Machine Learning Project 2

James Bardet, Nicolas Delamaide, Ilyas Benadada
CS-433 EPFL

***Abstract*—Image segmentation is a great tool to have insights into the growth of brain neurons in culture. For this, the U-Net neural network architecture can be used to perform neuron and axon segmentation. In this project, we used a modified version of the U-Net which uses a VGG pre-trained network as its encoder. We created two such models which we call UNet11 and UNet16, with the first one using VGG11 and the second one VGG16 as their encoders. We trained the models to detect at first only the neurons cell bodies, and then both neurons cell bodies and axons in microscopic optical images. The neurons were grown either on an empty plate, or on a nanostructured surface, which might influence the direction of the growth of the axons. We thus analyzed the directionality of the axons depending on the orientation of the grid formed by the nanostructure. We show that the models trained to detect the neurons perform well, while the models trained to detect neurons and axons could be improved with a larger dataset of microscopic images with expert labeling of neurons with their axons.**

I. Introduction

Neuroscience is an active field of research because we are far away from understanding all the brain's processes and functions. Thus, as a first step for global brain understanding, neuron analysis is primordial and can be studied through neuron culture. But the neuronal growth analysis is not trivial and recent advances in Machine Learning and Computer Vision, with the improvements of the processors' performance, are going towards a fully automated neuronal tracking.

In this context, the EPFL Galland's and Quack's Groups launched a project where the goal is to develop a new experimental platform for neuronal activity imaging, consisting of nanostructured diamond containing NV centers. At the head of this project, Dr. Losero was looking for ways to detect neurons and axons in grey microscopic images using Computer Vision. For this, we first used a model that detects only the neurons cell bodies. Then, we trained a second model that aims to find both axons and neurons cell bodies. Finally, we attempted to use these models to understand if and how the nanostructures affect neuronal growth. In particular, the nanostructures considered in this case consist of a diamond nanopillar array, potentially influencing the direction of the axons.

II. Models and methods

We first tried to use transfer learning from an already-trained network on a large-scale dataset of labeled cell segmentation called LIVECell [1]. The network architecture was based on CenterMask [2], an anchor-free one-stage architecture with a VoVNet2-FPN backbone using FCOS detection [3]. Unfortunately, the model was too heavy in memory to be trained on our CPU devices and thus we decided to focus our training on a less complex model.

Thus, we used two model architectures both based on the U-Net [4] but using a slight modification as suggested by [5]: the first one uses VGG11 as its encoder and the second one uses VGG16 [6], replacing the original U-Net encoder. Both VGG11 and VGG16 were pretrained on ImageNet [7]. The resulting architecture is called Ternausnet [5], but we will call our models UNet11 and UNet16 for simplicity. We used our models for two segmentation problems each, the first one being performing the segmentation of the neurons only (a binary problem), the second was to segment the neurons as well as the axons (a 3-class problem, including the background). An illustration of the UNet11 architecture for the binary problem can be seen in Fig.1.

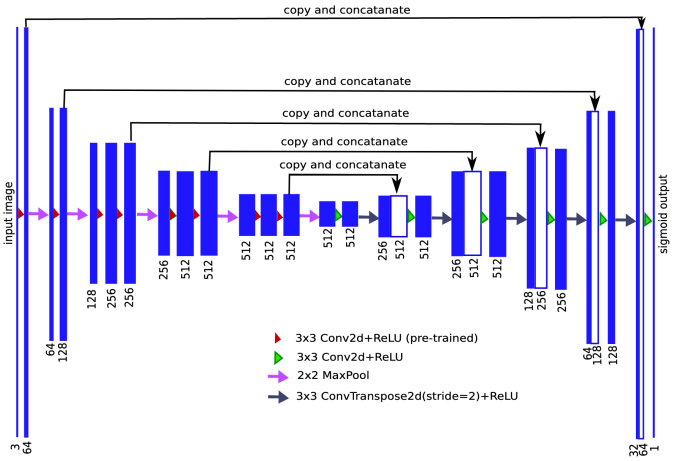

Fig. 1: U-Net architecture using the VGG11 neural network without the fully connected layers as its encoder, for the binary problem. Each blue rectangular block corresponds to a multi-channel feature map, with their width proportional to the number of channels. The number of channels is subscribed below each block. For the 3-class problem, a log softmax of the last copy and concatenation is performed. Therefore, the sigmoid output is replaced by an argmax along the 3 channels (one for each class). Illustration taken from [5].

The U-Net is a fully convolutional neural network developed for biomedical image segmentation, and was originally used for segmenting images of HeLa cells [4]. The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. In order to localize the upsampled features, the expanding path combines them with high-resolution features from the contracting path

using skip-connections. The output of the model is a pixel-by-pixel mask that represents the class of each pixel. A major benefit of U-Nets is that they are capable of learning from a relatively small dataset, which was very useful in our case as we only had 114 training images.

However, in a Ternausnet, the contracting path is replaced by a CNN of the VGG family [6] that serves as an encoder, in our case VGG11 and VGG16. The model architectures are shown in Fig.2. We can see that each have convolutional layers with 3x3 kernels followed by a ReLU activation function, with 2x2 max pooling operations. The two networks only differ in the number of convolutional layers. Though to function as an encoder, the fully connected layers were removed and replaced with a convolutional layer of 512 channels that serves as a bottleneck in the central part of the network [5].

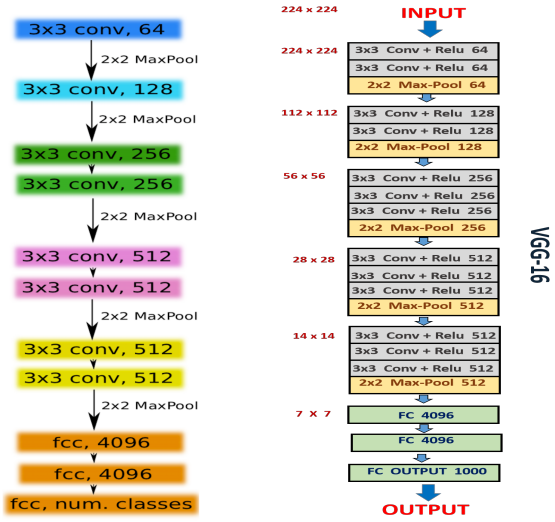

Fig. 2: Left: VGG11 architecture. Right: VGG16 architecture. The rectangles correspond to a 3x3 convolution + ReLU, followed by a Max-Pool (arrow for VG11, orange rectangle for VGG16), with the number of channels written on the right hand side. The green and orange rectangles correspond to the fully connected layers.

The loss function we used for our models is of the form :

$$L = (1 - \alpha)\, H - \alpha\, log\, J \tag{1}$$

where H is a loss function for classification, J is the Jaccard index (Intersection over Union) and $\alpha \in [0, 1]$ modulates the weight to give to the Jaccard index. Minimizing such a loss function thus allows maximizing the probabilities for correct pixels to be predicted and also to maximize the intersection between the original mask and the predicted mask.

For the binary case (neurons only) the Binary Cross Entropy Loss was used and for the 3-class case (axons and neurons) the negative log-likelihood loss was used, corresponding respectively to :

$$H = \frac{1}{N} \sum_{n=1}^{N} -w_n [y_n \cdot log(\hat{y}_n) + (1 - y_n) \cdot log(1 - \hat{y}_n)] \tag{2}$$

$$H = \frac{1}{N} \sum_{n=1}^{N} -log(\hat{y}_n) \tag{3}$$

The Jaccard index was explicitly computed as :

$$J = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i \hat{y}_i}{y_i + \hat{y}_i - y_i \hat{y}_i} \right) \tag{4}$$

where for each class y_i is equal to 0 if pixel i doesn't belong to that class and 1 otherwise, and $\hat{y}_i$ the predicted value (for each class either 0 or 1).

When training our model, we set $\alpha = 0.5$ and used the Adam optimizer using a learning rate of 0.001 and $\beta_1 = 0.9$, $\beta_2 = 0.999$. We used a batch size of 4 for the binary problem and 1 for the 3-class problem, and trained the models for 100 and 85 epochs respectively.

Our initial dataset consisted of only 41 images of size 1536 x 2048, which we divided in four 768 x 1024 to obtain 164 images that we split in a train - test - validation set(70% - 20% - 10%). To best leverage our training dataset of only 114 images, we applied a set of transformations with a given probability to images at each epoch. This ensures that the data for each training epoch is different from the other epochs. We applied the following transformations on the training data: a random crop to 512 x 768 pixels, a vertical flip with probability 0.5, a horizontal flip with probability 0.5 and finally the image was normalized. On the validation set, only a center crop to 512 x 768 was performed followed by normalization. For testing, we only perform normalization.

As the initial research question was to find out whether the axonal growth is impacted by the presence of the nanostructured diamond grid, we needed to find the relative orientations of the axons detected by our best resulting algorithm. For this, we used Principal Component Analysis (PCA) [8] on the segmented axons, where the first component represents the most explained variance and it is orthogonal to the axon's orientation. The first step of PCA is to standardize the dataset, x being the data, μ the mean of this data and σ its standard deviation:

$$x_{std} = \frac{x - \mu}{\sigma} \tag{5}$$

Then, the covariance matrix is calculated for each pair of features, X being the standardized data matrix:

$$COV = \frac{1}{n - 1} X^T X \tag{6}$$

We need to perform the eigendecomposition on this covariance matrix :

$$COV\, v = \lambda v \tag{7}$$

And we select only the first principal component which represents the normal direction of the most explained variance of the data.

To detect the orientation of the grid of the nanostructured diamond plates, we used the Hough Transform [9], which is a technique to isolate features of a particular shape within an image. For this, we first detected small round-shaped objects

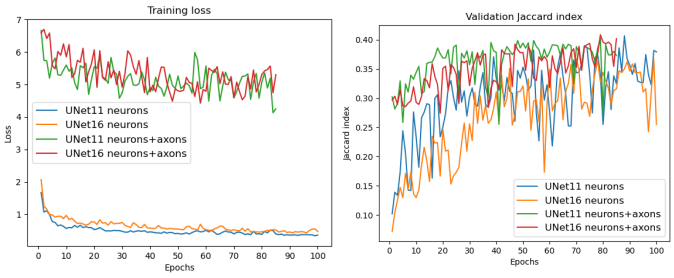

Fig. 3: Loss during training and Jaccard index during validation for the different networks. The jaccard index for the models with axons also takes into account the background.

corresponding to the diamond nanostructure with help of Canny Edge Detection algorithm [10]. Then, we apply the Hough Transform to find lines that pass through those dots and keep only the ones parallel representing the grid orientation.

III. Results

Because we didn't have access to dedicated GPUs, training was quite long, with one epoch taking between 30 to 40 minutes depending on the model. Overall, to train the models for 100 epochs it took us around 66 hours, while for 85 epochs it took about 55 hours.

For each epoch, we recorded the mean training loss as well as the mean Jaccard index. We also performed validation after each training epoch, again recording the mean loss and mean Jaccard index. The results of the validation metrics as a function of the epoch can be seen in Fig.3.

For the models trained on the binary problem, the output corresponds to an image where each pixel corresponds to a probability for the pixel to correspond to a given class. In this case, a probability of 0 means that the pixel belongs to the background while a probability of 1 means that it's a neuron. To convert our probabilities to a binary output, we set a threshold of 0.3 (as suggested by [5]) with values above the threshold set to 1 and the others set to 0. Then we multiply the pixel values by 255 to obtain as an output a black and white mask. For the 3-class problem, the output corresponds to an image where each pixel is assigned a value corresponding to its class label. For the background, it's 0, for the neurons it's 1 while for the axons it's equal to 2. To convert this output to a mask, we multiply the pixel values by 127 such that the background as pixel value 0, the neurons 127 and axons 254.

We evaluated our models on a test dataset of 32 images. As a performance metric, we used the Jaccard index that we computed independently for each class, except for the background. The results of the evaluation of our models on the test set can be seen in Table I, which shows the mean Jaccard index on the test data for each model as well as the highest Jaccard index.

We also overlayed the predicted mask on top of the input image, where the results for the neurons only are shown in Fig.4 and for the neurons and axons on Fig.5.

Regarding the grid orientation, we found out that the images have two different grid orientations with slopes being

	Mean Jaccard ± std		Highest Jaccard	
Model	Neurons	Axons	Neurons	Axons
UNet11 neurons	0.44 ± 0.17	-	0.71	-
UNet11 with axons	0.20 ± 0.09	0.05 ± 0.05	0.40	0.14
UNet16 neurons	0.33 ± 0.14	-	0.70	-
UNet16 with axons	0.27 ± 0.13	0.07 ± 0.17	0.48	0.23

TABLE I: Mean jaccard index ± std of the models as well as highest jaccard index obtained on the test set. Note: we used the second highest value for the jaccard on axons of the last model as the highest value was 1 due to one image having no axons.

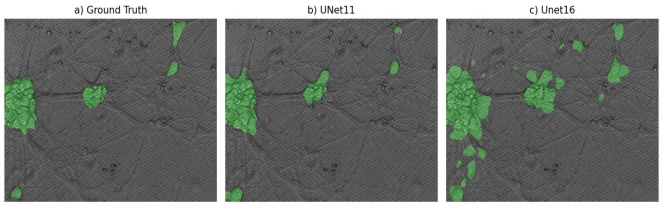

Fig. 4: Overlay of the predictions made by our models for the binary problem. a) Original image with ground truth overlay. b) Original image with predicted overlay by UNet11. c) Original image with predicted overlay by UNet16.

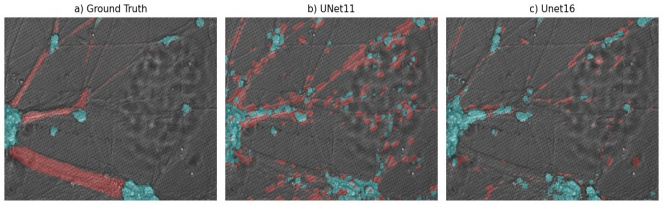

Fig. 5: Overlay of the predictions made by our models for the 3-class problem. a) Original image with ground truth overlay. b) Original image with predicted overlay by UNet11. c) Original image with predicted overlay by UNet16.

either 0.23 or 0.51. An example is shown in Fig.6, with the x-axis pointing horizontally to the right and the y-axis pointing upward vertically. Our technique was robust to detect grid in images when the spacing between nanopillars was 4μm but not with the 2μm nanostructure because then the grid was too small and thus undetectable with our pipeline (see Appendix Fig.8). The result of the PCA analysis to discover the axons' orientation is shown in Fig.6 for one mask, with 0° corresponding to vertical axis pointing upward and 90° facing horizontally to the right.

As we were not able to detect grid orientation in small grid spacing images, we classified by hand our image dataset into grid, small grid (not detected because of the small spacing, see appendix Fig.8) and no grid images. There were only 2 images that did not belong to any of the categories because one part of the image had a grid and the other didn't. As the axon segmentation prediction from the networks was not precise enough, we used the hand-labeled axons to find their orientations using PCA and plotted the results in Fig.7. It is hard to say from the histogram whether the images belonging to different classes come from the same distribution, thus we

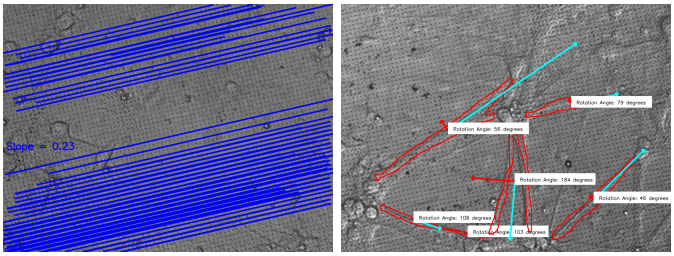

Fig. 6: Left: Slope detection algorithm using Hough Transform. Right: orientation of axons using PCA.

	grid vs small grid	grid vs no grid	small grid vs no grid
statistics	0.0904	0.2609	0.2074
pvalue	0.4006	0.0013	0.0158

TABLE II: Two samples Kolmogorov–Smirnov test results.

have done multiple two-sample Kolmogorov–Smirnov tests. This test aims to find the probability that two sets of samples were drawn from the same unknown probability distribution. It does so by quantifying the distance between the empirical distribution functions of the samples. The null distribution is calculated under the null hypothesis that the samples are drawn from the same distribution. The results of the test are shown in Table II. We can see that the p-value is smaller than 5% for every distributions except between the grid versus small grid test. Thus, we can reject the null hypothesis that the two samples are from the same distribution between the presence and absence of the grid. We can conclude that the presence of the grid with the 2 different pillar spacing (grid and small grid) has an impact on the axonal growth orientation.

Then, we plotted in Fig.7 the orientation of the axons relative to the grid orientation, a 0° orientation meaning the axon follows the grid orientation. We have tested whether the distribution was uniform using a one-sample Kolmogorov–Smirnov test and we found a p-value smaller than 5% meaning that we can reject the hypothesis that the distribution was uniform. Thus, the orientation of the axons are not uniformly distributed in space: the grid has indeed an impact on axons' orientations.

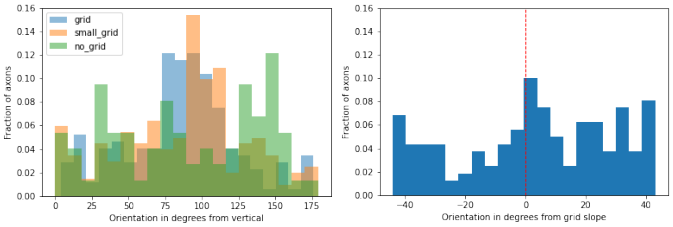

Fig. 7: Left: Histograms of the axons orientations distributions. Right: Distribution of the axons' orientation relative to the grid.

IV. Discussion

We can see from Table I and Fig.5 that the models to detect the axons under-perform. It is probably due to the fact that a really small amount of labeled data was given to the network and that the hand-labeling of axons was a difficult and time-consuming task. As we can see in the appendix (Fig.8 and 9), there was high variance in how the axons looked due to the different grid spacing. When there was a high grid spacing, the axons were clearly defined, while with low grid spacing, the axons appeared as thin dark lines, sometimes interleaved, which made it difficult and very time-consuming to label. That is why we used only the hand-labeled masks to detect axon orientation as the network outputs were not precise enough for this.

But even with those difficulties, we have compared different architectures and we can say that for the detection of neurons only, the UNet11 for the binary problem is the most suited model. It achieves a mean Intersection over Union (IoU) of 0.44 with the best prediction being an IoU of 0.71 (Table I). We also see that models trained for the 3-class problem have a poorer performance for neuron segmentation than neurons only trained for neuron cell body detection. For example, UNet11 for neurons and axons had half the mean IoU as UNet11 for neurons only (0.44 vs 0.20 respectively, Fig.I). This can be explained by the fact that it is easier for the network to learn a binary problem than a 3-class problem, especially here when neuron cell bodies and axons share very similar features on some images. Among the models trained for the segmentation of neurons and axons, UNet16 performed better with a mean Jaccard index on neurons of 0.27 and 0.07 on axons (versus 0.20 and 0.05 for UNet11).

Overall for neuron detection, the UNet11 model trained for the binary problem produces satisfying results as can be seen in Fig.4. Though the models for neuron cell bodies and axon segmentation could be improved. This could be done by using a bigger dataset of expert labeled data or using a dataset with fluorescence images of neurons and axons [11]. Indeed, this would have allowed us to use simpler image segmentation algorithms for robust labeling as it has been done before in [12]. Also, having better quality images when the grid was small ($2\,\mu m$) would have allowed us to better classify axons on the nanostructured diamond plate and the ones on the empty plate. Then the analysis of the distributions would have been more robust.

V. Summary

In this work, we compared different architectures for neurons and axons segmentation. We have seen that image processing pipelines were not able to detect neurons in those gray microscopic images but a deep neural network was able to detect neurons and axons with a certain accuracy. We were then able to detect labeled axons' orientations and compare them to the nanostructured grid orientation to conclude that the diamond plate has an influence on neuronal growth. We thus developed a ready-to-use pipeline that can predict axon orientation from microscopic images with help of CNNs and a small dataset of labeled images.

A link to our code and trained models can be found in the appendix.

References

[1] Christoffer Edlund et al. "LIVECell—A large-scale dataset for label-free live cell segmentation". en. In: *Nature Methods* 18.9 (Sept. 2021). Bandiera_abtest: a Cc_license_type: cc_by Cg_type: Nature Research Journals Number: 9 Primary_atype: Research Publisher: Nature Publishing Group Subject_term: Cell biology;Research data;Technology Subject_term_id: cell-biology;research-data;technology, pp. 1038–1045. ISSN: 1548-7105. DOI: 10.1038/s41592-021-01249-6. URL: https://www.nature.com/articles/s41592-021-01249-6 (visited on 12/22/2021).

[2] Youngwan Lee and Jongyoul Park. "CenterMask : Real-Time Anchor-Free Instance Segmentation". In: *arXiv:1911.06667 [cs]* (Apr. 2020). arXiv: 1911.06667. URL: http://arxiv.org/abs/1911.06667 (visited on 12/22/2021).

[3] Zhi Tian et al. "FCOS: Fully Convolutional One-Stage Object Detection". In: *arXiv:1904.01355 [cs]* (Aug. 2019). arXiv: 1904.01355. URL: http://arxiv.org/abs/1904.01355 (visited on 12/22/2021).

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *arXiv:1505.04597 [cs]* (May 2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597 (visited on 12/21/2021).

[5] Vladimir Iglovikov and Alexey Shvets. "TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation". In: *arXiv:1801.05746 [cs]* (Jan. 2018). arXiv: 1801.05746. URL: http://arxiv.org/abs/1801.05746 (visited on 12/21/2021).

[6] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv:1409.1556 [cs]* (Apr. 2015). arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556 (visited on 12/21/2021).

[7] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *arXiv:1409.0575 [cs]* (Jan. 2015). arXiv: 1409.0575. URL: http://arxiv.org/abs/1409.0575 (visited on 12/21/2021).

[8] Ian T. Jolliffe and Jorge Cadima. "Principal component analysis: a review and recent developments". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (Apr. 2016). Publisher: Royal Society, p. 20150202. DOI: 10.1098/rsta.2015.0202. URL: https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202 (visited on 12/23/2021).

[9] Paul C. Hough V. "Method and means for recognizing complex patterns". Pat. 3069654. Dec. 1962. URL: https://www.freepatentsonline.com/3069654.html (visited on 12/22/2021).

[10] J. Canny. "A computational approach to edge detection". eng. In: *IEEE transactions on pattern analysis and machine intelligence* 8.6 (June 1986), pp. 679–698. ISSN: 0162-8828.

[11] Anthony Brown. "Live-cell imaging of slow axonal transport in cultured neurons". In: *Methods in cell biology* 71 (2003), pp. 305–323. ISSN: 0091-679X. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3699318/ (visited on 12/23/2021).

[12] Ronald Wihal Oei et al. "Convolutional neural network for cell classification using microscope images of intracellular actin networks". en. In: *PLOS ONE* 14.3 (Mar. 2019). Publisher: Public Library of Science, e0213626. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0213626. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0213626 (visited on 12/22/2021).

APPENDIX

All our code, the labeled and original images as well as our trained models can be found on this link: https://github.com/jbardet/neuron_analysis.

On Fig.8 and Fig.9 we show examples of images with low and high grid spacing respectively. We can see that there is a significant difference in the appearance of neurons and axons. In particular, the axons on the small grid spacing images were very difficult to label, as they appear as very thin lines, sometimes hardly distinguishable from the background (Fig.8).

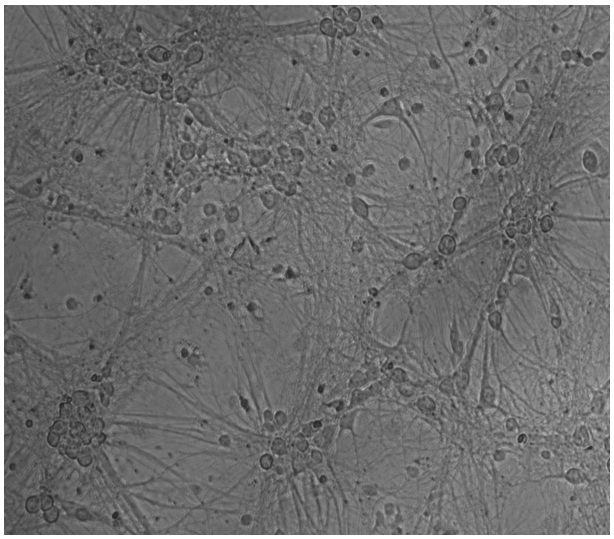

Fig. 8: Image of cells with low magnification.

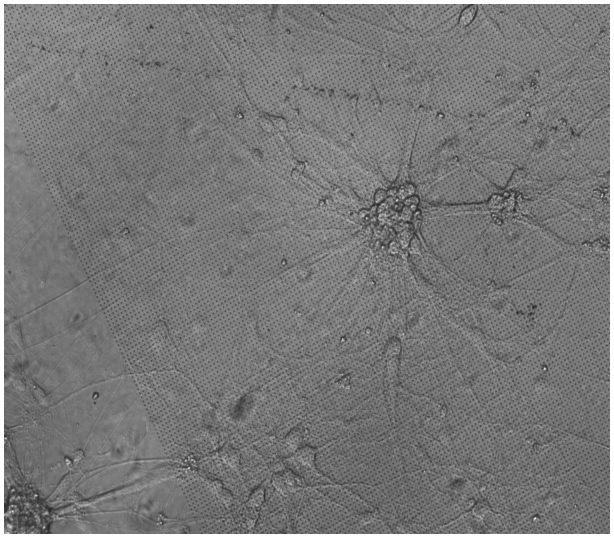

Fig. 9: Image of cells with high magnification.

Moreover, some images had a grid that was too small to be detected by our pipeline. An example of such an image is shown in Fig.10. Even with the naked eye, one needs to zoom in quite a bit to distinguish the grid.

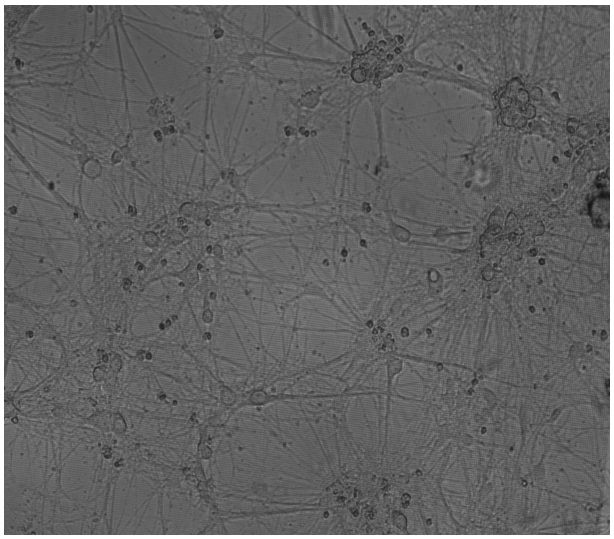

Fig. 10: Image of cells with low magnification and undetectable grid.