sugawara K, Myers G. Star-convex Polyhedra for 3D Object Detection and Segmentation in Microscopy. *2020 IEEE Winter Conf Appl Comput Vis WACV*. Published online March 2020:3655-3662. doi:10.1109/WACV45572.2020.9093435

22. Walter FC, Damrich S, Hamprecht FA. Multistar: Instance Segmentation Of Overlapping Objects With Star-Convex Polygons. In: *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*. ; 2021:295-298. doi:10.1109/ISBI48211.2021.9433769

23. Comellas E, Farkas JE, Kleinberg G, et al. Local mechanical stimuli correlate with tissue growth in axolotl salamander joint morphogenesis. *Proc R Soc B Biol Sci*. 2022;289(1975):20220621. doi:10.1098/rspb.2022.0621

24. O’Conor CJ, Leddy HA, Benefield HC, Liedtke WB, Guilak F. TRPV4-mediated mechanotransduction regulates the metabolic response of chondrocytes to dynamic loading. *Proc Natl Acad Sci*. 2014;111(4):1316-1321. doi:10.1073/pnas.1319569111

25. Duerr TJ, Comellas E, Jeon EK, et al. 3D visualization of macromolecule synthesis. Stainier DY, Sandoval-Guzman T, Sandoval-Guzman T, eds. *eLife*. 2020;9:e60354. doi:10.7554/eLife.60354

26. Schindelin J, Arganda-Carreras I, Frise E, et al. Fiji: an open-source platform for biological-image analysis. *Nat Methods*. 2012;9(7):676-682. doi:10.1038/nmeth.2019

27. Google Colaboratory. Accessed October 5, 2021. https://colab.research.google.com/github/MouseLand/cellpose/blob/master/notebooks/run_cellpose_GPU.ipynb

28. Bolte S, Cordelières FP. A guided tour into subcellular colocalization analysis in light microscopy. *J Microsc*. 2006;224(3):213-232. doi:10.1111/j.1365-2818.2006.01706.x

29. Comellas E, Kleinberg G, Lloyd K, Mueller T, Shefelbine SJ. *Pipeline for the 3D Shape and Cell Proliferation Analysis in a Regenerating Axolotl Humerus*. Zenodo; 2021. doi:10.5281/zenodo.5591984

30. Ouyang W, Le T, Xu H, Lundberg E. Interactive biomedical segmentation tool powered by deep learning and ImJoy. Published online February 24, 2021. doi:10.12688/f1000research.50798.1

31. Berg S, Kutra D, Kroeger T, et al. ilastik: interactive machine learning for (bio)image analysis. *Nat Methods*. 2019;16(12):1226-1232. doi:10.1038/s41592-019-0582-9

32. Sommer C, Straehle C, Köthe U, Hamprecht FA. Ilastik: Interactive learning and segmentation toolkit. In: *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. ; 2011:230-233. doi:10.1109/ISBI.2011.5872394

33. Borland D, McCormick CM, Patel NK, et al. Segmentor: a tool for manual refinement of 3D microscopy annotations. *BMC Bioinformatics*. 2021;22(1):260. doi:10.1186/s12859-021-04202-8

34. Stringer C, Pachitariu M. Cellpose 2.0: how to train your own model. Published online April 5, 2022:2022.04.01.486764. doi:10.1101/2022.04.01.486764

Figures

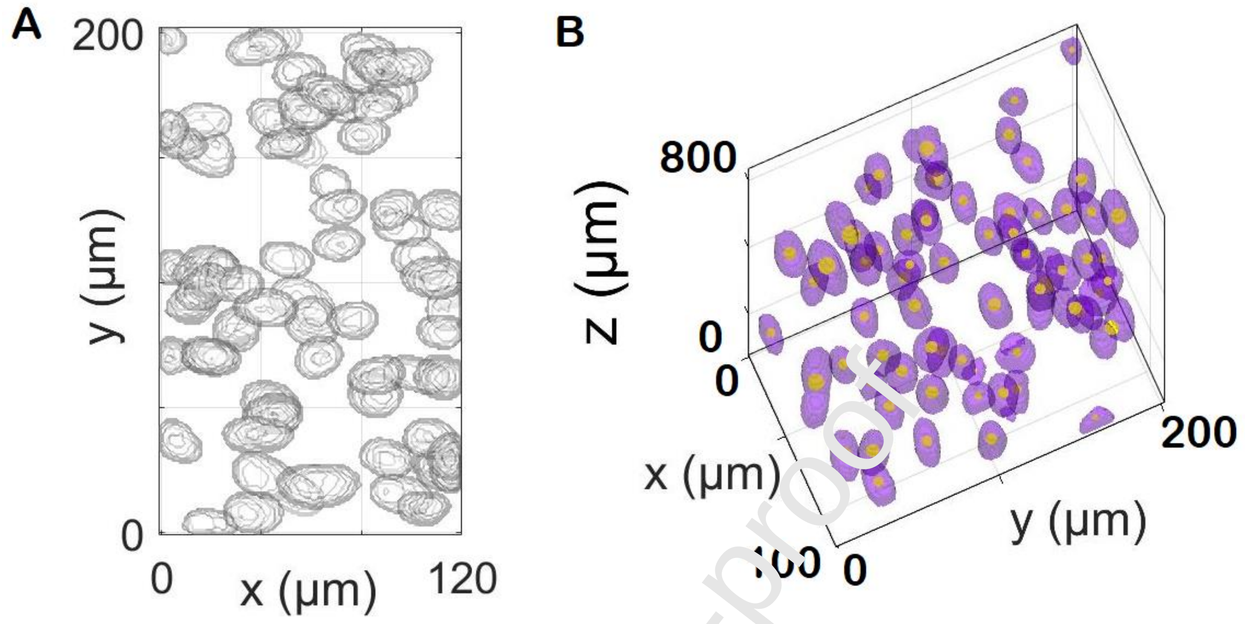


Figure 1: Surfaces of Cells Identified by Stardist

Segmenting cells in 3D images provides a much greater challenge compared to 2D segmentation as each 3D image can contain hundreds of 2D cross sections often referred to as slices. Each 2D cross section of 3D microscopy images are then often more difficult to manually segment than normal 2D images as the exact boundaries of cells in 3D images are not always well defined. As a result, if attempting to manually segment cells in 3D images, it is often necessary to look at adjacent slices in the 3D image stack in order to better visualize the boundaries of a cell. As shown above, this process can be made harder still when cells are clumped together. In a 2D view (A), these clumps of cells may look like one large object, and it is necessary to gather data from the adjacent slices in order to segment the individual cells as when viewed in 3D (B), where it is easier to see that these "clumps" are made up of a few individual cells.

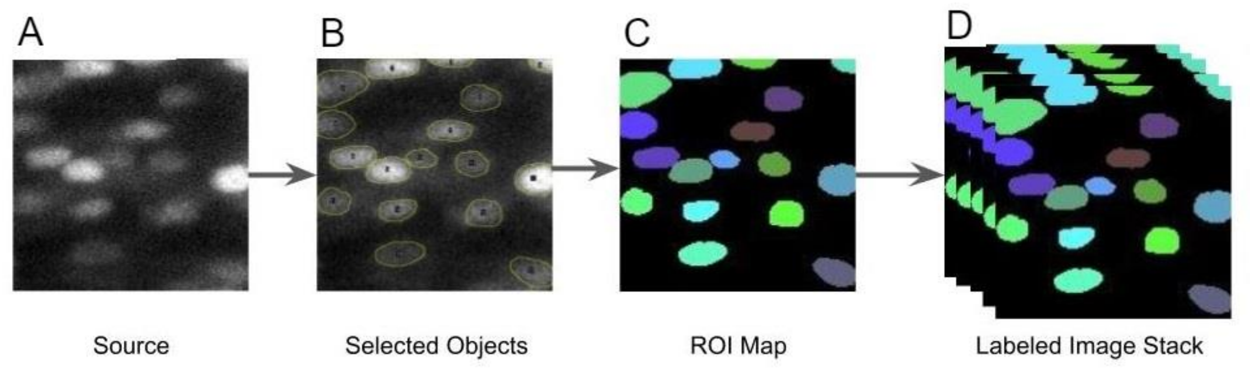


Figure 2: Creating Labeled Image Stacks for a Stardist Training Set

To create an element of a training set, a small, cropped section of one slice was taken from an image in the dataset (A). The selection tool was then used to add all objects in the slice to the ROI manager (B). This was followed by using the LOCI plugin to create a region of interest map of all the selections (C). The same process was then repeated for 15 neighboring slices which were then concatenated (D) and paired with their original source image.

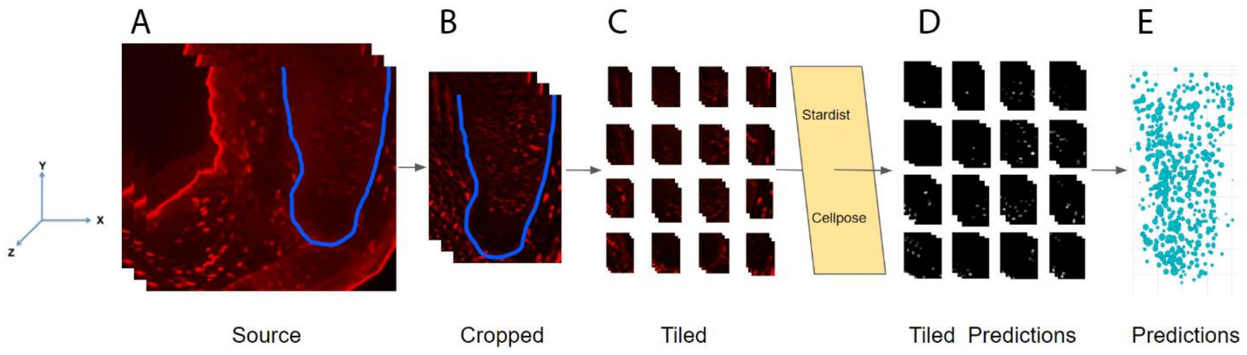


Figure 3: Image Analysis Pipeline

To process an image (A), a box around the area of interest was cropped out and background subtract was applied to all slices in Fiji (B). The cropped image was then split into 16 or 25 equally sized tiles based on the image size (C). The tiles were then processed using Stardist in Google Colab or Cellpose to obtain ROI maps of the objects detected by the algorithm's model (D). Fiji was used to collect data on the size and locations of each detected object and MATLAB was used to reassemble all the tiles in a plot to view the detected objects in 3D as well as compute the total number of objects (E).

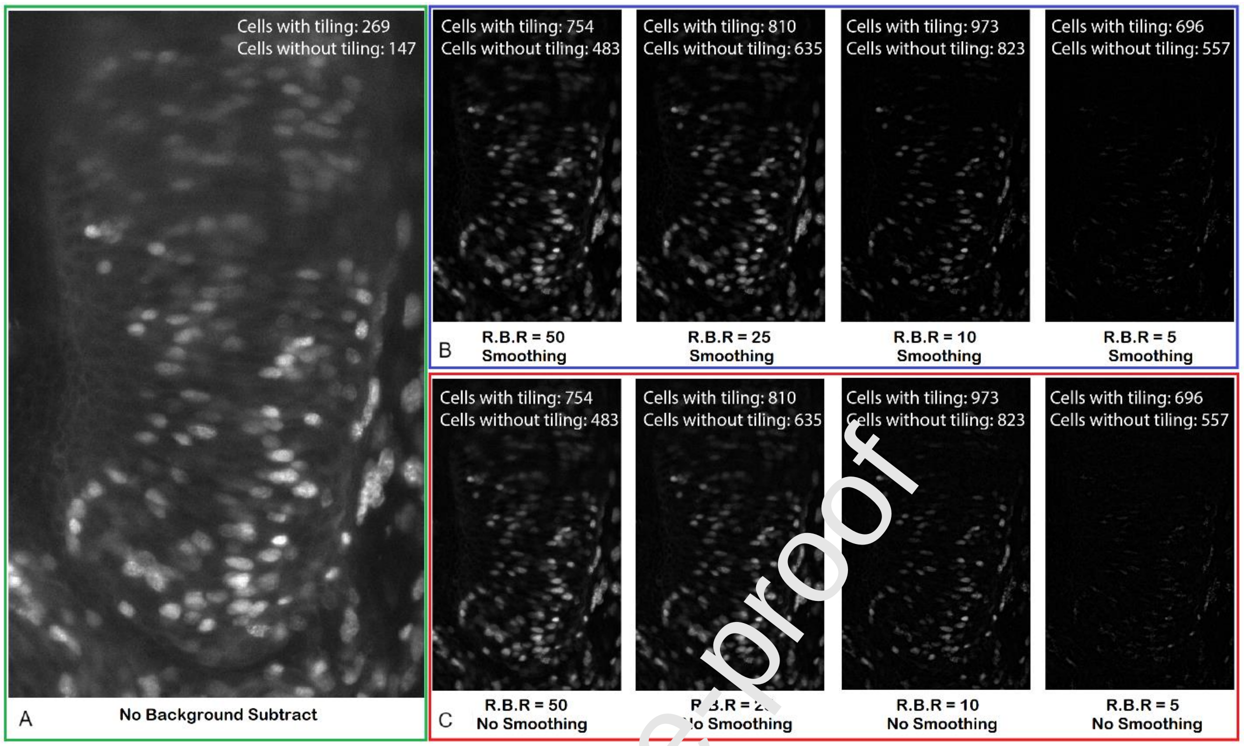


Figure 4: Effect of Background Subtract, Rolling Ball Radius, and Smoothing

The original image (A) was processed in Cellpose with background subtract and smoothing (B) as well as with background subtract with no smoothing (C). Background subtract enhances the contrast between objects of interest and the background by finding the average pixel intensity around each pixel in a circle (the size of which is the rolling ball radius [R.B.R.]) and subtracting it from the pixel's intensity (A to B, A to C). The rolling ball radius should be set to at least the size of the largest non-background object in the image to avoid mistakenly subtracting objects of interest from the image (B, C). Smoothing is another parameter, which reduces noise by averaging pixel intensities in sets of 3x3 which improves the background subtraction. It should be disabled with very small changes on intensity to prevent image data from being subtracted below the background. The low cell count on the original image shows the need for background subtract when predicting on high intensity images with low contrast.

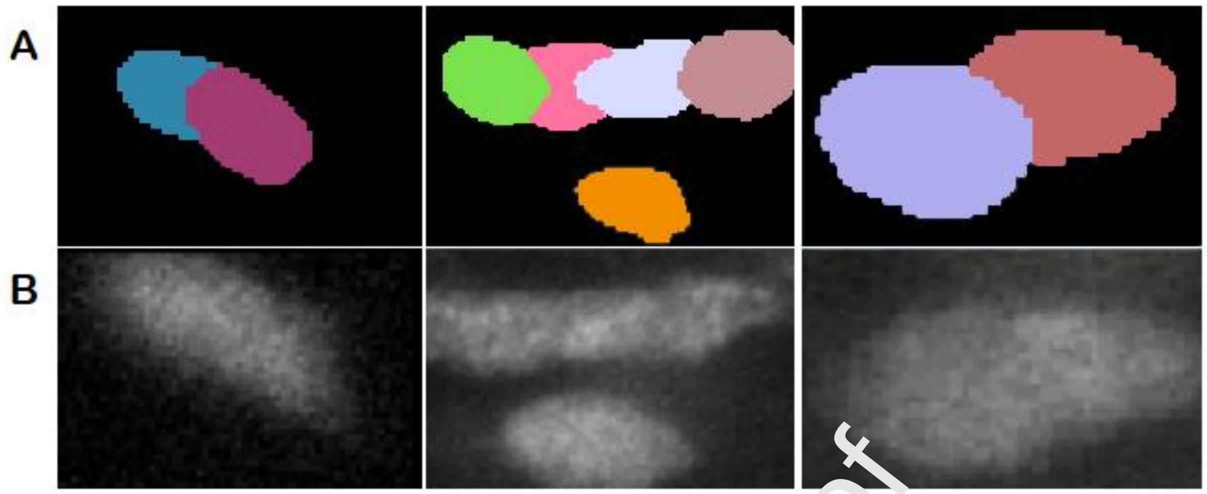


Figure 5: Stardist Predictions on Overlapped Cells

Shown is one slice of three different Stardist predictions (A) and their corresponding original images (B). Based on our results, Stardist excels at identifying overlapped cells and creating appropriate predictions in the form of an ROI map. This is especially important as in a 2D cross section of a 3D image, overlapped cells can appear as just one large object (B). By using data from adjacent slices, Stardist is able to determine which sections of the "large object" belong to each cell.

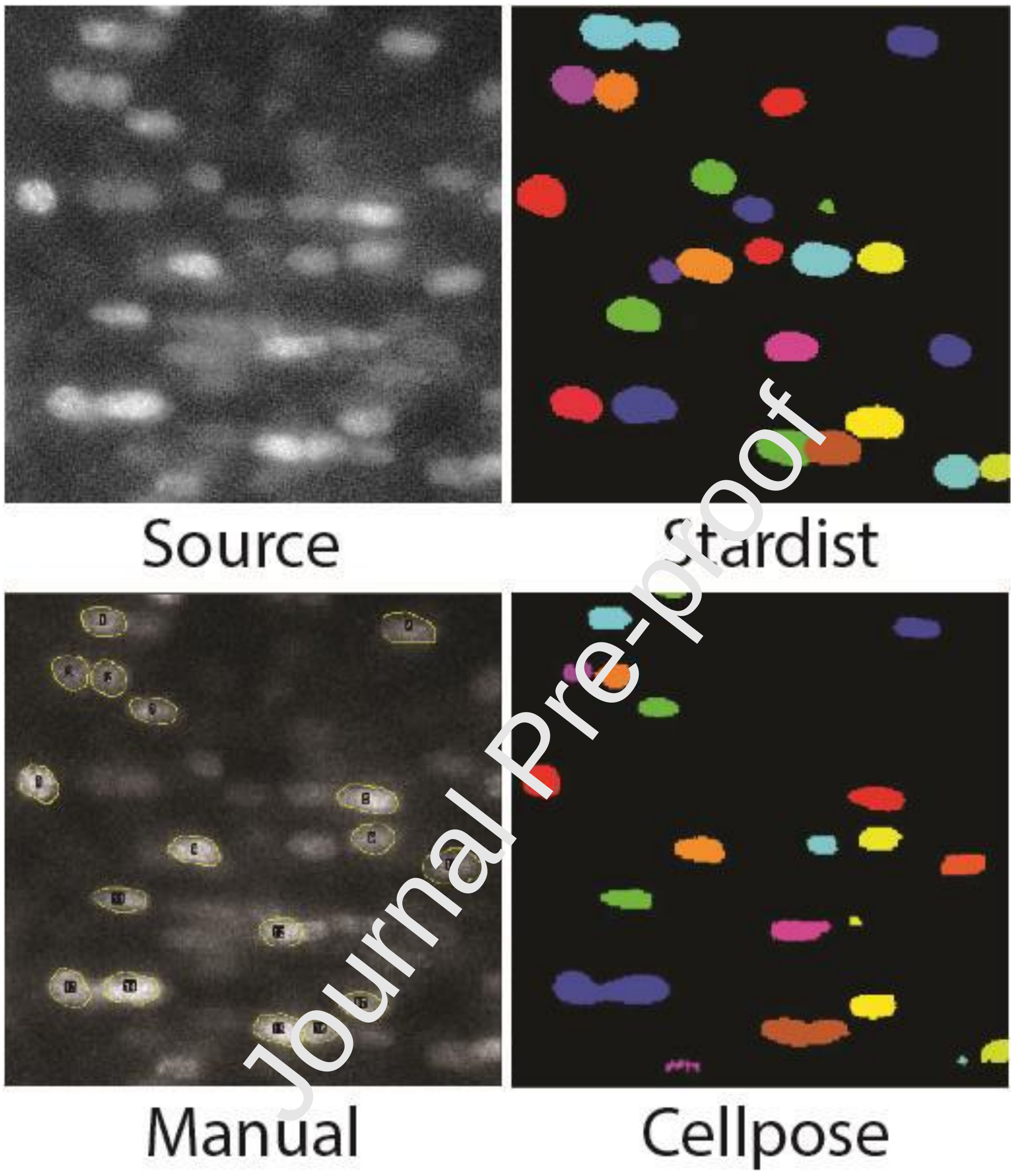


Figure 6: Direct Comparison of Stardist, Cellpose, and Manual Segmentations

Shown is the Cellpose and Stardist segmentations as well as one slice of the source image for a 256-pixel by 256-pixel subsection of the initial dataset. A manual segmentation is also shown. For the whole subsection, manual segmentation found 30 cells while Cellpose found 20 cells and Stardist found 24 cells.

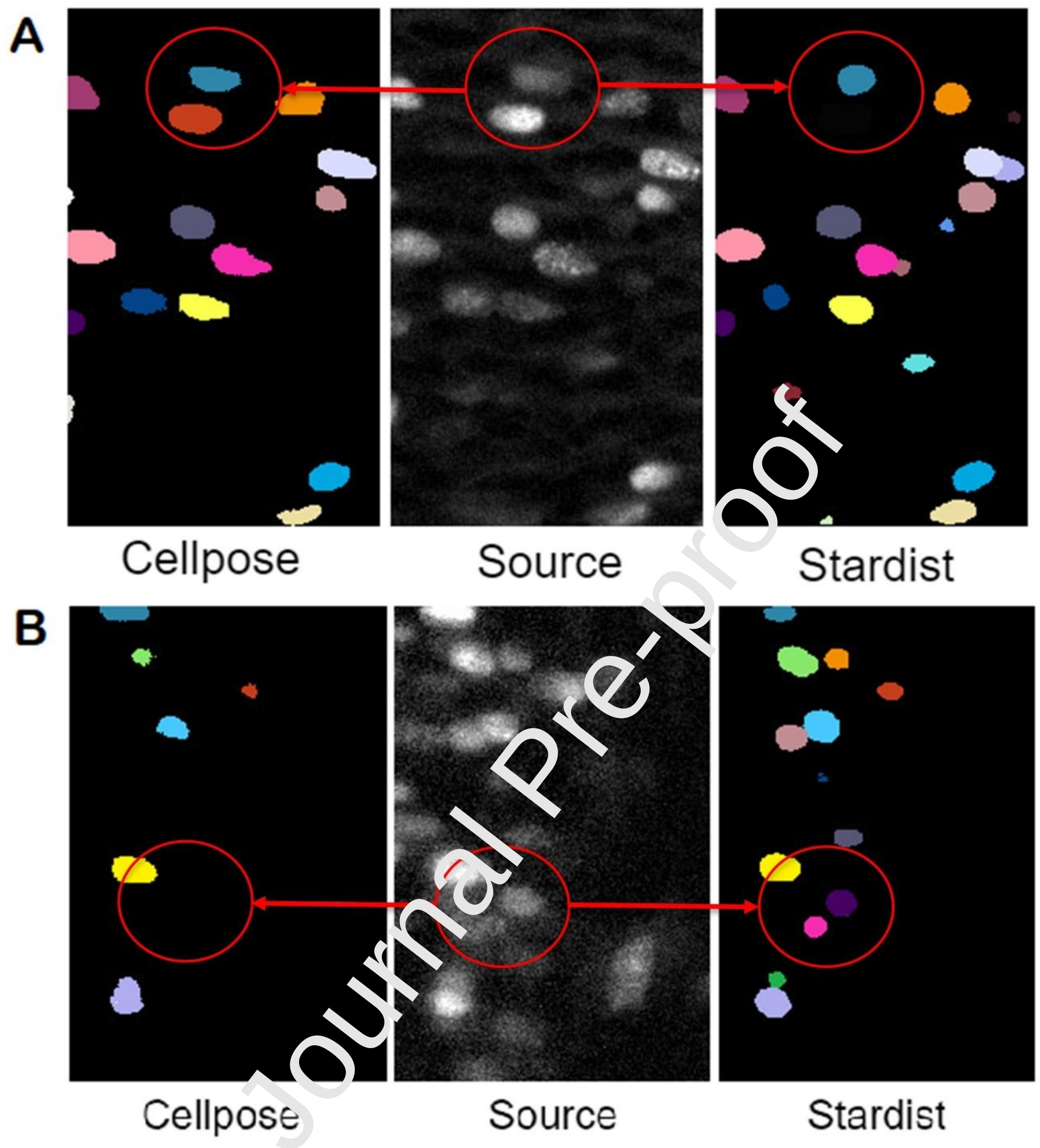
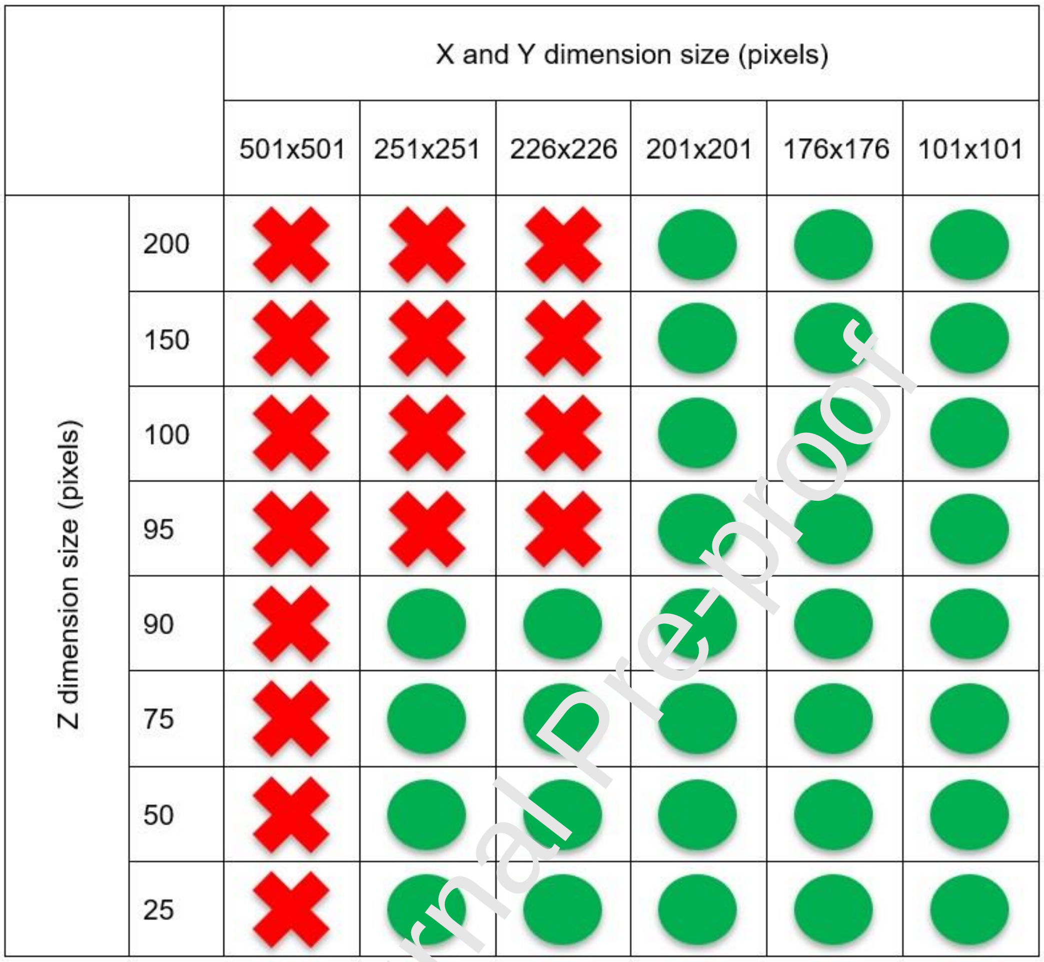


Figure 7: Cellpose and Stardist Predictions on High Contrast or Blurry Images

When images contain high contrast between objects of interest and the background, largely spaced-out objects, or objects with uniform shape and intensity, Cellpose will generate predictions that will segment objects consistently and correctly. For the source image (A, middle), the cells of interest overlap very little and are clearly visible against the dark background. As a result, the Cellpose predictions (A, left) are sufficient and training a Stardist model to create predictions (A, right) is not necessary and may even lead to less accuracy compared to Cellpose's widely pretrained model. In these cases, training a Stardist model from the ground up is not necessary and may actually miss more cells than Cellpose as neither model can be completely accurate. Cellpose's correct segmentation of objects decreases in consistency when images contain high amounts of noise, clustered cells, and unusual intensities. The source image shown (B, middle) is slightly blurry with areas of low contrast. As a result, the Cellpose predictions (B, left) which are found using a pre-trained algorithm are incomplete, missing a large amount of the cells in the source image. Due to being trained on a dataset containing similar blur and noise, the specialized Stardist predictions (B, right) find more of the cells. Due to the poor quality of the source image and the current progress in deep learning, neither algorithm will generate perfect predictions, but Stardist is preferred since it best mitigates the errors between the two algorithms

Appendix



		X and Y dimension size (pixels)					
		501x501	251x251	226x226	201x201	176x176	101x101
Z dimension size (pixels)	200	❌	❌	❌	🟢	🟢	🟢
	150	❌	❌	❌	🟢	🟢	🟢
	100	❌	❌	❌	🟢	🟢	🟢
	95	❌	❌	❌	🟢	🟢	🟢
	90	❌	🟢	🟢	🟢	🟢	🟢
	75	❌	🟢	🟢	🟢	🟢	🟢
	50	❌	🟢	🟢	🟢	🟢	🟢
	25	❌	🟢	🟢	🟢	🟢	🟢

Figure A.1: Testing Stardist Processing Image Size

Images were processed by Stardist at varying planar dimensions and varying depths in order to find the largest image size that Stardist could successfully (green) generate predictions for without incurring an error (red). Increments were determined by starting at very high and low dimensions and gradually decreasing the increment size while homing in on the critical image dimensions. Systematic narrowing of the image dimensions is a practical way to accomplish this however due to Stardist's quick image processing time, the process is rather short. Minimizing edges in the z dimension was prioritized.

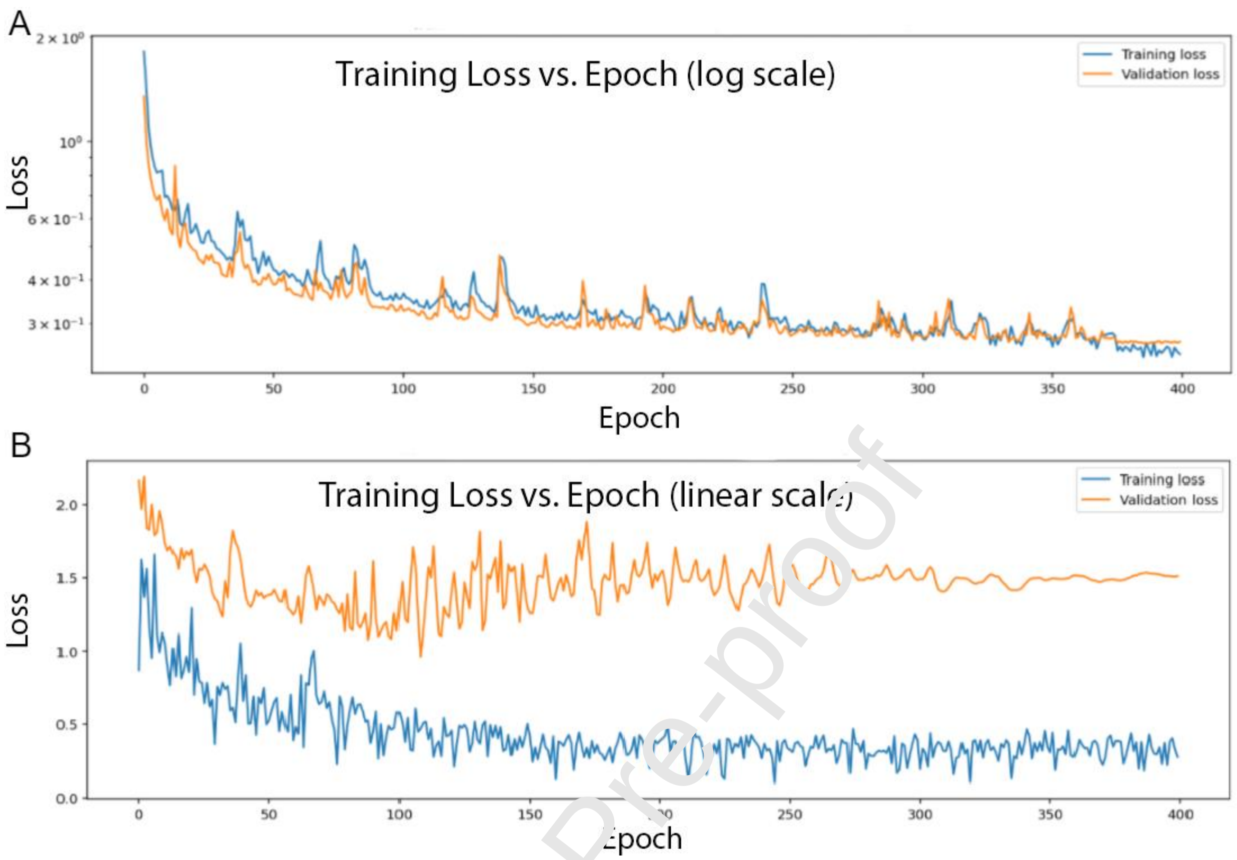


Figure A.2: Training Loss vs. Epoch Number in Stardist Training

Training loss (shown in blue) is a measure of the difference between a model's current prediction of a source image and the correct prediction based on labeled data within the training set. A smaller loss value indicates a more accurate prediction. Validation loss (shown in orange) is a similar measure but instead is based on the model's prediction accuracy on an image that is not within the training set. Low validation loss indicates a model that is more likely to succeed in processing images. A successfully trained model should have a training curve with a shape resembling exponential decay (A). Because of this, additional training has diminishing returns and may be unnecessary. If training loss is lower than validation loss by a large amount, the model is likely overtrained and needs a larger, more varied training set (B).

CRediTAuthorStatement
Giona Kleinberg: Data Curation, Formal Analysis, Investigation, Methodology, Software, Writing - Original draft preparation, Writing – Review & Editing.: **Sophia Wang:** Formal Analysis, Investigation, Methodology, Writing - Original draft preparation, Writing – Review & Editing.: **Ester Comellas:** Software, Data Curation, Formal Analysis, Funding Acquisition, Writing – Review & Editing.: **James R. Monaghan:** Conceptualization, Funding Acquisition, Writing – Review & Editing.: **Sandra J. Shefelbine:** Conceptualization, Validation, Resources, Data Curation, Supervision, Funding Acquisition, Writing – Review & Editing

Highlights

- Image analysis pipelines were used for cell quantification in 3D light-sheet images.
- Stardist outperforms Cellpose on large datasets that have high amounts of objects.
- Stardist outperforms Cellpose given a low signal-to-background ratio of intensity.
- Tiling of images results in more effective background subtract when pre-processing.