Capillary detector

ELEN0016-2 : Computer Vision

François Lievens, Julien Hubar, Matthias Pirlet

December 2020

https://github.com/francoislievens/ELEN0016-2-CV-Capillary-Detector

GitHub

Abstract

In this project we developed an architecture to count moving objects by analysing a video by combining computer vision methods and deep learning methods. The detection of the droplets was done by a background subtraction method using mixture of Gaussian (MOG) coupled with simple mathematical peaks detection. The link between object identity across different frames was done using euclidean distances threshold over its positions. The detection and counting of cells inside different droplets was done using cropped images over only one image of each detected droplet provided by the first part of the model. A U-Net deep convolutional network is used over this droplet images to generate a mask which contains peaks at each detected cell positions. A peak detector can then be used again to detect cell's positions. This combination of classical computer vision and the use of deep learning methods over the most interesting parts of frames permit to go beyond the 500 frames per second while keeping a good accuracy and precision.

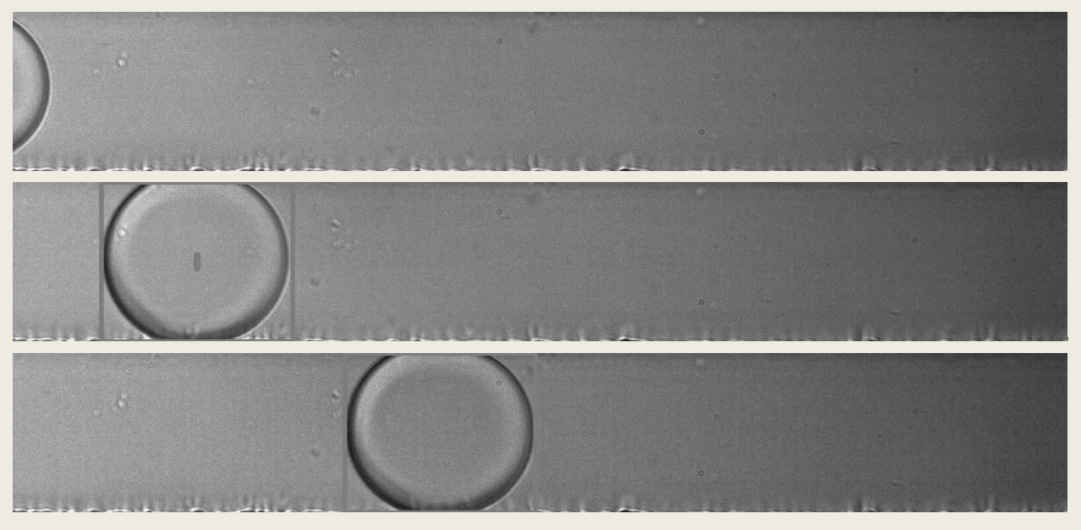
Objective

The goal of the project is to produce an object detection algorithm that can work in real time on a high frequency stream of images. This images are taken from a microscopic camera and represent liquid droplets who may content biological cells inside. As output, the model will provide:

- The total number of frames in the video
- The number of detected droplets + coordinates
- The number of detected cells + coordinates
- The histogram of the number of cells per droplets

The input frames have a dimensions of 1600 x 240 and the real time frame rate is considered as 300 frames per seconds.

Droplet Counting and selection



Width < 250: not considered

Firstly fully detected: troplet count increment.

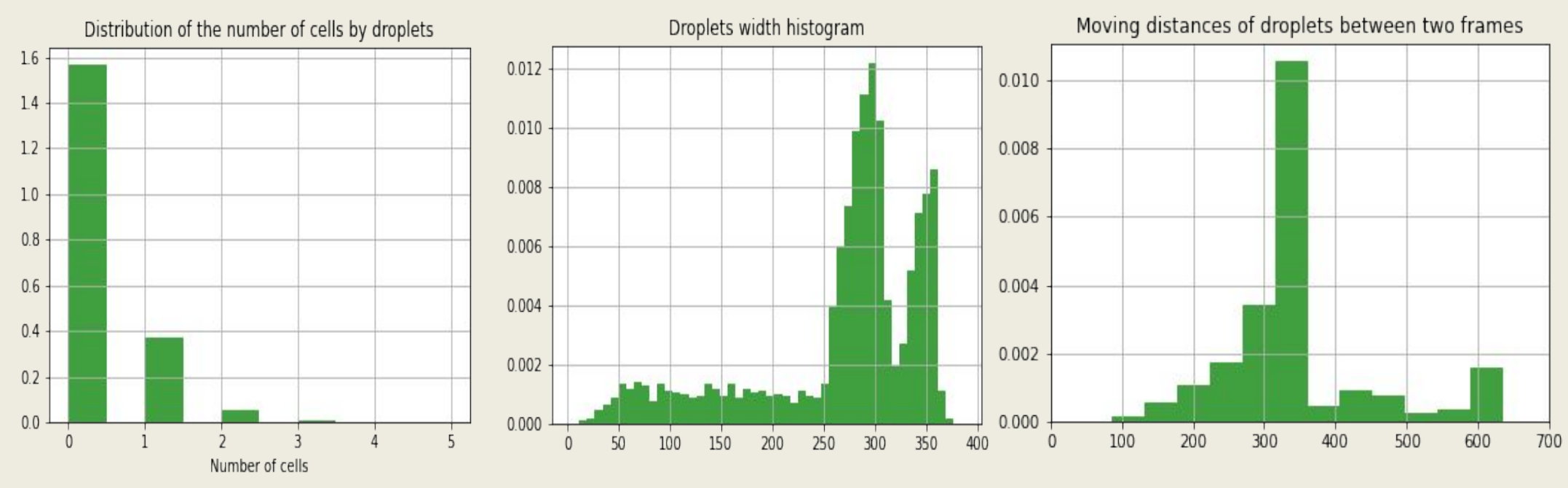
Secondly fully detected: Used as UNet input

From the cell detection results, we only consider droplets that fully appear for the second time in the video. By this way we guarantee to crop a droplet image of the best quality.

The content of the bounding box is cropped from the original frame, resized to 240 x 240 px and is feed into the UNet part.

In the droplet counting point of view, a droplet is counted

Statistical Analysis of the data

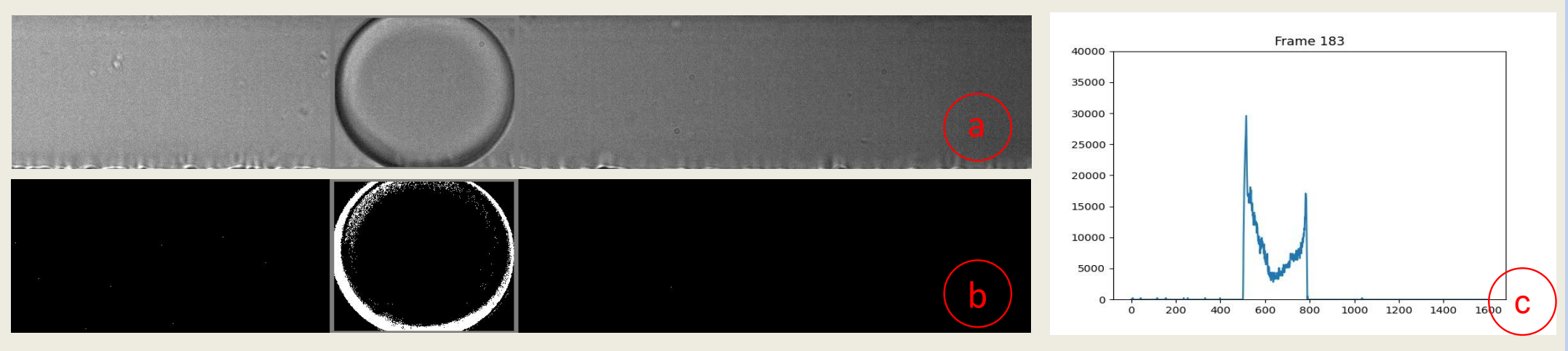


At the beginning, a first statistical analysis of our data was done in order to determine the different thresholds used in the algorithm.

- A distance < 750px for droplets between two successive frames are considerate as the same droplet
- A droplet of width < 250px is considered as a part of droplet and is therefore not considered

Droplets detection

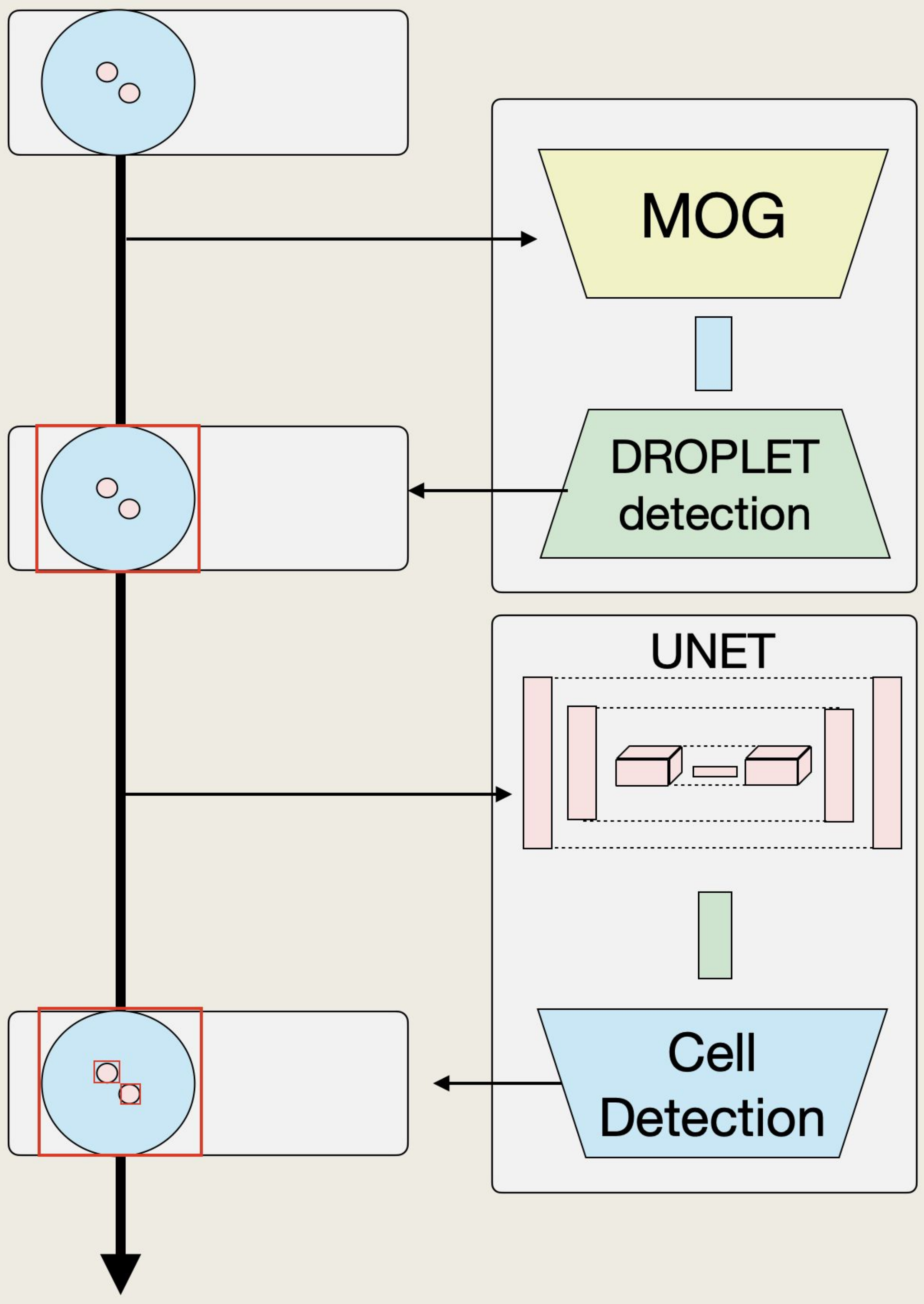
A mask is generated by using a mixture of Gaussian (MOG) implementation from OpenCV library.



In the figure (a.) and (b.) are show respectively the original frame and the corresponding mask from the MOG filter. In the figure (c.) is shown the sum of the mask values over pixels of the columns. From this values, the algorithm will travel the x axis and uses a specific threshold of 1000 to detect if it enters a droplet, and a threshold of 300 to detect if it goes out of a droplet. This last one is performed on an average of the next 20 values to avoid false exit of droplet detections.

Bounding boxes resulting from this process are draw in the figure (a.) and (b.).

Global Model

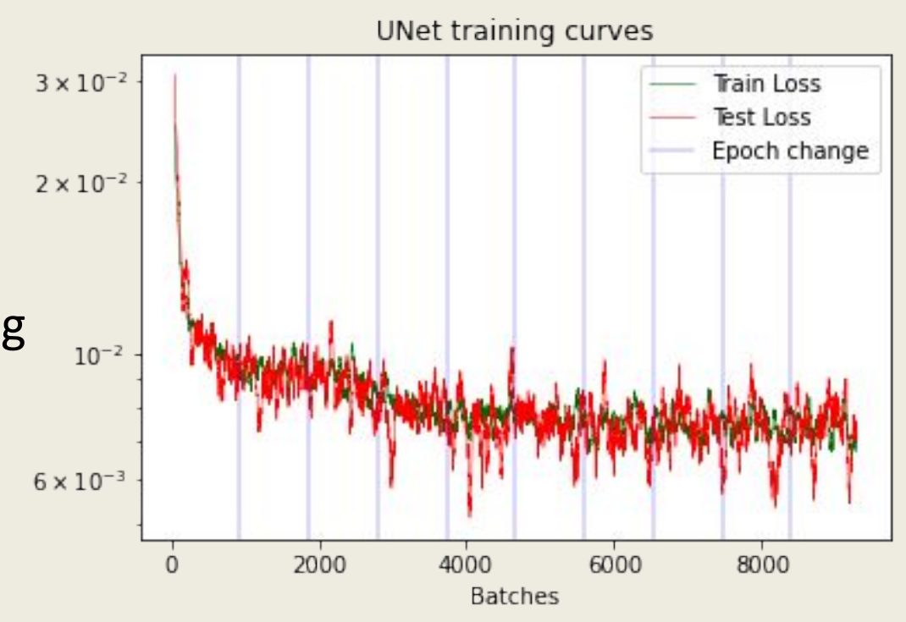


Deep Learning Part: UNet

Training data:

Our dataset consists of 2916 droplets cropped images coming from the provided dataset annotated by students on the cytomine platform. These images are resized to 240 x 240 px and the target masks are generated using matrices where we have zeros and 2D Gaussian distributions centered on each cell coordinates (heat map) such that the sum of values on the image sum to 100 * the number of cells.

UNet training:



- 80% Data for training
- 20% Data for testing
- 10 epochs
- Data balancing by over-weighting images with cells
- Learning rate: 1*10e-5
- Loss: Mean Squared Error (MSE)
- Optimizer: Adam
- Data augmentation: random crop, rotation, translation, distorsion

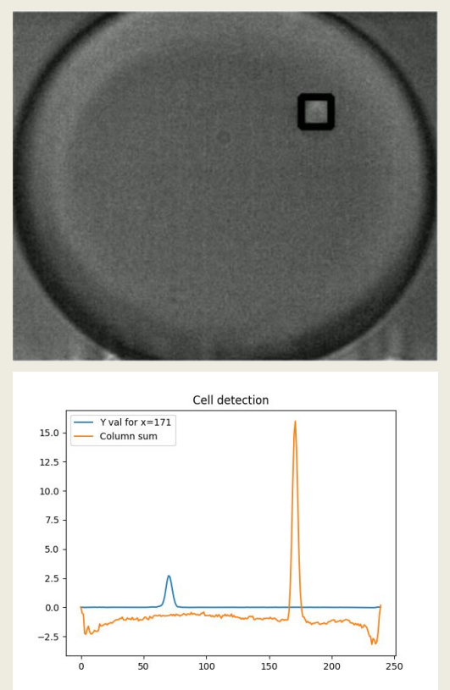
UNet Architecture:

- Downsampling: 3 x Convolutional block
- UpSampling: 3 x Convolutional upsampling bloc
- Skip Connection are added between each block of same size

Cells Counting

Cells detection is done thanks to peak detection on the masks provided by UNet predictions.

- A first peak detection on the sum of columns values
- A second peak detection on the values of the columns detected during by the previous step.



Results

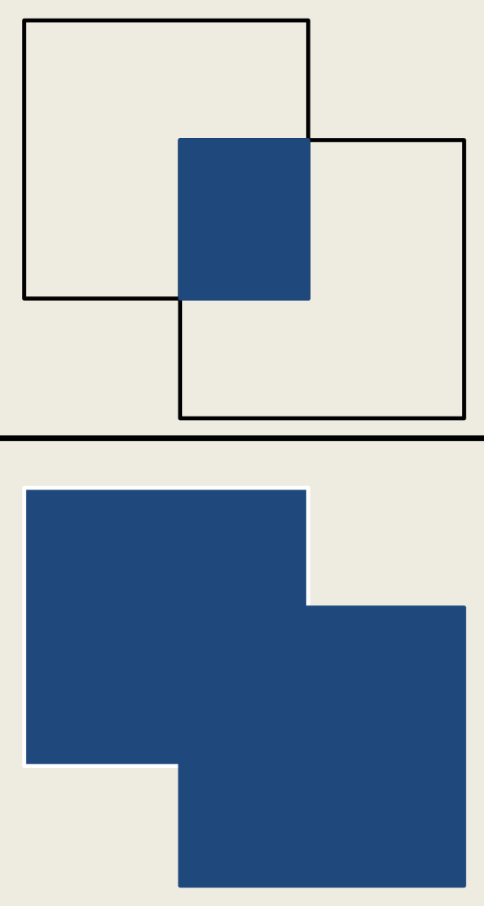
Comparison metrics used

- Confusion Matrix :

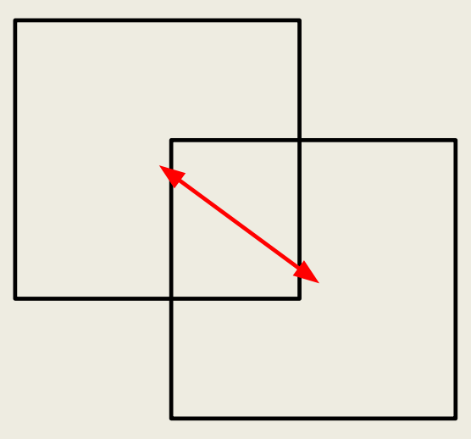
		Predicted class	
		Positive	Negative
Actual class	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

- Intersection Over Union (IoU) :

$$\text{IoU} = \frac{Area\ of\ overlap}{Area\ of\ union}$$



- Euclidean distance :

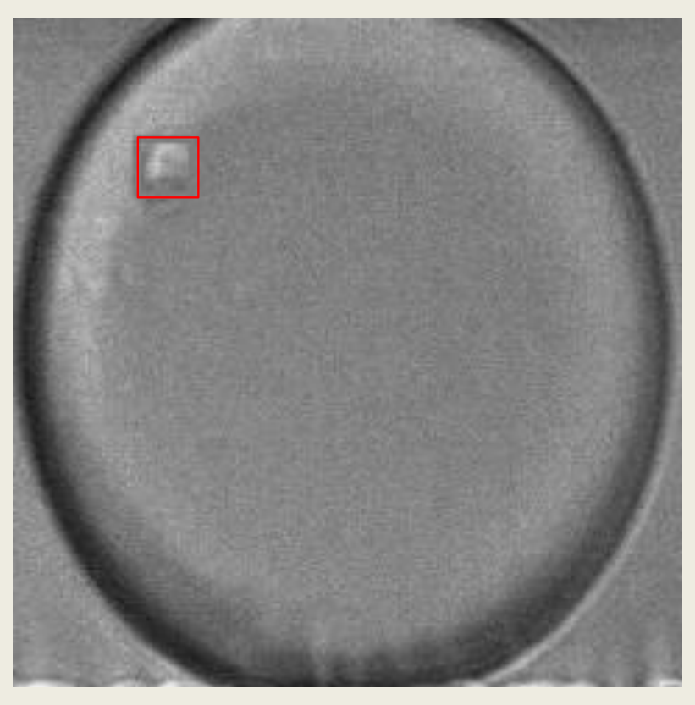


$$\text{Euclidean distance : } \sqrt{\left(\text{center\_x}_1 - \text{center\_x}_2\right)^2 + \left(\text{center\_y}_1 - \text{center\_y}_2\right)^2}$$

Comparison of our results

Comparison of the outputs of the two algorithms and the annotations provided on the cytomine platform :

Droplet		Predicted class	
		Positive	Negative
Actual class	Positive	2850	66
	Negative	285	/



Mean IoU : 0.95

Cells		Predicted class	
		Positive	Negative
Actual class	Positive	629	64
	Negative	55	/

Mean Euclidean distance : 4.7223

Mean IOU : 0.5587

Final results

Counted droplet : 3353

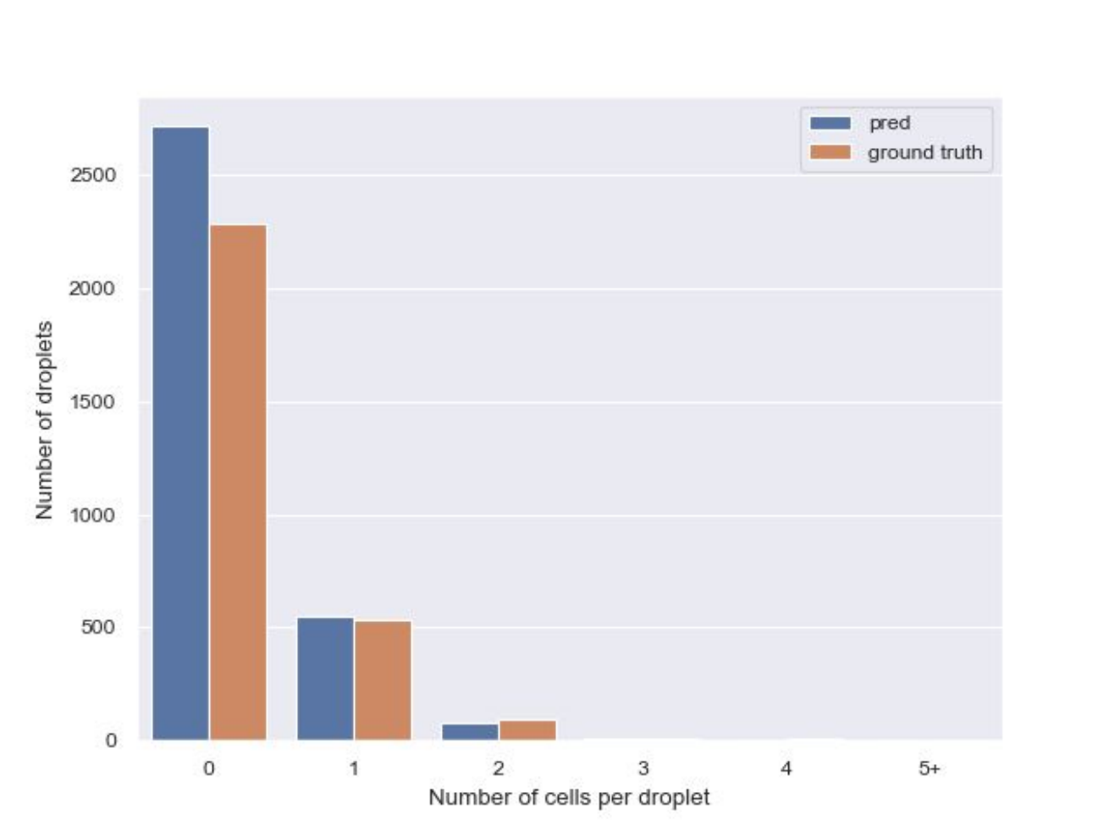
Counted cells : 737

Computing time (seconds) : 49.8275

Read frames : 24952

Frames per second :

- **500.7673:** Execution into the Jupyter notebook
- **545,9852**: Execution of *main.py* in console
- Reference: AMD Ryzen 7300X, Nvidia GTX1070)



Nb Cells	Ground Truth	Predictions
0	2287	2714
1	533	551
2	95	76
3	12	11
4	7	1
5 and more	2	0

Results from the 2 challenge trials :

	Jensen-Shannon distance	Manhattan distance	Euclidean distance
1st trial	0.016238127711396932	3.0	2.23606797749979
2nd trial	0.0064719438914195085	2.0	1.4142135623730951

Conclusion

This work provides an efficient way to count elements in capillary videos using the neural network only on sub-frames of interest. The deep learning approach seems to be necessary to obtain a good accuracy to detect cells with changing shape, can be superposed by other or confused with impurities. Despite a Python implementation, our code was optimized to parallelize operations by separating the different steps into 6 different threads and pre-compiling slow operations like detection loops before the execution. This implementation requires a lot of thresholds to choose. This choice can probably be improved to increase the accuracy. The precision of the UNet predictions also show a small lack of precision when the number of cells is high due to the small quantity of training data.