Nematode Counting

David May
NYU Center for Data Science
Masters Candidate

Zixuan Shao
NYU Center for Data Science
Masters Candidate

Eric Spector
NYU Center for Data Science
Masters Candidate

Abstract

The NYU Rockman Biology lab needs accurate counts of nematodes on petri dishes. The previous method proved insufficient so we implemented a machine-learning-based approach to apply to their data. We use a Linknet model for semantic segmentation followed by density estimation to determine the number of total worms. We estimate density using a combination of the thin-kernel and watershed methods. Generally, our model improves upon the lab's previously used method.

1 Introduction

Nematodes are microscopic, worm-like organisms that are often hailed for their exemplary ability to undergo biological experiments showcasing the relationship between a genome and survivability in various environments. In this project, we teamed up with a biologist to analyze time-lapse series of images containing Nematodes and ultimately count the number of worms in each petri dish in order to model population growth rates. Despite recent advances in the field of microscopy, it still remains a challenge to properly identify and count miniscule objects that often physically overlap each other. The purpose of the project is to identify one or more robust Computer Vision techniques that will allow researchers to efficiently glean worm counts from a variety of different images, regardless of clarity/density. We will begin by considering the baseline approach and its limitations, then move on to discussing our approach: a two stage model consisting of semantic segmentation followed by density estimation. The crux of this strategy is two subdivide the problem into two more digestible parts, the first being delineating between parts of an image that are "worm" or "not worm", and the second being an estimate of the number of worms in the "worm area" given its size and shape relative to that of an individual worm. We will then finish by discussing the results of our tests and the conclusions we've drawn.

2 Related Work

Prior to this project, the biologists we've been working with have been counting worms via a program called CellProfiler [Wählby et al., 2012]. This open-source application uses simple machine learning techniques to assist scientists with limited experience in this field to use Computer Vision for microscopic object detection. However, CellProfiler often has only moderate performance for a few reasons. Firstly, the image segmentation techniques are often based on thresholding on some capacity, often considered the most basic approach in image recognition. This means that individual pixels are either blackened or whitened depending on which side of a threshold value their own value falls on. Moreover, the series tend to be either high-resolution and low through-put (which is bad for a true time series analysis) or high through-put and low-resolution (which diminishes CellProfiler's efficacy). Our goal is to see if our methods are able to show an absolute improvement upon CellProfiler such that the latter is no longer necessary (at least for just this Nematode-related task).

In addition to the domain of microscopic item recognition, another related domain is the more statistical topic of density estimation. This problem obviously arises quite frequently when counting

Preprint. Under review.

microorganisms but is also not entirely dissimilar to crowd density estimation, i.e. how many people are in a single image. This particular subtopic has its differences to that of counting worms, however, it has been a recent area of interest and may likely lend or yield benefits when taken in conjunction with cell/worm density estimation.

3 Problem Definition and Algorithm

3.1 Task

The task is to count the number of worms on any given petri dish by any means necessary. Given that the petri dishes are tiff images in greyscale the most natural approach is image processing techniques. The output of any single petri dish will be a whole number count of worms suspected to be on the plate.

3.2 Algorithm

3.2.1 Baseline Method

The baseline model removes background noises from the original image by manually modifying the threshold which decides whether particular pixels are treated as the background. Then we compute the number of connected components in the preprocessed images. However, this method is not able to count the number of worms in a cluster nor generalize a variety of dish images.

3.2.2 Two Stage Method

The method consists of two stages. The first stage is a semantic segmentation task that classifies the plate image into background and worms at the pixel level. The second stage predicts the number of worms from the binary mask results in stage one. We use image processing algorithms to help estimate the worm density in the plate due to lack of sufficient labeled data to train a robust machine learning model.

3.2.3 Semantic Segmentation

Semantic segmentation is a type of segmentation task that classifies the input image into regions of different classes so that each pixel in the image corresponds to one class. Our semantic segmentation model aims to transform the original image into a binary mask where each pixel represents either worm or background. The original image of size 1152 x 1152 is equally cropped into 16 patches of subregion as input of segmentation models, and the final mask of the image is reconstructed from the 16 prediction masks output. We assume that a very deep network may not be needed due to the limited training sample and relatively small variation of the input images, and variety of encoders type could influence the model performance.

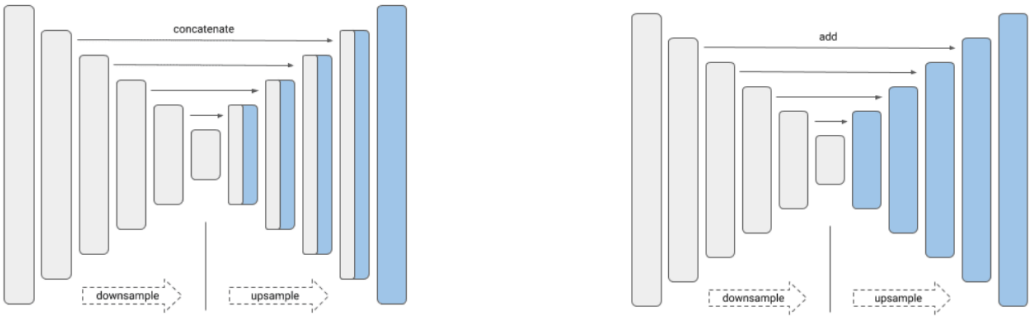

Figure 1: Unet and Linknet.

Two classical segmentation model architectures we use are Unet [Ronneberger et al., 2015] and Linknet [Chaurasia and Culurciello, 2017]. Both models consist of encoder and decoder part where encoder utilized pretrained resnet on ImageNet because we assume that pretrained weights are general

enough to perform well on this specific task and we could fine-tune the weights to leverage the result. The main difference between Unet and Linknet is that in the skip connection, Unet concatenates encoder and decoder blocks at the same level whereas Linknet adds the encoder blocks as residual connection to the decoder blocks.

3.2.4 Thinning Algorithm and Kernel Counting

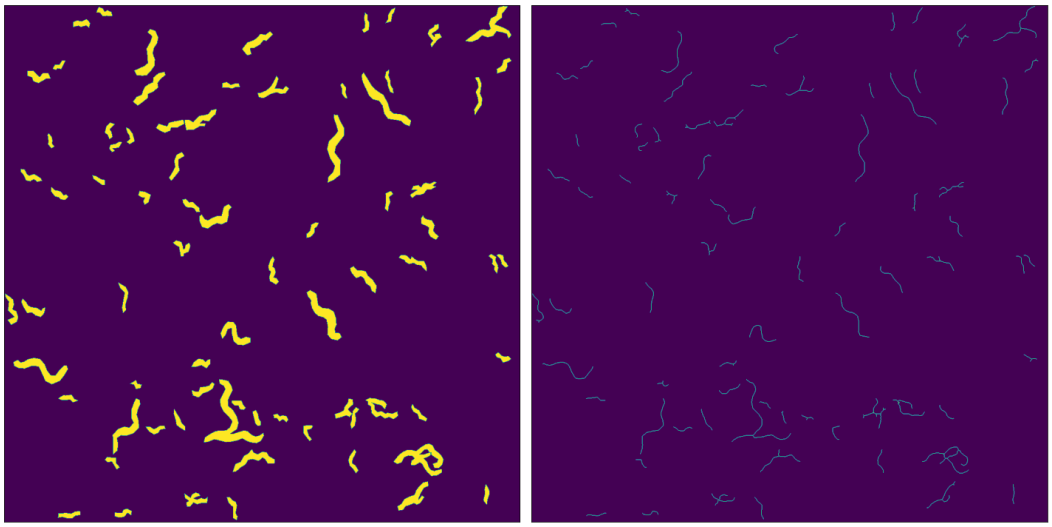

Figure 2: Mask after skeletonization.

Thinning algorithm [Zhang and Suen, 1984] is used in the method of counting heads and tails to estimate the number of worms in the binary mask of the plate. It extracts the skeleton of certain patterns in the image containing pixel value of ones and zeros such that the connectivity is preserved and width of the pattern is one pixel. In the case of our model, each worm in the mask is reduced to a single pixel wide line (Figure 2). Then pixels representing heads and tails only have one valid neighbor pixel if we omit the possibility that heads or tails of one worm perfectly overlap with those of other worms. To count such pixels, we apply a 3x3 kernel on the image such that each kernel represents one pixel. Two properties of the kernel ensure the pixel is a head or tail. (1) The sum of the kernel is two. (2) The center of the kernel is a valid pixel. The count number is half of the valid kernel number, which corresponds to the total number of heads and tails. We assumed this method would perform best in the case of evenly distributed worms.

3.2.5 Watershed Flooding Algorithm

Watershed flooding algorithm [Barnes et al., 2015] is applied on the images where many worms merge into large clusters because the watershed algorithm tends to overestimate the number of objects in a cluster compared to the thinning and counting algorithm. It first calculates the distance of each true pixel from the background and negates all the distance values. Local minima are labeled as marks as the initial condition of the algorithm. The watershed flooding algorithm then applies a priority queue to fill the basins from the bottom by the order of gradients of adjacent unfilled pixels. The algorithm stops when all the components reach the watershed lines where they meet other components. We take the final number of distinct components as the number of worms in the mask.

4 Experimental Evaluation

4.1 Data

The dataset comes in the form of scanned flatbed images of petri dishes with varying amounts of nematodes on them. There are two subcategories of the data: the raw tiffs set of data and the annotated set of data. The raw tiffs subcategory has six zipped folders which each contain a time course of

images. These images start with a few nematodes on the plate and the plate full of food with a new image taken every hour for a total of 200-300 images per time course. For a hand-picked 24 images from a combination of these time courses there are approximate hand counts of the number of worms on the slides. The annotated set of data is 150 images in various resolutions with 81 of these having annotations in the form of json files. In addition to these images there are approximate counts and graphs from the previously used method Cellprofiler on images from the raw tiffs.

The annotated set was turned into a training set of 81 images and the 24 hand-counted images were turned into a test set through a data preparation process. For the training set, the folders containing the images were scanned for images with accompanying json files of the same name. These images were read in with opencv to store the base images and the jsons were parsed to compute the ground truth. The json files contain point outlines of the worm locations and estimates of the number of worms within each outline in plain english. To convert these to the ground truth, the text was parsed into a number and summed for the image to store as a count and the filled in outlines were normalized to the same size and stored in a binary mask ground truth image.

Four of these 81 images in the training set were mislabeled with half of the image having no points for the worms in the bottom half of the image. In practice, however, the dataset with these mislabeled images performed better. This dataset with 81 images including the 4 mislabeled images was ultimately used. This dataset was also augmented for purposes of model training because 81 datapoints in a training/validation set is insufficient. For augmentation, the training dataset was cropped into subimages of size 288 by 288 and these subimages were flipped and rotated. This gave a total of 7776 images with 6000 being used for the training set and 1776 being used for the validation set.

To make the test set workable, the excel file containing the hand-counts for the 24 images was parsed to find the name of the folder and timestamp of each of the hand-counts and these images were extracted from the zipped folders for future reference. The Cellprofiler counts for the test set were also cross-referenced and stored for evaluation.

4.2 Metrics

For this project we used two metrics, the Jaccard Index and Mean Average Error. Jaccard Index was used to determine the similarity between two images. It is described as the size of the intersection of two sets divided by the size of the union of the two sets. Mean Average Error was used for evaluation of the difference in training and test set predicted vs true values. It is defined as the mean of the absolute difference between the counts of predicted vs true value.

4.3 Methodology

In our experiment on the segmentation models, we employ Segmentation Models package [Yakubovskiy, 2019] to accelerate the experiments and maintain the model consistency. Semantic segmentation often has a wide range of loss functions to choose from, and we compared dice loss and focal loss. The overall accuracy difference on the loss function is indistinguishable but focal loss tends to have faster training speed 0.15 iterations per second faster than that of dice loss. To evaluate the models on the validation set, we adopt the classical Jaccard Index (IoU) to measure the accuracy of model output on the true annotated mask. In particular, we run the models on individual images in the validation set and calculate the average of them. The training procedure uses Adam optimizer with learning rate 0.0001 and 0.9 exponential decay. The training set is randomly shuffled into batch size of 10, and the total training epoch is 10 because of the fast convergence.

Apart from the model architecture, we also concern about the depth of the networks and variety of encoders impact on the overall performance. After the experiment, we noticed that encoders with the same scale of weights have almost identical accuracy. Therefore, we stick to resnet due to its fast training time. We tested the influence of network depth and found out that resnet-50 negatively affects the model performance on both accuracy and efficiency, and resnet-34 obtains a moderate depth. So we finally benchmarked all the network architectures on resnet-18 and resnet-34, and chose the best one to train on the whole training dataset as the candidate model used for counting estimation.

Table 1: Mean Jaccard Index on Validation Set

Encoder	Unet	Unet++	Linknet	MANet	DeepLabV3	PSPnet	FPN
Resnet18	0.675	0.672	0.673	0.665	0.657	0.657	0.654
Resnet34	0.678	0.677	**0.682**	0.676	0.664	0.654	0.671

4.4 Results

As mentioned in the methodology several models were implemented and tested for the semantic segmentation stage. We tested Unet, Unet++, Linknet, MAnet, DeepLabV3, PSPnet, and FPN each with ResNet18 and ResNet34. Table 1 contains the Jaccard index achieved by these methods with a higher Jaccard index performing better on the validation set. The best semantic segmentation method we implemented achieved a Jaccard index of 0.682 as seen in Table 1. This was Linknet with ResNet34. The improvement on the Jaccard index for this method during training can be seen in Figure 3 and a sample output of this segmentation model can be seen in Figure 4.

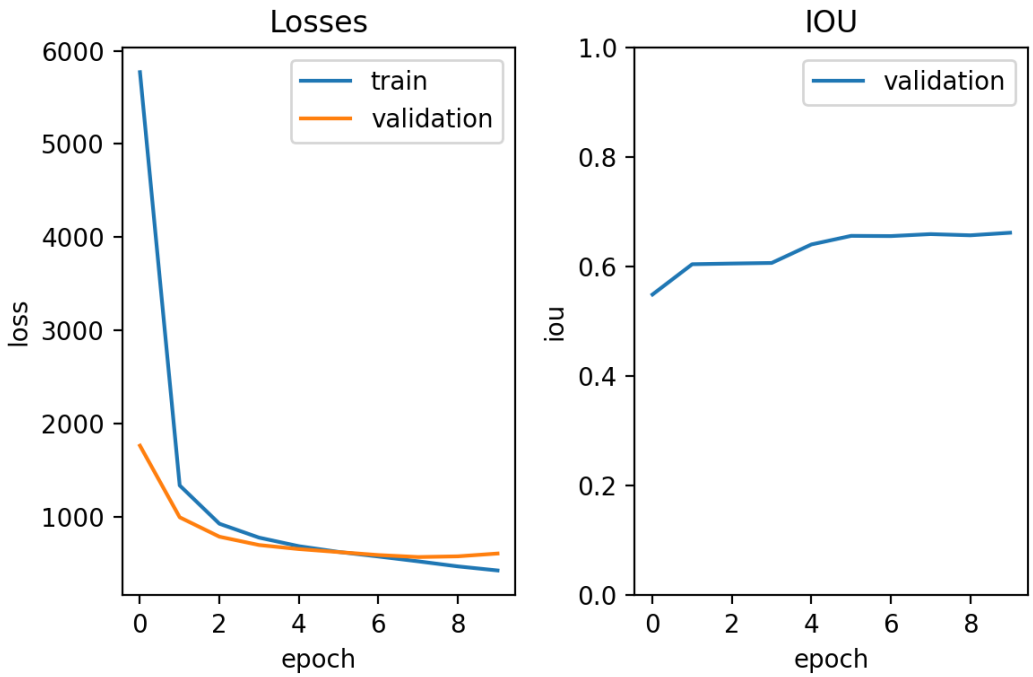

Figure 3: Linknet with ResNet34 as encoder training and validation results.

The first stage of our two-stage model produces the predicted masks of the nematodes in the image. We then apply the various algorithms described in the algorithm section on the training set and then finally the test set to evaluate the effectiveness of each one. All values on the training set have very low error due to the small size and relatively homogeneous nature of the training set. The thinning method has a mean average error of less than 1 on the training set while the watershed method has a mean average error of less than 13. Looking more specifically into the images that are more representative of the test set, which are those that have large clusters of worms, the watershed method more accurately determines the count. Even more accurate is the combination of watershed and the thinning method.

Evaluated on the test set, this watershed and thinning combined method performs best as seen in Table 2. The previous method Cellprofiler reaches a mean absolute error of 1655.6 on the test set. Worse results are achieved by the baseline with a mean absolute error of 1681.2. Our method improves on this accuracy by achieving a mean absolute error of 1421.2 using the thinning method, a 1248.1 using the watershed method, and ultimately a 1199.2 using the combined method.

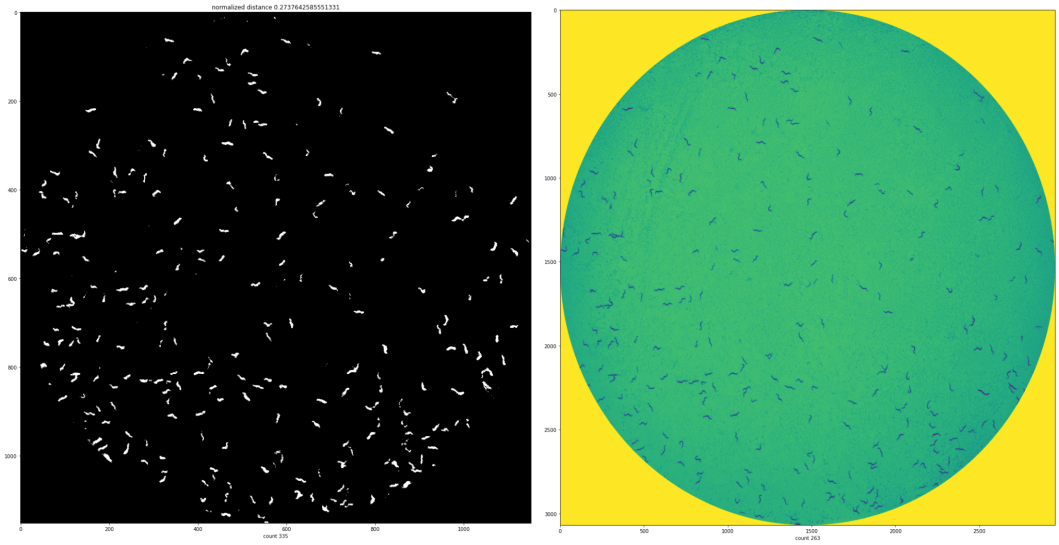

Figure 4: Left predicted mask, Right original image.

Table 2: Mean Absolute Error on Test Set

Cellprofiler	Baseline	Watershed	Area	Thin Kernel	Watershed + Thin
1655.6	1681.2	1248.1	1742.0	1421.2	**1199.2**

4.5 Discussion

The two stage method is a relatively good approach with better performance than the previously used method by the lab. In particular, using semantic segmentation tasks as a general approach to segment the worms and generalize all the worm patterns is a reliable direction to work on. From the result of the training set, we could notice that if we have sufficient training samples including all general cases, the segmentation model is able to identify most worms on the plate, which makes the algorithm on the second stage count the worm number a lot better.

Besides,compared to the watershed method, the thinning kernel counting method will perform very well in most cases if the first stage segmentation result is close enough to the true mask because the possibility of perfect overlapping heads and tails or hundreds of worms clustering into a single connect components, I think, is still relatively low. Therefore, we utilized the watershed method characteristics that overestimate the number of worms, to reduce the potential large error from such special cases. Another point to note is that the densely populated worms are also estimated by human eyes, which may contain a considerable amount of error.

Our approach is an improvement to the lab's previous method which means that it can likely be used by our client. The runtime is relatively fast and it gives additional metrics compared to that method as well. In terms of overall accuracy both methods still have over 1000 mean absolute error on the test set. This is a fairly large error, but is justified by the fact that the hand counts on the test set were approximate and even a human cannot truly count the number of worms. What is ultimately important for the lab is the growth curves of the worms and our method will be able to replicate those from the output. There is still room for improvement with our method, however, in a marginally better semantic segmentation model and with great room for improvement in the counting method. The watershed and thinning combined method proves good but perhaps a machine learning approach would improve upon this further. We did not have enough time or data to explore this as an approach. Another approach we did not have enough time or data to explore was a method that would take into account the sequential nature of the time-course dataset.

5 Conclusions

The two stage model consisting of semantic segmentation and density estimation works relatively well, with low error relative to the overall worm count as well as consistently outperforming the original CellProfiler method. The Linknet with pretrained resnet-34 architecture was particularly effective in the image segmentation task. For density estimation, the thin-kernel method proved to be a great standard for images with moderately full or evenly spaced worms. The watershed method excelled on highly dense plates; the area counting method (estimation through dividing image size by average worm size) was a reasonable solution for individual clusters of incredibly high density. Overall, the best density estimator appears to be a combination of the thin-kernel and watershed counting methods.

6 Lessons Learned

We learned a lot about working with image data and computer vision methods that can be used to interact with this data. We learned about the intricacies of Unet, Linknet, and ResNet. We learned how to communicate data science language with our client in an understandable way. We learned how to overcome difficulties in limited dataset size. We learned about communication within our group. We learned about how to deliver our project to a client in an easily runnable way.

7 References

Richard Barnes, Clarence Lehman, and David J. Mulla. Priority-flood: An optimal depression-filling and watershed-labeling algorithm for digital elevation models. *CoRR*, abs/1511.04463, 2015. URL `http://arxiv.org/abs/1511.04463`.

Abhishek Chaurasia and Eugenio Culurciello. Linknet: Exploiting encoder representations for efficient semantic segmentation. *CoRR*, abs/1707.03718, 2017. URL `http://arxiv.org/abs/1707.03718`.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL `http://arxiv.org/abs/1505.04597`.

Carolina Wählby, Lee Kamentsky, Zihan H. Liu, Tammy Riklin-Raviv, Annie L. Conery, Eyleen J. O'Rourke, Katherine L. Sokolnicki, Orane Visvikis, Vebjorn Ljosa, Javier E. Irazoqui, Polina Golland, Gary Ruvkun, Frederick M. Ausubel, and Anne E. Carpenter. An image analysis toolbox for high-throughput *C. elegans* assays. *Nature Methods*, 9(7):714–716, July 2012. ISSN 1548-7105. doi: 10.1038/nmeth.1984. URL `https://www.nature.com/articles/nmeth.1984`.

Pavel Yakubovskiy. Segmentation models. `https://github.com/qubvel/segmentation_models`, 2019.

T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Commun. ACM*, 27(3):236–239, mar 1984. ISSN 0001-0782. doi: 10.1145/357994.358023. URL `https://doi.org/10.1145/357994.358023`.

8 Student Contributions

Zixuan: preprocessing true mask from json file initial implementation, baseline implementation, segmentation models training and experiments, counting algorithms implementation, backbone of poster, part of report

David: preprocessing counts from json and image normalization, counting algorithms implementation, evaluation on training and test sets, code cleaning, deliverable implementation, contributions to poster, part of report

Eric: research on various methods, contributions to poster, part of report