Deep Learning Semantic Segmentation of Neuronal Cell Images

Daniel Corona*, Daniel Silva*, Mariana Calado*, Jaime Cardoso**, Wilson Silva**

***Abstract*—This report in the form of an article aims to describe the methods and results of a deep learning algorithm acting on neuronal cell images. The objective of this algorithm is to implement a segmentation of the different cells which are present in the images. The datasets for training and testing of the machine learning models were given by Sartorius [1], a partner of the biopharmaceutical industry. Results of the proposed task are shown for different deep learning algorithms. All the algorithms were written using *Python* and the main library used for the deep learning implementation was *PyTorch* [2].**

***Index Terms*—neuronal cell, semantic segmentation, deep learning.**

I. Introduction

SEMANTIC segmentation is the task of labeling each pixel of an image to a certain object or class of interest. In this project the objects that needed to be segmented were neuronal cell from images obtained via light microscopy. The observation of this type of image is very important for the quantification of the cellular response to some drugs which are used to treat neurological disorders. The hand-made task of segmenting these cells in microscopy images can be very challenging and highly time consuming. In order to overcome these challenges we propose the use of five different deep learning algorithms integrated with pre-processing and post-processing of the images that are able to automatically segment the neuronal cell in the images. The datasets were made available by Sartorius and we have used the intersection over union, the Dice coefficient and the accuracy metrics to assess the quality of our models. [1]

January 28, 2022

A. Image Segmentation

Image segmentation is one of the main process implemented in the image processing field. There are many different techniques that are used nowadays in order to partition regions of an image which have similar characteristics. The objects that are segmented dictates the best approaches, depending on the features of the objects one may prefer to use a specific technique than other. With the development of machine learning, specially deep learning, the task of finding the principal features of a certain object are performed by the algorithm itself, thus it facilitates and speeds up the process of segmentation. [3]

* Student Master in Bioengineering, FEUP/ICBAS
** Professor of Assisted Computer Diagnosis, Master in Bioengineering, FEUP/ICBAS

B. Deep Learning

1) Neural Networks: Neural Networks (NN) are inserted in the Machine Learning field of Artificial Intelligence, it is used in Deep Learning and its basic structure is called perceptron (figure 1), a group of perceptrons forms a layer. Layers in NN can have different structures and functions but the main framework of these layers is the receiving of inputs, which is the data that is being processed, the multiplication of these inputs by different weights and the output of the results, which are often called scores or logits. These logits may be transformed into probabilities by an activation function. Finally these outputs enter into another layer, being now the inputs of the following layer. More than one layer of perceptrons forms a Multilayer Perceptron (MLP).

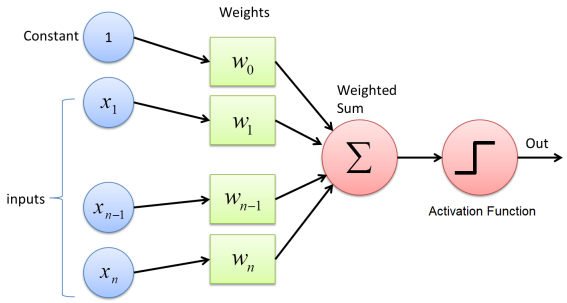


Fig. 1: Structure of a Perceptron

As we can observe, just one layer will create a linear model, thus, in order to have more powerful networks which are able to correctly classify data that is non-linearly separable, one must add hidden layers to the Neural Network. Usually, a neural network with less perceptrons and more layers have a better performance than one with more perceptrons and less layers. [4]

2) Convolutional Neural Network: Convolutional Neural Networks have a workflow very similar to the Neural Networks workflow, the difference is that the input layers are processed by a spatial filter (kernel) and the output layer is obtained using a convolution operation between the input layer and the filter. The convolution operation may be held using more than one filter, thus more than one output layer is going to be created.

Observing the figure 2 we can observe the two basic types of operations used in CNN models, one is the convolution already described, the another one is the max pooling which is basically a maximum rank filter operation, widely used in image processing techniques. The maximum rank filter operation uses a spatial kernel that convolutes through the input data and give as output the maximum value from part of the input data delimited by the kernel. CNN's are widely

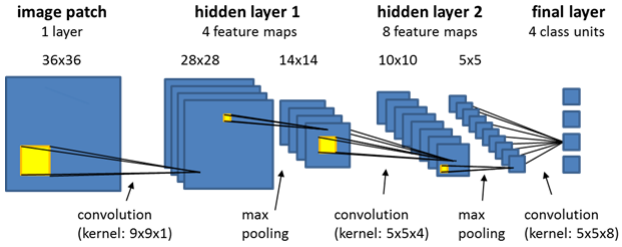


Fig. 2: Common Convolutional Neural Network Arquitecture

used to process images and that is because the weights used are shared between the input data and the logits depends on the neighborhood of the input defined by the kernel, this type of processing preserves the spatial information which is very useful in the case of images. [4]

3) Training Neural Networks: A neural network model is defined by its parameters or weights which multiply the different inputs of the layers. These weights are automatically learned by the model by two important steps which are Feedforward and Backpropagation. Firstly the weights are randomly initialized, usually with values near zero, with low values of weights the model starts as an approximately linear model and then learns the non-linearities where needed. In this project the goal of the training is to increase the similarity between the output image of the model and a certain target image (mask) associated with the input. Since these masks are the ground-truth, the learning task of the model is supervised and the quality of a certain model is translated into a loss function which decreases when the similarity between the output and the mask increases.

- Feedforward - Corresponds to the calculation of the output of the neural network. The input data is fed to the network and then multiplied by the different weights of the model, going through the layers until it reaches the end (output).
- Backpropagation - The measure of the quality of our model is defined by a loss function. This loss function is dependent of the weights and since we are interested in minimizing this function, the best model is defined by the following weights (w^*):

$$w^* = argmin_{w^*} Loss(w) \tag{1}$$

Since we cannot calculate the exact w^* in the case of Neural Networks where the Loss normally have more than one solution, the gradient descent algorithm is used. This algorithm corresponds to the Backpropagation step where all the weights are updated accordingly to the following expression:

$$w_i^{new} = w_i^{old} - \eta \frac{\partial Loss(w)}{\partial w_i} \tag{2}$$

Where η is a hyperparameter called learning rate, which defines the magnitude of the "step" taken in direction of minimizing the loss. There are a lot of different loss functions and one can change the learning rate during the training in order to obtain better results.

II. Methods

For the implementation of a Deep Learning algorithm which can perform the semantic segmentation of neuronal cells we have opted for using Convolutional Neural Network models by the advantages already described in the section I-B2 when working with images as input data. In this project the performance of different CNN models were assessed. The CNN models used were ones that have already been used in segmentation tasks and are well documented in the bibliography. For each image the goal is to label each pixel as being part of a cell or the background, hence our segmentation problem turns into a classification one where we have two classes:

- Class 0 (negative): Pixel that belongs to the background.
- Class 1 (positive): Pixel that belongs to a cell in the image.

A. Dataset

The Sartorius Dataset is composed of contrast microscopy images in PNG format with 520x704 dimensions, consisting of more than 1.6 million cells of eight cell types with distinct morphologies. There are 606 grayscale images for train and 122 for test. A cross validation with a random sub-sampling was performed on the data during the training so it was used 20% of the training set as the validation set. The training annotations file contains the pixel coordinates for each cell in the images, that is, the true label of each training image. Each image in the training set is associated with multiple annotations, therefore there will be multiple cells in one image [1]. All images have a designated class, based on the type of neuronal cell present. There are 3 distinct classes in this Dataset: astro (represents astrocytes), cort (represents cortical neurons) and shsy5y (represents a cell line differentiation of SK-N-SH neuroblastomas).

In the next graph, we can analyse the data distribution of each class:

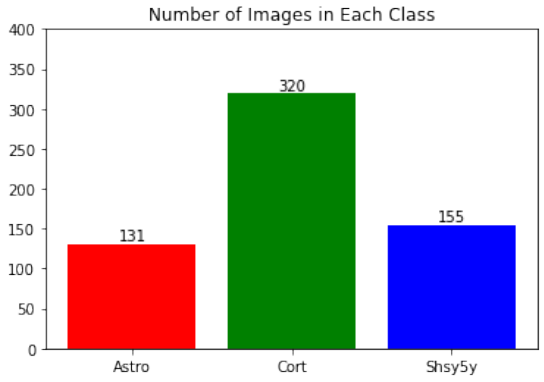


Fig. 3: Images' distribution across classes

This means that there might be problems with unbalanced data. This complication will be taken into account in the development of the work, and will be fixed if necessary.

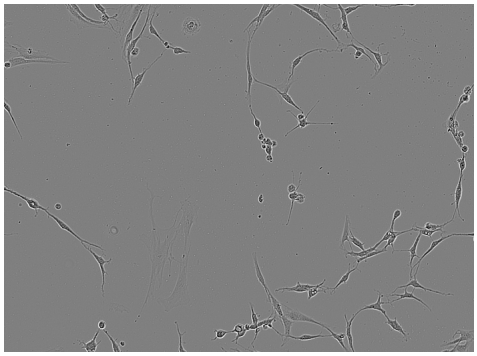


Fig. 4: Image from Astro Class

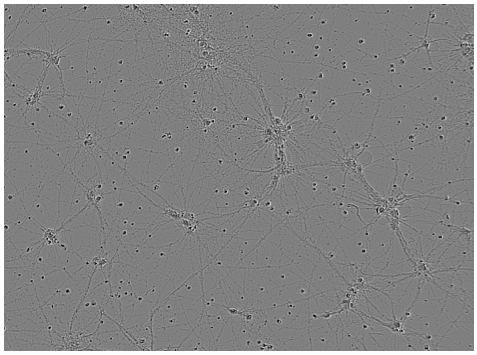


Fig. 5: Image from Cort Class

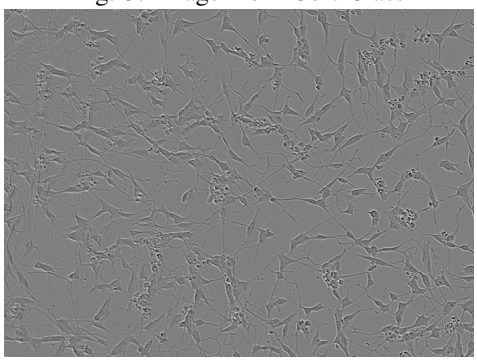


Fig. 6: Image from Shsy5y Class

B. Pre-Processing

The binary masks used to evaluate the outputs of most models were created using the annotations given by the Dataset for each image. In these annotations, the numbers can be splited as the location of the pixel and the length of the cell in the x-axis. The images were resized to 512x512 pixels in all the models executed.

C. Models and Hyperparameters

1) U-Net: It was used a U-Net model with batch normalization for the image segmentation. This model consists of four levels of blocks containing two convolutional layers with batch normalization (that is a way to speed up the training process and to improve the results by minimizing the internal covariate shift), Rectifier linear units (ReLU) is used as activation function of the hidden layers, since the speed of convergence of the gradient descent is much faster than hyperbolic tangent or sigmoid functions [9] and one max pooling layer in the encoding part and up-convolutional layers instead in the decoding part. The number of convolutional filters in each block is 32, 64, 128, and 256 as can be visualized in figure 7. The bottleneck layer has 512 convolutional filters. [7]

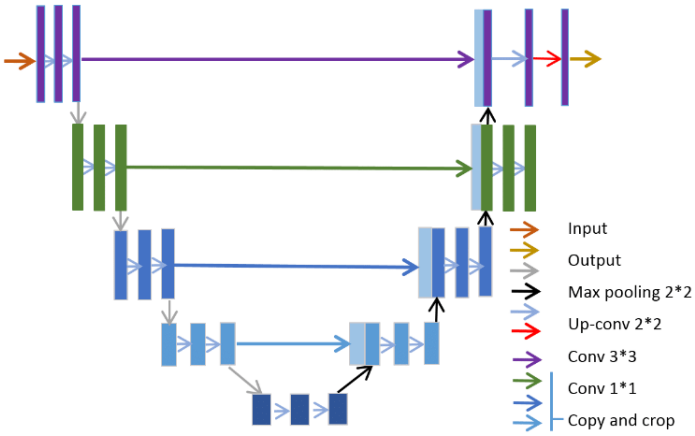


Fig. 7: U-Net Architecture [8]

2) Addition of One Hot Encoding - Hot U-Net: One Hot Encoding is an approach that will convert the categorical classes of the cell images (astro, cort or shsy5y) into binary combinations of tensors with the same size of the images. The procedure is simple: when the DataLoader gets an image from the Dataset, instead of receiving a 1x512x512 tensor, it will receive a 4x512x512 one. These 3 new channels consist of combinations of tensors filled by zeros or ones, depending of the categorical class: if the image represents cortical neurons, the combination of tensor will be (1,0,0); if the image is of astrocytes, the tensors are (0,1,0); in the case of neuroblastoma cells, the distribution will be (0,0,1). The figure 8 depicts this scheme. Using this technique, it is supposed to improve the segmentation performance, because each class of neurons will get a specific segmentation. This is relevant in this case due to the perceivable differences of size and shape of the cells in each class.

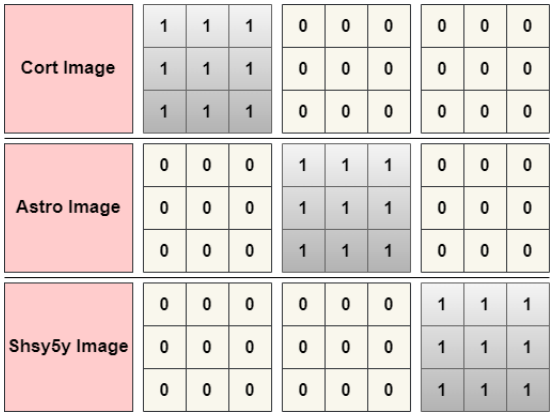


Fig. 8: Visual representation of the new input to the model

3) Addition of positive weights - Hot U-Net Weighted: While defining the loss function for this model, it was added

a parameter of positive weight of 3. This means that the ratio between negative and positive examples is 3. This will increase the weight of positive classification, and it may improve the model's performance because the images have more negative pixels than positive ones.

4) Addition of Cell Pose - Hot-Flow U-Net: CellPose is a deep-learning network algorithm that can segment cells and nucleous from an extensive range of images.[11] This algorithm was trained on a dataset of over 70,000 segmentation objects. Using this pre-trained model, it was possible to obtain the flow field representations of the Sartorius Dataset images. These representations can be translated as the gradient flows to each cell's center. By following these paths, all pixels belonging to a given cell will converge in its center. An image representing the flow field of figure 4 will be displayed bellow:

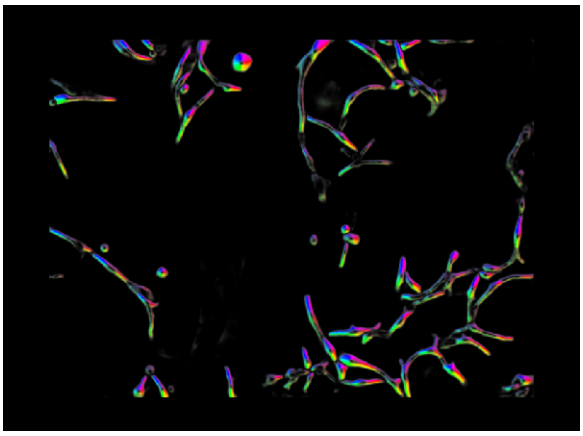


Fig. 9: Flow Field Representation of figure 4

The use of the grayscale images of the field representations as labels for the model (instead of binary masks) is said to be beneficial for its performance, because many top-scored teams of the competition used this method. The explanation provided by some members was that it was easier for the model to understand and delineate the limits of the cells, because these limits had a different intensity, comparing to the centers of the cells.[12] Consequently, it was proposed a model that used these images as labels, changing the loss function to the Mean Squared Error (due to the labels' image type). When evaluating the IoU and Dice metrics, the output image suffered a binarization following a certain threshold.

5) Addition of Dropout Layers and Data Augmentation - Hot-Drop-U-Net: In this approach, the U-Net model was modified, adding dropout layers after every convolutional layers. The Dropout layers randomly sets the inputs values to zero, given a certain probability (p=0.2). This procedure may improve the model overfitting, but it achieves significantly higher impact when applied after Flatten layers (not presented in this model). Therefore it might not make great differences. Finally, more transformations were added to the train set, such as the Affine transformation that makes the image center invariant [6] and the random Horizontal and Vertical flip that makes the given image flip in a randomly fashion with a given probability. These will add Data Augmentation to the model.

6) Epochs: All models were trained for 50 epochs. It is not an ideal number, but the time limit of the notebooks' runtime using graphical computing wouldn't allow for much more epochs.

7) Learning Rate: The learning rate used to train all the models started at $lr = 0.01$ and followed an exponential decay of $gamma = 0.5$, meaning that for every 10 epochs, the learning rate will get cut by half.

8) Loss Function: 2 different loss functions were used, depending of what the model was evaluating. When the output of the model was a binary image, it was used the Binary Cross Entropy With Logits, that incorporates the sigmoid activation function and the Cross Entropy Loss function (equation 3). When it was expected for the model to output a grayscale image, the Mean Squared Error Loss (equation 4) function was applied.

$$BCE = -(y \times log(\hat{y}) + (1 - y) \times log(1 - \hat{y})) \quad (3)$$

$$MSE = \frac{1}{2}(y - \hat{y})^2 \quad (4)$$

In the equations 3 and 4 the y corresponds to the ground truth for a certain pixel in the input image, that is 0 or 1 whether the pixel pertains to class 0 or 1, respectively. $\hat{y}$ is the class prediction of the model for the pixel.

9) Batch Size: In order to improve the training speed and achieve a faster convergence, the model was being fed by multiple images. The number of images simultaneously fed to the model corresponds to the batch size. In this work, due to graphical memory limitations, only 2 images could be incorporated, hence the batch size is 2.

D. Post-Processing

After obtaining some results of the trained models, some post-processing techniques were proposed and applied. The first one was the closing operation [5] which is a morphological operation widely used in image processing. It corresponds to a dilation followed by an erosion operation. Closing is able to remove black pixels (class 0) from inside the segmented objects (class 1) whilst connecting 'cracks' in the segmented object.

Another post-process applied was called 'boundary tuning'. This process consisted in applying a machine learning classic classifier in the pixels that were in the boundaries of the cells in the output of the deep learning model. The classifier then predicts if the pixels from the boundary are really from the positive or negative class.

Two classifiers were trained, a Support Vector Machine (SVM) and a Bayes classifier. The SVM algorithm used had a polynomial boundary with degree 3 and it works by trying to increase the minimum distance between the data and the boundary. This classifier maps a certain input data to a certain class, it is a discriminant function. The Bayes classifier takes into account the distribution of the data and maps a class for a certain data point by considering the conditional probabilities of this point being part of each class, therefore it is a generative

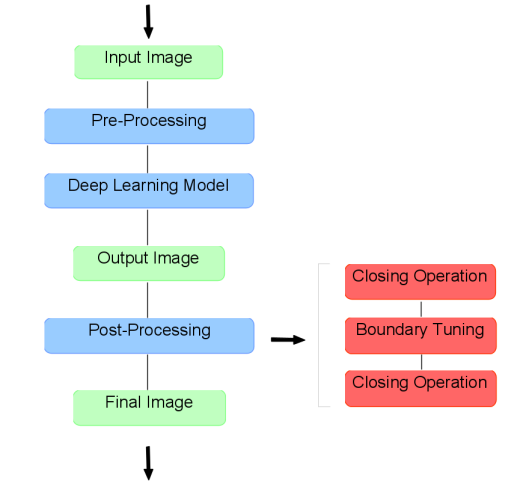


Fig. 10: Complete workflow with post-processing

model. The distribution of the data points was assumed to follow a gaussian distribution. Because our task is a binary classification, the classification rule used in the Bayes model is defined in equation 5.

$$Decide\ for\ class: \begin{cases} 1,\ if\ p(y=1|x) > p(y=0|x) \\ 0,\ otherwise \end{cases} \tag{5}$$

Being x a certain data point and y the class. The data used to train these classifiers had two features, one was the value of the pixel in the input image and the other one was the value of the same pixel after filtering the image with a sobel filter [10]. The dataset was constructed by means of 200 images from the Sartorius dataset. From each image we extracted features of 50 pixels lying in the cell boundaries, resulting in 10000 data points. In order to measure the quality of the model, a cross validation with random subsampling was implemented. 20% of the data points were used to validate the model in each of the 10 performed iterations.

The complete workflow of the image processing and segmentation when including post-processing is presented in figure 10. As presented in this image the order of post processing techniques applied is a closing operation followed by the boundary tuning with classifier and again a closing operation.

E. Model Validation

The metrics used to assess the performance of the models in the test dataset were the Dice coefficient, the Intersection over Union (IoU) and the accuracy which can be calculated as showed in the equations 6, 7 and 8.

$$Dice(A,B) = \frac{2 \times |A \cap B|}{|A| + |B|} \tag{6}$$

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{7}$$

$$Accuracy(A,B) = \frac{|A \cap B| + |(1-A) \cap (1-B)|}{|A \cup B| + |(1-A) \cap (1-B)|} \tag{8}$$

Being A the output image of the model and B the mask of the input of the model. The Dice coefficient gives us a metric which shows how good is the classification of the positive class (class 1) of our model. The IoU has the definition in its name, corresponding to the fraction of the output image segmented objects which correctly intersects the cells of the mask. The accuracy gives an overall measurement of the performance, that is, shows how well we have classified both the class 0 and 1.

III. Results and Discussion

A. Models Performance

During the training of the U-Net model as described in section II-C1, the loss was recorded. The values are presented in figure 11.

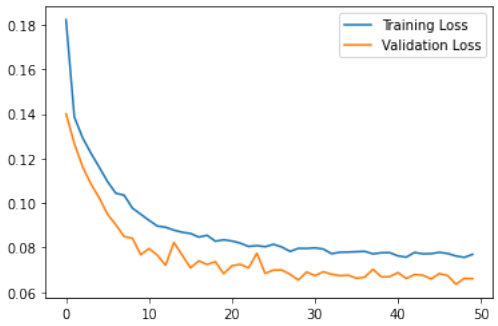


Fig. 11: Loss during U-Net model training

The figure 11 shows the common behavior of the loss during the training of all the models which were assessed. Firstly the loss rapidly decreases, then it continues to decrease but with a very low slope. In this case the validation loss was lower than the training loss, which is not common. The reason may be that when the validation set was randomly created, it created a set which our model has a good performance on it. If it were a different set, the validation loss could be higher than the training loss.

Furthermore, the observation of the behavior of the loss justifies the use of a negative exponential learning rate. Using a high learning rate at the start allows the model to rapidly adapt the random weights to a region of the loss hypersurface where the value of the loss is lower. After these initial steps, the reduction of the learning rate is crucial in order to search for narrow minimums.

Regarding the validation of the models (section II-E), the results for the IoU and Dice metrics are presented in table I.

TABLE I: Evaluation Metrics for Trained Models

Model	IoU	Dice	Accuracy
U-Net	0.528	0.068	0.94
Hot U-Net	0.522	0.067	0.94
Hot U-Net Weighted	0.583	0.092	0.942
Hot-Flow U-Net	0.22	0.04	0.889
Hot-Drop U-Net	0.098	0.014	0.879

Observing the table I, one easily can see that the Hot U-Net with a weighted loss matrix was the model with

best performance in terms of cell segmentation. This model presented the highest IoU value, which corresponds to a better intersection between the cells of the mask and the segmented cells of the model. This result can be justified by the presence of the higher weight for the false negative results, which will result in a increased reduction of this type of results, since it has a higher impact in the loss. Then, consequence of applying the weight is an increase of positive class pixels in the output of the model, which also increases the number of true positive values.

The better performance of the Hot U-Net weighted model cannot be caused by the addition of the one hot encoding data because the results presented by the U-Net model and the Hot U-Net model are very close to each other, thus the addition of the cell class information did not help the CNN architecture to better classify the pixels. Maybe the training of this model was not long enough in order to the CNN add the type of cell information as features in the hidden layers which may help the classification.

Despite the expectation, the CellPose model did not perform well. The output images, because they were trained to be grayscale, did not intersect well with the binary masks, even after applying a binarization with a threshold. Maybe it would be better if this model were pre-trained with a very large set of CellPose images, and then trained for this specific task. In that way, not only would it have a better understanding of the cells limits but also it would perform a better binarization.

The model with Dropout layers and Data Augmentation was the worst performer. What can be retrieved of this is that maybe the Dropout layers removed valuable information for the image binary classification.

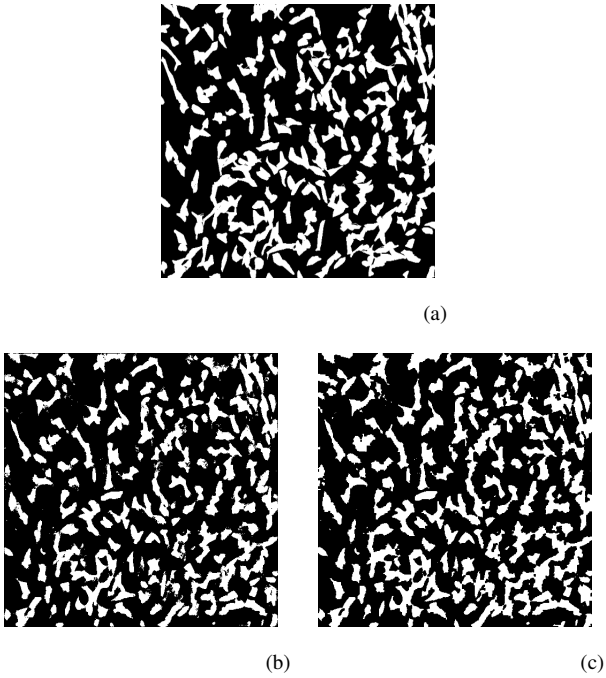
The first trained model (U-Net), without the addition of any information, had already given a good result, intersecting 52.8% of the mask. The evaluation of this model reinforces the robustness of the model and shows how powerful it is. U-Net is used in a lot of cell segmentation tasks, because the model has a good generalization, it can present an impressive performance for the segmentation of different cell types.

B. With Post-Processing vs Without Post Processing

Without applying any post processing technique, the output of the U-Net model for a test image (mask in figure 12a) can be visualized in the figure 12b. After applying the post-processing techniques (section II-D) the output obtained corresponds to the image presented in figure 12c. The value for the evaluation metrics are in table II.

The classifier used in the boundary tuning step was the SVM model, because it presented a higher accuracy in the validation set than the Bayes model. The accuracy for the SVM was 0.67 and for Bayes was 0.665.

The post process was applied in a single image and with a single model because of the elevated computational costs and time needed to run the algorithm when the close operation and boundary tuning steps were added. Although it increases the IoU by 4% and the Dice by 8% relatively to the U-Net without post-processing, the increased time to obtain the final segmented image shows that applying the closing operation and the boundary tuning is not good to segment a high amount of neuronal cell images.



(a) (b) (c)

Fig. 12: Mask (a) and output images of the U-Net Model without (b) and with (c) post-processing

TABLE II: Evaluation Metrics for U-Net model

Post-Processing	IoU	Dice
No	0.652	0.238
Yes	0.676	0.257

IV. Conclusion

In this work we have assessed the performance of different deep learning models in the task of segmentation of neuronal cell images provided by Sartorius [1]. The best model was the Hot U-Net Weighted, but in order to implement it in practice for automatically segmenting the images obtained by a biopharmaceutical enterprise, the metrics need to be improved. In conclusion, our goal of obtaining an excelent deep learning model was not reached, on the other hand the objective of testing different models was achieved and we can conclude that the U-Net is a powerful network for this type of task and its performance can be improved by addition of post image processing and other machine learning techniques, for example adding a weighted loss. For a future work we would like to optimize the post-processing techniques, because they had high computational cost which slowed the algorithm. Another model that can be assessed is the Mask R-CNN. This model works with bounding boxes for the objects which needed to be segmented (cells) and performs an instance segmentation for each cell. Finally, since we are dealing with images we could train the models using the Dice loss and check the evaluation metrics.

REFERENCES

[1] Kaggle. 2021. Sartorius - Cell Instance Segmentation. Retrieved February 3, 2022 from https://www.kaggle.com/c/sartorius-cell-instance-segmentation.

[2] Torch Contributors. 2019. PyTorch Documentation. Retrieved February 3, 2022 from https://pytorch.org/docs/stable/index.html.

[3] Gonzalez, R. C. & Woods, R. E. 2006. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., USA.

[4] Theodoridis, S. 2020. Machine Learning: A Bayesian and Optimization Perspective: Vol. 2nd edition. Academic Press.

[5] Sci-kit image development team. Module morphology - closing. Retrieved February 3, 2022 from https://scikit-image.org/docs/stable/api/skimage.morphology.html#skimage.morphology.closing.

[6] Contributors, T. (2022, February 1). PyTorch . Retrieved from Torch Contributors: http://pytorch.org/vision/main/generated/torchvision.transforms.functional.affine.html

[7] K, B. (2022, February 1). U-Net Architecture For Image Segmentation. Retrieved from PaperspaceBlog: https://blog.paperspace.com/unet-architecture-image-segmentation/?fbclid=IwAR23lR0XI5MEsG-KiqTnsRyA3JcbCCFhseeWCVrAXAMjSUjqi2uyMy-QwSg#tensorflow-implementation-of-u-net

[8] Ding, Yi & Chen, Fujuan & Zhao, Yang & Wu, Zhixing & Zhang, Chao & Wu, Dongyuan. (2019). A Stacked Multi-Connection Simple Reducing Net for Brain Tumor Segmentation. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2926448.

[9] Huang, H., et al. (2020, May). Unet 3+: A full-scale connected unet for medical image segmentation. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1055-1059). IEEE.

[10] Sci-kit image development team. Filters - Sobel. Retrieved February 4, 2022 from https://scikit-image.org/docs/stable/api/skimage.filters.html#skimage.filters.sobel.

[11] Stringer, C., Wang, T., Michaelos, M., & Pachitariu, M. (2020). Cellpose: a generalist algorithm for cellular segmentation.

[12] Sartorius - Cell Instance Segmentation — Kaggle. (2022). Retrieved 4 February 2022, from https://www.kaggle.com/c/sartorius-cell-instance-segmentation/discussion/297984