EPFL - Machine learning CS-433 Project 2 : Developing a ML pipeline to detect centrioles in human cells Under supervision of Gönczy Lab

Alexandre Bouillon SV-MA3
David Khatanassian GM-MA3
Grégoire Molas SV-MA3

***Abstract*—Calculation of the number of centrioles (Nc) per nucleus in fluorescence images is a tedious but necessary task to study cellular processes and division. It is currently achieved by manual calculation. A great and useful improvement is to develop an automated pipeline that is reliable and simple to understand. We developed a 3 block pipeline to compute Nc for typical laboratory images. The challenge is to reliably predict two kinds of objects : centrioles and nuclei. Thus, two separate blocks are dedicated to predict centrioles one hand and nucleus on the other. The last block is tasked to associated predictions to compute Nc.**

I. Introduction

The developmental biology field studies the fundamental cellular processes involved in organisms growth. An central mechanism of development is the cell division. It is crucial for a cell to maintain its number of centrioles (Nc), cylindrical organelles, as it could lead to division problems. They have been extensively studied through a combination of approaches. One of the characterization is the fact that the absence of certain proteins change the Nc. Yet, not all proteins involved have been described. Thus knowing the variation of Nc during division is a key step when analysing a protein of interest. Today, Nc is determined through manual calculation by looking directly at samples. It is a tedious task that is not reproducible, low throughput and biased because of human involvement. Automating this task could solve of these issues. We developed a machine learning pipeline to explore solutions to this automatization problem for the Gönczy lab[1] in the context of the CS-433 course. Our approach consists in using laboratory produced images in order to identify two kinds of objects : centrioles and nuclei. The first part of the pipeline is a neural network tasked with the detection of the centrioles. The second part, that detects nuclei, is a combination of an already existing neural network (namely *Sartdist*[2]) and another neural network. The third and last part is an algorithm that associate predictions of each previous block to compute the number of centrioles per nucleus in one sample.

A. The dataset

The dataset consists of fluorescence images of human cells of either centrioles or nuclei. They are saved as single channel 2048×2048 `.tif` images. Centrioles images are stained with different agents : such as CEP63, CP110, CEP 152 or GUT88. Nuclei images are only stained with DAPI. The dataset is splitted into three channels differentiated by the markers used to view centrioles. Each channel regroups 25 samples. The samples are subdivided into 4 images. One of them shows the nuclei, and the 3 others the centrioles marked with three different agents.

An additional `.csv` file listing centrioles positions in each images has been provided. Nothing is provided for the nuclei except the 75 DAPI images, whose 25 have been wrongly saved (CEP152+...). Although they are usable, major quality has been lost. Hence, the DAPI dataset is rather poor.

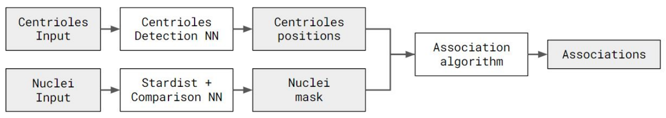

Fig. 1: Global pipeline for the computation of Nc per nucleus

Centrioles input corresponds to a 2048×2048 `.tif` stained image. Nuclei input corresponds to the same image format but stained to show nucleus.

II. Centrioles detection

The first sub-task we defined is centrioles detection in single channel images. In those images centrioles takes the form of bright luminescent spots that vary in size and shape. Centrioles can be close from each other in some cases (less than 10 pixels apart). Images can also contain noise due to the staining agents used. Additionally, some samples were not perfectly captured resulting in high brightness contrast between images. It might be caused by the staining agent or other parameters (microscope focal point, intensity adjustments etc). These challenges cannot be easily using traditional machine learning and image processing techniques such as filtering with a threshold. Thus, we decided to use a well known approach in the computer vision and image processing, that is neural networks.

Our problem can be summarized as circular spot detection with varying intensity, size and background noise, which makes neural networks a particularly good solution for their adaptability to different samples without having to define precise features.

A. Model

We designed our neural network based on a pixelwise segmentation approach. The idea is to classify each pixel of the image as part of two classes : 1, 0 *"is centriole"* or *"is background"*, and then to extract centrioles' positions from the generated map.

The neural network is given as an input a two channels 14×14 pixels patch cut around a pixel of interest, and the Sobel treated patch. It gives neighbourhood and edge information about the central pixel. In the case of border, we decided to apply a zero padding method.

As seen on Fig. 2, the neural network structure is the following : two convolution layers (with kernel sizes = 2), associated with max pooling and ReLU. It is supposed to extract features from the input. After that, the information is flatten into a single 1D vector and passed to multiples linear layers associated with ReLU. Finally, the information is concentrated into a single point by the output layer and passed into a sigmoid function, to obtain a binary classification as the final output.

For each candidate, the neural network will produce a prediction between 0 and 1 that is stored in a map accordingly to the pixel position. This score represents the probability of a pixel to be part of a centriole. We then filter out lower values using a simple threshold. It is defined based on the training set precision, f1 and recall scores. It can be set to reach our goal (maximizing precision, recall or both), assuming that the training set is representative for centrioles fluorescence images. The resulting map contains non zero blobs corresponding to centrioles. To differentiate close centrioles within a blob, we use local maxima detection on the prediction map. After that, we associate maximum scores and their position as detected centrioles.

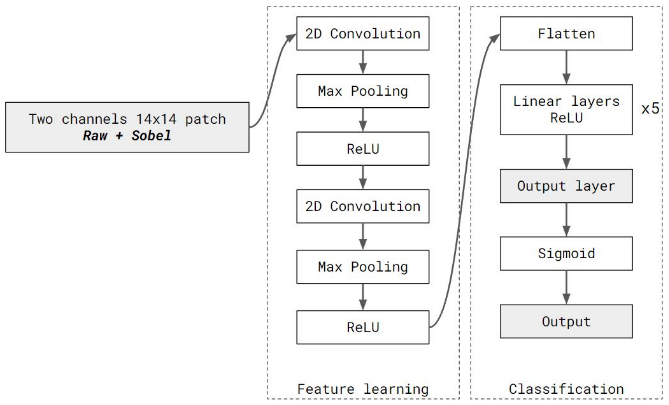

Fig. 2: Centrioles detection NeuralNet structure

1) Loss function & class imbalance: We used the Binary Cross Entropy loss (BCE) as the loss function for our model. It is a well suited loss function for binary classification problem and regular loss function for image segmentation problems.

Another issue faced was the overwhelming number of negative examples used to train the neural network. As centrioles are small spots in an image, it leads to heavy class imbalance. To solve this issue, we added weights to each class equal to the candidate pixel class ratio. It increases the network capacity to predict centrioles.

2) Hyper-parameters tuning: We realized 5-fold Cross Validation (CV) to set model hyper-parameters. For reduced computation time, CV has been done on an random subsample of 5 images. The hyper-parameters we tuned are the following : of epochs at 5, learning rate of BCE at $1e^{-3}$.(Appendix AA)

We have also realised the CV on the number of hidden layers and dimensions of hidden layers of our model. As it does not impact significantly performances, we decided to keep 5 hidden layers of dimension 168.(Appendix A)

3) Evaluation of precision & recall: Precision and recall are estimated by doing a linear assignment between predictions and annotated centrioles. It consists of associating one annotations to the closest prediction while minimizing the annotation-prediction overall.

B. Feature processing & engineering

1) Normalization: To have a similar data treatment and inputs for the neural network over all images, we normalize the images between 0 and 1 using the minimum-maximum method.

Due to memory limitation, we preferred to proceed one image at the time for training our model.

2) Use of Sobel filter: We found during our research that feeding the network with the sobel filtered image as well as the raw image lead to increase in geometric pattern learning instead of the intensity pattern. And it performed better using this supplementary information.

3) Foreground selection: Pixel-wise segmentation requires computation time and memory space as it looks at one pixel at a time. To quicken the process, we opted to select pixels that could be candidates for being centrioles. It is achieved through a foreground selection by keeping only the most bright pixels. We suppose that centrioles have the highest intensity in fluorescence images. A general threshold is not relevant because of sample variation, so we keep a certain percentage of the brightest pixels in an image. From our observations, we can remove a significant proportion of background pixels (99%) and remain generous on the zones were centrioles are located at the same time.

4) Ground truth definition: We need to define a less restrictive ground truth for the model to segment an image as using a single positive pixel is not enough for training. To generate it, we iterate over the annotated centriole positions and cut a large patch (28x28) around them. The brightest pixels are filtered and placed as ones in a zeros image at the centrioles' position. The threshold is defined based on the intensity distribution of the centrioles. Additionally, the position annotated is always placed in this map by default to avoid any problem.

5) Data augmentation: To augment the number of samples and avoid a possible asymmetry, we also use rotation. By rotating the samples by 90°, 180°, and 270°, it varies the pattern recognition capability of the model.

C. Results

We only tested a few models only on one staining at a time. We tested our procedure on CEP63, CP110 and GTU88. We selected a threshold value for each based on its best F1 score on the training set.(Appendix A) As expected, when the threshold is restrictive (low values), the precision is maximized and the recall is weaker. It can be seen in supplementary Fig. 7, 9 & 11.

Table I (and Supplementary Fig. 8, 10 & 12) shows that threshold selection based on training set performances is relevant as we do not see any major difference in F1 score.

Marker used	Train F1	Test F1	Threshold
CEP63	0.94	0.91	0.09
CP110	0.89	0.92	0.30
GUT88	0.91	0.94	0.27

TABLE I: Results of models trained on different markers

III. Nuclei detection

Nuclei detection is a redundant task in life science. It exists several models, which can be obtained pre-trained. The Gönzy lab oriented us toward Stardist,[2] a U-net based neural network, localizing cell nuclei via star-convex polygons.

A. Adaptation to the model

Several pre-training are available. The one that most fits our dataset is 2D_versatile_fluo. It has been trained over about 500 images. As we have only 75 images, it was not possible to train this model with our dataset. On the training dataset, a nucleus is about 50 pixel large. To fit this scale, we first tried to reduced our input from 2048x2048 to 128x128, but this led to poor image quality. We then chose 256x256, which works fine.

B. Model limitations

Stardist outputs a mask representing each nuclei together with their confidence score and the (x, y) coordinates. They are represented by the dots on figure 3, displayed as green if $p \geq 0.75$, red otherwise. As we can see, it works fine but not perfectly. As the aim of this project is to consider automated detection and association between centrioles and nuclei, wrong nuclei detection is a major issue. We highlight some of the case we would like to avoid in figure 3. *a)* The shape of nucleus 1. is similar to the composed shape 2.. However, in both cases it predicts 2 nucleus. *b)* The mask of nucleus 3. does not represent correctly the nucleus. This might be an issue since we assume that the centrioles are generally close to their nucleus. *c)* The nucleus 4. should be labeled as uncertain. *d)* Nucleus half on the picture are labeled as good, but their centrioles might not be on the image. Furthermore, some images are not as clean as this one, which leads to obvious false positive.

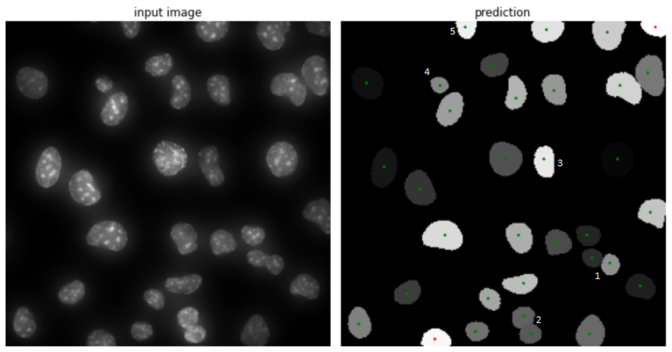

Fig. 3: Stardist prediction on RPE1wt_CEP63+CETN2+PCNT_1_000_000_max_C0.tif

C. Output classifier

To cope with those issues, we built a simple model that compares each identified nucleus to its mask and outputs *"0: mainly out of the image"*, *"1: good prediction"*, *"2: requires human intervention"*.

1) Input preprocessing: The network is fed with two 64x64 patches centered on nucleus coordinates (outlined in fig. 4). The first patch comes from the raw image and the second from the outputted mask. To help the network focusing on the centered nucleus, we collected the outputs of 3 inputs given to Stardist: [sobel, sobel+raw, raw], stacked them (A), and merged them (B) to form an augmented mask. We then filter this mask by keeping only the central blob and those connected to it (D). By doing so, the region of interest becomes larger and we can dim everything around (E). The second patch is the outputted mask from the raw input for the nucleus of interest.

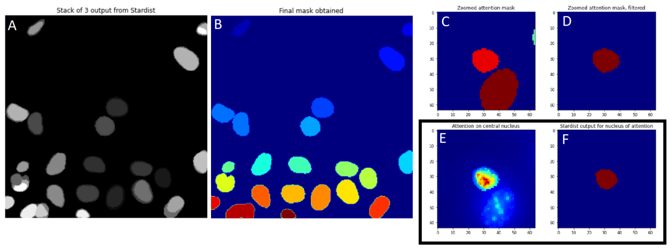
Stack of 3 output from Stardist
Final mask obtained
Zoomed attention mask
Zoomed attention mask, filtered
Attention on central nucleus
Stardist output for nucleus of attention
A
B
C
D
E
F

Fig. 4: Preprocessing of classifier input

2) Data augmentation: As most of the nuclei are *good*, the classes are heavily unbalanced (541 vs 31). We augmented the weaker classes by flipping/rotating/stretching them. We augment between 4 and 16 times some data

3) Classifier structure: The figure 5 shows the structure of the network. The structure has been adapted from a network seen in a previous course (EE-559). A Softmax activation function is chosen together with a CrossEntropyLoss. The output can be then interpreted as probability.

4) Training / testing: We trained our model with the data from channel CEP152 and CP110 and trained on the channel CEP63, which is approximattely a 70/30 split

5) Results: 87% of the output is above a confidence 0.85 score. In this output, 87.8% is classified successfully. This could be further improved by using auxiliary loss for example, but also by having more data.

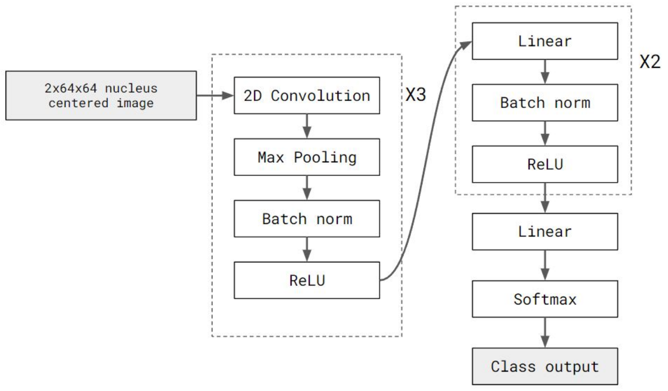

Fig. 5: Nucleus classifier NeuralNet structure

We show here the results of our classifier on the previously shown sample. The targets are displayed as follow: Border, Good, Requires intervention. The nuclei 2,3,5 provides satisfying results. Most of the ”Border” class are classified successfully, as well as for the ”Good” class. The nuclei 1 and 4 don’t deserve a ”Require intervention” but it is not an issue to have them in this class. Furthermore, their shape are off the norm, which justify this misclassification. The nucleus pointed with the red arrow is has a shape and size that could lead to a ”Requires intervention”, but our network tends to classify to ”Border” too much. This comes from the preprocessing that dim heavily everything out from the region of interest. This creates net cuts, which are interpreted as border.

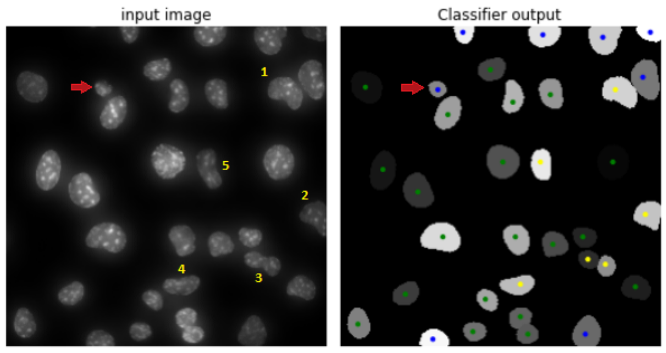

Fig. 6: Classifier prediction on RPE1wt_CEP63+CETN2+PCNT_1_000_000_max_C0.tif

IV. Centrioles nuclei association task

The last part of our pipeline is associating centrioles and nucleus prediction. In order to keep the pipeline simple, we designed a simple association algorithm that searches the closest nucleus in the nucleus mask obtained in the Nuclei detection block. It looks in a square window around the centriole of interest. The window size can be adjusted but it has been found that 97%(Supplementary Fig. 29) of the annotated centrioles are within 36 pixels to their nucleus. Thus the default window size is a 12 pixels width. Our aim was to maintain complexity low after the previous steps. It was important for us to understand the liabilities of our approach.

A. Output structure

The output format of the pipeline is a `.csv` file containing all predicted centrioles with the image name they come from and many information concerning the associated nucleus. Notably, the nucleus id, class, position and scores of both Stardist and classifier network.

V. Discussions

Neural Networks should not rely on features such as filtered images as they can build similar filters internally. Using Sobel filtered image as additional input in the centrioles NeuralNet leads it to not only consider intensity as an important feature but geometry as well.

We introduced an arbitrary threshold to deal with the centrioles NN output. We hypothesized that, based on training set scores, it would perform similarly on testing set and future samples. It is to be noted that this threshold can be set in a way to maximize precision or recall.

We have only performed cross validation on a 5 images subset. Thus, results obtained may not represent a large scale behaviour of the model.

The aim of this project was to develop a general tool for centriole detection. Then we tried to create a simple tool specific to this task, and handled easily . We decided to keep exploring this network architecture as no other architecture seemed particularly fitted for our problem. In our opinion, the nucleus detection classifier could perform better if supplemented with more training data, due to large class imbalance largely.

Unfortunately, we lacked any real baseline model to compare our approach to. But we had a goal of 90% precision to reach for centrioles detection if possible. Furthermore, it is the purpose of this paper to study the possibility of an automatization process.

VI. Conclusions

The combination of existing model together with simple model can already lead to satisfying results. A important part is to handle correctly the data and preprocess it with care. By doing so our centrioles detection block achieves a high level of precision while staying in the lower part of the complexity spectrum.

References

[1] Gonczy Lab Cell and Developemental Biology.
[2] U. Schmidt, M. Weigert, C. Broaddus, and G. Myers, “Cell detection with star-convex polygons,” in *Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II*, 2018, pp. 265–273.

Appendix

Part A : Centrioles detection

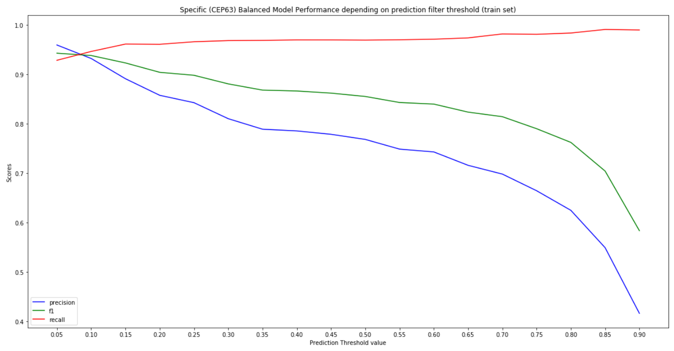

Fig. 7: CEP63 performances depending on prediction filter threshold on train set

On the train set, CEP63 model shows very good recall score always above 0.9. Precision and f1 score are low for high values as expected, and tend to increase the lower threshold is. Around 0.1, we observe that the three scores converge and that there is a trade off between recall and precision and f1 scores.

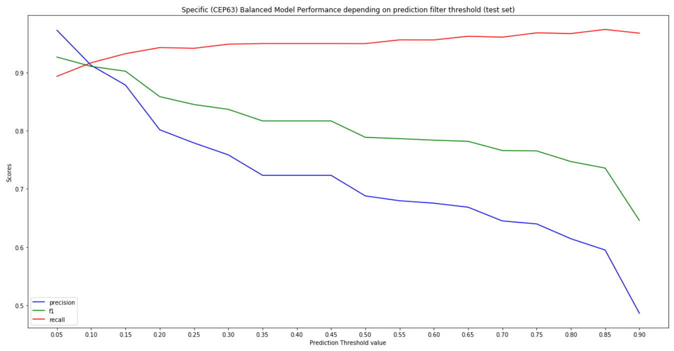

Fig. 8: CEP63 performances depending on prediction filter threshold on test set

Overall, the test set seems to have a similar behavior as the train set, but with slightly different score values. This might also be explain by the size of the sets. On the test set, CEP63 model shows very good recall score always above 0.9, except at very low values. Precision and f1 score are low for high values as expected, and tend to increase the lower threshold is. Around 0.1, we observe that the three scores converge and that there is a trade off between recall and precision and f1 scores.

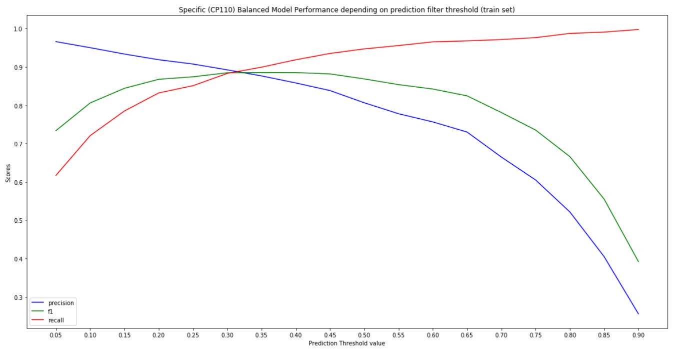

Fig. 9: CP110 performances depending on prediction filter threshold on train set

On the train set, CP110 model shows a good recall score for high threshold, but this score tend to decrease quickly with lower values (under 0.9 at 0.3 value). The precision has a surprising bell curve shape, with the best values around 0.3. The f1 score only increase with lower threhsold. This behaviors of the scores can be explain by the presence of artifacts that are detected as centrioles at lower values. the scores converge at 0.3 where there is a trade off between the scores.

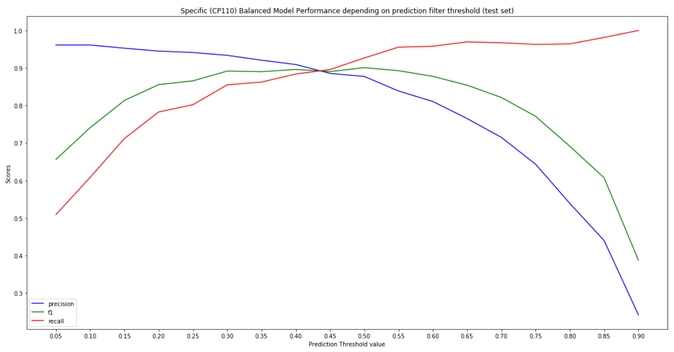

Fig. 10: CP110 performances depending on prediction filter threshold on test set

On the test set, CP110 model shows a similar behavior as the train set. However, the peak of precision is more around 0.5 here.

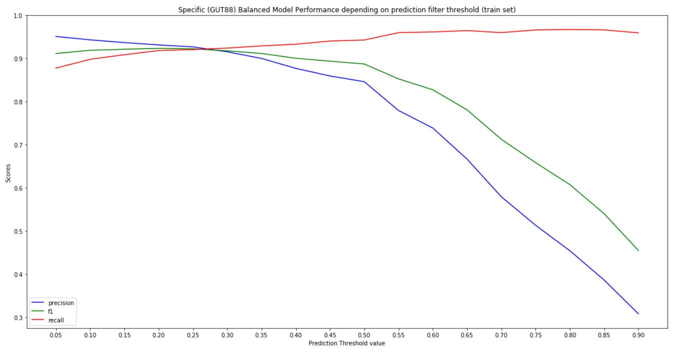

Fig. 11: GTU88 performances depending on prediction filter threshold on train set

On the train set, the GTU88 model shows good performance for the recall that seems quite stable. The recall goes under 0.9 only for the lowest values. the f1 and precision scores are low for high thresholds, and increases with lower values as expected. the scores converge at 0.27 where there is a trade off between the scores.

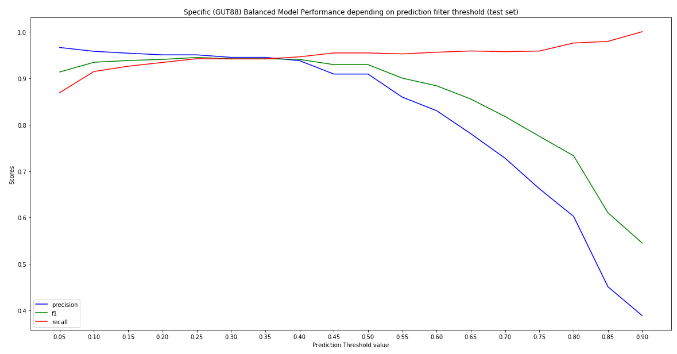

Fig. 12: GTU88 performances depending on prediction filter threshold on test set

On the test set, GTU88 model shows a similar behavior as the train set. However, the peak of precision is more around 0.35 here. Those results shows that GTU88 would be good to use as the score change slowly around the trade off regions.

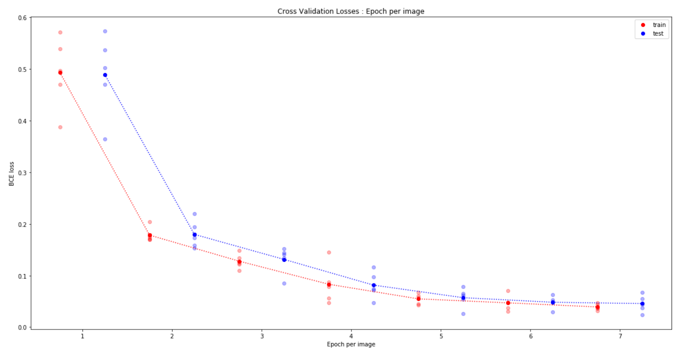

Fig. 13: Cross Validation (5 folds) Losses over the number of epoch

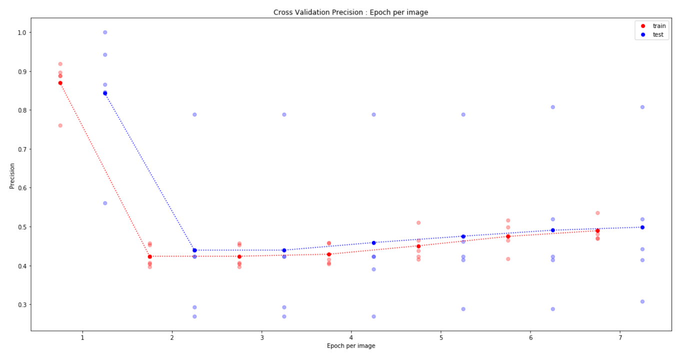

Fig. 14: Cross Validation (5 folds) Precision over the number of epoch

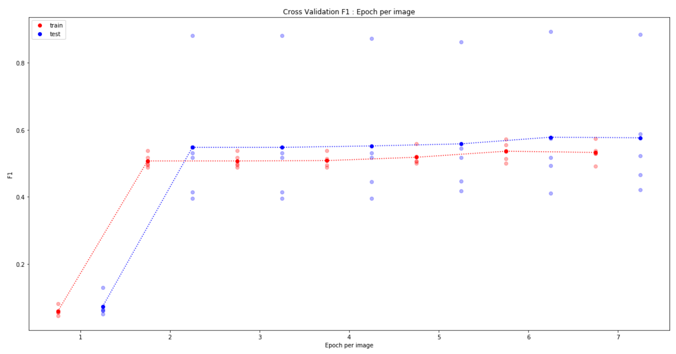

Fig. 15: Cross Validation (5 folds) f1 over the number of epoch

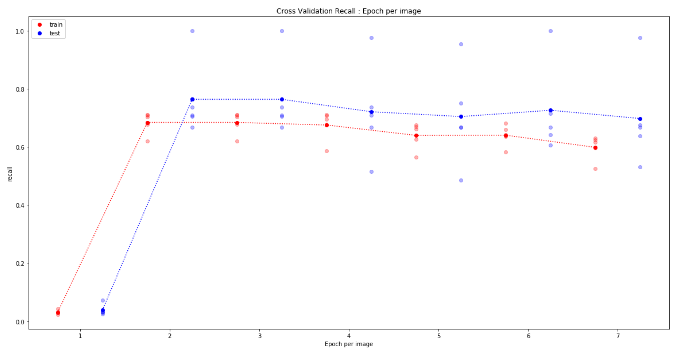

Fig. 16: Cross Validation (5 folds) Recall over the number of epoch

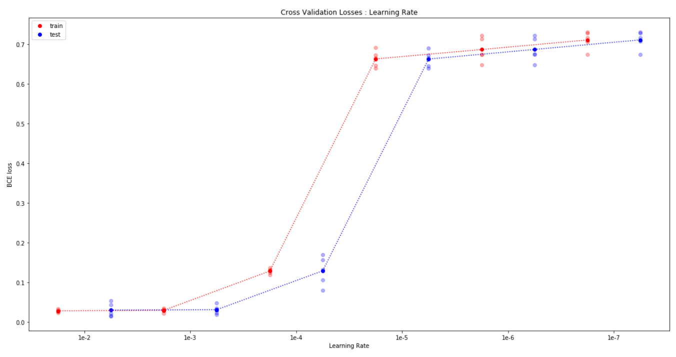

Fig. 17: Cross Validation (5 folds) Losses over the Learning rate

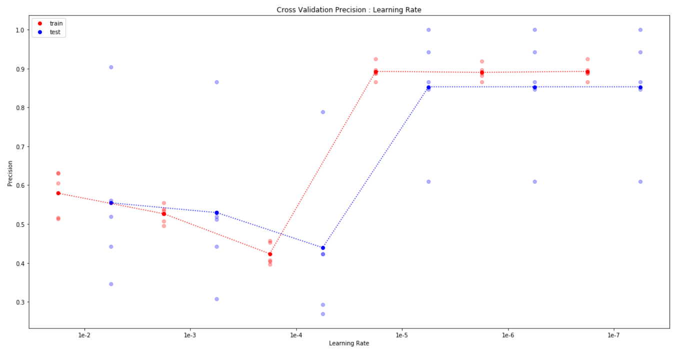

Fig. 18: Cross Validation (5 folds) Precision over the Learning rate

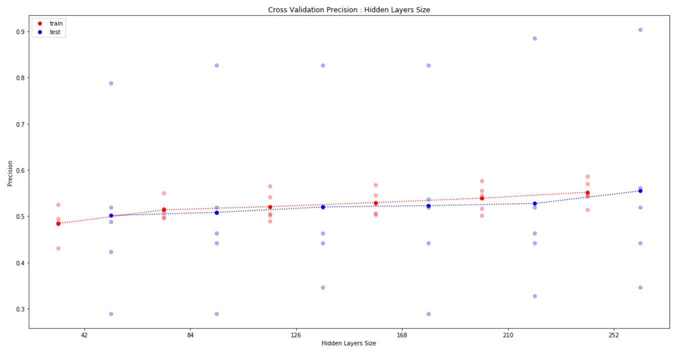

Fig. 22: Cross Validation (5 folds) Precision over the hidden layers size

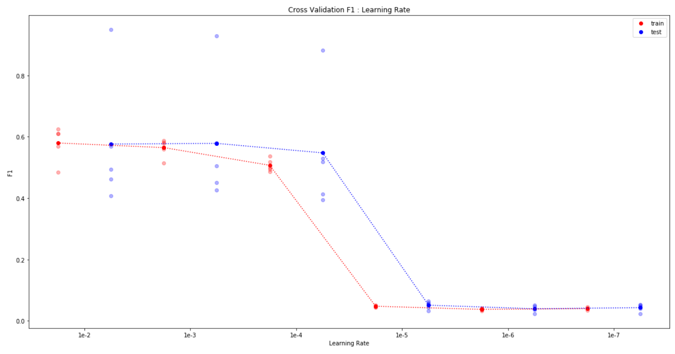

Fig. 19: Cross Validation (5 folds) f1 over the Learning rate

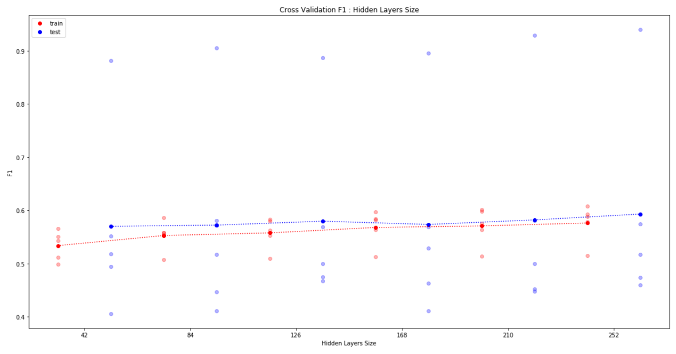

Fig. 23: Cross Validation (5 folds) f1 over hidden layers size

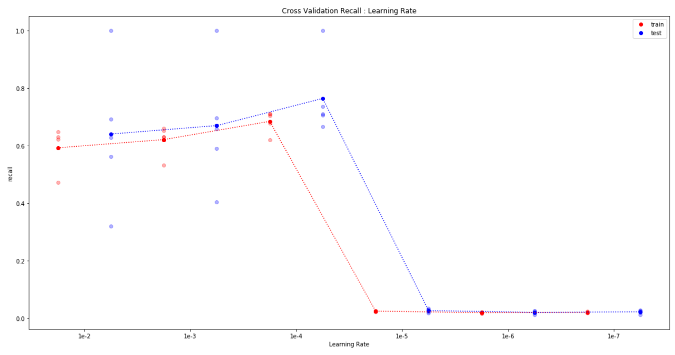

Fig. 20: Cross Validation (5 folds) Recall over the Learning rate

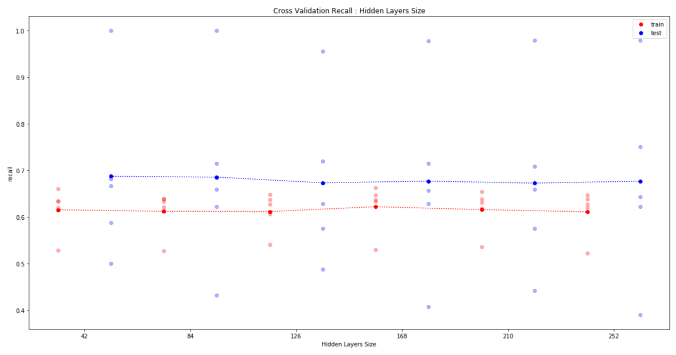

Fig. 24: Cross Validation (5 folds) Recall over the hidden layers size

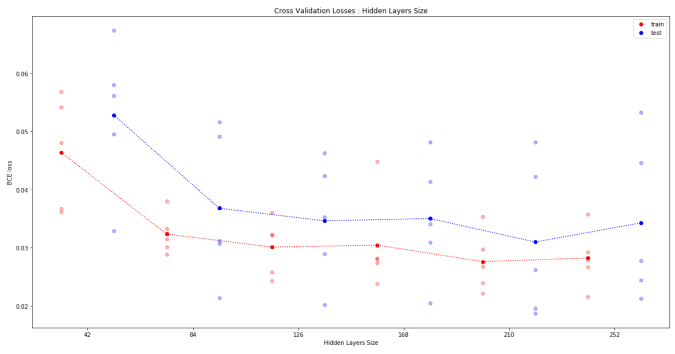

Fig. 21: Cross Validation (5 folds) Losses over the hidden layers size

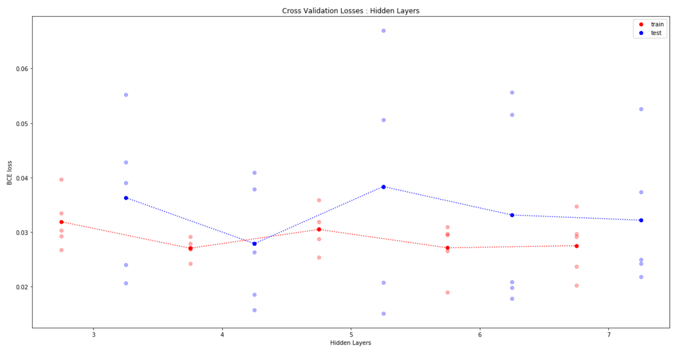

Fig. 25: Cross Validation (5 folds) Losses over the hidden layers number

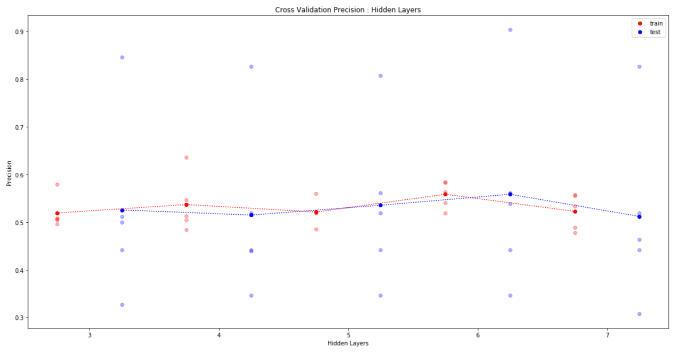

Fig. 26: Cross Validation (5 folds) Precision over the hidden layers number

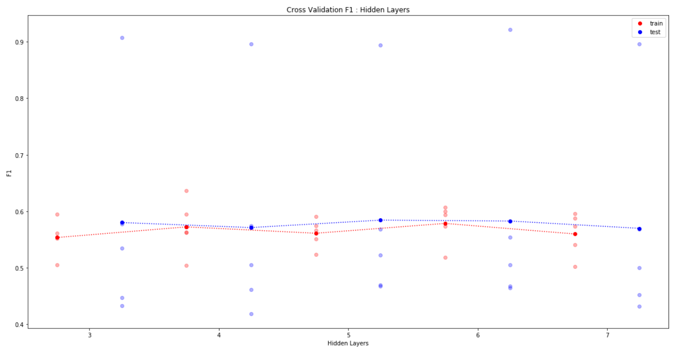

Fig. 27: Cross Validation (5 folds) f1 over hidden layers number

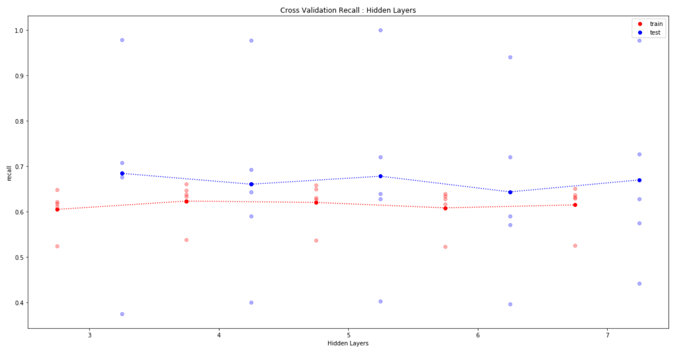

Fig. 28: Cross Validation (5 folds) Recall over the hidden layers number

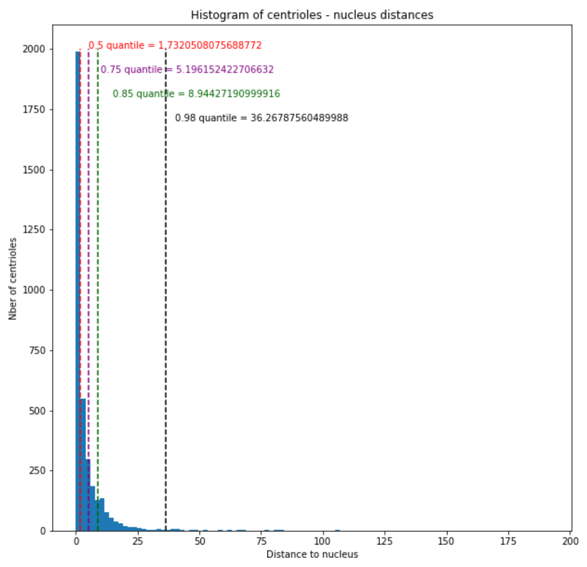

Fig. 29: Histogram distribution of the centrioles depending on their distance to the closest nucleus

0.5, 0.75, 0.85 and 0.98 quantile appears as different colored doted lines

Part B : Nuclei detection Part C : Centrioles - Nucleus assignement