jove

Automated Analysis of *C. elegans* Fluorescence Images using SegElegans

Konstantinos Kounakis[1,2], **Pablo E. Layana Castro**[3], **Antonio Garcia Garvi**[3], **Antonio-José Sánchez-Salmerón**[3], **Nektarios Tavernarakis**[1,2]

[1] Department of Basic Sciences, Faculty of Medicine, University of Crete [2] Institute of Molecular Biology and Biotechnology, Foundation for Research and Technology-Hellas [3] Instituto de Automática e Informática Industrial, Universitat Politècnica de València

Corresponding Author

Nektarios Tavernarakis
tavernarakis@imbb.forth.gr

Citation

Kounakis, K., Layana Castro, P.E., Garvi, A.G., Sánchez-Salmerón, A.J., Tavernarakis, N. Automated Analysis of *C. elegans* Fluorescence Images using SegElegans. *J. Vis. Exp.* (224), e69094, doi:10.3791/69094 (2025).

Date Published

October 10, 2025

DOI

10.3791/69094

URL

jove.com/video/69094

Abstract

Microscopy, particularly of the fluorescent kind, is a frequently used tool in *C. elegans* research. The analysis of data from microscopy experiments can, however, be quite tedious and time-consuming. Thus, automation is desirable. We developed SegElegans, a two-headed U-net-based convolutional neural network system that is specifically designed for the automated segmentation of worms, even in images with large numbers of touching or overlapping individuals. The first part of SegElegans consists of one encoder and two decoders. The encoder, based on the SmaAt AT model, applies double convolution layers followed by a Convolutional Block Attention Module (CBAM). Both decoders use convolutional LSTMs: one performs semantic segmentation of worm images (body, edge, background, or overlap), while the other extracts a linear skeleton along each worm. The second part is a post-processing algorithm that combines the outputs of the two decoders and uses them to generate accurate instance segmentations. These segmentations can then be fed to ImageJ or other appropriate image analysis tools. Here we present instructions on how to access and run this system. We provide an online, cloud computing-based implementation as well as two methods to use the SegElegans models offline, on a local machine, should the required hardware be available.

Introduction

Caenorhabditis elegans is a eukaryotic model organism with wide use in the fields of cellular and molecular biology[1,2,3]. It is a small (~1 mm for wild type adults) soil-living nematode worm with a fast life cycle (~three weeks), and a large number of progeny (typically genetically identical to the self-fertilizing hermaphrodite parent). Despite its relative simplicity, the animal is still a complex multicellular organism with defined and distinct tissues and organs, which, thanks to its small size and transparency, can be studied *in vivo* without any need for fixing or other invasive treatments[4]. The capacity of the worm for *in vivo* study through microscopy can be further amplified through the use of fluorescent reporters,

that permit the easy distinguishment of individual cells or even the direct visualization of sub-cellular organelles/components such as mitochondria, lipid droplets, or protein aggregates, and of processes such as signal transduction, gene expression, vesicle fusion, autophagy, neuronal and muscle action potentials, etc.[5,6,7,8,9,10,11]. In addition, the nematode is also highly amenable to genetic manipulation through mutagenesis[12], transgenesis via microinjection or microparticle bombardment[13], CRISPR editing[14], and, most importantly, when it comes to easy and cost-effective genetic screening, RNAi[15]. This manipulation permits the extensive study of genes and their role in the nematode itself, while simultaneously, thanks to the significant genetic homology that *C. elegans* exhibits to other model organisms, making foundational discoveries with relevance all the way to humans[16,17]. Finally, this homology/conservation also makes the worm an ideal initial testing ground for drugs and chemical agents, allowing the elucidation of the mechanisms of drug activity, the identification of potential activity-modifying genetic variations, and the discovery of potential unwanted interactions and off-target effects[18,19,20].

All the above-mentioned advantages have rendered *C. elegans* an attractive model for cellular and molecular biology studies, particularly for scientists interested in the *in vivo* and often real-time monitoring of processes via microscopy. Such studies typically involve a significant amount of image analysis through special software, with the most widely used option being ImageJ[21,22,23]. A common characteristic of the analysis through such software is the need for the user/researcher to specify Regions of Interest (ROIs) for analysis through the use of selection tools. Quite frequently, each individual worm in an image will be selected as its own ROI, with the aim to acquire per-worm information about simple morphological features (such as body dimensions) or more "advanced" experimental readouts such as the expression levels of a gene (measured with a fluorescent reporter), the readout of a ratiometric reporter, the number and size of lipid droplets, the formation of protein aggregates, the morphology of individual organelles and their networks etc.[5,8,10,24,25,26,27]. These per-worm selections are typically made manually, since automatic selection methods that are based on standard image processing algorithms are usually not capable enough to distinguish the shapes and features of the worm, especially in darkfield images where the outlines of the worms are not clearly visible or in brightfield images with significant numbers of animals that touch and overlap. The manual process is, however, slow (20 s to 1 min per worm depending on the precision needed and the experience of the user), laborious, and subject to user bias and error.

An alternative and much more powerful approach for the generation of individual *C. elegans* ROIs (a task typically referred to as segmentation in the field of computer vision) is to automate it with the help of deep learning/neural network techniques. Convolutional neural networks based primarily on the Mask R-CNN[28] and the U-net architectures[29] have produced decent results in segmentation tasks on various biological model systems[30,31,32] including in *C. elegans*[33,34,35,36,37,38,39], but none have provided a satisfactory solution to the problem of generating full body segmentations that correctly distinguish individual animals (instance segmentation) in high resolution images with large numbers of touching or outright overlapping worms. In order to meet this need, we developed SegElegans, a deep learning model specifically designed and optimized for this task[40].

SegElegans is comprised of two major parts (**Figure 1**). The first part is a two-headed U-net variant convolutional

neural network. It is composed of one encoder block and two decoder blocks. The encoder block is based on the SmaAt AT model[41] and uses double convolution blocks for each layer feeding into a Convolutional Block Attention Module (CBAM)[42]. The two decoder blocks are based on the convolutional long short-term memory (LSTM) network[43]. One decoder is responsible for categorizing each pixel of the images as a part of the main body of a worm, a part of the edge of a worm, a part of the background, or a part of an area where worms overlap (this is typically referred to as semantic segmentation). The other decoder is responsible for drawing a linear "skeleton" along the length of each worm. The second part is a post-processing algorithm that combines the outputs of the two decoders and uses them to generate accurate instance segmentations. It initially identifies the segments of true overlaps by comparing the semantic segmentation output to the skeleton one. Then it outputs the instance segmentation directly for worms without overlaps or by assembling it from segments for overlapping ones[40]. These segmentations are saved as binary masks as well as ImageJ compatible ROIs.

Protocol

The following sections include detailed instructions on how to make effective use of SegElegans (**Figure 2**). They include instructions on preparatory image acquisition (Section 1), how to run the model online (Section 2) or offline (Sections 3-5), and the import of ROIs to ImageJ or other tools (Section 6). Since the evaluation of images through the model requires a PC with a CUDA compatible Graphics Processing Unit (GPU) that has at least 6 GB (ideally more) of video random access memory (VRAM), most users (provided they have access to Google's services) are advised to utilize the online version of the model which satisfies and surpasses these hardware requirements through cloud computing (Section 1 > Section 2 > Section 6). Alternatively, users who have access to appropriately powerful hardware and a basic understanding of command line/terminal use (or users who lack access to the aforementioned cloud computing options) may find running the model locally more convenient (Section 1 > Section 3 > Section 4 or 5 > Section 6).

1. Acquisition of sample and guide images

1. Use SegElegans to acquire segmentations of adult worms from any worm strain, regardless of what fluorescent reporters they may or may not express. If the phenotype(s)/data of interest are measured in brightfield images, acquire them normally using a widefield microscope and compatible software, using a 4x or similar objective lens.

   NOTE: SegElegans can analyze images with a wide range of brightness and contrast, although it is preferable that there are no saturated bright or dark pixels. In regard to dimensions, the system has been trained with 1328 x 1048 images but should work with any image that is at least 512x512 in size.

2. If the phenotype(s)/data of interest are measured in darkfield images (typical when using one or more transgenic fluorescent reporters or dyes), acquire brightfield images alongside the data to be used as guide images (as explained in step 1.1). Ensure the guide images perfectly match the darkfield images of the fluorescence channel(s) spatially and as close as possible timewise, ideally by using the multichannel acquisition options that are provided by the microscope's operating software. Save the guide images in a separate folder from the corresponding darkfield data images, using the same name for both.

2. Running the online version of SegElegans

1. Log in to a Google account on a web browser. Creating a fresh dedicated lab account for this purpose is recommended.
2. Enter Google Drive (https://drive.google.com) and upload the folder with the brightfield images (data or guide). As mentioned before, if guide images are used to generate ROIs for darkfield images, do not include said darkfield data images in this folder.
3. Go to https://github.com/KonstantinosKounakis/SegEleganSOnline/tree/v1.0 and click on the **SegElegans Body Prediction Interface.ipynb** file. Click on the **Open in Colab** button at the top of the file that opens. The Jupyter notebook/interface for SegElegans will open on Google Colab.
4. Execute code block 1 by pressing the play (▶) button under the title. Grant the runtime permission to run the code and observe the output after execution is over (a green checkmark appears next to the play button). Colab should automatically run the correct, CUDA-compatible runtime and display an NVIDIA-SMI output table. Otherwise, force the use of a T4 GPU runtime through the **Runtime > Change runtime type** menu.
5. Execute code block 2 to load the contents of Google Drive into the runtime. Accept all confirmation dialogues and grant all requested permissions.
6. Execute code blocks 3 then 4. Block 3 will take a couple of minutes to finish, so ensure the green checkmark has appeared before proceeding to 4.
7. Open the icon with the folder tab to the left of the Colab interface. This will display the files that are loaded into the runtime, including the imported contents of Drive as **/content/drive/MyDrive**. Right-click on the folder with the images, copy the path, and paste it into the **test_images** input form of code block 5. Ensure it looks like this: **/content/drive/MyDrive/Imagefolder**. Similarly, specify a path for the analysis output (e.g., **/content/drive/MyDrive/Imagefolder/Output**). Note that paths in this environment use a forward slash to separate folders (**/**).
8. Execute code block 5.
9. Specify the exact extension of the images to be analyzed by the model in the provided form of code block 6. For the purpose of this input ".tif", ".TIF", and ".tiff" are treated as different extensions.
10. Do not adjust the **batch_crop_img** input in code block 6 if the system is running in the default T4 GPU runtime (which is provided for free to all users for a limited number of hours per day, depending on usage levels). Reduce it if the runtime raises memory availability issues.
11. Execute code block 6. This will take some time, so ensure the green checkmark has appeared before proceeding.
12. Execute code block 7. Do not adjust its inputs for the default T4 runtime, but reduce the number of subprocesses or outright disable parallel processing if memory availability issues emerge. This will also take significant time.
13. Access Google Drive in a different browser window or tab and navigate to the output folder designated in 2.7.

    NOTE: At this point, SegElegans has concluded the initial evaluation and post processing of segmentations and has created several subfolders with outputs. **0_summary results** contain graphs summarizing the output for each image, with an index number assigned to each worm. **1_complete_mask** contains the curated binary masks for all segmentations the algorithm has

decided are good for analysis. **1_edge_small_mask** contains binary masks for all segmentations the algorithm has decided to reject because the animals are too small or partially obscured at the edge of the image. **1_overlap_mask** contains binary masks for all segmentations of worms that exhibited real overlaps, and are, by default, not added to the curated output. **1_all_rois_results** contains the segmentations of ALL 3 types, regardless of curation, in the ImageJ format (zip files with ROIs).

14. At this point, choose one of the three options:
    1. Use the ROIs provided in **1_all_rois_results** in ImageJ and reject unwanted ROIs after the import there (see protocol section 6).
    2. Accept the curated good masks without any manual correction (and without the inclusion of overlapping worms) (skip to 2.16).
    3. Manually adjust the curation by selecting the worms (including the overlapping ones) to be included in the output (continue to 2.15).
15. In order to manually adjust the curation, use code block 8. Examine the results of the initial curation from the summary graphs in the **0_summary results** folder. For each image that needs correction, input the full name (with extension) of the original input image in the "**name_image_change**" form and the numbers of the masks to be kept (from the summary graph), separated by commas in the "**index_images**" form. Execute code block 8. Repeat this step for any other image that needs correction.
16. After correction or if users choose to accept the initial curation without corrections (and without the overlapping worms), execute code block 9. A new subfolder will be created in the outputs folder in Google Drive called **2_curated_rois_results**. It includes all the final curated segmentations in the ImageJ format (zip files with ROIs).
17. In order to execute the process again for a different folder of images, reset the runtime through the menu **Runtime > Restart Session** and start from the beginning.

3. Preparing the offline version of SegElegans before the first use

1. As an alternative to running SegElegans online on Colab, an offline version of SegElegans is available for users who cannot or prefer not to use Google's cloud computing services. This requires a PC (Windows or Linux) with a CUDA-compatible GPU (https://developer.nvidia.com/cuda-gpus) that has at least 6 GB of VRAM (although 16 is recommended if available and affordable).

   NOTE: The following instructions in sections 3-5 are focused on Windows, but the software can also run on Linux-based systems with some small changes in the commands used.
2. Download the CUDA toolkit installer from https://developer.nvidia.com/cuda-downloads following the instructions to get the version that is appropriate for the device used.

   NOTE: SegElegans has been tested with versions 12.9 and 13.0 of the toolkit and CUDA Version 11 in Windows, however future versions are expected to retain backwards compatibility.
    1. Install the toolkit, following the instructions of the installer itself. Visual Studio is not needed.
3. Download the appropriate version of the Python installer for the device from https://www.python.org/downloads/.

Install Python, following the instructions of the installer itself, and ensuring python.exe is added to PATH.

NOTE: SegElegans was built and tested on Python 3.13, but future versions are expected to retain backwards compatibility.

4. Go to https://github.com/KonstantinosKounakis/SegEleganSOffline/releases/tag/v1 . Click on the **Source code (zip)** link to download the package with the necessary files to set up and run SegElegans locally.

5. Extract the zip file. Place it in a high or even top-level folder, such as C:\SegElegans. The contents of the zip file already come inside a subfolder **SegEleganSOffline-1**.

6. Open the Windows command prompt (cmd.exe). Type the command **python -m venv "fullsegeleganspath"** where "fullsegeleganspath" is the full path, in quotes, into the **SegEleganSOffline-1** subfolder.

   NOTE: For example, if the zip file was extracted in C:\SegElegans, the full path is C:\SegElegans \SegEleganSOffline-1 and the command will be **python -m venv "C:\SegElegans\SegEleganSOffline-1"** . Note that paths in this environment use a backslash to separate folders ("\").

7. In the command prompt, use standard controls to navigate inside the SegEleganSOffline-1 folder and execute the command **Scripts\activate**. A (SegEleganSOffline-1) label should appear before the next prompt cursor, indicating the Python environment is now active.

8. While inside the active (SegEleganSOffline-1) environment, run the command **pip install -r requirements.txt** . The process will take some time.

9. Input and execute the command **python assemblenetworks.py** .

   NOTE: SegElegans is now ready to run. Two ways to achieve this are provided: a quick script that will automatically provide all segmentations as ImageJ ROIs without the option to manually correct the curation (section 4), and a local Jupyter notebook that will provide the option to overview and correct the curation (section 5). The latter approach is recommended for most users.

4. Running the offline version of SegElegans with a quick script

1. Open the Windows command prompt (cmd.exe). As described in step 3.7, navigate to the SegElegans environment folder and activate it.

2. In order to use the quick script, while in the command prompt with the active (SegEleganSOffline-1) environment input and execute the command **python SegEleganSBodyQuickEval.py -i "pathtoinputfolder" -x <fileextension>** where "pathtoinputfolder" is the full path, in quotes, to the folder with the brightfield images (e.g. "D:\Data\Experiment12365\Condition 1") and <fileextension> is the precise file extension of the images users wish to analyze (Example ".TIF"). Type the command **python SegEleganSBodyQuickEval.py -h** for instructions on additional parameters that can be changed to accelerate the processing by the script if the device can handle it.

   NOTE: The output folder in this case will be created inside the input folder (So for the above example: "D:\Data\Experiment12365\Condition 1\Output"). Inside, there is a series of subfolders. **0_summary results** contain graphs summarizing the output for each image, with an index number assigned to each worm.

1_complete_mask contains the curated binary masks for all non-overlapping segmentations the algorithm has decided are good for analysis. **1_edge_small_mask** contains binary masks for all segmentations the algorithm has decided to reject because the animals are too small or partially obscured at the edge of the image. **1_overlap_mask** contains binary masks for all segmentations of worms that exhibited real overlaps. **1_all_rois_results** contains the segmentations of ALL 3 types, regardless of curation, in the ImageJ format (zip files with ROIs). Since there is no way to manually correct curation here, **2_curated_rois_results** includes by default all segmentations deemed good by the algorithm AND all overlapping segmentations in the ImageJ format (zip files with ROIs).

5. Running the offline version of SegElegans with a Jupyter notebook

1. Open the Windows command prompt (cmd.exe). As described in step 3.7, navigate to the SegElegans environment folder and activate it.
2. In order to use the Jupyter interface, while in the command prompt with the active (SegEleganSOffline-1) environment, input and execute the command **jupyter notebook**. This will open a web browser window/tab with a locally hosted site. In the initial screen, select the file **SegEleganSOfflineJupyterInterface.ipynb**.
3. Execute the code in code block 1 by selecting it (click on the left outside of the actual code) and pressing the play (▶) button on the toolbar above. A confirmation message will appear below the code block when the execution is complete.
4. Execute code block 2. This will generate some input forms in the readout underneath the code block. Fill in the generated forms with the path to the input folder with the brightfield images (Example: "D:\Data\Experiment12365\Condition 1"), the desired path for the analysis output (Example: "D:\Data \Experiment12365\Condition 1\Output"), and the precise file extension of the images to be analyzed (Example ".TIF"). Adjust the additional provided settings (increase the number of subcrops per batch to 9 or 16, and run multiple post-processing parallel processes) if the device is sufficiently capable.
5. Execute code block 3.
6. Execute code block 4. This will take some time, so ensure it is complete before proceeding.
7. Execute code block 5. Again, this will take some time. After this, navigate to the output folder designated above. NOTE: At this point, SegElegans has concluded the initial evaluation and post processing of segmentations and has created several subfolders with outputs. **0_summary results** contain graphs summarizing the output for each image, with an index number assigned to each worm. **1_complete_mask** contains the curated binary masks for all segmentations the algorithm has decided are good for analysis. **1_edge_small_mask** contains binary masks for all segmentations the algorithm has decided to reject because the animals are too small or partially obscured at the edge of the image. **1_overlap_mask** contains binary masks for all segmentations of worms that exhibited real overlaps, and are by default not added to the curated output. **1_all_rois_results** contains the segmentations of ALL 3 types, regardless of curation, in the ImageJ format (zip files with ROIs).

8. Similar to the online version of SegElegans, use any of the three options listed below:
    1. Use the ROIs provided in **1_all_rois_results** in ImageJ and reject unwanted ROIs after the import there (see protocol section 6).
    2. Accept the automatically curated good masks without any manual correction (and without the inclusion of overlapping worms) (skip to step 5.11).
    3. Manually adjust the curation by selecting the worms (including the overlapping ones) to be included in the output (continue to step 5.9).
9. In order to manually adjust the curation, use code blocks 6 and 7. First execute 6. This will create input forms in the readout underneath the code.
10. Examine the results of the initial curation from the summary graphs in the **0_summary results** subfolder. For each image that needs correction, input the full name of the INPUT image (with extension) in the **Image to correct**: form and the numbers of the masks to be kept (from the summary graph) separated by commas in the **Masks to keep**: form. Execute code block 7 and ensure execution is complete. Repeat this step for any other image that needs correction.
11. After the curation correction is done or if users choose to accept the initial curation without corrections (and without the overlapping worms), execute code block 8. A new subfolder will be created in the designated outputs folder called **2_curated_rois_results**. It includes all the final curated segmentations in the ImageJ format (zip files with ROIs).
12. In order to execute the process again for a different folder of images, reset the runtime through the menu **Kernel > Restart Kernel and Clear Outputs of all Cells** and start from the beginning.

6. Importing the segmentations to ImageJ (or alternatives)

1. Open one of the actual data images on ImageJ.
2. Open the corresponding zip file with the ROIs of that image. This will load the selections into the ImageJ ROI manager. If they are ROIs from the **2_curated_rois_results** output they are ready for analysis with the desired methods normally used (preferably with the use of macros to further automate and accelerate the process).
3. If they are ROIs from the **1_all_rois_results** output, remove unwanted segmentations from the ROI manager. Select them and press the **Delete** button on the manager window itself. Do not press the **Delete** key on the keyboard, as that deletes the contents of the selection in the image instead.
4. If software other than ImageJ is needed for the analysis, the ROI format will most likely not be compatible. In that case, import segmentations in the universally utilized form of binary masks, which are provided in the **1_complete_mask** and **1_overlap_mask** folders. Consult the instructions of the receiving software on how to perform that import.

Representative Results

By following this protocol, researchers should be able to extract high-quality worm segmentations, even for the analysis of fluorescence images without visible worm borders. As discussed in section 6 of the protocol, these segmentations can be imported directly to ImageJ and used for the quick measurement of relevant properties

such as the intensity of fluorescent reporters (commonly used for expression quantification) or the number, size, and morphology of fluorescently tagged areas (used frequently to study protein aggregates or organelles). Furthermore, since SegElegans is designed to acquire segmentations from brightfield images in a fluorescence agnostic manner, it can assist with multichannel analysis as well, including the measurement of intensity ratios (for instance, to quantify the autophagic degradation of specific organelles) or co-localization metrics (such as when trying to identify interactions between organelles).

SegElegans achieves a segmentation IoU (Intersection over Union) score of over 93% (**Table 1**), surpassing alternatives at the time of publication[40]. In practice, this means that, after running all images through the described protocol, there will be some worms that get segmented incorrectly and, depending on the user's needs, should be ignored from the analysis or re-selected with manually drawn selections if their inclusion is deemed necessary.

Our tests suggest that a highly precise manual selection, such as those used to train SegElegans (following the actual edge of the animals as closely/tightly as possible), can take between 30-60 s per individual worm, depending on the user, leading to an average analysis time of ~245 s per image. SegElegans can produce segmentations of that quality for all worms in an image (even if there are 7+ of them) at under a minute per image, with the time used for manual corrections of the curation included. On average, effective use of SegElegans should cut the time needed for worm segmentation before analysis to a quarter or even a fifth of what is needed manually (**Figure 3**).

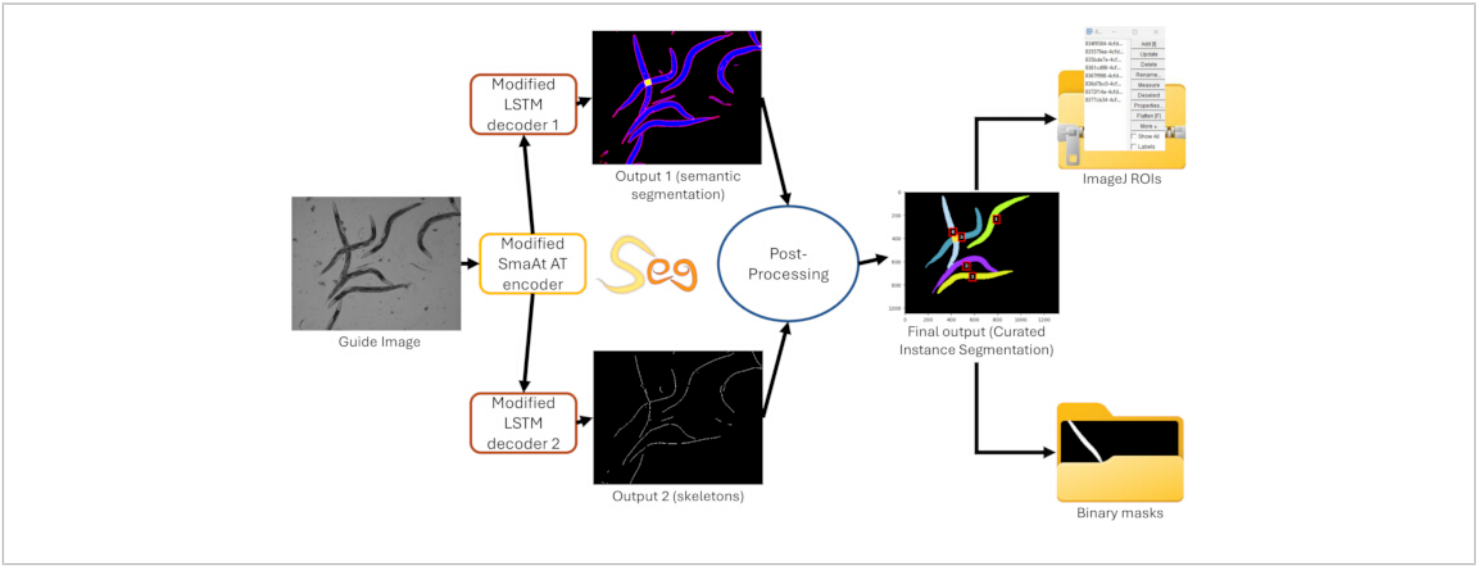

Figure 1: Overview of SegElegans. It is built on a two-headed U-net architecture with 1 encoder and 2 decoders that separately generate a semantic segmentation of the image and segmentations of the worms' skeletons. These two segmentations are fed into a post-processing system and are used to generate the final instance segmentation, which is then output in the form of ImageJ-compatible ROIs and binary masks. Please click here to view a larger version of this figure.

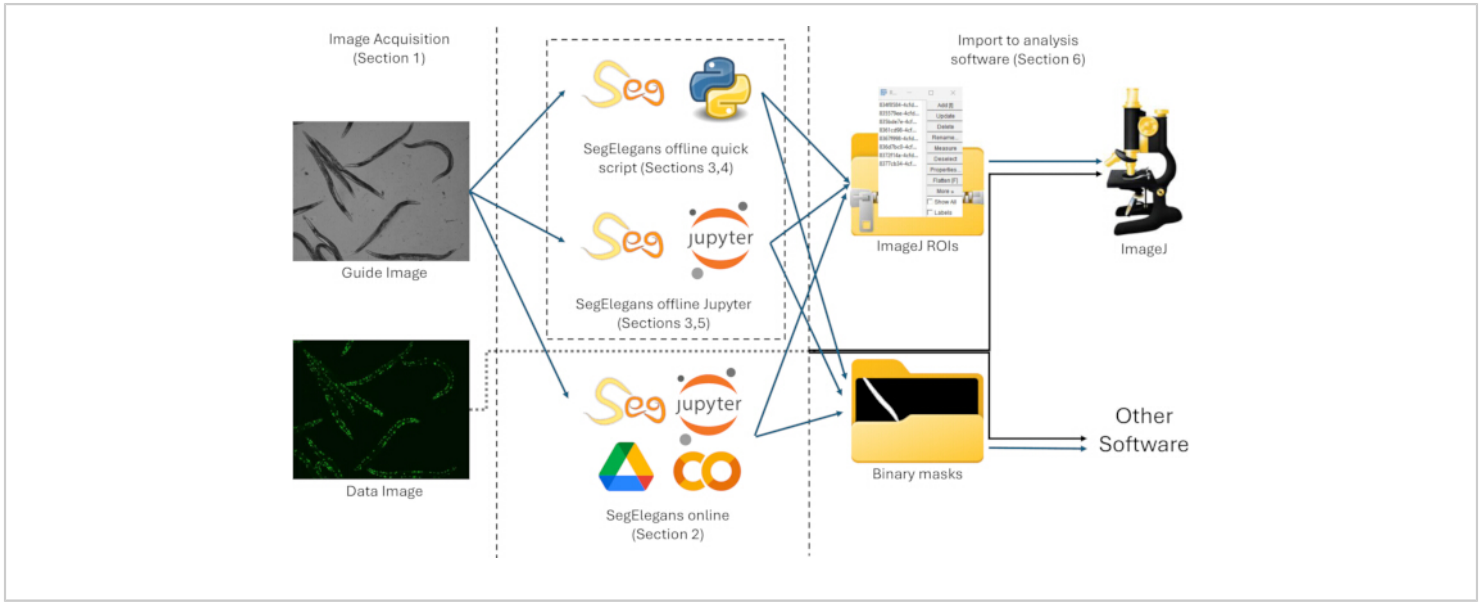

Figure 2: Summary of the protocol and the multiple alternative options provided. After image acquisition (Section 1), users can acquire segmentations from guide images by using SegElegans through an online Jupyter interface (Section 2), an offline Jupyter interface (Sections 3 and 5) or an offline script (Sections 3 and 4). The segmentation output can then be used in ImageJ or other software to analyze the images containing the actual data (Section 6). The worms used in this figure as an example are of the AM141 strain[10]. Please click here to view a larger version of this figure.

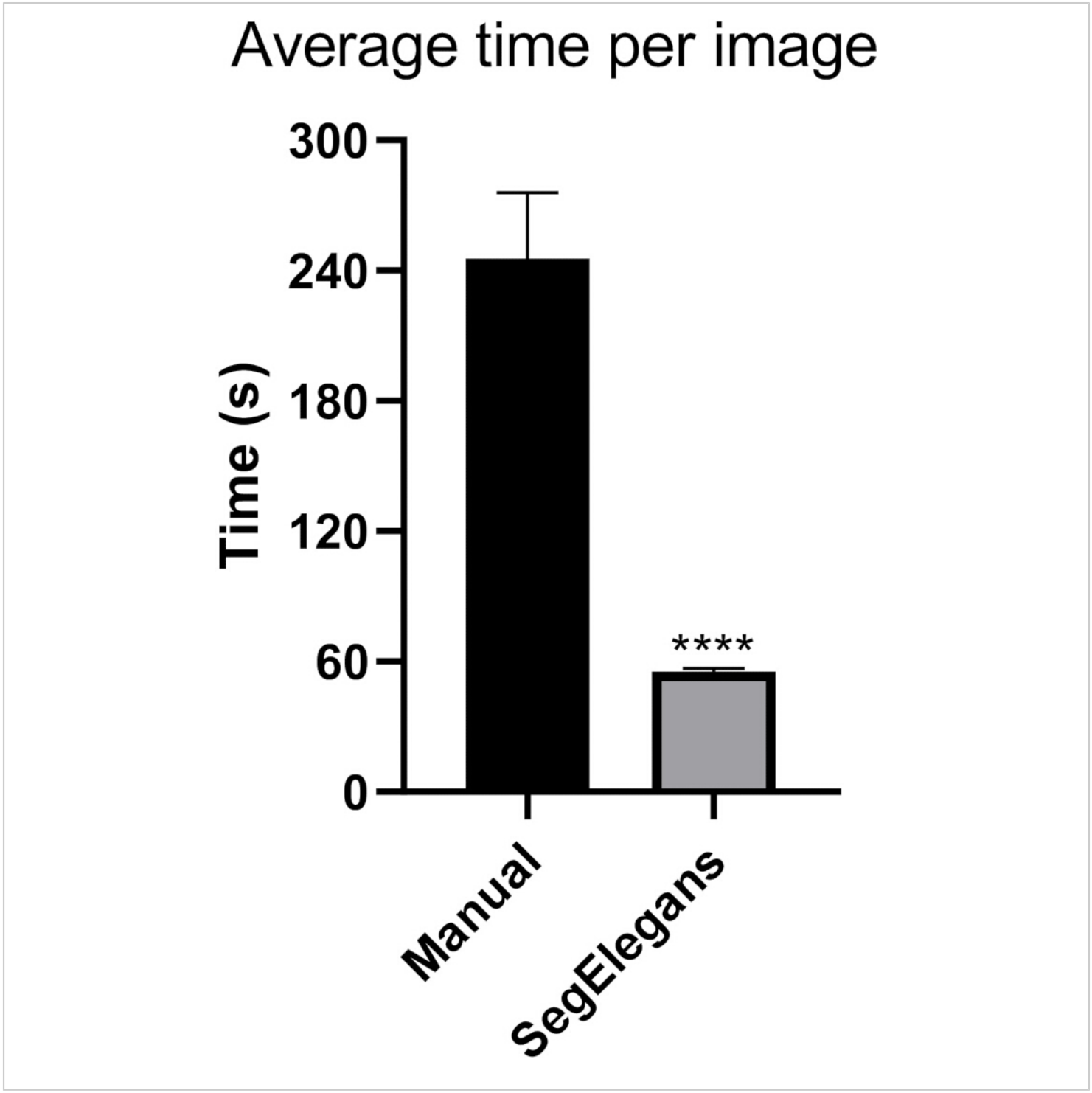

Figure 3: Average segmentation time (measured in seconds) per image. Error bars indicate SEM. **** indicates p-value <0.0001 in Welch's t-test. N = 53 images. Please click here to view a larger version of this figure.

	Avg IoU [0.5 cuttof]		Avg IoU [0.7 cuttof]		Avg IoU [0.9 cuttof]	
Model	Whole Image	Per Worm	Whole Image	Per Worm	Whole Image	Per Worm
EmbedSeg[30]	0.8775	0.9017	0.823	0.8867	0.6891	0.7247
UMF U-Net[32]	0.9266	0.9382	0.9266	0.9088	0.9266	0.9088
SmaAt DS[41]	0.9272	0.9238	0.9272	0.8895	0.9272	0.8895
SmaAt AT[41]	0.9343	0.9498	0.9343	0.926	0.9343	0.926
SegElegans[40]	0.9355	0.9627	0.9355	0.9461	0.9335	0.9461

Table 1: Intersection over union (IoU) score comparisons at different cutoffs for full body segmentations between SegElegans and other published *C. elegans* convolutional neural networks[32,34,40,41].

Discussion

The methodology presented here should allow users to analyze *C. elegans* microscopy experiments in a significantly faster timeframe without any loss in accuracy. Since it extracts segmentations from brightfield guide images independently of the actual fluorescence (as discussed in protocol section 1) it can be used with any strain and for any application that requires measuring phenotypes on a per-worm basis. These can include single-channel applications such as the quantification of the formation of abnormal protein inclusions in disease models[10] (like in the example shown in **Figure 2**), the activity of transcriptional reporters[26], or the size and number of lipid droplets[44]. They can also include multichannel assays, such as the measurement of hydrogen peroxide levels with a ratiometric sensor[27], the assessment of organelle and protein co-localizations[11], the detection of macromolecule modifications[45], and/or quantifications of various types of autophagy[8,46]. The segmentations generated by SegElegans are (as discussed in section 6) provided in both the ImageJ format, permitting quick and easy usage in the "golden standard" ecosystem for biological image analysis[21,22,23], and in the universal binary mask format, permitting import into any more bespoke solution a lab may utilize. Finally, in addition to the fact that at the time of publication SegElegans already inherently achieves the best segmentation quality available on the overlapping full body task compared to alternatives (**Table 1**)[32,34,40,41], the implementations provided in this protocol also allow the user to easily correct some of the possible mistakes in the curation step (as discussed in sections 3,5, and 6), ensuring that all correct segmentations can be included in the output while the bad ones are discarded.

SegElegans has been designed to work for all worms from the late L4 stage and older at a magnification where multiple full bodies can be observed (~4x objective). Animals with dramatically abnormal body phenotypes (such as dumpy or mutlivulva) may, however, not be segmented correctly. In addition, worms that are too small (L3 and younger) should be segmented correctly, but will be rejected during the automatic curation step. Users who wish to work with such animals will need to add them to the output during the curation correction step, or use the outputs from the **1_edge_small_mask** and

1_all_rois_results folders. The system has been trained with images from multiple microscopes and a variety of brightness and contrast settings to ensure maximum compatibility for users with different equipment, but it is preferable for users to utilize their acquisition software's features (such as brightness histograms, which are widely used) to ensure there are at least no saturated dark or bright pixels in their guide images. In all cases, it is important that the images used are in perfect focus. Images that are generally blurry or images that blur due to worm movement will not provide good results. Finally, it is absolutely critical to ensure that the brightfield guide images perfectly match the fluorescence darkfield images spatially and as closely as feasible temporally (especially if the sample worms are not completely immobilized). This is necessary to guarantee that the segmentation generated by the guide image can perfectly apply to the data. Thankfully, most modern microscopy acquisition software provides tools that allow the operator to acquire 2 or more channels almost simultaneously with a single button press, provided that the software has actual mechanical control over the microscope's light sources and filter wheels.

In addition to the obvious advantages SegElegans provides in expediting analysis, we argue that its use can also lead to tangible improvements in the overall quality of research, as time is often a limiting factor that constrains what can be done and can force a scientist to make concessions in their experimental designs. Fast automation tools can allow researchers to conduct more assays with more experimental conditions and with larger sample sizes, leading to more informative results and to more secure and well-founded conclusions. In addition, usage of SegElegans can lead to better data reproducibility via replacing the part of data analysis that is most susceptible to user bias and differences in experience.

The incorporation of deep learning techniques in life sciences is still in a relatively early stage, in part due to the fact that it requires coding experience that is somewhat outside the scope of a standard Biology curriculum, and specialized hardware that can be outside the scope of standard laboratory equipment. We have no doubts that the field will quickly adapt and catch up in both ways, as deep learning becomes increasingly widespread and major microscopy and camera manufacturers incorporate artificial intelligence (AI) driven capabilities into their products. For now, however, we have developed SegElegans with maximum accessibility in mind, providing a network model that is small and efficient enough to be run in freely available cloud computing services such as Colab without requiring the user to type a single line of code or perform any training themselves. The current version of SegElegans is limited to providing full body segmentations, but we plan to expand it with models for segmenting specific areas or tissues of adult animals in the future (as tools for such segmentations are currently limited to worm embryonic stages and dependent on the use of specific fluorescent proteins as segmentation guides[37,38,39]).

Disclosures

The authors have no conflicts of interest to declare.

Acknowledgments

Work in the authors' laboratory is funded by the General Secretariat for
Research and Innovation of the Greek Ministry of Development, the
European Union - NextGenerationEU (project code: TAEDR-0535850, Acronym:

BrainPrecision) in the framework of the Action "Flagship Actions in
Interdisciplinary Scientific Areas with Special Interest in Connection
to the Productive Fabric," under the National Recovery and Resilience
Plan "Greece 2.0", the European Commission Research Executive Agency
Excellence Hub "CHAngeing" (GA-101087071) and the Hellenic Foundation
for Research and Innovation (H.F.R.I.) under the "1st Call for H.F.R.I.
Research Projects to support Faculty members and Researchers and the
procurement of high-cost research equipment" (Project Number:
HFRI-FM17C3- 0869, NeuroMitophagy).

References

1. Kaletta, T., Hengartner, M. O. Finding function in novel targets: C. elegans as a model organism. *Nat Rev Drug Discov.* **5** (5), 387-398 (2006).
2. Markaki, M., Tavernarakis, N. *Caenorhabditis elegans* as a model system for human diseases. *Curr Opin Biotechnol.* **63**, 118-125 (2020).
3. Zuryn, S. C. elegans as a model for health and disease. *Semin Cell Dev Biol.* **154** (Pt A), 1-3 (2024).
4. Corsi, A. K., Wightman, B., Chalfie, M. A transparent window into biology: A primer on *Caenorhabditis elegans. WormBook.* 1-31 (2015).
5. Kerr, R. A. Imaging the activity of neurons and muscles. *WormBook.* 1-13 (2006).
6. Feinberg, E. H. et al. GFP reconstitution across synaptic partners (GRASP) defines cell contacts and synapses in living nervous systems. *Neuron.* **57** (3), 353-363 (2008).
7. Hammarlund, M., Jin, Y. Axon regeneration in C. elegans. *Curr Opin Neurobiol.* **27**, 199-207 (2014).
8. Daniele, J. R. et al. UPR(ER) promotes lipophagy independent of chaperones to extend life span. *Sci Adv.* **6** (1), eaaz1441 (2020).
9. Yemini, E. et al. NeuroPAL: A multicolor atlas for whole-brain neuronal identification in C. elegans. *Cell.* **184** (1), 272-288.e11 (2021).
10. Sinnige, T. et al. Kinetic analysis reveals that independent nucleation events determine the progression of polyglutamine aggregation in C. elegans. *Proc Natl Acad Sci U S A.* **118** (11), e2021888118 (2021).
11. Gkikas, I., Daskalaki, I., Kounakis, K., Tavernarakis, N., Lionaki, E. MitoSNARE assembly and disassembly factors regulate basal autophagy and aging in C. elegans. *Int J Mol Sci.* **24** (4), 4230 (2023).
12. Kutscher, L. M., Shaham, S. Forward and reverse mutagenesis in C. elegans. *WormBook.* 1-26 (2014).
13. Praitis, V., Maduro, M. F. Transgenesis in C. elegans. *Methods Cell Biol.* **106**, 161-185 (2011).
14. Kim, H. M., Colaiacovo, M. P. CRISPR-Cas9-guided genome engineering in *Caenorhabditis elegans. Curr Protoc Mol Biol.* **129** (1), e106 (2019).
15. Kamath, R. S., Ahringer, J. Genome-wide RNAi screening in *Caenorhabditis elegans. Methods.* **30** (4), 313-321 (2003).
16. Lai, C. H., Chou, C. Y., Ch'ang, L. Y., Liu, C. S., Lin, W. Identification of novel human genes evolutionarily

conserved in *Caenorhabditis elegans* by comparative proteomics. *Genome Res.* **10** (5), 703-713 (2000).

17. Kuwabara, P. E., O'Neil, N. The use of functional genomics in C. elegans for studying human development and disease. *J Inherit Metab Dis.* **24** (2), 127-138 (2001).
18. Ranganathan, R., Sawin, E. R., Trent, C., Horvitz, H. R. Mutations in the *Caenorhabditis elegans* serotonin reuptake transporter MOD-5 reveal serotonin-dependent and -independent activities of fluoxetine. *J Neurosci.* **21** (16), 5871-5884 (2001).
19. Lackner, M. R. et al. Chemical genetics identifies Rab geranylgeranyl transferase as an apoptotic target of farnesyl transferase inhibitors. *Cancer Cell.* **7** (4), 325-336 (2005).
20. Yoon, S. et al. Microfluidics in high-throughput drug screening: Organ-on-a-chip and C. elegans-based innovations. *Biosensors (Basel).* **14** (1), 55 (2024).
21. Schindelin, J. et al. Fiji: An open-source platform for biological-image analysis. *Nat Methods.* **9** (7), 676-682 (2012).
22. Schindelin, J., Rueden, C. T., Hiner, M. C., Eliceiri, K. W. The ImageJ ecosystem: An open platform for biomedical image analysis. *Mol Reprod Dev.* **82** (7-8), 518-529 (2015).
23. Rueden, C. T. et al. ImageJ2: ImageJ for the next generation of scientific image data. *BMC Bioinformatics.* **18** (1), 529 (2017).
24. Valente, A. J., Maddalena, L. A., Robb, E. L., Moradi, F., Stuart, J. A. A simple ImageJ macro tool for analyzing mitochondrial network morphology in mammalian cell culture. *Acta Histochem.* **119** (3), 315-326 (2017).
25. Ramki, K., Thiruppathi, G., Ramasamy, S. K., Sundararaj, P., Sakthivel, P. An aggregation-induced emission-based ratiometric fluorescent chemosensor for Hg(II) and its application in *Caenorhabditis elegans* imaging. *Methods.* **221**, 1-11 (2024).
26. Boulin, T., Etchberger, J. F., Hobert, O. Reporter gene fusions. *WormBook.* 1-23 (2006).
27. Braeckman, B. P., Smolders, A., Back, P., De Henau, S. *In vivo* detection of reactive oxygen species and redox status in *Caenorhabditis elegans*. *Antioxid Redox Signal.* **25** (10), 577-592 (2016).
28. He, K., Gkioxari, G., Dollár, P., Girshick, R. Mask R-CNN. *arXiv.* arXiv:1703.06870 (2017).
29. Ronneberger, O., Fischer, P., Brox, T. U-Net: Convolutional networks for biomedical image segmentation. *arXiv.* arXiv:1505.04597 (2015).
30. Alom, M. Z., Yakopcic, C., Taha, T. M., Asari, V. K. Exploring deep learning architectures in NAECON 2018 - IEEE National Aerospace and Electronics Conference. *Proc IEEE NAECON.* 228-233 (2018).
31. Micallef, N., Seychell, D., Bajada, C. J. Exploring the U-Net++ model for automatic brain tumor segmentation. *IEEE Access.* **9**, 125523-125539 (2021).
32. Plebani, E. et al. High-throughput segmentation of unmyelinated axons by deep learning. *Sci Rep.* **12** (1), 1198 (2022).
33. Fudickar, S., Nustede, E. J., Dreyer, E., Bornhorst, J. Mask R-CNN based C. elegans detection with a DIY microscope. *Biosensors (Basel).* **11** (8), 257 (2021).
34. Lalit, M., Tomancak, P., Jug, F. EmbedSeg: Embedding-based instance segmentation for biomedical microscopy data. *Med Image Anal.* **81**, 102523 (2022).

35. Layana Castro, P. E., Garvi, A. G., Sanchez-Salmeron, A. J. Automatic segmentation of *Caenorhabditis elegans* skeletons in worm aggregations using improved U-Net in low-resolution image sequences. *Heliyon.* **9** (4), e14715 (2023).

36. Chen, L. et al. A convolutional neural network model for worm imaging in 2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI). *Proc IEEE ISBI.* 508-512 (2020).

37. Guan, G. et al. Cell lineage-resolved embryonic morphological map reveals signaling associated with cell fate and size asymmetry. *Nat Commun.* **16** (1), 3700 (2025).

38. Cao, J. et al. Establishment of a morphological atlas of the *Caenorhabditis elegans* embryo using deep-learning-based 4D segmentation. *Nat Commun.* **11** (1), 6254 (2020).

39. Zhao, C. et al. EmbSAM: Cell boundary localization and Segment Anything Model for fast images of developing embryos. *bioRxiv.* 2024.09.07.611795 (2024).

40. Castro, P. E. L. et al. SegElegans: Instance segmentation using dual convolutional recurrent neural network decoder in *Caenorhabditis elegans* microscopic images. *Comput Biol Med.* **190**, 110012 (2025).

41. Trebing, K., Stańczyk, T., Mehrkanoon, S. SmaAt-UNet: Precipitation nowcasting using a small attention-UNet architecture. *Pattern Recognit Lett.* **145**, 178-186 (2021).

42. Woo, S., Park, J., Lee, J. Y., Kweon, I. S. CBAM: Convolutional block attention module. *arXiv.* arXiv:1807.06521 (2018).

43. Shi, X. et al. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *arXiv.* arXiv:1506.04214 (2015).

44. Stuhr, N. L. et al. Rapid lipid quantification in *Caenorhabditis elegans* by Oil Red O and Nile Red staining. *Bio Protoc.* **12** (5), e4340 (2022).

45. Kuzmic, M. et al. In situ visualization of carbonylation and its co-localization with proteins, lipids, DNA and RNA in *Caenorhabditis elegans. Free Radic Biol Med.* **101**, 465-474 (2016).

46. Fang, E. F. et al. *In vitro* and *in vivo* detection of mitophagy in human cells, C. elegans, and mice. *J Vis Exp.* (129), e56301 (2017).