C. Elegans Brain Segmentation: Machine Learning Project 2 Report

Johan Barthas, Ruben Janssens, Nathan Sennesael

***Abstract*—The Laboratory of the Physics of Biological Systems (LPBS) at EPFL produces many 3D microscopy videos of the brain of moving C. elegans worms. A challenging problem is the segmentation of these images into certain anatomical structures, and the identification of these structures. This process is very time-consuming to do manually, so various solutions to perform this task using machine learning models were explored. First, the similar task of segmenting neurons in 2D images of yeast cells is explored. Then, we explored the 3D-U-Net model for 3D images. We implemented this model to work with the provided C elegans data. More advanced models, a Res-U-Net and Dense-U-Net, are also explored.**

Good results are obtained with these models: the 3D-U-Net, Dense-U-Net and the Res-U-Net obtain a validation DICE coefficient of 0.9, with the latter reaching this faster, in only 13 epochs. Our main conclusion on how our results could be improved in the future is the lack of diversity in the training set. It may be more interesting to train the model on a large set of videos with perhaps more sparsely annotated time frames. This is needed to evaluate how well the model generalizes to other input data, and thus could also be used to train the network in a more generalized way. The models currently also does not incorporate the position of the neurons in past time frames, which could significantly improve results.

I. Introduction

Volumetric C.Elegans brain data is very abundant in the Laboratory of the Physics of Biological Systems (LPBS) at EPFL. As part of their research, they observe the brain of moving C. elegans worms through a microscope, and record 3D videos of this.

Each such video contains 286 time frames, which each consist of a 3D matrix of recorded pixels. However, not all time frames were annotated correctly and thus some had to be removed from the training set. The spacial dimensions (resolution) of the volume matrix are different for different videos, approximately 512 x 312 x 36, which had to be scaled to powers of two in order to feed the data into the network. Each video was rescaled to a size of 256 x 256 x 32. These 3D images show many different individual anatomical structures within the worm's brain, these were annoted within a given segmentation mask which could be used as group truth for our training set. The goal is thus to identify those different anatomical structures within the given 3D videos. An example of two z-slices in such an image, with some of the anatomical structures indicated, is shown in Figure 1. More specifically, the problem our group tried to solve was identifying the AIA central nodule and AIA terminal nodule in two given C. elegans videos.

Annotating the data is difficult and time consuming since the 3D data has to be annotated by going through 2D slices.

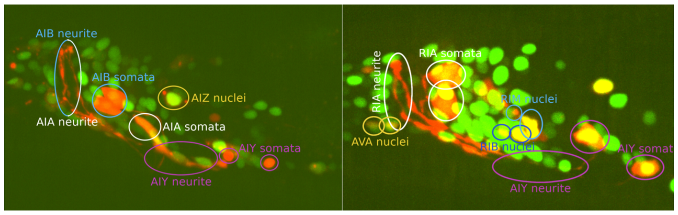

Fig. 1. Example of two z-slices in a C. elegans video recording, with different structures within this frame annotated.

Moreover, 3D data that is manually segmented can't represent overlapping neurons. Our goal is to automate the process using cutting edge deep learning methods such that annotation is no longer a bottleneck in the data analysis pipeline of this research group.

Before we explore the C. elegans problem, we first consider a similar problem the lab encounters: segmenting 2D images of yeast cells. Since these images are only 2D, but the problem that is to be solved is similar, the experience we obtained from this problem was useful when solving the C. elegans problem.

We provide all our implementations on our public Github repository[1] for reproducibility, and we facilitate the understanding of our code by using Jupyter notebooks.

In this report, we first explain the yeast cell segmentation problem and show the methods that were used to solve this. After that, we move on to the segmentation of 3D C. elegans data. We first explain the 3D U-Net convolutional neural network design [1]. Then, more details are given regarding how we implemented this model to work with the provided C. elegans data, how we evaluated it and what results it obtained. Lastly, we explore a more advanced model and give suggestions for further developments.

II. Yeast cell segmentation

Besides C. Elegans, the LPBS group also does research on yeast cells. For this yeast cell data they cope with similar problems, a large amount of yeast cell micropscopy data is available, but segmenting the images by hand is time consuming. Since the C. elegans data was delayed at the start of this project, we started by working on yeast cell data. A 2D U-Net was implemented in Keras and was briefly trained. However, these microscopy images solely contained yeast cells that had to be segmented as a whole and individually counted and quanitified by area, but did not need to be classified.

[1]https://github.com/cweo/3DElegansTracking

Because of this, we started doubting that machine learning methods were truly neccesary for this specific application.

Thus, out of curiousity we tried implementing some classic segmentation methods. This included basic operation such as: taking the gradient of the image, threshholding, binary dillations and corrosions, etc.. Finally, watershedding was applied to distinguish the individual cells. As can be seen in fig. 4, each cell is labelled by a different integer value (color).

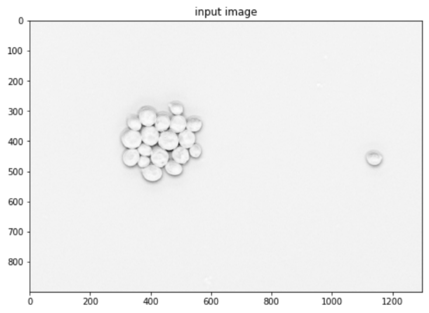

Fig. 2. Micropscopy image (negative)

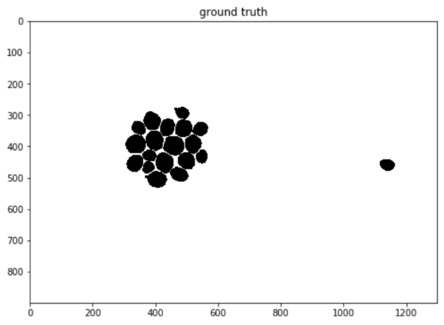

Fig. 3. Hand-made segmentation

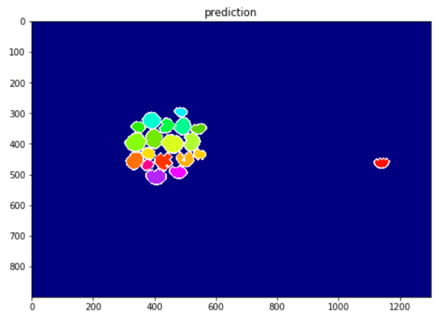

Fig. 4. Output of classic image processing methods

Although the segmentation seems to be quite chunky as a result of dilations, we can conclude that classical methods can indeed perform this segmentation task with reasonable accuracy. An accuracy of 99.3% and an F1-score (dice coefficient) of 91.0% was achieved in the depicted example. This method was applied to the full dataset of 73 images, achieving an average accuracy of 99.4% and an average F1-score of 84.3%. Perhaps a combination of a U-Net with these classical methods would give optimal result. However, this was not further investigated as we focused our attention on the 3D C. Elegans data once it became available.

III. 3D segmentation Network Architecture

The architecture of a 3D-U-Net [1] is similar to the Unet [2] structure used for 2D segmentation.

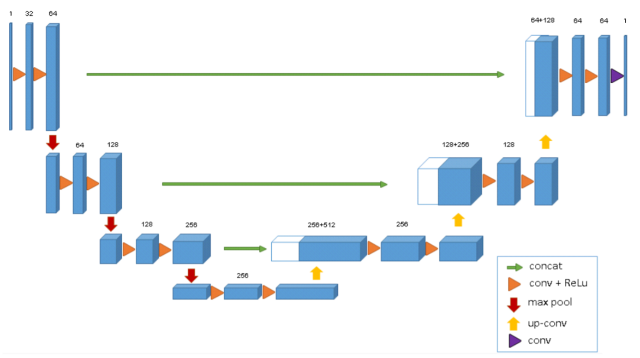

Fig. 5. 3D U-net architecture

In fact, it consists also with a contracting path (left side) and an expansive path (right side). The difference with the original U-net is that we add one dimension for each convolution. Thus, the network is as shown in the figure 5: in the contracting path, each layer is composed of two 3x3x3 (unpadded) convolutions applied first, each followed by a rectified linear unit (ReLU) and a 2x2x2 max pooling operation with stride 2 for each dimension for downsampling. In the expansive path, each layer is composed of two upconvolution 2x2x2 with stride 2 in each dimension followed by two 3x3x3 convolutions each followed by a ReLu. We concatenate the output from the contracting path at the beginning of each layer's input of the expansive path in order to keep the high-resolution features that we lost in the contracting phase. The final layer is a 1x1x1 convolution which allows us to reduce the number of labels which is 1 for us: we will train different networks for each neuron because some neurons may be actually overlapping in some images. The model has 22575329 parameters in total. The last layer is inevitable to be able to train on noisy images and to only train to segment the neuron that we focus on.

IV. Implementation

A. Data processing

The data provided by the lab, which can be made available for you if you send us an email, was unfortunately not directly prone to train our network. In fact the annotation data was stored in csv files. Therefore a first step was to convert it to npz to further be able to load it easily. Moreover the videos themselves were in the nd2 format which was also challenging to load directly into keras without having to make a data generator, thus we converted it to npz also. These .nd2 files were actually 5 dimensional as this contained 2 different colors: red green. These come from two different types of fluorescent proteins which the worms were genetically modified to synthesize. These two seperate signals were thus detected by two seperate detectors at the corresponding wavelengths.

The videos provided don't have a consistent shapes and were resized to a slightly lower resolution of (256,256,32).

Only the red channel was used as input for the model, since this is the only data relevant when looking for the AIA central and terminal nodule, according to the researchers involved.

Also note that we did not apply any augmentation of the input data since the number of frames provided was quite large. Also, optimal results were obtained when using the original input data and using a set of later time frames as true validation set.

B. Evaluation

The network output and the ground truth labels are compared using the minus DICE loss (eq. 1), which should be minimized. [3].

$$\texttt{dice\_loss} = \frac{2|X \cap Y|}{|X| + |Y|} = \frac{2 * TP}{2 * TP + FP + FN} \quad (1)$$

The DICE coefficient measure is used often in image segmentation, and is similar to the Jaccard coefficient in that it measures the similarity between two sets. It is equal to the F1 score [4]. A higher value shows that the network performs better.

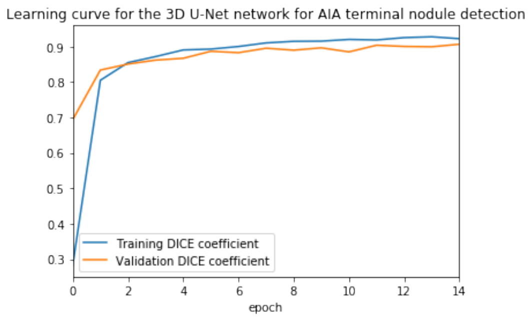

Fig. 6. Dice coefficient evolution during the training of the 3D-U-Net for 15 epochs on the AIA terminal nodule

V. Results

We used Keras with a Tensorflow GPU backend to efficiently train our network. We did 15 epochs for each neuron with a batch size 2, using a GTX 1080 Ti and it took us approximately 107 minutes for each network (430 seconds per epoch).

We trained two different networks using the same network architecture, one to segment AIA central and the other to segment AIA terminal nodules, training on 428 3D-images, and validating on 107 samples. As we explained before, we used two networks instead of one because the two nodules could overlapping/touch one another. This allowed the network to properly track the individual neurons without being perturbed.

Fig. 7 and fig. 8 show a 3D scatted plot of the output of the network next to the ground truth after thresholding the prediction at a value of 0.9. Thresholding is required as the network does not output a binary array and has a lot of values that are very close to zero and result from computational approximations. We obtained a 0.93 dice coefficient on training for both networks and a 0.9 validation dice coefficient.

The learning curve of the AIA terminal nodule segmentation network is shown in Figure 6. The one for the central nodule looks very similar. Both learning curves show that both the training and validation loss are still rising at the end, albeit not quickly, indicating that the model is still underfitting and has more capacity left.

However, it needs to be taken into account that the model was only trained on two videos. Although each of those videos did contain a rather large amount of frames, which should normally be enough to train a 3D-U-Net, the conditions in other real-world videos might be different, so it is unclear how well the model generalizes to other real-world data. Perhaps conditions might be more variable in other videos, so the model capacity could be needed to handles these variable conditions. More annotated videos are need to assess how the model generalizes and to optimize the model for more real-world data. The "shakiness" of the validation learning curves shows that there might not be enough validation data, or that the validation data was not well enough distributed.

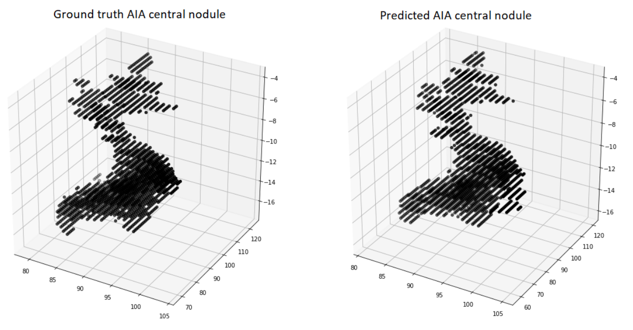

Fig. 7. AIA central nodules

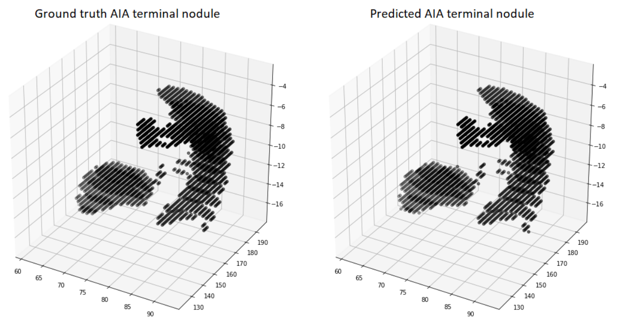

Fig. 8. AIA terminal nodules

VI. Advanced models

To see if the performance can be improved, a more advanced method was tested. For this, the Residual 3D-U-Net (Res-U-Net) model proposed by Kolařík et al. was used. This model implements residual interconnections in the 3D U-Net segmentation model [5].

A 3D-Res-UNet model for the terminal nodule was trained for 15 epochs, on the same data. The validation DICE coefficient obtained was 0.88, and the training DICE coefficient was 0.93. However, when looking at the learning curves (shown in Fig 9), the train and validation score are diverging in the end, indicating that the model might be overfitting. At epoch 12, the best validation DICE coefficient is reached, at 0.90.

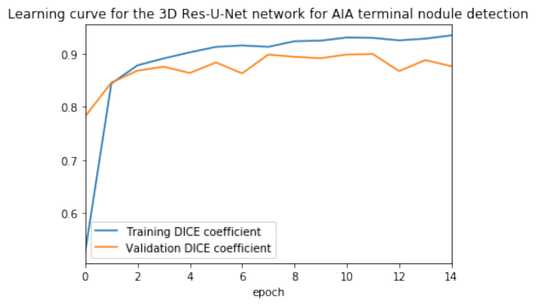

Fig. 9. Dice coefficient evolution during the training of the Res-U-Net for 15 epochs on the AIA terminal nodule

We can conclude that this model reaches a similar performance as the standard 3D-UNet. However, the model reaches this performance faster, so the capacity of this model might be higher. So, to minimize the training time and to possibly get a better performance on a noisier dataset by using this larger model capacity, the Res-U-Net model is to be preferred over the standard 3D-U-Net.

Kolařík et al. also developed a dense 3D-U-Net, (a 3D-U-Net with dense interconnections) which obtained a slightly better performance [5]. This was also implemented and tested for this problem and obtained a validation DICE coefficient of 0.90 and a training DICE coefficient of 0.94. The learning curve of the Dense 3D-U-Net did now show overfitting as strongly as the one for Res-U-Net.

VII. Suggestions for further development

Although the performance that was obtained is already quite satisfactory, the team sees two possible options to further improve it.

First of all, the current methods do not take into account the position of the nodules in previous timeframes to predict their position in the next timeframe. Possibly, expanding the model by using video object tracking methods like nearest-neighbor and correlation matching [6] and others [7], or using of combined segmentation and tracking methods like seeded watershedding [8] could improve on this model. Possibly, the 3D-U-Net model could also be expanded to become a 4D model and take the time dimension directly into account, instead of creating a concatenation of two models. Mryonenko et al. have already explored the use of a 4D CNN for semantic segmentation of volumetric sequences [9].

However, to properly evaluate a model that takes the time dimension into account, much more than two videos are definitely required to train and evaluate the model, so this was not feasible for the training data provided.

What is also interesting, is that according to Çiçek et al. [1], the 3D-U-Net model still performs well for sparsely annotated training data. This means that it is possible that the annotators do not have to annotate every z-slice of every timeframe in a video, but can instead annotate more videos, where not every z-slice is always annotated. This way, more training videos could be generated to be able to properly train models that take into account the time dimension, with less time investment than annotating a complete video.

VIII. Conclusion

We implemented a 3D-U-Net model for the two anatomic structure that were to be segmented. The 3D-U-Net models we implemented and trained on the provided videos both obtained a training DICE coefficient of approximately 0.93 and a validation DICE coefficient of 0.9, which are satisfactory. The learning curves for these models show that improvement is still possible if the model were to be trained for a longer time.

Providing more annotated data for training and validation would of course further improve the model. In our opinion the main improvement that could be made is not necessarily the architecture of the model but instead the data that the model is trained on. The model was trained on a small set of videos with a large set of time frames. It may be more interesting to train the model on a large set of videos with perhaps a smaller set of annotated time frames, since 3D-U-Net performs quite well on sparsely annotated data (ofcourse more data is always better, but time is precious). In other words, the functioning of the network could be generalized to work on different data by feeding it a more diverse set of training videos. Similar results could also be achieved by performing augmentation of the input data, especially if annotations of a larger variety of videos is not an option.

The residual 3D-U-net we trained on the data showed a similar performance (validation DICE coefficient 0.9), but reached this faster, so it shows a higher model capacity and is thus preferable to the regular 3D-U-Net. A dense 3D-U-Net was also explored and obtained similar results to the residual 3D-U-Net.

More advancements could not only be obtained by training the models longer and with more data, but possibly also by expanding the model to take into account the time dimension. The obtained results are already quite promising and machine learning methods are certain to be a significant time-saver in the analysis of C. elegans brain videos.

References

[1] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: Learning dense volumetric segmentation from sparse

annotation," *CoRR*, vol. abs/1606.06650, 2016. [Online]. Available: http://arxiv.org/abs/1606.06650

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[3] L. Nieradzik, "Losses for image segmentation," 2019. [Online]. Available: https://lars76.github.io/neural-networks/object-detection/losses-for-segmentation/

[4] B. O'Connor, "F-scores, dice, and jaccard set similarity," 2012. [Online]. Available: https://brenocon.com/blog/2012/04/f-scores-dice-and-jaccard-set-similarity/

[5] M. Kolařík, R. Burget, V. Uher, K. Říha, and M. K. Dutta, "Optimized high resolution 3d dense-u-net network for brain and spine segmentation," *Applied Sciences*, vol. 9, no. 3, p. 404, 2019.

[6] K. Althoff, J. Degerman, and T. Gustavsson, "Combined segmentation and tracking of neural stem-cells," in *Scandinavian Conference on Image Analysis*. Springer, 2005, pp. 282–291.

[7] R. Yao, G. Lin, S. Xia, J. Zhao, and Y. Zhou, "Video object segmentation and tracking: A survey," *arXiv preprint arXiv:1904.09172*, 2019.

[8] A. Pinidiyaarachchi and C. Wählby, "Seeded watersheds for combined segmentation and tracking of cells," in *International Conference on Image Analysis and Processing*. Springer, 2005, pp. 336–343.

[9] A. Myronenko, D. Yang, V. Buch, D. Xu, A. Ihsani, S. Doyle, M. Michalski, N. Tenenholtz, and H. Roth, "4d cnn for semantic segmentation of cardiac volumetric sequences," *arXiv preprint arXiv:1906.07295*, 2019.