

# Transfer Learning for classification of subcellular mRNAs localization patterns

Remy Dubois

CBIO - Mines Paristech, Institut Curie.

This manuscript is of public access.

April 2017 - Aug 2017

# Contents

<b>1</b>	<b>Biological and Mathematical context</b>	<b>1</b>
1.1	Biological context . . . . .	1
1.1.1	Gene Expression . . . . .	1
1.1.2	Expression patterns . . . . .	2
1.1.3	Place of this project . . . . .	4
1.2	From Microscopy to gene mapping . . . . .	4
1.2.1	Analysis of smFISH data . . . . .	4
1.2.2	Why deep learning . . . . .	5
1.2.3	Transfer Learning as a key . . . . .	7
<b>2</b>	<b>From raw bitmaps to spatial distributions: preprocessing and data preparation</b>	<b>9</b>
2.1	Spot detection . . . . .	9
2.1.1	Gaussian kernel approaches . . . . .	9
2.1.2	Thresholding . . . . .	11
2.2	Segmentation . . . . .	11
2.2.1	Thresholding segmentation . . . . .	12
2.2.2	Morphological Mathematics tools . . . . .	17
<b>3</b>	<b>Generative Model</b>	<b>20</b>
3.1	Identified localization patterns . . . . .	20
3.2	Hand crafted features . . . . .	22
3.3	Image generation process . . . . .	22
3.3.1	Morphological structure . . . . .	22
<b>4</b>	<b>Deep Learning for classification of mRNAs localization patterns</b>	<b>24</b>
4.1	Model choice . . . . .	24
4.1.1	Architecture . . . . .	24
4.1.2	Training . . . . .	25
4.2	Training on synthetic images . . . . .	26
4.2.1	Approach . . . . .	26
4.2.2	Discussion . . . . .	29
4.3	Unsupervised domain adaptation . . . . .	29
4.3.1	Domain Invariant feature representation . . . . .	30
4.3.2	Optimization strategy . . . . .	30
4.3.3	Application to the SVHN dataset . . . . .	31
4.3.4	Application to mRNAs localization patterns classification. . . . .	32
<b>5</b>	<b>Conclusion and next steps</b>	<b>34</b>

<b>A</b>	<b>Generative model</b>	<b>35</b>
A.1	Localization patterns . . . . .	35
A.2	Feature engineering . . . . .	40
A.2.1	Simplified features . . . . .	41
A.2.2	Reconstruction of RNA foci . . . . .	41
A.2.3	Features related to the cell landscape . . . . .	41
A.2.4	Features based on the KRipley function . . . . .	42
A.2.5	Features based on morphological operations . . . . .	42
A.3	Attempts to generate cell landscapes . . . . .	43
<b>B</b>	<b>Transfer Learning strategies</b>	<b>46</b>
B.1	Model choice . . . . .	46
B.1.1	Inception Model . . . . .	46
B.1.2	Comparison with SqueezeNet . . . . .	47
B.1.3	Training details . . . . .	49

# 1 Biological and Mathematical context

## 1.1 Biological context

### 1.1.1 Gene Expression

Gene expression is the process by which the DNA sequence gives rise to functional products in the cell. DNA consists in a long sequence of nucleotides, representing all the information needed for a cell to regulate its basic processes and to fulfill its specific function. A gene corresponds to a stretch of DNA with a specific sequence: the blueprint of a functional molecule, such as a protein. Depending on the function of the cell, different subsets of genes are expressed, i.e. activated. Gene expression is thus one of the most fundamental processes of life and its deregulation plays an important role for diseases. For these reasons, the scientific community has been interested in studying this process for many years.

**Gene Expression process** The Gene expression process can be decomposed into three sub steps: DNA Transcription - RNA splicing - mRNA Translation. See fig. 1 for illustration.

1. DNA Transcription: This first step focuses on turning the DNA code of a single gene into another nucleic acid called RNA (or transcript). All along the cell's life, DNA remains cloistered into the nucleus, and RNA will move the information stored in a gene out of the cell nucleus, where it is actually needed by the cell (along the cell membrane for instance, for the production of reparative proteins).
2. RNA splicing: Once the RNA molecule has been created, it is then stripped off the parts that are not actually needed for the the product of the final functional product. This heavily processed RNA is called mRNA (*m* stands for *messenger*), as it delivers a message on protein construction to the corresponding machineries.
3. mRNA Translation: This third step takes as input the mRNA molecule described above and translates it to a protein. Once produced in the nuclei, each mRNA molecule moves out of the nuclei and will be read by the *Ribosome*, in order to produce the intended protein.

We note that not all the RNA are mRNA: some RNA fulfill a function themselves other than delivering a blueprint for protein production. This study however focuses exclusively on mRNA.

**Expression Level** *Expression Level* is used to characterize a single gene's expression within a cell. Expression level is simply based on a count (at a given timestamp) of mRNA molecules of a given gene in the cell. Therefore, it characterizes "activity" of the gene. Until recently, expression level was the

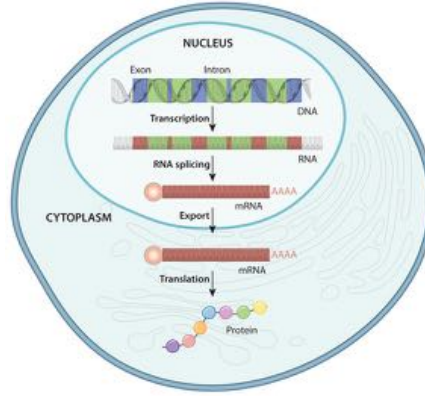


Figure 1: Global overview of the whole gene expression process

most popular metric to study gene expression and infer knowledge such as gene function, cell function, etc. For instance, a gene which expression level is very high will likely be coding information proper to the cell's essential functions, while another one whose expression level is low will likely be coding information for functions not targeted by the cell.

**RNA Localization** Until the early 2000s, translation was thought of occurring at the Endoplasmic Reticulum (ER) or on free ribosomes in the cytoplasm. The generated proteins are then transported to the place in the cell where they should fulfill their function. On the contrary, nothing was known about the active transport and localization of mRNA molecules. Only recent studies [1], [2], [3], showed that mRNAs do not distribute randomly in cells. Therefore, RNA localization is becoming an interesting factor to look at. The process of RNA localization is thought of playing an important role for the local control of gene expression; the reasons, implications and mechanisms however are not well understood. In particular, we still do not know which RNAs localize to specific locations and which do not.

### 1.1.2 Expression patterns

However there have existed several ways to analyze gene expression at the cell population level, conducting such study at the cell level is a more challenging task. First because of the technological challenge it represents: techniques such as *microarray* or *RNAseq* will be able to point out mRNAs from several genes but will show a large variance in the results because the base material is a population of cells. In order to look at the single cell level, other methods have to be invoked. *smFISH* (for Single Molecule FISH) allows to quantify mRNAs of a single gene but at the cell level. Our project heavily relies on smFISH for marking individual mRNAs within a cell.

**Localization patterns** In our project, the study of mRNA localization patterns is based on photographs of plain cells with fluorescent labels attached to mRNAs. As an example, genes whose mRNAs, once produced within the nu-

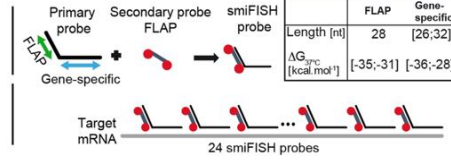


Figure 2: Marking mRNA with fluorescent probes

clei, would migrate toward the cell extensions, and other ones whose mRNAs would remain close to the nuclei, somehow sticking to the nuclei membrane, were identified in [16]. Figure 3 illustrates three different localization patterns with strong characterization.

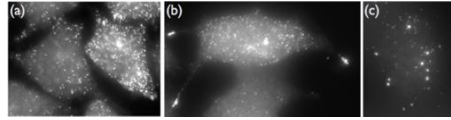


Figure 3: Three examples of localization patterns: (a) Random, (b) Cell Extensions, (c) Agglomeration into bright spots

Taking a step back and trying to sum up this approach would roughly produce such protocol:

1. Gather a large amount of cultured cells.
2. Mark each of those cells with the same fluorescent marker.
3. Take photographs of those cells through fluorescent microscopy.
4. Perform an approach which allows one to locate all the molecules of a given mRNA within a cell.
5. The result of such observation might look like this:

Cell	Gene of interest
$cell_1$	positions of $mRNAs$
...	...
$cell_n$	positions of $mRNAs_1$

6. Once such table has been produced, it is one's job to find a sufficiently concise (in order to be achievable in reasonable computation time) but still precise enough method to characterize a gene.

7. To finish with, the last step of such analysis could be to identify the different localization patterns in the cell population and group genes which showed similar mRNAs' localizations. One's hope would be to come up with a representation which allows him to clearly separate different groups of genes in something that might look like fig. 4.

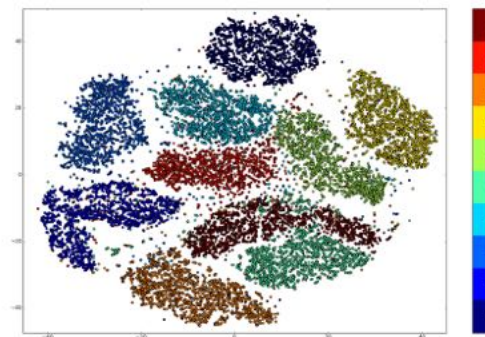


Figure 4: A 2-dimensional representation of a population, where each group would represent a "cluster", or group of similar individuals (gene in our case).

### 1.1.3 Place of this project

This project's goal is to conduct a large screen for localization patterns. [17] suggest that as many as 60% of the *Drosophila*'s genome showed up atypic subcellular localization patterns, and other studies such as [1] suggest that a large proportion of genes do not distribute randomly. Producing a mapping of a large amount of genes to their localization pattern (although we know that some genes distribute randomly) would be an interesting material to study gene expression. Therefore, our goal is to conduct a large-scale screening of genes to their localization pattern(s).

## 1.2 From Microscopy to gene mapping

The specificity of this project being to work at a much larger scale than the studies previously conducted, the care for robust and scalable methods has been a central matter all along its development.

### 1.2.1 Analysis of smFISH data

If we look in further details at the work flow described in 1.1.2, and focus on the last four steps, we can picture our approach as follows:

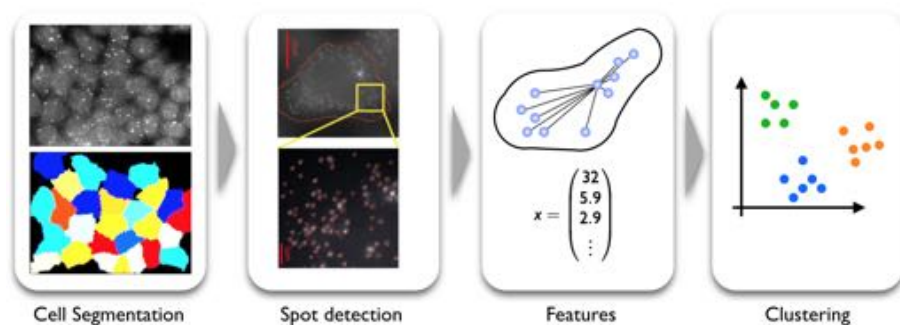


Figure 5:

The first step consists in identifying and separating the individual cells in the images we have. Once all the cells have been separated, the next step is to locate, for each individual cell, the mRNAs. We now have for each cell, a collection of positions  $X, Y, Z$  corresponding to each mRNA's localization. The next task is to build *features* which will characterize each cell's collection of positions. Such feature will take the form of scalars and can be for instance the average distance of mRNAs to the cell's membrane:

Cell	mRNA Positions
$Cell_1$	$position_{1,1}, \dots, position_{1,n_1}$
...	...
$Cell_p$	$position_{p,1}, \dots, position_{p,n_p}$

Cell	Average distance to membrane	Average distance to nucleus
$Cell_1$	235nm	435nm
...	...	...
$Cell_p$	335nm	135nm

Extracting such features allows us to compare cells between each others. Of course, this comparison is directly dependant on the features we chose. Choosing such representation is therefore a very sensitive step because a badly chosen feature might bring non discriminant information. The next step is to compute distances between cells based on the previously described features in order to identify similar cells with small distances, and declare them as a *class*: we identified several classes, and, for instance, one of them is the class of genes whose mRNAs will tend to locate close to the cell borders.

In order to propose a robust and scalable way to classify cells, we decided not to build those features ourselves but rather to rely on automated methods. The next section introduces such methods and their place in our project.

### 1.2.2 Why deep learning

Deep learning is a subfield of Machine Learning, whose main strength is to automatically learn optimal data representations in models known as neural networks. We will focus here on *supervised* learning which again is a sub group of Machine Learning.



**Pros** Deep learning became famous for exceeding at classifying images between a set of identified classes with models known as Convolutional Neural Networks. They can be seen as non linear, parametric models which are optimized against an objective function. In today's state of the art models, the number of parameters can count in millions. Their main strength is to build discriminant features regarding the objective they are asked to optimize against. Since 2012, they became simply ubiquitous in this task of image classification. Both popularity and efficiency of such methods can be assessed on figure 6 which shows the results timeline of the ImageNet challenge (a worldwide famous image classification challenge):

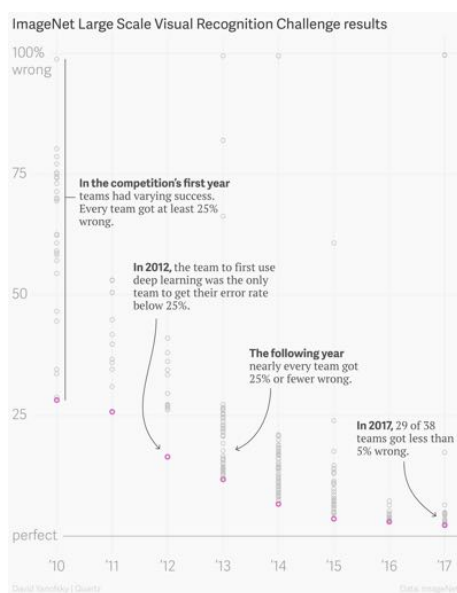


Figure 6: Results obtained on the ImageNet challenge, showing the raise of Deep Learning in the field of Computer Vision

**Source:** Courtesy of David Yanofsky, Quartz magazine

Furthermore, if trained on sufficiently diverse data, such models are known for their robustness. It is particularly valuable in the field of bioimaging where the data can show large variance between the different types of microscopy, or even different experimentalists.

**Cons** As stated above, the models tailored to our study are *supervised algorithms*. It means that they need to be presented a lot of cell images, along with the corresponding expected class for each image. This requires one to have in its possession a large amount of cell images, and for each cell, the class they belong to. Those models are *trained* on such labeled images and then assessed

on images never seen before (of which we know the class as well). We can then assess the *accuracy* of the model by comparing its own predictions and the expected predictions. Even if it is hard to estimate the number of labeled images necessary to train a given model, it is a common heuristic that the more the classification task is complicated, the more data is necessary. As an example, the ImageNet challenge is about classifying daily life images and contains millions of images. Such database was constituted by hand annotations, mainly through crowd sourcing.

**Application to bio imagery** If such methods are very appealing, their appetite for annotated data becomes a true hurdle when annotated data is scarce. In our case, one would need to have in its possession thousands of cell images, annotated by hand of the class of the mRNAs' localization pattern. It is quite obvious that the access to such database is impossible, mainly because annotating images would require a strong expertise and knowledge in bio imagery, which would ask experts to spend valuable hours of their time in annotating images. Therefore, applying Deep Learning methods to the specific case of bio imagery requires one to adapt to data scarcity. Transfer Learning can help in building a bypass, as described in the next section.

### 1.2.3 Transfer Learning as a key

From a general point of view, Transfer Learning is a way to apply knowledge gained from solving one problem (e.g. one classification task) to another problem (e.g. another classification task). It can be particularly helpful when one wants to solve a supervised learning problem with a low amount of annotated data, or when one does not want to train a model from scratch. For our project, we will focus on a specific family of transfer learning techniques, when one needs to perform the same task (e.g. classification) on two dataset but those datasets do not follow the same marginal distribution  $P(Y|X)$ . This approach is also known as Domain Adaptation. See fig 7 for illustration.

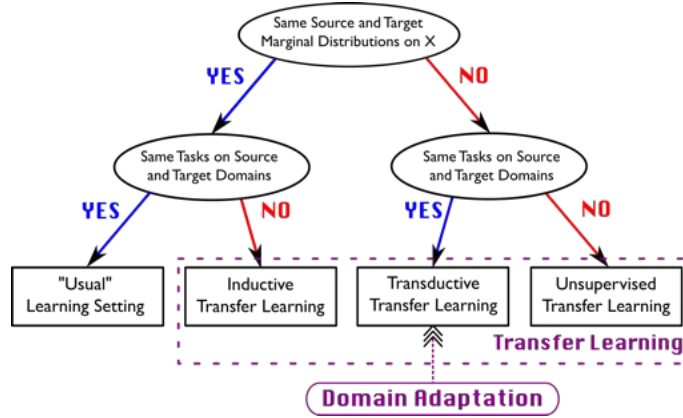


Figure 7: Different transfer learning configuration. This leftmost case is the canonical machine learning situation (no transfer learning). Our study focuses on Domain Adaptation.

**Training on the source** Those methods are based on the idea that in any deep neural network, during the training phase, the top layers (close to the output) will learn very specific and precise features for the asked task, while the bottom layers will learn more general features, which might not be so specific to the current task. Therefore, one could think that the overall model could be good when applied to another distribution, as some features are not very specific to the distribution it was trained on. This approach could be used the following way:

1. Based on observation of natural data, one could propose a simulation framework able to generate synthetic data.
2. Using the synthetic data to train a deep learning model at classifying synthetic images.
3. Use this trained algorithm to predict natural data classes.

Of course, due to this simplicity, this approach heavily relies on the similarity between the training and testing data, and of the model’s ability to learn generalizable features while keeping a low variance.

**Domain Adaptation** If the method described above do not show satisfying results, one could try a finer approach by training a model on synthetic data, while taking into account the fact that distributions are not identical (i.e. that the synthesized data is not a perfectly faithful representation of reality). This approach could take many form but the one we chose was to modify the model itself, in order to enforce it not to adapt too much to synthetic data.

Doing so, we expect our models to perform worse on synthetic data but in the meantime, to learn more discriminant features for real data. This approach will be described in section 4.

## 2 From raw bitmaps to spatial distributions: pre-processing and data preparation

The ultimate goal of this project is to classify spatial distributions of RNA, as observed by smFISH, into distinct categories, called localization patterns. For this, we need to preprocess the raw image data: we need to segment cells and nuclei and we have to detect the spots corresponding to individual mRNA molecules. While in principle, one could omit this last step, it is still useful to actually detect mRNAs and to use these detections subsequent analysis, as they provide us in addition with important features such as the expression level (number of spots).

Importantly, we also used the presented work flow in order to derive the SNR for different types of microscopy, which was important at the beginning of the project, and which was the base for an important series of experiments that is currently being conducted. This demonstrates that such an intermediate biologically interpretable representation has a value in itself.

### 2.1 Spot detection

Spot detection has been an open problem in the field of immunofluorescent images and even if several methods have been identified, no one has appeared as the gold standard for solving this task (see [5], [6]). However, a general work flow has been well identified and remains the core of a majority of spot detection methods. Figure 8 summarizes this approach.

1. First, the image is slightly denoised in order to get rid of all the very small frequency variations, this can easily be done with a low-pass filter such as a Fourier Filter or a Debauchie Wavelet filter. Even simpler, it can be a local minimum filter or a Gaussian filter with a small kernel.
2. Then, the signal is enhanced. This can be done through a contrast enhancement such as Histogram Equalization.
3. Finally, the signal is thresholded in order to keep only the highest values which are identified as spots.

#### 2.1.1 Gaussian kernel approaches

We decided to combine steps 1 and 2 of the above work flow into a single step through a LoG approach. However, for speed and scalability reasons, we approximated it with a DoG which is faster and yields reliable outcome. The idea behind LoG is to detect fast and strong changes in neighbor pixels value by computing the second derivative of the signal.

$$LoG(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \quad (1)$$

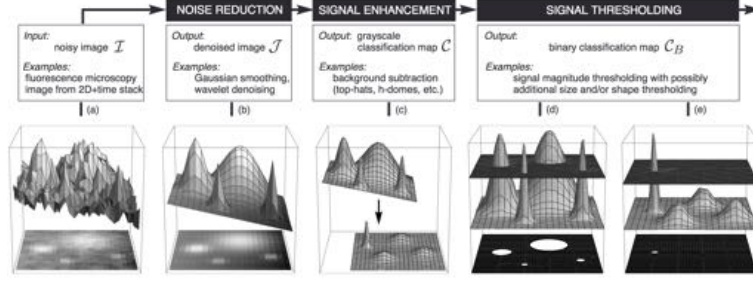


Figure 8: General work flow of spot detection systems

**Source:** [6]

For speed reasons, such derivative is computed by convolving a well chosen kernel to the image, such as:

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
1	-1	-1

As this operation is pretty unstable, the image is smoothed by a Gaussian Filter before computing its second derivatives. The LoG equation therefore becomes:

$$LoG(x, y) = -\frac{1}{\pi\sigma^4} \left[ 1 - \frac{x^2 + y^2}{2\sigma^2} \right] \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad (2)$$

This can be approximated by (where  $*$  denotes convolution product):

$$\begin{aligned} LoG(I) &= K_{laplacian} * (K_{gaussian} * I) \\ &= (K_{laplacian} * K_{gaussian}) * I \\ &\approx (K_{gaussian_1} - K_{gaussian_2}) * I \end{aligned} \quad (3)$$

This last operation is known as the Difference of Gaussian (DoG) which is faster than the LoG while keeping reliable results. For the sake of scalability, we adopted the DoG approach which yielded satisfying results on the images we had.

For our specific study, we do not only need to locate the spots but also to estimate their size. To do so, we iterate 3 with several successive pairs of kernels. The pair which returns the highest value at spot locations can be taken as a rather good approximation of spots width. The idea behind the method is that larger kernel will return higher result on noise than the smaller kernel, and the subtraction of 3 will then result in a negative signal on noise, which can then be clipped to zero. Therefore, we are able to estimate for each spot its precise location (center) and its width.

### 2.1.2 Thresholding

Choosing the detection threshold value is a sensitive step but it can be done with less risks once the image has been processed with a DoG. In parallel, one has to set  $k$ , the maximum authorized number of spots per neighborhood (eg: 1 spot per cube of size  $10 \times 10 \times 10$  pixels). This is another way to prevent false positives in case the threshold was wrongly chosen. In our approach, the threshold is fixed by estimating the probability distributions of pixels intensities in order to distinguish camera noise, autofluorescence noise, and the actual signal of interest. Part of the difficulty resides in the fact that mRNA can illuminate with a very wide range of intensities. To bypass this difficulty, we took the problem backward and tried to identify noise rather than spots. Pixel which were not classified as noise are therefore considered as spot candidates. Figure 9 highlights the different types of noise that have to be separated from the signal of interest.

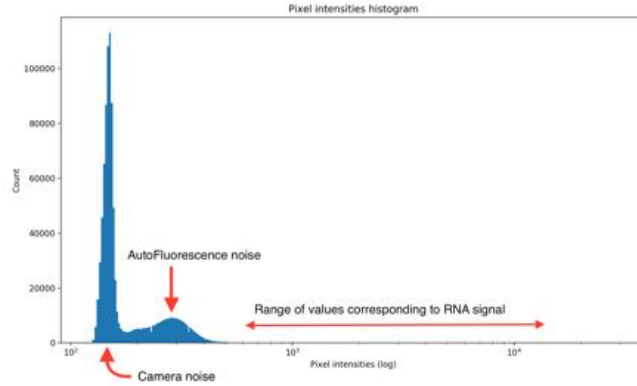


Figure 9: Histogram of pixel values, highlighting sources of noise

Those signals can easily be identified through a mixture model fitting or a standard KMeans algorithm with three classes.

A highly zoomed version of an image with detected spots circled in red is disclosed in figure 10.

## 2.2 Segmentation

Once fluorescent probes have been identified, their respective position is already a source of information (see Appendix B for classification attempts using only the mRNA positions) but they are more meaningful when confronted with morphological information we can extract from the cell itself, i.e. the nuclei boundaries and cytoplasm boundaries. The screening material is of very high quality (in terms of pixel resolution and luminosity conditions), hence, some rather basic unsupervised segmentation tools worked well on the images in our possession.

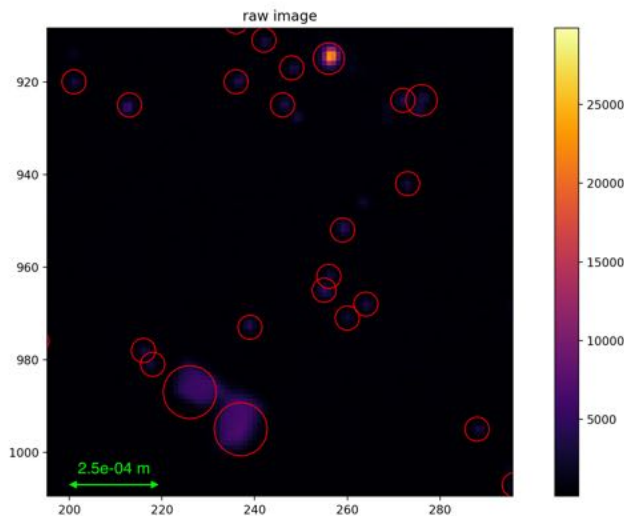


Figure 10: Zoomed image with mRNAs circled in red

### 2.2.1 Thresholding segmentation

While thresholding is one of the simplest image segmentation methods, it can still be sufficient for the segmentation of certain cellular compartments. This is particularly true for nuclei segmentation in fluorescent images, where thresholding remains the most popular method today.

Most thresholding segmentation techniques will try to minimize the intra-class variance of each of the created classes, so that each class will look as homogeneous as possible in terms of pixel intensity.

**Otsu method** Otsu's threshold is one of the most popular method for binary segmentation of images. Otsu's threshold was used in this project to segment nuclei images (DAPI marked). An example of such image can be seen in figure 11. At a simple glance it is obvious to tell what is a nuclei and what is not because the contrast on this image is excellent. As in the rest of the project, such image is the result of a projection of a 3 dimensional image onto one plan, thereafter called *MIP* (Maximum Intensity Projection). For nuclei information, we only perform a 2D segmentation because the DAPI is a highly residual signal which means that even outside the focus plan, one nuclei will keep illuminating several plan below and above which would blurry segmentation in other plans.

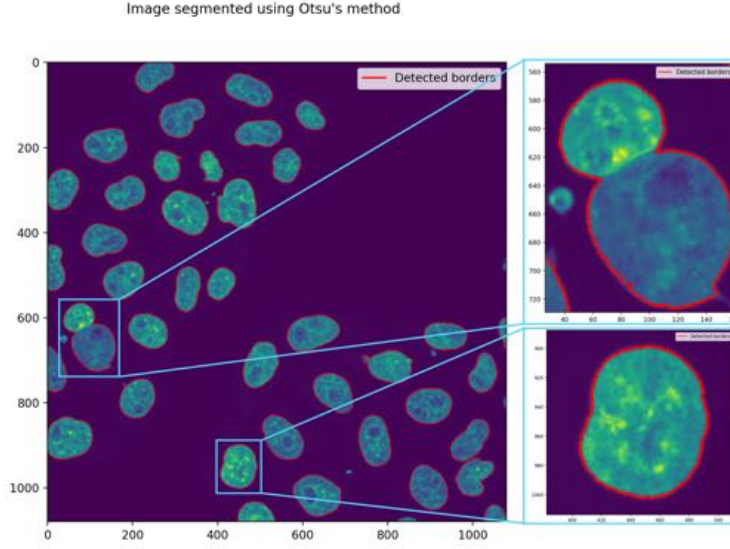


Figure 11: An example of successful nuclei segmentation using Otsu's method, with details magnified.

In order to identify the signal of interest, a simple bi-class thresholding will be much enough. Otsu's threshold method aims at defining classes by minimizing the intra-class variance which is equivalent to maximizing the inter-class variance. Indeed, if  $\sigma^2$  is the image variance,  $\sigma_B^2$  the inter-class variance, and  $\sigma_W^2$  the intra class variance is:

$$\sigma_W^2 = w_0\sigma_0^2 + w_1\sigma_1^2 \quad (4)$$

Where  $w_0, w_1$  are proportions of pixels of class 0 and 1 respectively. On a DAPI Image, Otsu's method seeks the maximum of the red curve:



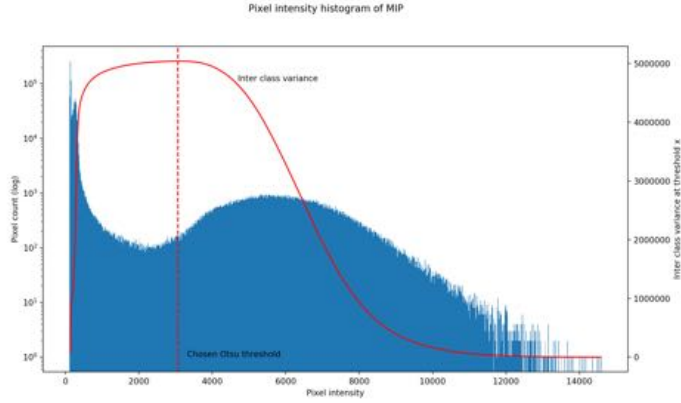


Figure 12: Histogram of pixel values (blue), along with the inter-class variance (red)

In practice, the threshold is computed on the logarithmic image because  $\log$  will spread the small values and condense high values (especially the right tail of the distribution, which increases the threshold arbitrarily with only a very low amount of pixels) which makes this method more robust to intensity variations between images. Furthermore, the image is slightly post-processed to remove small parasites. As a consequence, image presented in fig.11 will be segmented like the following: The two enlarged nuclei show:

1. A difficult case where nuclei show a great variance (i.e. different intensities), which means that the intra class variance will be higher and the threshold will be 'pulled' toward high values by the shining nuclei, at the risk of fixing a threshold too high which might let some darker nuclei aside.
2. The unsolvable issue of siamese nuclei. It is actually almost impossible to know whether those two nuclei belong to two distinct cells or if they are tied by a few chromosomes (in which case it is a unique nuclei). The only way to decide is to look at cytoplasm boundary.

In the first case scenario, one could compute a multi class Otsu which will seek several threshold to form compact classes. In this case (if the standard Otsu returned bad results), one might try a three class Otsu: one class for the background, one for very bright nuclei, and a last class for darker nuclei.

In practice, the quality of the images in our possession and the natural homogeneity of nuclei made the classical Otsu's method robust enough.

**Gaussian Mixture Model threshold** Once nuclei have been accurately segmented, one has to identify cell borders. Unfortunately, segmenting cytoplasm (thereafter called cell masks, the marker used to highlight it) is a much harder task than segmenting nuclei because cells are in contact with each other, unlike

nuclei. The main difficulty comes from separating those touching objects. A raw (MIP) cell mask image is shown, along with the segmented cells and nuclei, on figure13.

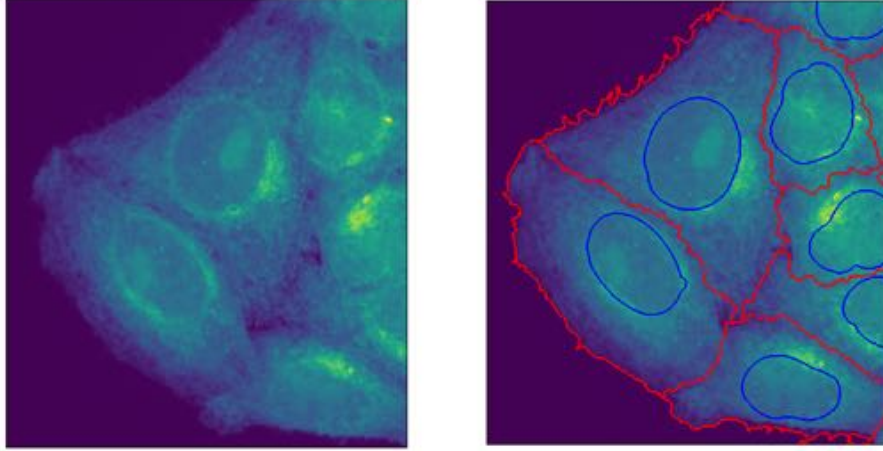


Figure 13: Raw cell mask image (left), and the segmented cells (red, right).

The first step is to distinguish cell pixels from background pixels. To do so, we perform a thresholding just as before but with a different method than Otsu's. Otsu's method is not adapted to distributions with very long tails, because such long tails induce a high intra class variance, which tweaks the optimal threshold setting criterion. We apply Gaussian Mixture Models to the histogram of pixel values, because GMM can adapt to a mixture distribution where components are very different. On the other hand, it tries to fit Gaussians on each mixture component, which might not always be adapted.

$$M(x) = \sum_{k=0}^m w_k \mathcal{N}(x|\theta_k, \sigma_k) \quad (5)$$

This model is fitted with the Expectation-Maximization (EM) algorithm. The procedure will iteratively update  $\{\sigma_k\}$  and  $\{\theta_k\}$  while keeping a lower bound on the maximum likelihood function computed on the data, which ensures convergence of the parameters. The advantage of this method over Otsu's threshold is that we value the prior knowledge that noise follows a normal distribution. Therefore, we ensure that we fit the noise component properly in order to identify it with high accuracy. Once the parameters have been fitted, the GMM classifies the pixel values based on the weight  $w_k$  attributed to each mode. Papers such as [10] investigated this method with further details and reported high segmentation accuracy on the same type of images as we have.

Once the model has been fitted to the data (i.e.  $\Sigma$  and  $\Theta$  has been determined), the GMM predicts posterior probability for each pixel (i.e. to which

class it belongs). The posterior probability  $p(y_i|v)$ , where  $v$  is the intensity value, is simply computed by evaluating the posterior probability of each component independently, which are then weighted by the  $W$  vector:

$$p(y|v) = \{w_i \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp(-\frac{1}{2}(v - \theta_i)^t \sigma_i^{-1} (v - \theta_i))\}_{i=1\dots m} \quad (6)$$

And the predicted class results in a simple maximization of this above vector (i.e. finding the index of the maximum value). In practice, instead of predicting class of each pixel with the GMM, we rather determine first the threshold by predicting classes of each element of a linear ramp  $[min\_pixel\_intensity : max\_pixel\_intensity]$  and determine the threshold which is the class shifting value. Figure14 highlights the GMM fitted to the data and the threshold determined as of, versus Otsu's method described earlier, and a hand-chosen threshold based on this pixel values histogram. As in Nuclei detection, we usually perform this segmentation on the logarithm of the image for better robustness. We also chose this method because we rather want our cell segmentation method to be permissive.

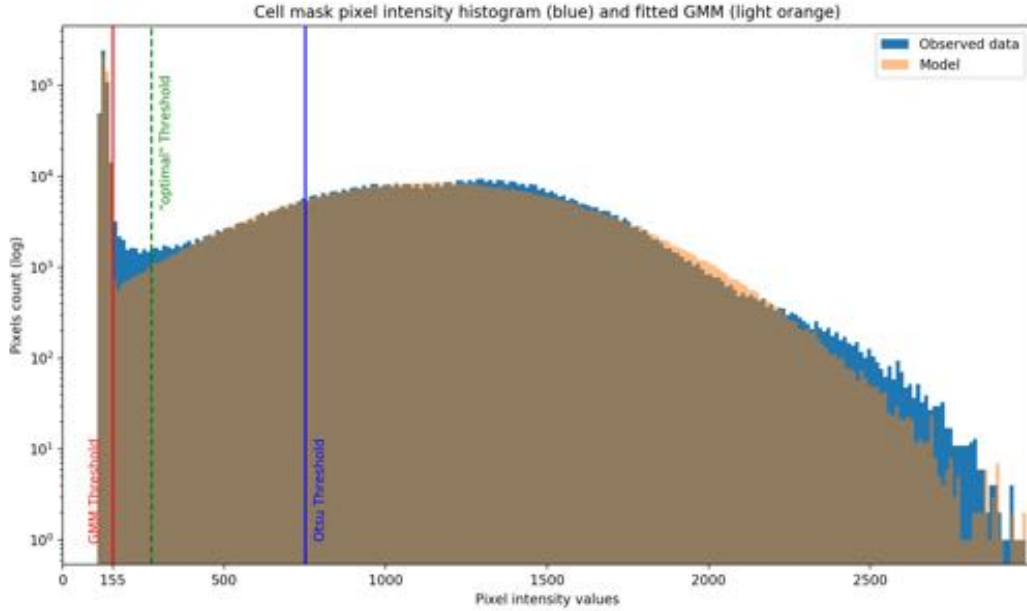


Figure 14: Thresholding through a Gaussian Mixture Model fitting, Otsu threshold in blue for comparison

Because some localization patterns depend on the detection of cell extensions, we rather want to over detect than under detect, this is a second reason why GMM threshold is better tailored to this segmentation.

## 2.2.2 Morphological Mathematics tools

We need now to actually segment cells, i.e. separate the merged cells, which usually is the hardest part. To do so, we apply a watershed segmentation, and enhance it by taking advantage of the 3rd dimension (depth) of our images.

**Watershed segmentation** The watershed segmentation algorithm can be imaged by a dynamic flooding of the 2 dimensional image, seen as a geodesic map (i.e. high value pixels represent crests and low value pixels represent valleys). This flooding starts at several points indicated by the user and water rises at the same speed everywhere in the image. At some point the whole image will be flooded and at this point the process stops. However, when two waters issued from different sources meet, it gives a border between two segments. Figure 15 shows such 3d representation of a 2d image of two merged cells.

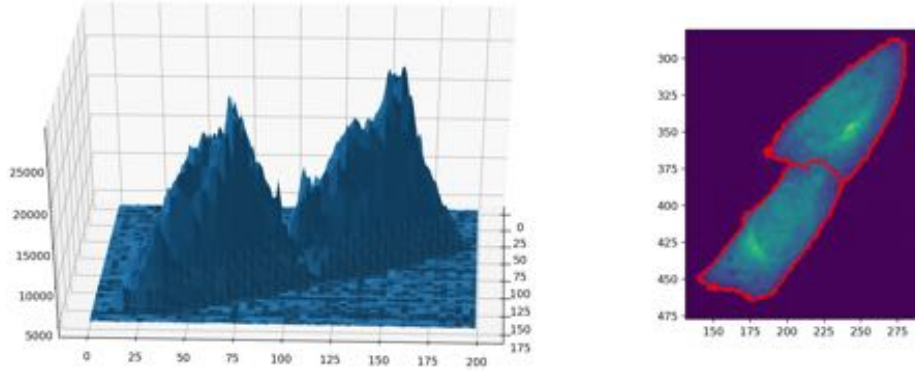


Figure 15: 3D visualization of a 2d slice of a cell mask image, showing two merged cells

Usually, the watershed segmentation is applied to the euclidean distance map of the mask image in order to separate touching elements. This approach requires elements to be rather regular with similar shapes, which is not our case as cells can have extremely various shapes and arrangements. To curb this problem, we figured out that as the cells grow on a plan surface, we could exploit that two touching cells would be in contact by their bottom, while it is very likely that the cell tops will all be distinct. Therefore, we applied the watershed transform to the inverse of the z-sum of the images. Doing so, we transform the 3d image into a 2d image where a pixel of high value is likely to be located somewhere under the top of a cell, and a pixel of low value will likely be located close to the boundaries of two touching cells. See fig. 15 for illustration.

If we invert the image and look at the crest line it becomes much clearer how the watershed segmentation works (see fig.17).

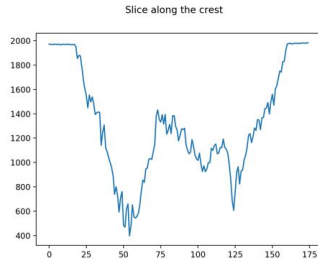


Figure 16: Crest of the 3d surface

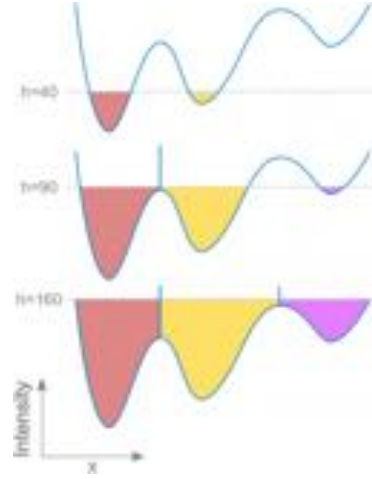


Figure 17: Watershed flooding

As illustrated on the figures above, the watershed segmentation is a very powerful tool for segmenting touching elements. In practice, the starting points of the flooding are positions of the nuclei of the same image because we suppose that each individual nuclei denotes one distinct cell.

**Merged cells segmentation** This segmentation routine is very satisfying considering that it works also at the slice level (in which case we only z-sum the slices located above the considered slice). It can be used at every slice successively and act as a 3D segmentation tool. Furthermore, in order to ensure a consistent behavior regarding nuclei (ensure that one nuclei is consistently attributed to a single cell, and that no nuclei actually sits on the border of two watershed-segmented cells), the image is put to its minimum value everywhere a nuclei is located (fig.18).

However, this method requires one to have in its possession a segmented DAPI image associated with each cell mask image

The routine described above is fairly satisfying in terms of computational complexity as the only extra operation performed by the user else than the watershed is a depth-wise sum and a slicing to set nuclei's positions values to zero. The watershed itself is efficiently implemented in a wide variety of languages and packages.

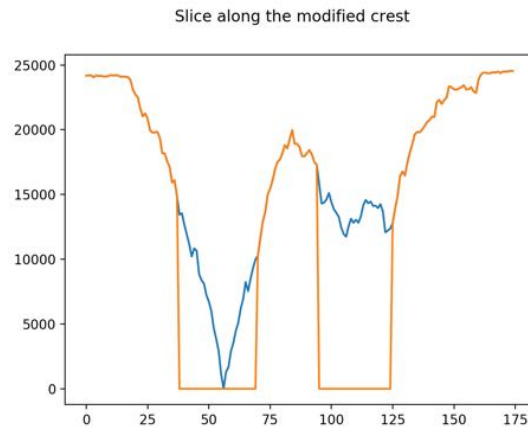


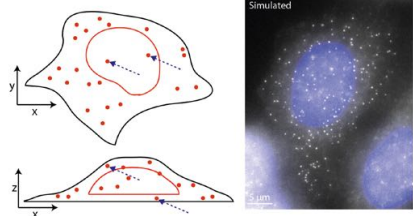
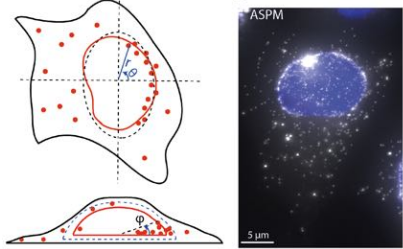
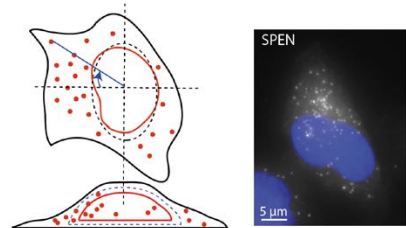
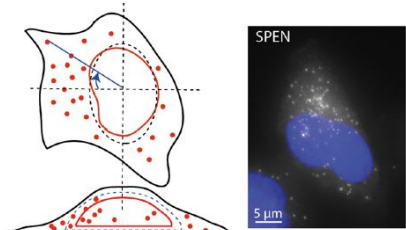
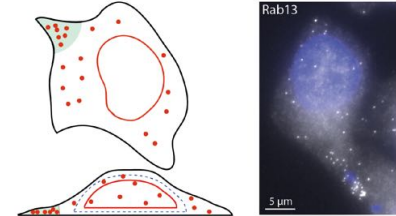
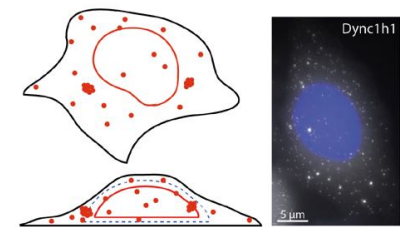
Figure 18: Original crest (blue) with the nuclei positions values put to minimum (orange).

### 3 Generative Model

The two main benefits of deep learning in the context of this work is that we do not need to engineer handcrafted features and that we can expect a much better accuracy than with traditional methods. On the downside, we need to provide large annotated data sets. As annotation for this type of data is tedious and error prone, Samacoïts et al. have developed a simulator that can generate simulated smFISH data with known spatial distributions. These data either mimic existing localization patterns or generate data with plausible localization rules. The idea is thus to learn a classifier from these simulated data and apply it to real data. If this approach is successful, it would address one of the major bottlenecks in the use of deep learning to bioimaging data today: the lack of annotated ground truth data. In this section, we describe the different patterns that were simulated. This work was performed during a PhD thesis prior to this master thesis, and details about the protocol are given in Appendix B.

#### 3.1 Identified localization patterns

We will list here six of the height identified localization patterns and briefly describe them. Our study focuses on those six patterns only. Details can be found in *Appendix A.1*, along with a full list of localization patterns.

Name	Description	Illustration
Random	In this case, mRNA are placed in the cell following a pure 3 dimensional random distribution, within the 3d volume of the cytoplasm.	
Nuclear edge	mRNAs locate selectively on some regions of the nuclear membrane.	
Cell edge	mRNAs show localization towards the edge of the cell as seen in 2D.	
Polarized localization	In this pattern, mRNAs occupy preferentially a certain part of the cytoplasm only, rather distant from any membrane.	
Cell extension	mRNAs will preferentially locate within cytoplasm extensions.	
Foci	For some genes, the corresponding transcripts aggregate and thus form foci, i.e. small areas of very high RNA density, where single RNA molecules can no longer be resolved.	



### 3.2 Hand crafted features

In order to sample sets of mRNA positions, hand crafted features were built upon biological expertise and data observation. Such features characterize the patterns listed in the previous section. For instance, one of those hand crafted features is the average distance of the mRNAs to the cell membrane. Such features are described in [16] and listed in *Appendix A.2*. Those features allowed us to model distribution of mRNA for each localization pattern, and therefore to sample individual from those distributions. Below an example of a cell2D pattern naturally observed, along with a synthetically generated set of mRNA positions mimicking this pattern.

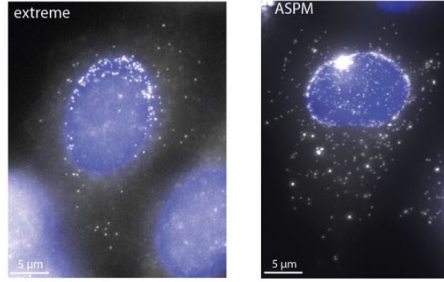


Figure 19: Comparison a naturally observed (right) and a synthetic (left) Cell2D pattern

### 3.3 Image generation process

As mRNA positions alone are not meaningful, one has to make sure that the pattern described above are used to generate mRNAs with regard to a certain cell landscape, therefore, it is one's obligation to be able to generate (or in our case provide) cell morphologies.

#### 3.3.1 Morphological structure

As mRNAs are synthetically created and placed in each simulated image, one could argue that having a similar dispositive for creating cell landscapes is a necessary step in our approach.

Unfortunately, generating random but meaningful cell landscapes (i.e. cytoplasm and nucleus inside of it) is a much more complicated task due to obvious constraints such as: the cell needs to be rather convex, but must allow extensions to exist. In the meantime, the nuclei should always be fully included inside the membrane, not be too big, and let some space for mRNA to locate outside of it but inside the membrane.

For now, our simulation framework includes a library of a few hundreds of segmented cells, and for each simulation, we pick one cell from this library and artificially add the mRNAs at the sampled positions. Such library includes for

each cell, the boundaries of the nuclei and the membrane in 2D. Therefore, one can argue that the generation process is not random and that the feature we will extract by training new algorithms on this dataset might actually be related to this morphological information because this one (being less varied than localization patterns) is easier to learn. To oppose this point, one can see that by generating large amounts of cells, each cell shape will be seen every time with a different set of synthetic mRNAs, and it is very likely that each cell landscape will be seen under all the possible classes. This last point ensures that the algorithm will not be able to classify mRNAs localization patterns based on the cell landscape only. Reader can refer to section 4 where we actually conducted experiments on this point to confirm validity of our approach.

An approach to randomly generate cell landscapes through Variational AutoEncoders was formulated but has not reach satisfying result yet. This approach is described in *Appendix A.3* for completeness.

## 4 Deep Learning for classification of mRNAs localization patterns

This section describes the core of our study, namely, extracting features from synthetically generated data and assess whether those features help in separating classes in real data.

### 4.1 Model choice

This subsection details the model we have used for the task of classifying cell images into one of the six chosen localization patterns. We present only one model, but *Appendix B.1* details a comparison of two models, along with the reasons which lead us to choosing this model in particular.

#### 4.1.1 Architecture

The SqueezeNet is based on the same idea as other famous Convolutional Neural Networks such as Inception, of using similar blocks of operation stacked one upon the other, each block acting as a *Network in Network* (NiN). It retains itself to a much lower number of parameters than its pairs and really focuses on portability and deployment. The original paper was issued in 2016 and announced state of the art accuracy with 50x less parameters than its main challengers. The initial observation was that for a given accuracy level, there existed many different architectures that achieved this accuracy, and that among those architectures, some were lighter than the others.

The SqueezeNet architecture, described in [14], mainly implements three strategies to reduce the number of parameters used while keeping a fairly large depth (26 convolutional layers) and high performances: use more 1x1 filters, decrease the number of channels inputted to 3x3 filters, and down sampling lately in the network in order to keep large activation maps (based on the heuristic that large activation maps help keeping high accuracy).

The two first strategies are revealed through the design of the *fire modules* which are repeated to compose the overall architecture. Such a module will first compress the information through 1x1 filters with a heuristically fixed number of channels. This compression will be followed by two convolution operations, one with 1x1 filters and the other with 3x3 filters, those operation are performed in parallel and merged back into a unique tensor which is then activated through a ReLu. Note the absence of any down sampling operation in the module.

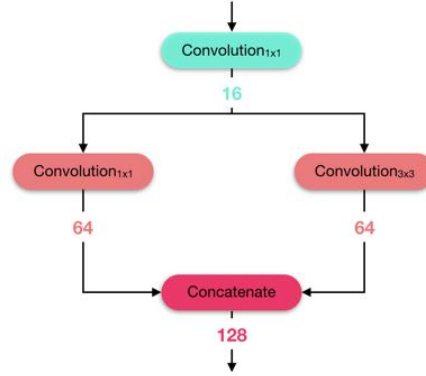


Figure 20: Example of a fire module, which compresses the input information into 16 feature maps, and then expands it into 128 feature maps built with 1x1 and 3x3 filters

The overall architecture is composed of stacked *fire modules*, alternated with some down sampling operations. Note that the SqueezeNet is a fully convolutional architecture, which makes it tailored to varying-sized input. In practice, this architecture is mixed with some *skip layers*: where the input of layer  $n$  is not only the output of layer  $n - 1$  but also the output of layers  $n - k$  where  $k$  is taken here equal to 2. Such strategy has proved consistent improvement in training convergence (see [18] for completeness). Refer to figure 21 for illustration.



Figure 21: Ensemble architecture with skip layers: input to module 7 is output of module 6 added to module 5's output. Colored figures along connections indicate the number of channels.

#### 4.1.2 Training

Our model was trained on a database of roughly 300,000 synthetic images, which was split between a train, a test, and a validation set in a standard fashion (60%, 20%, 20%), resulting in 180,000 images in the train set. A class-weighted categorical cross-entropy was used as the loss function. A single Dropout operation

with a rate of 0.5 was used before the penultimate layer in order to avoid overfitting. The images in our possession are  $600 \times 600$  images, split between three channels (representative of the acquisition procedure) i.e. one channel showing the nuclei, one channel showing the cell membrane, and another channel for the mRNAs. We observed a clear improvement in performances when the channels are split as explained, versus all the cell elements concatenated into a unique channel (see appendix B.1.3 for details). A few data augmentation operations are performed on the images, namely: rotating, stretching, zooming in and out, and translating the images, performed on the fly before feeding the network. Our SqueezeNet was trained with an Adam optimizer and a batch size of 60, implemented with the Keras library for Deep Learning. The final model took around 24 hours to train on a NVIDIA<sup>®</sup> TESLA P100, along with 56 CPU cores for data preprocessing and augmentation.

## 4.2 Training on synthetic images

### 4.2.1 Approach

The first step we took in our project was to train a classifier algorithm on synthetic data and assess it on real data. Evaluating such approach is not straightforward as the lack of annotated natural data retains us from computing the performance metrics we used during training. In order to do so, we decided to visualize the features extracted by the model. Those features are 512-dimensional vectors built by the model and accessed to just before the ultimate convolutional layer. Such features are expected to contain enough discriminant information on one image to decide to which class this image belongs to. By visualizing them, one's hope would be to see tendencies of genes' localization in the feature space. Conversely, one's hope would be that genes which show similar localization patterns have important overlaps in terms of distribution in the feature space. Figure 22 shows a 2-dimensional representation of the features extracted from a 10,000 synthetic images dataset never seen by our model.

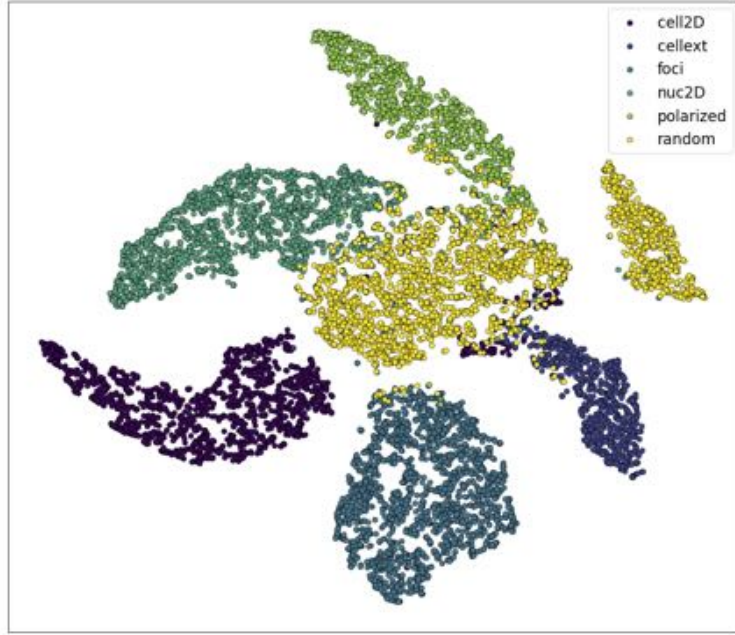


Figure 22: 2-dimensional visualization of features extracted of the synthetic data by our model. The different classes are clearly separated except the Random class, which is harder to classify.

This plot can be coupled with a similar scatter plot, color coding the pattern strength of each image. We can see that for two patterns (polarized and foci), images with strong pattern are more easily distinguished from images with randomly-placed mRNAs.

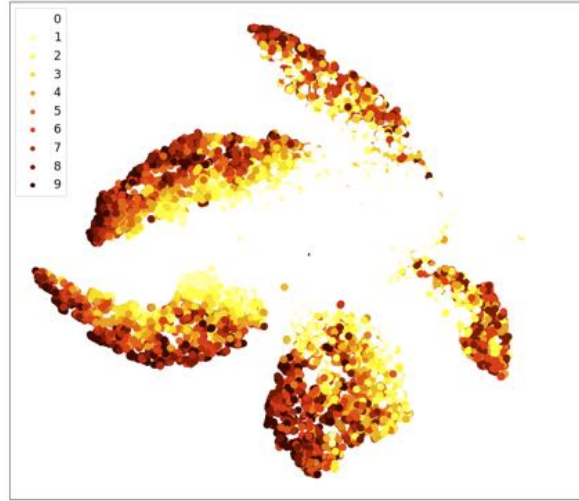


Figure 23: Pattern-strength color-coded TSNE of synthetic images

If we perform a similar analysis on the real data, we see that the result is not as clear. Below is shown a similar representation for a dataset of 1,300 real images of four genes.

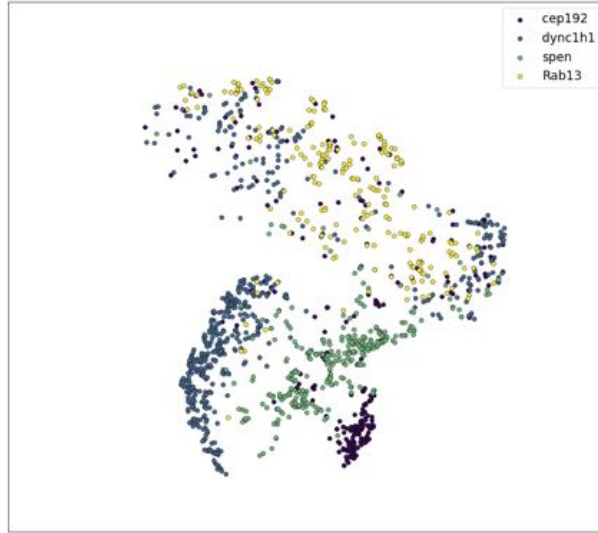


Figure 24: 2-dimensional visualization of features extracted of the real data by our model.

Some genes such as *dync1h1* and *cep192* show rather distinct localizations in this plot, but others such as *Rab13* are much more spread over the all map.

Blurrier color clouds than figure 22 were expected, as one gene will not systematically show a localization pattern. In other words, *dync1h1* might be *random* in one image and *cellex*t in another image, without meaning that the classifier made a mistake.

In order to get a better understanding of this map, we can look at the class predicted for each of the four genes described above.

	cellex	foci	nuc2D	polarized	random	All
gene						
<b>Rab13</b>	16.0	36.0		35.0	128.0	215
<b>cep192</b>		36.0		164.0	43.0	243
<b>dync1h1</b>		433.0		31.0	85.0	549
<b>spen</b>		147.0		160.0	7.0	314

Figure 25: Classes predicted for four genes. Each gene is classified into several patterns at least once.

We can see that *dync1h1* shows a very strong preference for the *cellex*t pattern. In the meantime *cep192* is in majority classified as *polarized*. Such strong preferences explains why we could distinguish the images of those genes creating clear clusters on image 24. On another hand, gene *spen* does not show a unique localization preference between *polarized* and *cellex*t, which might explain why *spen* images were located right between *cep192* and *dync1h1* on figure 24.

#### 4.2.2 Discussion

We see from 25 that difficulty can come from the biological reality of mRNAs localization which is not always consistent, but one can not occult that the data we trained our classifier on and the data we apply it to follow different distributions. The results we have obtained so far encourage us to push our Deep Learning approach further, by trying to tackle this last issue.

### 4.3 Unsupervised domain adaptation

This approach was introduced in *Unsupervised Domain Adaptation by Back Propagation* ([15]), where the authors tackle the issue of training an algorithm on data following a distribution  $P(X)$ , while this algorithm is meant at being applied to a data following another distribution,  $Q(X)$ .

Ganin and Lempitsky describe a real-life situation where one’s desire would be to classify data without having access to an annotated database. As a solution, they propose to train models on synthetic data while enforcing the learned feature representation to be domain invariant.



### 4.3.1 Domain Invariant feature representation

The main idea of Domain Adaptation is to adapt features learned on one domain (the source domain, here synthetic data) to the other domain (the target domain, here real data). This can be done by an innovative architecture of neural networks where a given model will be asked and trained for the additional task to tell whether an individual is synthetic or real. This extra task is achieved by a new branch appended at the end of the feature extractor and classifies *domain* in parallel with the original classification task (in our case, localization patterns). This architecture is completed by a so-called *Gradient Reversal Layer* (GRL thereafter) which turns the optimization process into a *Gradient Ascent* for some parameters. Refer to figure 26 for an illustration.

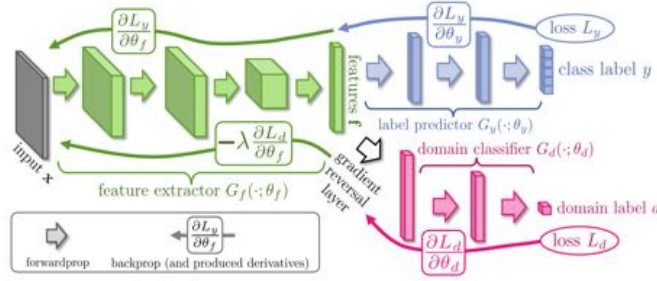


Figure 26: Illustration of our domain adaptation framework with a gradient reversal layer applied to the domain classifier branch of our model

**Source:** [15]

We can explain the role of this GRL as follows: we want to train a classifier on a given dataset (synthetic) in order to predict on another dataset (real) that we know do not follow the same distribution. Therefore, the risk is that our classifier learns latent representations of  $X$  which are discriminant in the synthetic dataset but not in the real dataset. In order to avoid that, we want to forbid our classifier to learn a feature representation too specific to the synthetic dataset.

Such bad feature representation would be *domain specific*, and, hence, would allow the classifier to discriminate domains (i.e. tell for each image whether it is real or synthetic). Equivalently, if our classifier is unable to discriminate domains, we can assert that the feature representation it learned are not domain-specific.

### 4.3.2 Optimization strategy

During the Optimization process, the following loss will be optimized against:

$$Loss = L_{pattern}(\Theta, y_{pattern}) + L_{domain}(\Theta, y_{domain})$$

And computing the gradient with regard to the parameters yields:

$$\frac{\partial L}{\partial \Theta} = \frac{\partial L_{pattern}}{\partial \Theta} + \frac{\partial L_{domain}}{\partial \Theta}$$

Furthermore, we can write  $\Theta = \Theta_{feature\_extractor} \cup \Theta_{domain\_classifier} \cup \Theta_{pattern\_classifier}$ . Looking at the update step during the Gradient Descent:

$$\begin{aligned} \Theta_{feature\_extractor}^{(t+1)} &= \Theta_{feature\_extractor}^t - \lambda * \frac{\partial L}{\partial \Theta_{feature\_extractor}} \\ &= \Theta_{feature\_extractor}^t - \lambda * \frac{\partial L_{domain} + L_{pattern}}{\partial \Theta_{feature\_extractor}} \\ &= \Theta_{feature\_extractor}^t - \lambda * \left( \frac{\partial L_{domain}}{\partial \Theta_{feature\_extractor}} + \frac{\partial L_{pattern}}{\partial \Theta_{feature\_extractor}} \right) \end{aligned} \tag{7}$$

We can re-write:

$$\begin{aligned} \frac{\partial L_{pattern}}{\partial \Theta_{feature\_extractor}} &= \frac{\partial L_{pattern}}{\partial \Theta_{domain\_classifier}} * \frac{\partial \Theta_{domain\_classifier}}{\partial \Theta_{feature\_extractor}} \\ &= (-1) * \frac{\partial L_{pattern}}{\partial \Theta_{domain\_classifier}} \end{aligned}$$

7 can then be re-written by injecting 4.3.2, which will yield two terms in the loss function:

1. One term decreasing  $L_{pattern}$  by going opposite way of the gradient
2. One term increasing  $L_{domain}$  by going same way as the gradient

Therefore, this optimization strategy ensures that the feature extractor will not learn to distinguish domains as  $\Theta_{feature\_extractor}$  is optimized as opposed of.

#### 4.3.3 Application to the SVHN dataset

In [15], authors demonstrate efficiency of their method by experimenting it on an enriched version of the SVHN dataset. The source domain is composed of Windows font numbers, and the target domain is composed of house numbers captured from Google Street view, examples shown in fig.27.

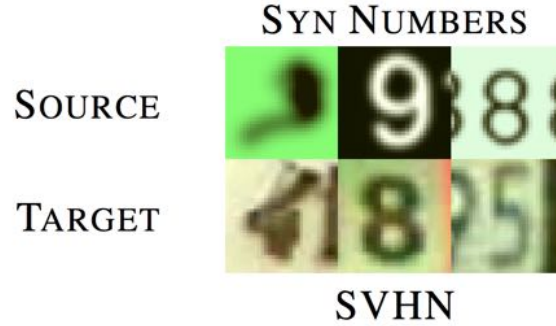


Figure 27: The model will be trained on the source domain in order to predict classes of images from the target domain.

**Source:** [15]

The network used for this task was composed of three convolutional layers, upon which were plugged: (1) two fully connected layers for the digit classification task, (2) two fully connected layers for the domain classification task. We are principally interested in seeing the impact of this method over the feature representation learned by the model. Figure 28 shows the feature representation of the test dataset.

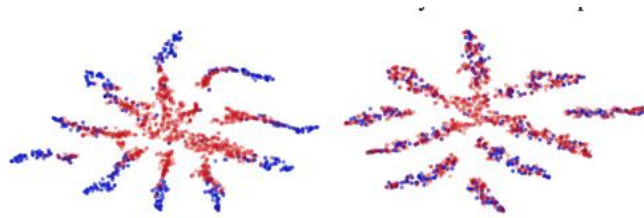


Figure 28: Feature representation of source domain images (blue) and target domain images (red), with (right) and without (left) domain adaptation.

**Source:** [15]

#### 4.3.4 Application to mRNAs localization patterns classification.

This figure shows well that the learned feature representation seems domain invariant. This better invariance comes along with an improved accuracy on the target domain (+ 4 pts% accuracy versus pure train-on-source strategy). The above results encourage us to implement this approach for our project, with a simple modification of our architecture (see fig.29)

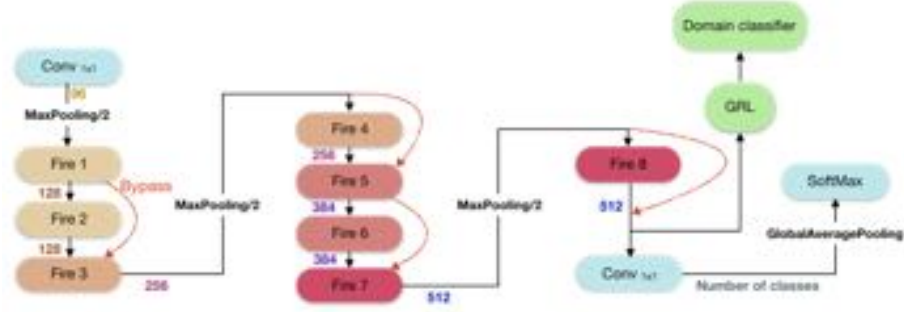


Figure 29: The modified architecture of our model, with an additional domain classifier in parallel of the standard localization pattern classifier

However simple in appearance, this approach still requires a careful parameter tuning, especially when designing the loss function. It is our next objective in this project to apply this Deep Adaptation approach in order to build more meaningful features of the real data.

## 5 Conclusion and next steps

On this project, our primary goal has been to develop a robust and scalable method to classify gene's localization patterns from immunofluorescent images. Our motivation is based on several studies asserting that a large proportion of genes would induce non random localization patterns for their mRNAs, and producing a large scale screen of mRNAs localization patterns would constitute valuable information in the study of gene expression. We heavily leveraged a statistical framework developed by A. Samacoits in *Development of a statistical analysis framework for sub-cellular mRNA localization*, Ph.D thesis, in order to experiment supervised learning algorithms, and compare them with the unsupervised methods he developed. This approach lead us to facing an issue which is a true hurdle for many bioimagery studies, namely, the difficulty to access large annotated database.

This obstacle usually retains bioinformaticians to leverage the proven robustness and efficiency of Deep Learning approaches, as the associated models require access to considerable amounts of such data. Hence, the guideline of this study has been to take advantage of a simulation framework developed by Samacoit, in order to assess the ability of such models to generalize feature representations learned on synthetic data to real data. The first results we have come up with were encouraging and invite us to pursue our efforts.

If our approach showed satisfying results on new sets of genes, it would be promising for extending it to a larger scale. In parallel, it would show that the lack of annotated data in bioimagery studies could be bypassed through the use of simulated data. More and more deep learning methods make use of such *transfer learning* techniques to reach the outstanding results they are known for, which encourages us to go further in our approach, and assess its efficiency in a field where it is not as popular yet.

\*\*

## A Generative model

### A.1 Localization patterns

The simulation framework allows the generation of 8 different localization patterns, they are listed below along with their respective specificities, as described in [16]

**Random** In this case, mRNA are placed in the cell following a pure 3 dimensional random distribution, no information regarding cell membrane or nuclei position are taken into account. mRNAs are placed within the 3d volume of the cytoplasm of course. See figure 30 for details.

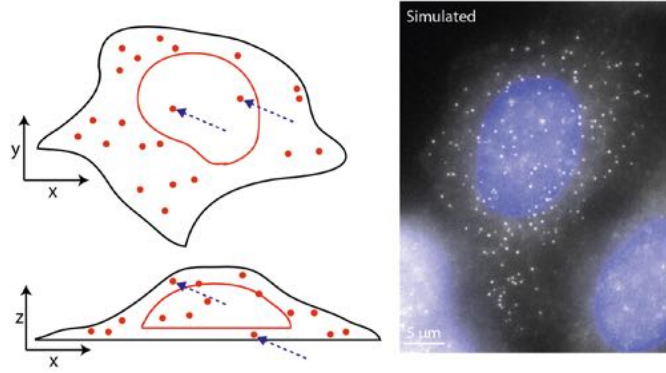


Figure 30: Detail of randomly placed mRNA, note that mRNAs are not placed inside the nuclei

**Source:** [16]

**3D Nuclear membrane** A mRNA is considered to be localized at the nuclear membrane if it is below a fixed distance (800 nm) from the membrane. This pattern is difficult to distinguish from other nuclear patterns by inspection of 2D projections. Strength of this pattern can be tweaked by varying the proportion  $p$  of mRNA actually close to the nuclear membrane. In our simulations,  $p$  varies from 0.6 (weak pattern strength) to 0.9 (strong pattern strength). See figure 31 for details.

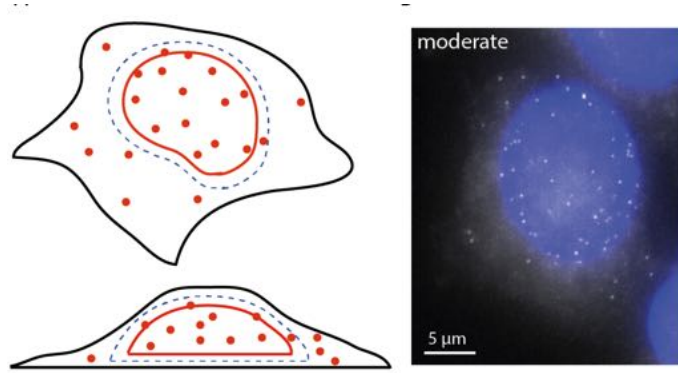


Figure 31: Detail of mRNA placed close to the nuclear membrane (between the dashed blue line and nuclear envelope)

**Source:** [16]

**Nuclear edge** This pattern was motivated by observations from experimental data. We observed localization patterns, where mRNA remain close to the nuclear membrane in 2D, often enriched on one side of the nucleus. The resulting patterns looked strikingly different from the simulations shown in. Apparently, RNAs did not localize uniformly on the entire nuclear membrane, but rather selectively to some parts. mRNA positions for this pattern are simulated using spherical coordinates with the origin at the center of the nucleus. The nucleus is fit with an ellipse to get the two main axes. The polar angle is picked from a normal distribution centered at one of the four possible nucleus axes direction with a fixed standard deviation. We choose the standard deviation such that the mRNAs localize at one half of the nucleus, which corresponds to the observed pattern. The azimuth angle is picked from a normal distribution with a low mean to force the mRNA to locate at the periphery of the nucleus. The radius is picked so that the mRNA is closer to the nuclear membrane than a fixed distance (800nm). The three different degrees of the pattern were simulated with different values of the parameter  $p$ .

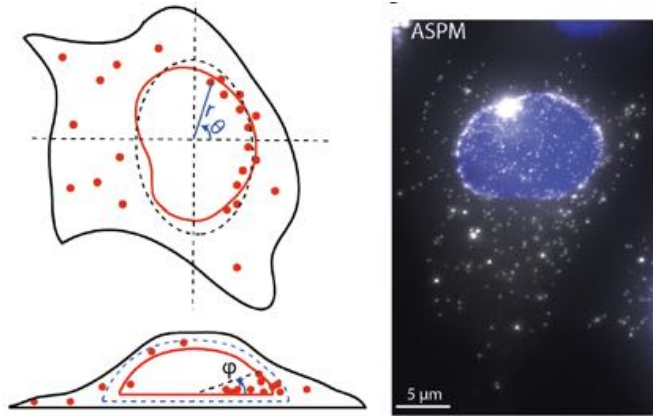


Figure 32: Different pattern strength of the 2d nuclear edge pattern, along with an image of the ASPM gene which was observed to locate the same way.

**Source:** [16]

**Cell membrane in 3D** An mRNA is considered to be localized at the cell membrane if it is below a fixed distance (800 nm) from the cell membrane. The three different degrees of the pattern were simulated with different values of parameter  $p$  (weak:  $p = 0.7$ , moderate:  $p = 0.8$ , strong:  $p = 0.9$ ). An intriguing feature of this pattern is that it is very difficult to be spotted from 2D projections, which is highlighted in figure .

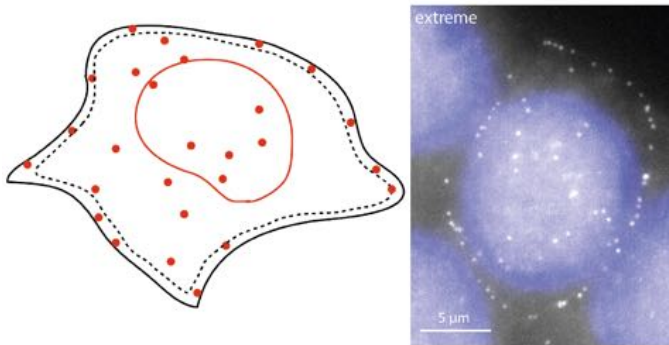


Figure 33: Vizualizing the cell in only 2D drastically limits the chance to distinguish this pattern from *Random*

**Source:** [16]



**Cell edge** Here, the mRNAs show localization towards the edge of the cell as seen in 2D. Positions are simulated with polar coordinates. The polar angle is picked from a uniform distribution between 0 and  $2\pi$ , and the radius is chosen such that the mRNAs are closer to the cell membrane than a fixed distance (800 nm). The three different degrees of the pattern were simulated with different values of parameter  $p$  (weak:  $p = 0.4$ , moderate:  $p = 0.5$  strong:  $p = 0.6$ )

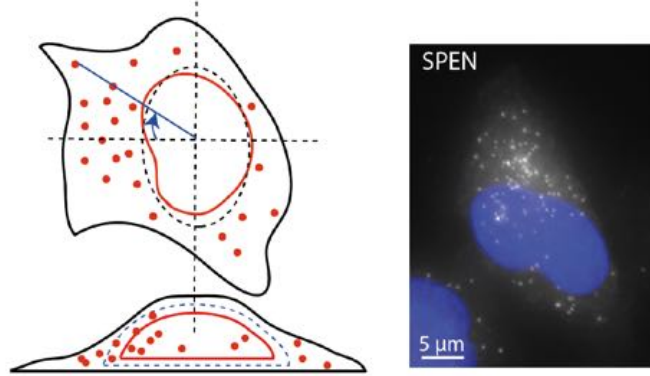


Figure 34: Density of RNAs is highest in between the cell membrane and the blue dashed line. This pattern is 2D pure as opposed to *3D nuclear membrane*  
**Source:** [16]

**Polarized localization** In this pattern, mRNAs occupy preferentially a certain part of the cytoplasm. The cell is fitted with an ellipse in order to get the two main axes. mRNA positions are simulated using polar coordinates with origin at the centroid of the cell. The polar angles are picked from a normal distribution centered at one of the four possible main axes with a standard deviation determined by the user. The smaller this standard deviation is, the stronger the pattern will be. The three different degrees of the pattern were simulated with different values of parameter  $\sigma$  (weak:  $\sigma = \pi/10$ , moderate:  $\sigma = 3\pi/10$ , strong:  $\sigma = \pi/2$ ).

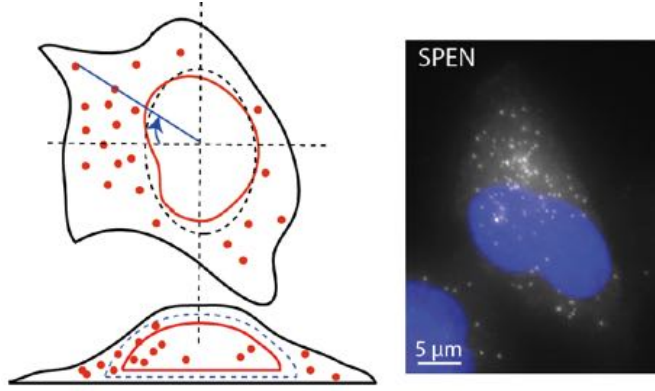


Figure 35: mRNAs are preferably placed in one part of the cytoplasm  
**Source:** [16]

**Cell extension** We found that some mRNAs are enriched in cell extensions. There is no formal definition of cell extension, but it is straightforward to give examples for extensions from image data. Hence, in order to model this pattern in a biologically meaningful way, we decided to annotate cellular extensions in our data set and to simulate this pattern as an accumulation in these annotated locations. More precisely, we extracted the annotated part of the cell border and we calculated the distance to the center of the cell. We considered the pixel on the border with largest distance to be the tip of annotated cellular extension. In some cells, many possible extensions were annotated. For the simulation, we randomly picked a maximum of three extensions for a given cell. We then simulated this localization pattern as uniform RNA distribution around the cell extension tip within a fixed radius  $r_{tip}$  (2000nm). The user specifies the ratio  $R$  between RNA densities at the cellular extension and the cytoplasm respectively. The simulation then proceeds as follows: first, a number of RNA locations is drawn from a uniform distribution inside the cells. Second, additional mRNAs are localized in the cell extensions in order to satisfy the density ratio. The three different degrees of the pattern were simulated with different values of parameter  $R$  (weak:  $R = 10$ , moderate:  $R = 15$ , strong:  $R = 20$ ).

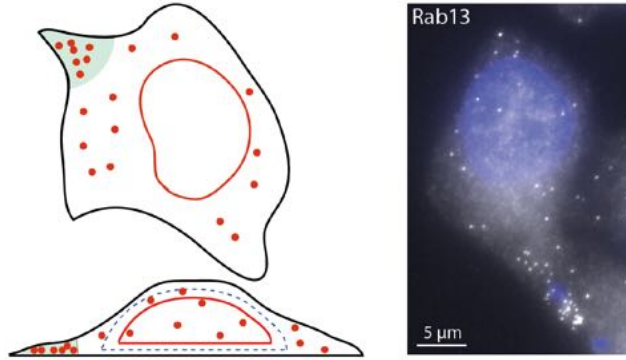


Figure 36: Details of cell whose extension have been manually enriched with mRNAs, along with the *RAB13* gene whose mRNA have been observed to locate following the same pattern

**Source:** [16]

**mRNA Foci** For some genes, the corresponding transcripts aggregate and thus form foci, i.e. small areas of very high RNA density, where single RNA molecules can no longer be resolved. We extracted from experimental data (DYNC1H1) the distribution of blob number per cell and the number of mRNAs per blob (using a Gaussian Mixture Model). In the simulation, we choose randomly from these distributions how many foci and how many mRNAs per blob are simulated. The spatial extent of these foci was set to a size-range that visually corresponded to experimental data (radius of the sphere picked between 500-1000 nm). The foci were randomly placed in the cytoplasm in 3D. Cells simulated with the parameters extracted from Dync1h1 were considered as strong. For the moderate pattern, we reduce both the number of foci and the number of mRNAs per blob by 25% compared the strong case. For the weak pattern, we reduced both numbers by 50

## A.2 Feature engineering

In a general context, a *feature* is a characteristic if a probability distribution which can be measured and quantified. Therefore, features allow one to compare individuals between each other. Of course, features can be of various types but the ability of a given feature to separate, or be discriminant is a usual way to assess its efficiency.

e.g.: In our case, a good feature would be a computed number which would be very different for two genes whose mRNA locate very differently.

The features described below are the results of experimented biologists and bioinformaticians, it was not the object of this project to build them but they are included in this section for the sake of completeness.

### A.2.1 Simplified features

As we are interested in an ensemble of points and their colocalization, it looks like an elementary to look at distances in-between the points, as well as distances between the points and specific cell points (such as the center of the nuclei).  
e.g.:

RNAs of gene $g_1$	distance to nuclei	distance to cell membrane
$RNA_1$	200 nm	100 nm
...	...	...
average $g_1$	300 nm	100nm

One can then apply this protocol to several genes. This *feature*, however very simple, will be crucial for distinguishing genes whose RNAs locate at the cell boundary, from those whose RNAs locate at the nuclei boundary. At this point, raises an important warning about distance computations. One has to ensure that at every step of the process where computations are actually performed, that such calculus are *normalised* regarding the cell morphology. Those distances can for instance be divided by the radius of the circle that covers the same area as largest slice of the cell. It could also be divided by the longest segment which goes through the nuclei center, etc.

In [12], as much as 32 localization features were extracted from the cell images to characterize the mRNA's positions within the cells.

### A.2.2 Reconstruction of RNA foci

Some genes will see their mRNA locate in a non random manner and gather themselves into medium size aggregations of mRNAs, which will give very bright spot in smFISH images. Foci raises an important issue as the agglomeration of mRNA retains one to use the previously described spot detection method. Therefore foci are a complicated case to handle, which is the reason one has to model this pattern carefully.

Foci, being an agglomeration of individual mRNA can be modeled faithfully by a Gaussian Mixture Model, already described in eq.5. In this case, the number of components in the mixture model will be the number of foci in the cell, the average of each component will be a three dimensional vector of the foci's center position, and the  $\sigma^2$  of each component will represent the 'spread' or roughly the width of the foci. Implicitely, such GMM renders a density map of the cell from where points will be randomly sampled. mRNA will therefore concentrate on the chosen  $(\mu_x, \mu_y, \mu_z)$  points and density will decrease as distance to those points increases.

### A.2.3 Features related to the cell landscape

This family of features is based on the Euclidean distance between individual mRNAs and different parts of the cell (cell membrane, centroid of the cell, centroid of the nucleus, nuclear membrane), calculated in 2D. For instance, the

distance between a mRNA and the cell membrane is the the minimum euclidean distance between this point and the membrane points.

- $d_{nucleustocentroid,i} = \sqrt{(x_{nucleuscentroid} - x_{mRNA,i})^2 + (y_{nucleuscentroid} - y_{mRNA,i})^2}$
- $d_{celltocentroid,i} = \sqrt{(x_{cellcentroid} - x_{mRNA,i})^2 + (y_{cellcentroid} - y_{mRNA,i})^2}$
- $d_{celltoborder,i} = \min_{x,y \in CellBorder} \sqrt{(x - x_{mRNA,i})^2 + (y - y_{mRNA,i})^2}$
- $d_{nucleustoborder,i} = \min_{x,y \in NucBorder} \sqrt{(x - x_{mRNA,i})^2 + (y - y_{mRNA,i})^2}$

For each cell, we then obtain four sets of distance, to turn those sets into features (i.e. scalars) we compute their means and several quantiles along the cells. Those scalars are then normalized by meaningful quantities describing the cell morphology.

#### A.2.4 Features based on the KRipley function

The K-Ripley function allows one to quantify the level of aggregation or sparsity of an ensemble of points in the nd space. It basically compares the observed set of points with a completely randomly distributed distributed dataset (in other words, respecting the Complete Spatial Randomness, CSR, assumption). From an ensemble of  $E = \{x_i\}_{i \in [1, \dots, n]}$  points, the KRipley function can be computed as:

$$K(r) = \frac{1}{n} \sum_{i=1}^n \frac{|E \cap Disk(radius = r, center = x_i)|}{\lambda} \quad (8)$$

This feature is very useful when distinguishing foci of mRNA in cells.

#### A.2.5 Features based on morphological operations

The cell extension pattern was chacterized using morphological tools. A *morphological opening* was used here to "crop" the details of the cell. Put in very simple words, an *opening* will somehow remove the thinnest parts of the cell mask. This feature is built by successively *open* the cell mask and at each iteration count the number of mRNA which where let aside with regard to the previous iteration. See fig.37 for an illustration of the process.

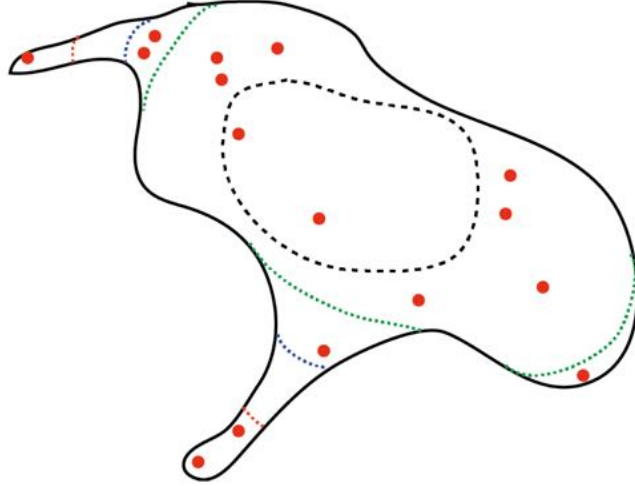


Figure 37: An illustration of the successive opening iterations on a cell mask.  
**Source:** [16]

### A.3 Attempts to generate cell landscapes

**Overview** Over the last four years, the field of Deep Learning has gained a lot of interest regarding *Generative models*. Such model are meant at generating new samples (never seen before) after trying to learn the probability distribution of given observation.

A famous approach is to build two models, one generating samples and the other deciding whether a sample is synthetic or truth, and to train such model face to face, iteratively, where at every timestep, samples generated by the so-called *generator* are mixed with real samples, and in the meantime, the *discriminator* tries to detect false samples. Doing this, one will hopefully end up with two models, one very good at generating trustworthy samples, and the other one good at detecting false ones.

Instead, we chose and experimented another approach, whose pitfall is to be less good at modeling complex distributions than the above, but is easier to train and implement. The family of model we chose is named *Variational Auto Encoder* and is highly inspired from standart AutoEncoder whose architecture can be described as below:

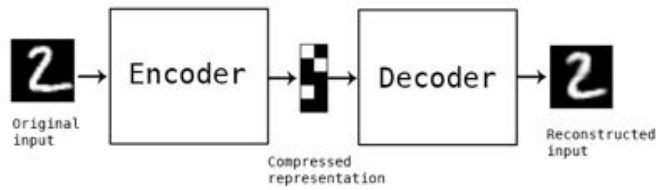


Figure 38: The encoder part compresses the data, while the decoder part decompresses it. The model is optimized so that the decompressed image is close to the original one.

**Source:** [keras.blog.io](https://keras.blog.io)

Encoder and decoder are often (for image data) a succession of convolutional and down sampling (or up-sampling) operations. Variational autoencoders, in addition, will try to enforce the compressed representation to follow a Standard Normal distribution. Therefore, one can hope to end up with a decoder which, when given a sample from  $\mathcal{N}(0, 1)$ , will return a real-like image.

**Obtained results** The hard part in such a generative model is that it needs to be very reliable. If for every generated cell, one has to build a test routine to ensure that the generated cell is good enough to be considered real, then the added value of such method is null. The results we obtained are, of course, far from this ideal but are encouraging enough to be shown in this part. Figure 39 illustrates successful and unsuccessful attempts to generate cytoplasm boundaries.

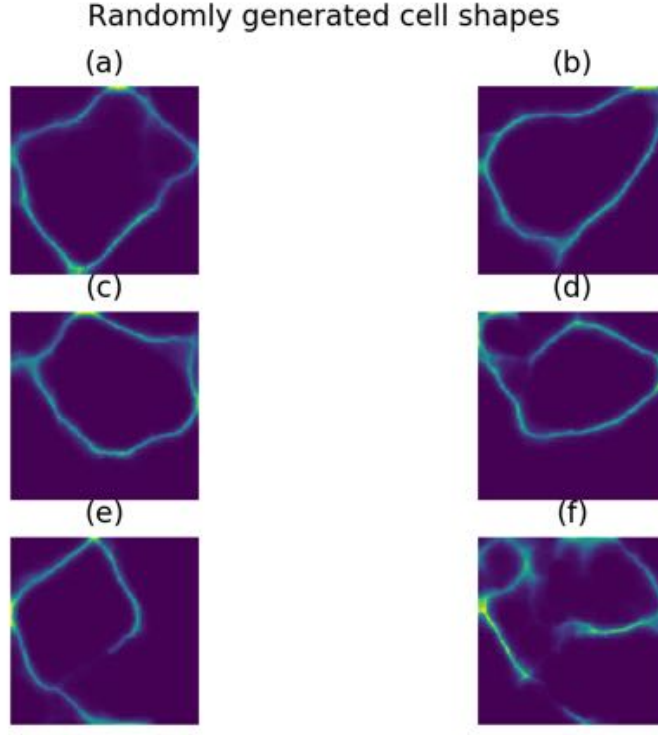


Figure 39: Examples of generated cells: (a) and (b) show very satisfying cell shapes. (c) and (d) show interesting but not necessarily successful attempts to generate cell shapes with cell extensions, while (e) and (f) show completely failed shapes where the boundary is not closed.

**Validity of the existing cell Library approach** In order to assess whether the model is learning features from the cell morphological landscape instead of the mRNAs positions, we ran tests where the network was successively fed with the cell mask signal only (CM\_only), DAPI signal only (DAPI\_only) and the smFISH signal only (RNA\_only). We witnessed that from the two first signal the network was unable to extract any discriminant features, asserting that the limited amount of cells is not tweaking our results. Furthermore, we saw that the network could not perform as good when only fed with smFISH signal versus when fed with the three signals, what shows that both cell landscape and mRNA positions matter in determining the localization patterns.





Figure 40: Overview of the network performances when fed with selected channels

## B Transfer Learning strategies

### B.1 Model choice

#### B.1.1 Inception Model

The Inception architecture was conceived based on the idea that in deep convolutional neural network, most of the neurons would be either unactivated most of the time, or would co adapt between each others. Therefore, according to this heuristic, the best way to transmit information from one layer of the network to the next one would actually be through sparse connections.

**Architecture** The Inception architecture was aimed at achieving state of the art performances while keeping the number of parameters relatively low. The version we used (version 3) contains a total of 23 million trainable parameters, while this figure depends on the number of possible classes at the output of the network. It remains a very deep neural network but stays very far behind its direct contenders (such as VGG16 or VGG19) with up to 155 million trainable parameters.

It is mainly composed of a chaining of similar modules which can be seen as "inner neural networks". Each of those modules will use convolutional layers at various scales (filters of sizes 1, 3, and 5) along with down sampling operations. In order to keep the number of parameters low, each module will reduce the number of input channels through 1x1 convolution filters. 1x1 convolutions plays the role of the described sparse convolutions by learning very low scaled features, while 3x3 and 5x5 convolutions will transmit blocks of information (dense). The module is depicted on figure 41.

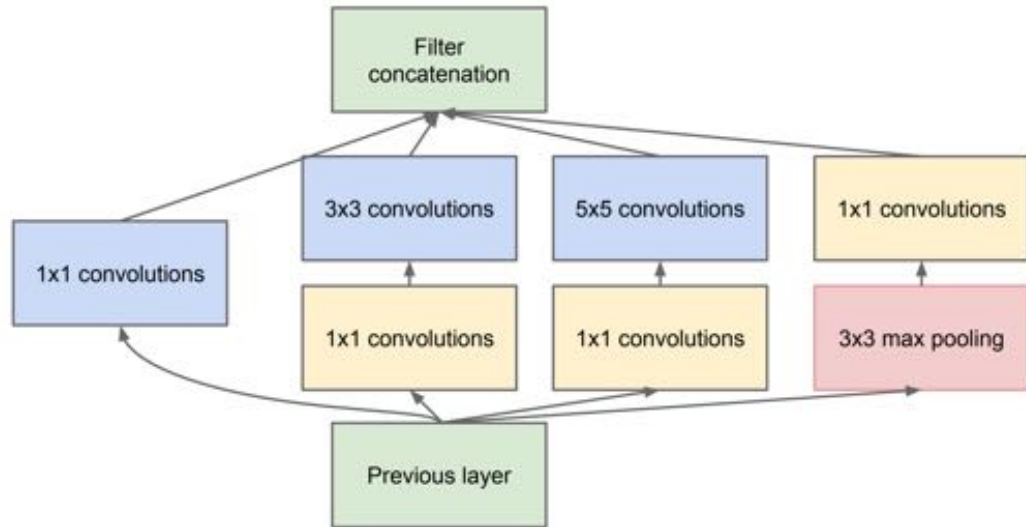


Figure 41: The module will apply convolutional filters of various sizes, after compressing the output of the previous layer through 1x1 convolutional layers which will hopefully 'select' useful features from the previous layers  
**Source:** [13]

The Inception model is made of a stack of such modules with occasional max pooling modules to halve the size of the grid. All in all, the ensemble is depicted in figure 42

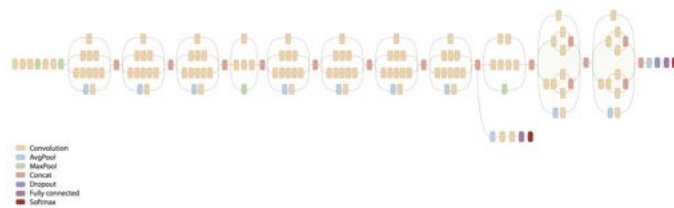


Figure 42: The whole inception architecture

### B.1.2 Comparison with SqueezeNet

**Versatility** Both model gained popularity on the ImageNet challenge but the Inception is, as of today, at the top of leader board of this challenge. Inception has become so popular that fully trained versions of the network are available

publically and can serve as a pre-trained base for many tasks. This aspect is very interesting as training such model takes hours on GPU machines which are not at anyone's reach. The Squeezenet do not show this big advantage but on the other hand, it can handle input of any size (down to a minimum size which is defined by the number of down sampling operations) and can be built with much more transparency than the Inception. Fire modules can be stacked as much times as desired and the only limitant factor is the input size (an input of small size should be treated with a low number of fires modules  $X$  downsampling while a big image can lead one to experiment bigger depth).

**Scalability (deployment)** A models' ease of deployment highly depends on its size.

**Inception** The total number of parameters of the Inception model we used (with height output classes) was 23,907,110. Such a high number of parameters implies a model's size (in storage) of roughly 300 MB. In the training phase, This model will eat as high as 5 GB of RAM memory which is very limitant because many machines (even GPUs) do not ship with enough RAM to train it properly (keep in mind that the machine's RAM also need to host data, software, python environment etc.)

**Squeezenet** On the other hand, SqueezeNet relies on a tiny 725,598 parameters (roughly 35 times less than Inception), weights around 9MB to store on disk and is below 500MB on machine's RAM during training phase. Due to this lower number of parameters, the SqueezeNet is also much faster at predicting and do not necessarily needs GPUs for this phase.

**Performance** Both model revealed surprisingly high accuracies for task. Input images are images where only boundaries are shown (boundary of nuclei and cell) and single-pixel points represent mRNAs. The following accuracy have been obtained after training both Squeezenet and Inception on a dataset of 130,000 synthetic images of six classes: Cell2D, Cellext, Foci, Nuclei2D, Polarized and Random. Those patterns where generated with varying strength in order to make the task difficult for the network. For each cell, a proportion  $p$  (varying as well) of mRNA was generated following the chosen pattern's feature, and the rest of the mRNAs where randomly generated. For some cells, up to 90% of the mRNAs were randomly located (while labeled as one of the 5 patterns different from *Random*). The graphs presented on figure 43 shows accuracy achieved by both models. Note that the Inception achieved the same accuracy than the SqueezeNet in just a couple of hours, due to the pretrained weights. Note also that the Inception's accuracy shows high volatility, what can be explained by the very small batch size (10 images) we had to use because of the model's size.

This graph was obtained by training both models on two identical NVIDIA

TESLA P80 (16GB RAM). For indicative purposes, such machine usually leverages a computational speed 15 times higher than recent laptop’s CPU.



Figure 43: Validation accuracy (%) obtained on a synthetic dataset,  $x$  axis indicates training time (in hours)

### B.1.3 Training details

When training our model, we tried different strategies regarding how to handle the different channels. Namely, we tried:

1. Concatenating all the information into one channel (nuclei, cell and mRNAs in the same 600x600 image)
2. Concatenating as above and duplicating this channel three times.
3. Splitting the biological channels into three different channels (nuclei, cell and mRNAs in this order)

The latter strategy yielded much better performances and a faster convergence, we believe that this strategy allows the model to better choose information. Indeed, in this case, giving importance or not to the nuclei signal can be taken care of in the channels dimension (third dimension of a convolutional layer) which is faster than computing the 2 dimensional kernels of each channel.

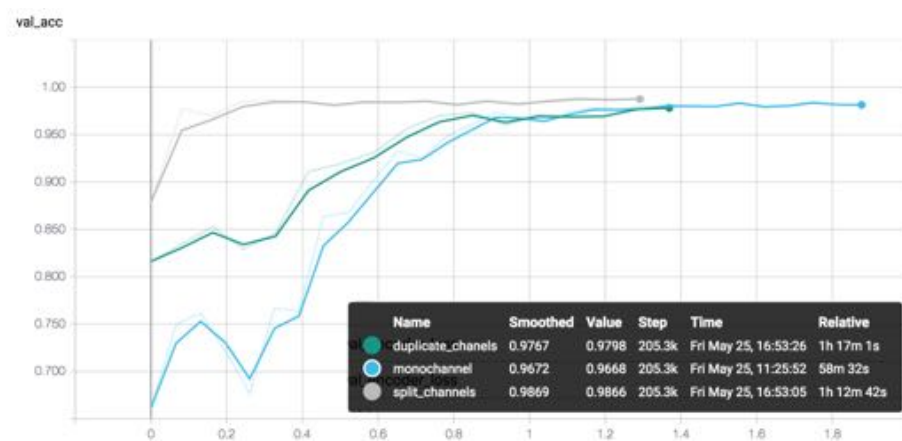


Figure 44: Validation accuracies yielded by the three different channel strategies on a light dataset (25,000 images). The x axis represents the training time in hours.

## List of Figures

1	Global overview of the whole gene expression process . . . . .	2
2	Marking mRNA with fluorescent probes . . . . .	3
3	Three examples of localization patterns: (a) Random, (b) Cell Extensions, (c) Agglomeration into bright spots . . . . .	3
4	A 2-dimensional representation of a population, where each group would represent a "cluster", or group of similar individuals (gene in our case). . . . .	4
5	. . . . .	5
6	Results obtained on the ImageNet challenge, showing the raise of Deep Learning in the field of Computer Vision . . . . .	6
7	Different transfer learning configuration. This leftmost case is the canonical machine learning situation (no transfer learning). Our study focuses on Domain Adaptation. . . . .	8
8	General work flow of spot detection systems . . . . .	10
9	Histogram of pixel values, highlighting sources of noise . . . . .	11
10	Zoomed image with mRNAs circled in red . . . . .	12
11	An example of successful nuclei segmentation using Otsu's method, with details magnified. . . . .	13
12	Histogram of pixel values (blue), along with the inter-class variance (red) . . . . .	14
13	Raw cell mask image (left), and the segmented cells (red, right). . . . .	15
14	Thresholding through a Gaussian Mixture Model fitting, Otsu threshold in blue for comparison . . . . .	16
15	3D visualization of a 2d slice of a cell mask image, showing two merged cells . . . . .	17
16	Crest of the 3d surface . . . . .	18
17	Watershed flooding . . . . .	18
18	Original crest (blue) with the nuclei positions values put to minimum (orange). . . . .	19
19	Comparison a naturally observed (right) and a synthetic (left) Cell2D pattern . . . . .	22
20	Example of a fire module, which compresses the input information into 16 feature maps, and then expands it into 128 feature maps built with 1x1 and 3x3 filters . . . . .	25
21	Ensemble architecture with skip layers: input to module 7 is output of module 6 added to module 5's output. Colored figures along connections indicate the number of channels. . . . .	25
22	2-dimensional visualization of features extracted of the synthetic data by our model. The different classes are clearly separated except the Random class, which is harder to classify. . . . .	27
23	Pattern-strength color-coded TSNE of synthetic images . . . . .	28
24	2-dimensional visualization of features extracted of the real data by our model. . . . .	28

25	Classes predicted for four genes. Each gene is classified into several patterns at least once. . . . .	29
26	Illustration of our domain adaptation framework with a gradient reversal layer applied to the domain classifier branch of our model	30
27	The model will be trained on the source domain in order to predict classes of images from the target domain. . . . .	32
28	Feature representation of source domain images (blue) and target domain images (red), with (right) and without (left) domain adaptation. . . . .	32
29	The modified architecture of our model, with an additional domain classifier in parallel of the standard localization pattern classifier . . . . .	33
30	Detail of randomly placed mRNA, note that mRNAs are not placed inside the nuclei . . . . .	35
31	Detail of mRNA placed close to the nuclear membrane (between the dashed blue line and nuclear envelope) . . . . .	36
32	Different pattern strength of the 2d nuclear edge pattern, along with an image of the ASPM gene which was observed to locate the same way. . . . .	37
33	Vizualizing the cell in only 2D drastically limits the chance to distinguish this pattern from <i>Random</i> . . . . .	37
34	Density of RNAs is highest in between the cell membrane and the blue dashed line. This pattern is 2D pure as opposed to <i>3D nuclear membrane</i> . . . . .	38
35	mRNAs are preferably placed in one part of the cytoplasm . . .	39
36	Details of cell whose extension have been manually enriched with mRNAs, along with the <i>RAB13</i> gene whose mRNA have been observed to locate following the same pattern . . . . .	40
37	An illustration of the successive opening iterations on a cell mask.	43
38	The encoder part compresses the data, while the decoder part decompresses it. The model is optimized so that the decompresses image is close to the original one. . . . .	44
39	Examples of generated cells: (a) and (b) show very satisfying cell shapes. (c) and (d) show interesting but not necessarily successful attempts to generate cell shapes with cell extensions, while (e) and (f) show completely failed shapes where the boundary is not closed. . . . .	45
40	Overview of the network performances when fed with selected channels . . . . .	46
41	The module will apply convolutional filters of various sizes, after compressing the output of the previous layer through 1x1 convolutional layers which will hopefully 'select' useful features from the previous layers . . . . .	47
42	The whole inception architecture . . . . .	47
43	Validation accuracy (%) obtained on a synthetic dataset, <i>x</i> axis indicates training time (in hours) . . . . .	49

44	Validation accuracies yielded by the three different channel strategies on a light dataset (25,000 images). The x axis represents the training time in hours. . . . .	50
----	---	----



## References

- [1] *Global Analysis of mRNA Localization Reveals a Prominent Role in Organizing Cellular Architecture and Function*. EricLécuyer, HidekiYoshida, NeelaParthasarathy, ChristinaAlm, TomasBabak, TanjaCervova, Timothy R.Hughes, PavelTomancak, Henry M.Krause.
- [2] *mRNA Localization: Gene Expression in the Spatial Dimension*. Kelsey C.Martin, Anne Ephrussi.
- [3] *The intracellular localization of messenger RNAs* St Johnston D.
- [4] *smiFISH and FISH-quant – a flexible single RNA detection approach with super-resolution capability* Tsanov N, Samacoits A, Chouaib R, Traboulsi A, Gostan T et. al.
- [5] *Quantitative comparison of spot detection methods in fluorescence microscopy* Smal I Loog M Niessen W Meijering E.
- [6] *QUANTITATIVE COMPARISON OF SPOT DETECTION METHODS IN LIVE-CELL FLUORESCENCE MICROSCOPY IMAGING* Smal I, Loog M, Niessen W, Meijering E, Imaging B, et. al.
- [7] *Dermatologist-level classification of skin cancer with deep neural networks* Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau & Sebastian Thrun
- [8] *A Deep Learning Approach for Tumor Tissue Image Classification* Xulei Yang, Si Yong Yeo, Jia Mei Hong, Sum Thai Wong, Wai Teng Tang, Zhen Zhou Wu, Gary Lee, Sulin Chen, Vanessa Ding, Brendan Pang, Andre Choo, Yi Su.
- [9] *Deep learning for tumour classification in homogeneous breast tissue in medical microwave imaging* Branislav Gerazov, Raquel C. Conceicao.
- [10] *Fast and accurate automated cell boundary determination for fluorescence microscopy* Arce S, Wu P, Tseng Y.
- [11] *Spatially resolved, highly multiplexed RNA profiling in single cells*. Kok Hao Chen, Alistair N. Boettiger, Jeffrey R. Moffitt, Siyuan Wang, and Xiaowei Zhuang. Science, page aaa6090, April 2015.
- [12] *Image-based transcriptomics in thousands of single human cells at single-molecule resolution*. Nico Battich, Thomas Stoeger, and Lucas Pelkmans. Nature methods, 10(11):1127–1133, November 2013.
- [13] *Going deeper with convolutions* Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich

- [14] *Squeezenet: AlexNet-level accuracy with 50x fewer parameters and 10.5MB model size* Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf1, William J. Dally, Kurt Keutzer
- [15] *Unsupervised Domain Adaptation by Backpropagation* Ganin Y, Lempitsky V.
- [16] *Development of a statistical analysis framework for sub-cellular mRNA localization* Aubin Samacoïts, Ph.D thesis.
- [17] *Global Analysis of mRNA Localization Reveals a Prominent Role in Organizing Cellular Architecture and Function* Eric Lécuyer, Hideki Yoshida, Neela Parthasarathy, Christina Alm, Tomas Babak, Tanja Cerovina, Timothy R.Hughes, Pavel Tomancak, Henry M.Krause.
- [18] *VISUALIZING THE LOSS LANDSCAPE OF NEURAL NETS* Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, Tom Goldstein