POLITECNICO DI TORINO

**Master's Degree Course in Biomedical Engineering**

Master's Degree Thesis

# IMPROVING RECALL OF *IN SITU* SEQUENCING BY SELF-LEARNED FEATURES AND CLASSICAL IMAGE ANALYSIS TECHNIQUES

Centre for Image Analysis - Uppsala University
Politecnico di Torino


**Supervisor:**
*Prof. Elisa Ficarra*

**Candidate:**
*Giorgia Milli*

**Co-supervisors:**
*Prof. Carolina Wählby*
*Gabriele Partel*

A.Y. 2017/2018

# Abstract

Image-based sequencing method to decode mRNA fragments directly in fixed tissue samples allows to carry out the gene expression profile preserving morphological and spatial information of cells and tissues. This approach called *in situ* sequencing makes it possible to directly visualize, at sub-cellular resolution, where in a tissue sample a given gene is active, to quantify its expression, and to distinguish among many different cell types at the same time. Such information are fundamental to gain a better understanding about tissue and disease development (such as cancer) and cells interplay. Since each gene is composed by a specific sequence of bases, the search is addressed to targeted sequences which must be decoded over multiple staining and imaging cycles, and retrieved by processing multichannel fluorescent biological images of the analyzed samples. However, signal density, high signal to noise ratio, and microscopes' resolution limits make decoding challenging. The state-of-art approach for signal decoding has led to low signal recall in efforts to maintain high sensitivity. The main issues related to the state-of-art technique concern difficulties in distinguishing signals in more dense regions, the lack of a proper handling for local misalignments among signals belonging to different sequencing cycles and the inability of processing 3D datasets. In this thesis a new approach has been implemented in order to face the state-of-art issues and increase recall at maintained sensitivity. Here signal candidates are included in the first processing steps and provided with their true-signal probability by an opportunely trained classifier. Signal candidates and their probability predictions are then fed to a decoding approach searching for signal candidates across sequencing cycles. Finally, the decoded sequences are provided with a quality measure indicating their reliability based on the classifier probabilities. In order to find the best solution, either a support vector machine and convolutional neural network have been tested as classifier. A window-based search has been designed for the

sequence decoding. The developed sequence decoding method looks for the optimal paths representing the decoded signal sequences by combining intensity, probability and spatial distance. Multiple quality metrics have been tested to find out which one allowed to obtain the highest signal recall. All the possible combinations of the new proposed pipeline have been evaluated in relation of the state-of-art. Using the support vector machine as classifier has led to a consistent decrease in signal recall (20%) compared to the state-of-art pipeline. On the other hand, using the convolutional neural network has led to an improvement (31%). The obtained results demonstrated an evident advantage in using a classifier based on self-learned features and the need of a sequence decoding approach less dependent on signal probability predictions. The new proposed approach solves all the state-of-art issues and has the potential of significantly improve further analysis of spatial statistics in *in situ* sequencing experiments.

*To my Family*

# Contents

# Chapter 1

# Introduction

Cell behaviours, functions and nature determine the most of the organism-level functions. In turn, each of these cell characteristics depends on cell gene makeup. Consequently, measuring gene expression is the key to gain a better understanding of organ functions and of a tissue or a disease development. The information retrieved from this analysis opens the possibility towards the definition of new targets for therapy and prognosis [1]. The necessity of gene expression profiling is widely recognized and important progress has been made in terms of specificity, sensitivity, multiplexing, throughput and cost reduction.[2] Traditional next generation sequencing (NGS) technologies attempted to extract gene expression profiling through bulk analysis of tissue samples. However, this approach can only give an average of the gene expression profile of a tissue sample and cannot be used to deduce information at transcript-level resolution [3]. NGS technologies limitations pushed towards the invention of methods addressing the analysis at single cells level, thus decomposing the complexity. These techniques, called Single-cell RNA sequencing (scRNA-seq) methods, allow studying the whole transcriptome of individual cells, but they require the extraction of the targeted cells from their natural context. Isolating cells leads to loose spatial and morphological information, which are fundamental to understand the existing connections between the morphology and the mRNA sequences [1]. The issues related to these techniques highlighted the need of more comprehensive *in situ* transcriptional analysis, in order to obtain results at transcript-level resolution also preserving morphological and spatial information. In this sense, nowadays a certain number of techniques for spatially resolved transcriptomics [4] are available to fulfill this not trivial task. In

particular, *in situ* sequencing [5] method turned out to be a powerful tool to quantify the expression of targeted genes directly in biological tissue samples without loosing morphological and spatial information. In this technique a computerized image analysis pipeline is necessary to analyze and process the output in order to map the gene expression profile of the targeted genes across the tissue slide. However, some difficulties such as low signal-to-noise ratio (SNR), weak intensity values of the objects to analyze, and noise components like those related to the different fluorophores used during sequencing, make the gene expression decoding challenging. Hence, distinguishing if signal candidates are true signals or noise is not an easy task, and this choice cannot depend on criteria based only on intensity values.

The actual state-of-art (SoA) image analysis pipeline [2] for *in situ* sequencing uses classical image analysis methods, such as global thresholding and watershed segmentation, to extract single fluorescent signals composing the sequences and decode the gene expression of the targeted genes. These methods, uniquely based on the intensity values of single pixels, make the SoA image analysis pipeline very straightforward, and allow to achieve the intended goal of sequence decoding. However, the major issue of this approach is that lot of the available information is lost in order to maintain a high sensitivity, leading as consequence to poor performances in terms of recall.

The work carried out in this thesis aims to overcome the issues of the current SoA pipeline, thus increasing signal recall at maintained sensitivity. Improving recall of *in situ* sequencing technique means significantly improve further analysis of spatial statistics on *in situ* sequencing experiments [6].

## 1.1 Fluorescent microscopy

In fluorescent microscopy, the observed samples are stained with a specific fluorescent dye and then illuminated with an excitatory light having a shorter wavelength. Once excitation is over, the fluorescent dye emits light of a longer wavelength which results in fluorescence [7]. This response is therefore measured to produce an image. Fluorescent microscopy is generally used to image sub-cellular structures such as DNA, RNA, proteins etc. These entities can be specifically labeled with fluorescent probes and biomarkers (for

instance fluorescent stains such as DAPI and Hoechst are used to stain cell nuclei). Hence they can be observed through fluorescence microscope as bright spots emerging from a dark background [8]. This technology allows to map in detail subcellular structures and to estimate their locations. The information about subcellular structures locations can be taken as starting points to track these entities over the time, which is useful to determine their biological function. Fluorescence microscopy makes use of the difference in excitation and emission wavelengths to block the incident light. This results in an image with high contrast between sample and background [9]. The digital image of the sample is obtained thanks to a camera mounted on the microscope. A digital image is the digital version of a captured scene, and consist of very small entities, called pixels, whose number depends on the sensor used to capture the scene. Once the digital images have been acquired, the following challenge is to analyze them. To achieve this purpose computerized analysis represents a powerful tool. Computerized analysis uses programs and algorithms on digital computer to analyze the data and it is fundamental to obtain a quantitative and unbiased analysis [8].

## 1.2 Challenges

The major challenges in image analysis are related to different sources of noise characterizing fluorescence microscopy that can be summarized in:

- **Poisson noise:** this noise origins from the stochastic arrival of the photons at the camera chip which follows Poisson counting statistics. It is equal to the square root of the number of the photons collected, and it cannot be eliminated but only minimized. In particular, its effect can be reduced by augmenting the exposure time, hence increasing the SNR [10];

- **cross-talk:** if two or more fluorescent dyes are excited by or emit at the same wavelength [11], then more than one emission dye will be imaged resulting in a mix of information. This phenomenon is called signal cross-talk;

- **background noise:** this noise is given by the sum of Poisson noise and camera noise. Camera noise mainly consists in read noise, which derives from imprecision of

each photoelectron charge packet by the read amplifier of the camera. Background noise results in a unpredictable variation in the gray values of the image;

- **autofluorescence:** many organic molecules are naturally fluorescent, and thus even unstained biological samples can emit fluorescence in the visible domain. Autofluorescence is an important source of noise when it overlaps with the emission of a selected fluorophore, especially when the latter is sparsely expressed or exhibits weak fluorescence. This interference can render the detection of a signal very difficult [12];

- **residual fluorescence:** this source of noise can occur during sequential fluorescence labeling on the same sample. Fluorophores that are inaccessible to washing procedures are not cleavaged out, and will emit fluorescence during the next excitation round.

## 1.3 Thesis outline

In this thesis will be presented a new image analysis pipeline to overcome the issues related to the SoA pipeline of the *in situ* sequencing context. First of all, in Chapter 2 will be given some notions concerning the methods available today for spatially resolved transcriptomics. In this chapter the *in situ* sequencing technique will be presented both in methodological approach used to acquire images and in SoA image-based pipeline used for data analysis. Moreover, the issues related to the SoA image-based pipeline will be discussed in detail. In Chapter 3 will be given some notions regarding the computational technologies used in this thesis. In particular, will be presented the support vector machine (SVM) classifier and fundamentals of deep neural networks, focusing on a specific type of deep net called convolutional neural network (CNN). In addition, will be introduced some useful techniques for parameter tuning. Chapter 4 will address to the pipeline implemented in this thesis, discussing each step in detail. Chapter 5 will be dedicated to present and discuss the obtained results after a detailed presentation of the *in situ* sequencing datasets used to implement and evaluate the developed pipeline. Furthermore, a paragraph will be dedicated to the results related to parameter setup and optimization. Finally, conclusions and future works will be discussed in Chapter 6.

# Chapter 2

# Biotechnological background

The majority of organism-level functions are driven by the action of cells having different nature. In turn, each cell behaviour, function and type are determined by its gene makeup [5]. Thus, measuring gene expression allows to better understand organ functions, and a tissue or a disease development (such as cancer) consequently defining new targets for therapy and prognosis. However, fulfilling this task is not trivial. In this chapter will be presented the technologies available today to carry out gene expression profiling on fixed tissue samples, focusing in particular on *in situ* sequencing technique [5] upon which this thesis has been developed. Finally will be introduced the SoA image analysis pipeline [2] for gene expression decoding and its related issues to highlight the need of a new image-based approach.

## 2.1 Introduction to spatially resolved transcriptomics

Traditional next-generation sequencing (NGS) methods attempt to study the biological structures functions by measuring gene expression in extracts after tissue homogenization. These approaches bring to imperfect outcomes, because the bulk analysis can only give an average gene expression profile of tissue samples. Thus, this result cannot be used to deduce accurate information concerning the transcript-level molecular state of the various cell types composing a tissue [3]. To overcome this issue it is necessary to address the analysis to single cells, thus decomposing the complexity. In this sense, NGS-based scRNA-seq technology makes it possible to study the whole transcriptome of individual

cells. However, the majority of scRNA-seq methods require to isolate single cells from their natural context by enzymatic or mechanical dissociation [1]. Moreover, the contextual information is only retrieved for the targeted analyzed cells and not for their surrounding neighbor cells forming the microenvironment. Hence, all the spatial information and the connection between the morphology and the mRNA sequences are lost. Furthermore, since the sample might bias the results and the isolation procedure itself, this might affect cellular gene expression. Thus, the outcomes from the scRNA-seq should not be taken as representative of the cell compartment targeted for expression profiling [5]. The issues related to these techniques highlight the need of more comprehensive *in situ* transcriptional analysis to achieve a better understanding of how complex biological systems operate. Recently a variety of technologies have been developed to analyze high throughput of RNA transcripts across fixed cells in their natural context. These methods for spatially resolved transcriptomics are being applied successfully, making it possible to relate molecular characteristics of different cell types with their morphological and functional environment. Spatially resolved transcriptomics techniques [4] are presented below divided in two categories.

### Targeted approaches

The techniques belonging to this group aim to resolve gene expression at sub-cellular resolution by tagging the sequences of interest with different approaches. A widely used technique to localize nucleic acid targets is the *in situ* hybridization (ISH). This approach allows to visualize DNA and RNA molecules in their cellular context by hybridizing target complementary probes to nucleic acids in fixed tissues. An improvement of ISH is the single-molecule RNA fluorescence ISH (smFISH) technique, which provides a direct estimate of the number of RNA molecules present in a cell by labelling the target complementary probes with a fluorophore [13]. The limited multiplexing capacity of these techniques is augmented by increasing the number of spectrally distinct fluorophores available through combinatorial labeling using two-colour spectral barcodes [1] [14]. Another

[1] by combining more fluorophores it is possible to greatly expand the number of different targets distinguishable in a field of view (i.e. if the number of dyes is 6 and the combination size is set to 2 there exist 15 different combinations).

approach for multiplexing smFISH is sequential hybridization (seqFISH) together with combinatorial labeling. In this case a single transcript is detected multiple times using different coloured probes in each hybridization cycle [15][16]. However increasing the multiplexing capacity means that a huge number of target-specific FISH probes are required. Furthermore, it must be considered that the sequential hybridization rounds are time consuming. A further targeted approach is the multiplexed error-robust FISH (MERFISH) [17]. In this case, the encoding-probes label the RNA thanks to the presence of a specific combination of readout sequences flanking the RNA-target complementary sequence. Indeed, these readout sequences create an error robust binary code with minimum Hamming distance of 4. Since each binary code is uniquely associated to an individual RNA, the fluorescent probes are used to detect the readout sequences in several hybridization rounds.

### *In situ* RNA sequencing

The approaches belonging to this category carry out the RNA sequencing *in situ* at single-nucleotide resolution. Differently from the targeted approaches previously described, sequences are revealed by retrieving the bases composing them one at a time. Hence, fluorophores are associated each one to a different base and not to whole sequences. The first method is the targeted *in situ* sequencing [5], which is a technique promoting the generation of targeted, multiplexed expression profiles within fixed cells or tissues at subcellular resolution. It consists in an image-based approach that allows to directly visualize where in a tissue sample a given gene is active, to quantify its expression and to distinguish among many different cell types at the same time. To decode the sequences, fluorescent signals are collected over multiple staining and imaging cycles and retrieved by processing multichannel fluorescent biological images of the analyzed samples. The study carried out in this thesis refers to this method, hence the sequencing technique to retrieve information and the state of art (SoA) image analysis approach used for sequence decoding will be presented in detail in the following paragraphs. A limitation of this approach is its moderate detection efficiency which derives from the low efficiency during the reverse transcription step *in situ*. However, some meaningful advantages compensate for this leak. Indeed, this technique has a general high specificity that allows the target recognition with a single-

nucleotide resolution, and a high signal to noise ratio in the fluorescence images which permits using wide-field imaging consequently reducing experimental costs and increasing the imaging throughput. The second available approach is called fluorescent *in situ* sequencing (FISSEQ) and it is an offshoot of the previous method [18]. The difference is that in this case the sequencing libraries are generated in an untargeted way and longer sequencing reads are produced. The limitations of this method rely in low-sequencing depth, due to rRNA contamination, and on optical crowding, due to the size of amplified RNA molecules which limits the number of reads per cell and consequently the sensitivity of this technique.

## 2.2 *In situ* sequencing analysis

The *in situ* sequencing method [5] is based on padlock probing, rolling-circle-amplification (RCA) and sequencing-by-ligation chemistry. The combination of RCA with padlock probe circularization is used to create a high density of clonally amplified rolling-circle products (RCPs) in cells and tissue sections at micrometer spatial resolution. This allows detecting and genotyping individual mRNA molecules *in situ*. In particular, two targeted approaches, called gap and barcode targeted, can be used. In figure 2.1 both approaches are represented. The procedure consists in three main steps, repeated as many times as the length of the sequence of interest.

(a) mRNA is copied to cDNA by Reverse transcription **(i)** and then degraded **(ii)**. In case of gap-targeted sequencing **(iii)** the padlock probe binds to the cDNA template with a gap between its two ends, in correspondence of the bases to be sequenced. By filling this gap through DNA polymerization and DNA ligation, a DNA circle is obtained. In case of barcode-targeted sequencing **(iv)**, instead, a DNA circle from the padlock probe carrying a barcode sequence is obtained simply through ligation. **(v)** the DNA circle is amplified through target-primed RCA generating an RCP. This RCP is then subjected to sequencing by ligation using interrogation probes in order to decipher it. Hence, an anchor primer sequence, which consists of four libraries (one for each base, each labeled with a different fluorescent dye) of 9-mers with eight random positions (N) and a fixed one ($A$, $C$, $G$ or $T$), is hybridized next

to the targeted sequence (red fragment). The interrogation probe that best matches the fixed position is incorporated by ligation with its flourophore.

(b) the sample is then imaged through brightfield fluorescence microscopy using dye-specific filters. This is because each RCP displays the dye color associated to the library of the matched base. A series of fluorescent images are recorded at different focal plans. In particular, in 2D datasets, those images are then combined via maximum-intensity projection (MIP) to capture signals from RCPs belonging to different focal plans. After the image acquisition, the interrogation probe is washed away before the application of the interrogation probes for the following base. These steps of ligation, imaging and washing are repeated until the desired number of bases has been read.

(c) base-calling is finally carried out by an image analysis approach. Fluorescent signals are decoded across the fluorescent channels and hybridization cycles and the gene expression is mapped onto the tissue sample (paragraph 2.4).

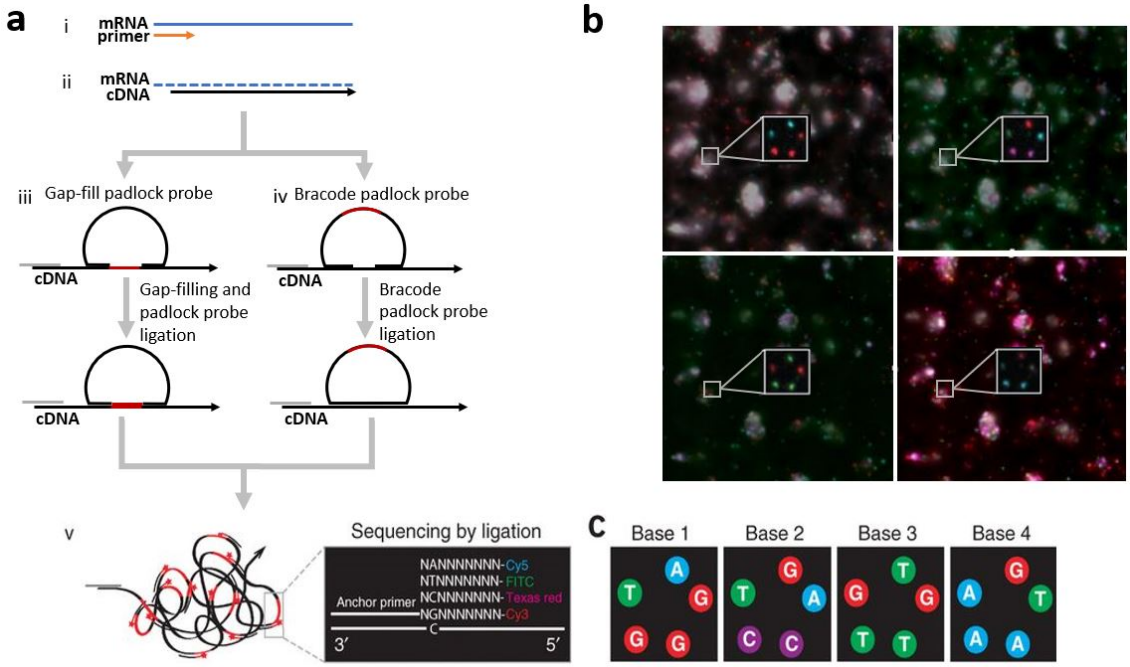



**Figure 2.1:** Procedure for targeted *in situ* sequencing for four-base-long sequence

## 2.3 *In situ* sequencing dataset

Before discussing about the SoA image analysis pipeline used to carry out the base calling, it is important to understand how a typical *in situ* sequencing dataset is structured. Generally, it is composed by six fluorescent channels:

- four channels each one related to a different biological base (*T, G, C* and *A*). In particular, since each base is related to a different flourophore (respectively, FITC, Cy3, TexasRed, Cy5), those channels are called colour channels;
- one DAPI channel to visualize cell nuclei. This channel carries no signal information, but it can be useful to localize cells;
- a last channel, called general stain channel, marks all sequenced RCPs with a different independent fluorophore (Cy7). Consequently, this channel carries the combined information contained in the four colour channels. The general stain channel is optional, hence it can be either present or not.

A fluorescent image is acquired for every channel at each hybridization cycle. Hence, these channels will be filled by their relative image as many times as the length of the sequence to be sequenced (i.e. four-base-long sequence consists in a dataset of $6 \times 4$ fluorescent images, where 6 is the number of the channels and 4 the number of hybridization cycles) (Figure 2.2). The information of interest in these images consists in bright fluorescent spots (generally round shaped) with average radius of 1 $\mu m$ that, due to the diffraction limit of the microscope, results in 3 to 7 px size each. Starting from this dataset, the decoding task consists in connecting among them all the signals in the colour channels along the sequential hybridization cycles according to their position in order to identify the targeted sequences. However, high signal density, noise (such as auto-fluorescence of the tissue, signal crosstalk [2] and fluorescence residual [3]) and local misalignments of signals among hybridization cycles make decoding challenging. Furthermore, signals have

[2] due to overlapping spectra of fluorescent channels

[3] due to imperfect washing procedures among hybridization cycles

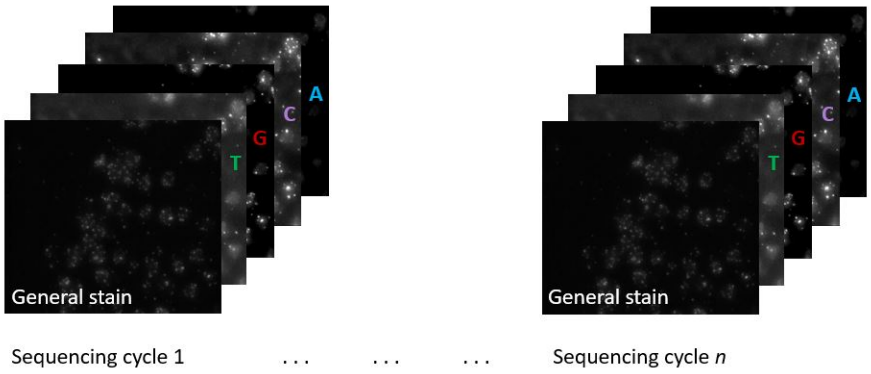



**Figure 2.2:** dataset structure shown through General stain and colour channels for each sequencing cycle

typically a blurry appearance, non clear border and weak intensity that makes it easy to confuse them with noise.

## 2.4 State of art pipeline for base calling

Once the dataset has been acquired, it is processed through the image analysis pipeline in order to decode targeted sequences and spatially resolve the gene expression in the tissue slice. The SoA pipeline for sequence decoding [2] consists in the following steps.

1. the images belonging to the general stain and to the four colour channels are filtered through a morphological grayscale Top-Hat filter using a disk as structural element. This step is meant to attenuate the background contribution while enhancing signal information;

2. after being filtered, images from each hybridization cycle are registered to the general stain of the first sequencing cycle taken as reference image. For each cycle, the combined image ($CH$) of the colour channels is obtained as

$$CH = \max(A, T, C, G) - average(A, T, C, G) \tag{2.1}$$

and aligned to the reference image. From this registration the transformation matrix is retrieved and multiplied for each colour channels belonging to the current cycle.

The registration is carried out through an automatic sub-pixel approach minimizing the mean square difference of the intensities between the reference and the moving images;

3. at the end of the registration procedure, the individual fluorescent spots are extracted according to the filtered image of the general stain through a watershed segmentation. Foreground/background threshold was set empirically;

4. the resulting segmentation mask is then used to measure the fluorescent intensity while extracting the coordinates of RCPs centers in each registered colour channel;

5. for each hybridization cycle, each RCP is assigned the base (*A, T, C* or *G*) having the highest intensity within the segmentation mask defining the RCP positions. RCPs located at the same position along the hybridization cycles are merged to form the sequence.

6. at this step, false signals (i.e. deriving from background noise like auto-fluorescence) may have been included. Thus, to extract only the most reliable signals, a quality score $Qb_{ik}$ is assigned to each base $j$ of each RCP $k$ at each cycle $i$ as following:

$$Qb_{ik} = \frac{I_{ik}}{\sum_{j\in(A,C,G,T)} I_{ikj}} \tag{2.2}$$

where

$$I_{ik} = \max(I_p), \forall p \in \Omega_{ik}$$

where $\Omega_{ik}$ is the set of pixels belonging to RCP $k$ at the cycle $i$. Once each base composing the RCP $k$ is provided with its own quality, the quality score of the full sequence $Qs_k$ of RCP $k$ is defined as the quality score of its weakest base as:

$$Qs_k = \min(Qb_{ik}), \quad \forall i \in s \tag{2.3}$$

where $s$ is the full sequence.

The above presented approach allows to decode sequences and to obtain the local genetic expression. However, it also leads to loose lot of the available information. Indeed,

the watershed might result in an undersegmentation in order to avoid picking up noise and to ensure that all the RCPs detected and used to form sequences are real signals. Consequently, regions of clustered signals result merged into a single detection, and weak signals (i.e. having an intensity value similar to noise) are lost. Another issue that can lead to loose information relies into the method used to merge the RCPs to form the sequences. The segmented mask of the general stain of the first hybridization cycle is used as reference to locate RCPs and merge them into sequences. This means that RCPs coordinates are strictly defined by the segmented mask and if the peak of an RCP results to be even just one pixel shifted respect to the reference coordinates of the mask it might be not considered and excluded. Even if the registration procedure gives good results, local shifts of RCPs between sequencing cycles cannot be compensated, thus local misalignments can occur and must be handled to improve sequence decoding. One last source of error is related to the quality metrics. It is worth noting that it is only based on intensity values and this can lead to wrong evaluations. Indeed, there are lot of artifacts having high intensity value that will be considered as high quality signals and, on the other hand, weak signals that will be considered as low quality signals. Moreover, the whole sequence quality is determined by its weakest signal. This means that one weak signal in a sequence is sufficient to make the sequence not reliable because of its $Qs$ value and thus to be discarded. In conclusion, it is evident that these issues cause a huge lost in the amount of sequences decoded, revealing the need of a finer and more flexible approach.

# Chapter 3

# Background on computational technologies

Machine learning and deep learning algorithms are demonstrating to be among the most powerful technologies available today for a large variety of different tasks. Among the interested fields there is also image analysis, for which the contribution of those techniques is becoming more and more fundamental.
This chapter is meant to give some basic technical concepts concerning the two classifiers used in this thesis. First, it will be presented a classical machine learning classifier, called support vector machine, and second, some generalities of deep learning models with a particular focus on convolutional neural networks.

## 3.1 Support vector machines

SVM [19] are non-probabilistic linear classifiers specially intended for binary classifications. Depending if a dataset is linearly separable in its classes of belongingness or not, a linear SVM or a non-linear SVM classifier is used. Both cases are discussed in detail below.

### 3.1.1 Linear support vector machines

Considering a two-classes problem with linearly separable data, the way those classifiers get the optimal classification consists in maximizing the area (**margin**) lying between the

two classes. The width of this region is the distance allowing the discrimination of the two classes, and the datapoints from each class which are hitting the discriminating surface are called **support vectors**. Thus, the discrimination function is only determined by the support vectors, while other training examples aren't meaningful for this purpose.
An example of two-classes discrimination is given in figure 3.1. Plotting the data on the

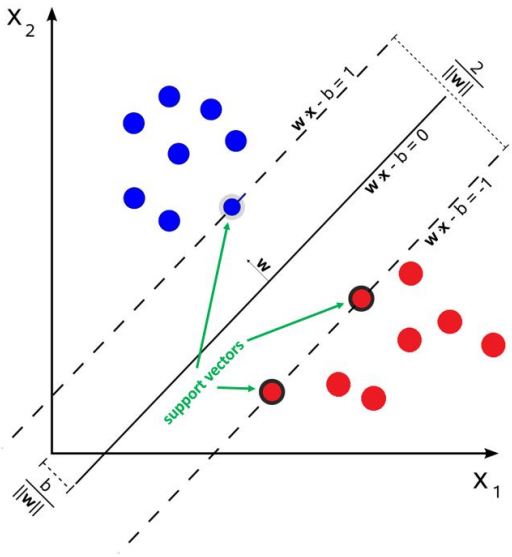



**Figure 3.1:** SVM Hyper-planes with 2 example classes

plane defined by their feature vectors $x_1$ and $x_2$, discrimination is achieved by defining a separating hyperplane that maximizes the margin between the two classes in the feature space. Defining a separating hyperplane as

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{3.1}$$

To maximize the margin, two parallel hyperplanes are defined as

$$\mathbf{w} \cdot \mathbf{x} + b = 1, \qquad \mathbf{w} \cdot \mathbf{x} + b = -1 \tag{3.2}$$

In particular, these hyperplanes must pass through the support vectors without including training samples. Assuming $x^+$ to belong to the positive plane and $x^-$ to be the closest point on the negative plane, then $x^+ - x^-$ is normal to the plane(s) and is equal to $2/\|\mathbf{w}\|$. Thus, in order to maximize the margin and obtain the optimal discrimination of the two

classes, $\|\mathbf{w}\|$ must be minimized without including training samples. The main strength of this classifier is the ability to identify the optimal discriminating surface while better handling the over-fitting issue during the training phase. However, typically, classification problems involve data not linearly separable. Hence, on the majority of real cases the linear separability assumption does not hold.

### 3.1.2 Non-linear support vector machines

The simplest way to handle data non-linearity issue is allowing a softer margin, therefore tolerating some mis-classifications. However, most of the times this solution is not sufficient to obtain a good data partition.

Another way, smarter and working, is to map the non-linearly separable dataset into another parameter space with higher dimension, as shown in figure 3.2. Indeed, it is always possible to map the original input space to some higher dimensional feature space where the training set results to be linearly separable. Once the discriminating hyper-plane has been found, it is sufficient to project it back to the original space to obtain a non-linear decision boundary able to separate the data even at lower dimension level.

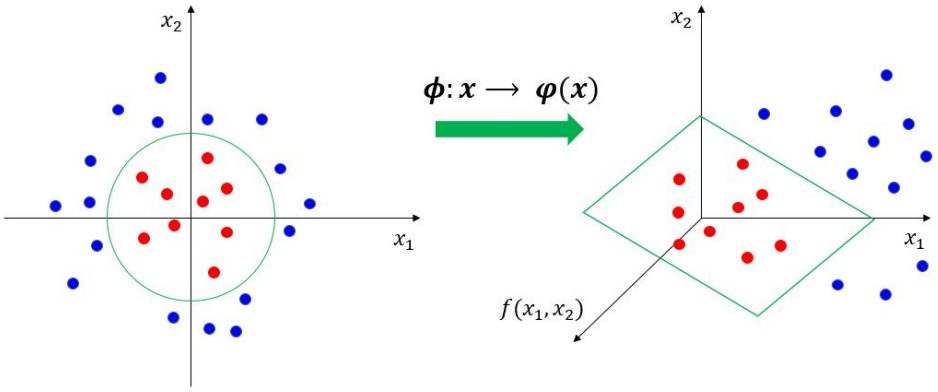



**Figure 3.2:** non-linear SVM hyperplanes with 2 example classes

However, increasing dimensionality adding new features is not priceless: computational and memory consequences of this action may not be feasible. Indeed this procedure means handling a more complex space and an increased number of parameters. Luckily, there is a way, called the **kernel trick**, which allows to obtain the same result without move the

analysis in a higher-dimensional space. Thanks to the introduction of a kernel function, all the dot products between vectors of training examples $\langle \vec{x_i}, \vec{x_j} \rangle$, where $\vec{x_i}, \vec{x_j} \in \mathbb{R}^{\mathbb{N}}$, can be computed in a higher dimensional space $\mathbb{R}^{\mathbb{M}}$ without moving from the original space. The non-linearity is given by the kernel function, hence

$$\forall \vec{x_i}, \vec{x_j} \in \mathbb{R}^{\mathbb{N}}, K(\vec{x_i}, \vec{x_j}) = \langle \Phi(\vec{x_i}), \Phi(\vec{x_j}) \rangle_M \tag{3.3}$$

where $\langle ., . \rangle_M$ is an inner product in $\mathbb{R}^{\mathbb{M}}$ and $\Phi(\vec{x_i})$ is the transformation function mapping $\vec{x_i}$ from $\mathbb{R}^{\mathbb{N}}$ to $\mathbb{R}^{\mathbb{M}}$.

There are several kernel functions that can be chosen, such as Polynomial kernels, Radial basis function (RBF) kernel, sigmoid kernel. However, the choice of the 'correct' kernel is quite tricky and strictly dependent on the specific task. Moreover, each kernel has some parameters to be tuned in order to obtain good performances. Two useful techniques particularly useful for parameter tuning will be discussed in paragraph 3.3.

## 3.2 Deep neural networks

Deep neural networks consist in neural networks composed by more than two hidden layers, whom number defines their depth. Each hidden layer creates a mapping of the data given as inputs to the required outputs. Below it will be presented how a deep neural network works and it will be discussed in detail the main blocks to build a CNN [8], which was used in this thesis and which is particularly meant for images.

### Data and Model

A typical dataset is composed by raw data, such as the input images, and their classes of belongingness, which represent what are called ground truths or annotations. In order to train a neural network, the entire dataset is divided into three parts of different sizes:

- training set: contains most of the entire dataset and it is used for the training process;
- validation set: consists in a small part of the entire dataset and it is used to tune the network parameters during the training phase, until the optimal ones are found;

- test set: it is usually bigger than validation set but smaller than training set. It is used to evaluate the performance of the built model once training phase has ended.

Typically, if the dataset is composed by images, those are normalized before to be passed to the network in such a way that they lie in the range of $[-1, 1]$. Depending on the inputs given for the training phase, the neural network model learns how to treat future unannotated data. However, this learning process also depends on further factors such as the designed architecture, which greatly affects network performances and requires many trials with different configurations.

### Training phase of deep neural networks

The training procedure is meant to find the best parameters able to map the input data into the desired output space. In particular, this process consists in three major steps: a forward pass, a backward pass and a parameter update.

1. Forward pass

   This is the first part of the training phase. All input, hidden and output layers are composed by units called neurons which receive one or more inputs $x_i$ from the respective previous layer. Neurons compute a weighted sum of the inputs using weights $w_{ij}$ and finally add a bias term $b_j$. To obtain the output $y_i$, the final value is usually passed to a non-linear activation function $f$.

   All those computational steps can be summarized through the following operation

   $$y_i = f(\sum_i w_{ij} x_i + b_j) \tag{3.4}$$

   The weights characterizing each layer represent the model parameters, the outputs represent the feature maps that will be the inputs of the following layer.

   Once the outputs from all the hidden layers have been obtained, the output layer is reached. The final output $o$ and its related annotation, called target $t$, are provided to a loss function $l$ to compute the loss of the network. This final computation is meant to quantify how well the network at the current state can map the input to

the given target.

$$loss = l(o, t) \tag{3.5}$$

2. Backward pass

   The backward pass process begins with the result of the network loss. Thus, the gradient of the output $o$, with respect to the input, is computed through the chain rule of differentiation in order to determine the contribution of each parameter to the resulting loss.

   The partial derivatives of the loss with respect to the weights are computed, obtaining each component as

$$\frac{\partial l}{\partial w_{ij}} = \frac{\partial l}{\partial o}\frac{\partial o}{\partial y_j}\frac{\partial y_j}{\partial w_{ij}} \tag{3.6}$$

   where $\frac{\partial l}{\partial o}$ is the partial derivative of the loss with respect to the final output of the network, $\frac{\partial o}{\partial y_j}$ is the partial derivative of the output with respect to the $j^{th}$ neuron, and $\frac{\partial y_j}{\partial w_{ij}}$ is the partial derivative of the $j^{th}$ neuron with respect to the $i^{th}$ weight.

   In case of more hidden layers, to obtain the total partial derivative for the current neuron, this result is multiplied for the partial derivative of the following hidden layer neuron.

3. Parameter update

   Once Backward Pass has ended, the network weights are modified in order to decrease the loss at the next iteration. The parameter values are updated following the opposite direction of the gradient and accordingly to a learning step $\eta$ which defines how big the changes to make must be. In general, it is preferred proceeding with small changes a time. This approach of making changes iteratively depending on the direction of a negative gradient is called **gradient descent**.

   Parameter update is made for all the weights at the same time.

### 3.2.1 Convolutional neural networks

The name of convolutional neural network is due to the presence of one or more convolutional layers which implement the mathematical function of **convolution**. The 2D

convolution of two functions I and K can be expressed as **cross-correlation**, hence we can interpret it as

$$S(i,j) = \sum_m \sum_n I(m,n)K(i+m, j+n) \tag{3.7}$$

where $I$ is the image and $K$ the convolutional kernel of size $m \times n$ translated over the image. Since the kernel size is smaller than the image size, the kernel interacts only with a small region of the entire image identified by its own size $m \times n$.
Furthermore, another meaningful difference from fully connected neural networks is that all nodes of the same layer share kernel parameters and, consequently, the number of parameters is highly reduced. This reduction allows to speed up the CNN training phase and to better prevent overfitting.

### Convolutional layers

The convolution function is specified by the number of inputs, the number of outputs and the width and height of the kernel. It is common in deep neural network to use stacks of 2D kernels. In particular, the number of inputs determines the number of 2D kernels per each stack, while the number of outputs determines the number of stacks or 3D kernels. Also there are some other parameters like step size, to specify the convolution operation step, or zero padding to handle the uncertainty at the borders. Indeed, kernels usually apply the convolution operation to every pixel of the input image but this can be changed modifying the step size, while zero padding is needed to avoid limiting network depth because of the decrease of feature maps size.

### Fully connected layers

In those layers all the input neurons are connected to all the output neurons. Typically, fully connected layers are inserted towards the end of the CNN to perform the classification, while convolutional layers are located at the beginning acting as feature extractors. Before to be passed to a fully connected layer, the 2D feature maps are rearranged in a 1D vector.

The parameters of a fully connected layer are determined by the number of inputs and outputs required. Thus, if the the input vector has size $m$ and $n$ is the number of desired outputs, then a parameter matrix having size $m \times n$ is required.

### Pooling layers

Pooling layers aim to down-sample feature maps to reduce their spatial dimensions. This intention has two main reasons: first, to allow a GPU with limited memory to store deeper network models, second, effectively increase the spatial area on which convolution kernel acts.
The operations applied in pooling layers are usually linear, like averaging, or statistic, such as maximum extraction. In particular, Pooling operations are applied on feature map values belonging to the region which interact with the kernel.

### Activation function layers

After linear layers, such as convolutional and fully connected layers, an activation function layer is added. In this layer, a non-linear function, called activation function, is applied. There are various activation functions that can be used, but it is important to take into account the problem of vanishing gradient. Activation functions such as *sigmoid* and *tanh* bond responses to values close to 1. This results in a very small gradient that becomes extremely small proceeding backward. Thus, in presence of large responses and very deep networks, this small gradient goes below the machine precision after few layers, leading quickly the network to stop learning. To face this problem the Rectified Linear Unit (ReLU) [20] activation function has been introduced. This function passes all values above 0 as they are and sets to 0 all the negative ones. It is then expressed as

$$f(x) = max(0, x) \tag{3.8}$$

The absence of an upper limit of the response of this function helps avoiding vanishing gradient issue.
However there can be the problem of exploding activations in which activation increases

because of several consecutive convolutions. To handle this problem a proper regularization of the layers can be useful. Some regularization techniques will be discussed in the 3.2.3 paragraph.

### 3.2.2 Loss functions

The learning process is highly dependent on the loss function, which aims to quantify the error of the network prediction. The main goal is indeed to minimize the difference between the output provided by the network and its related annotation during the training phase. Some metrics for loss computation are presented below.

#### Cross-entropy loss

The cross-entropy loss function is called softmax loss. It can be expressed as

$$Cross - entropy = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log p_{model}(y_i|x_i;\Theta) \tag{3.9}$$

where $p_{model}$ is the deep neural network model to estimate the $p_{data}$ distribution, $x_i$ is the $i^{th}$ example drown from the $p_{data}$, $\Theta$ is the distribution of network parameters and $y_i$ the label of the $x_i$ example. A particular case of cross-entropy is called binary cross-entropy and it is used in case of two classes classification.

#### Softmax activation

The classifier output is typically a categorical prediction involving $n$ unnormalized outputs $z_i, ..., z_n$. Thus, if a cross-entropy loss function is used, a normalization procedure is needed to obtain probabilities as outputs. This purpose is achieved through softmax activation that, for the $k^{th}$ output, can be expressed as

$$softmax(z_k) = \frac{exp(z_k)}{\sum_{j=1}^{n} exp(z_j)} \tag{3.10}$$

Taking the logarithm of the softmax function, cross-entropy between the model distribu-

tion $p_{model}(y_i|x_i;\Theta)$ and the data distribution $y_i$ is obtained

$$\log softmax(z_k) = z_k - \log \sum_{j=1}^{n} exp(z_j) \tag{3.11}$$

Thus, to write equation 3.9 in a more compact fashion and to represent the $n$ log softmax outputs by $\mathbf{z}$, $\log p_{model}(y_i|x_i;\Theta)$ is replaced, and equation 3.9 becomes

$$Cross - entropy = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \mathbf{z}_i \tag{3.12}$$

Equation 3.11 carries some advantages in the learning phase. Indeed, the first term $z_k$ pushes the activation of the correct output to increase while the second term, $-\log \sum_{j=1}^{n} exp(z_j)$ makes all the outputs to decrease. Thus, if $z_k$ is the largest activation while all other outputs are small, 3.11 can be approximated as

$$\begin{aligned} \log softmax(z_k) &\approx z_k - \max_j(z_j) \\ &\approx z_k - z_k = 0 \end{aligned} \tag{3.13}$$

It is then clear that the majority of the contributions will derive from wrongly classified examples, while the correct ones will influence very little the loss function.
However, the softmax output saturates when the differences among input values are consistent. To handle this instability, the maximum output is removed from all the output making the softmax function numerically stable [21].

### 3.2.3 Regularization

It might happen that an implemented deep network shows high performances during the training phase but notably lower in the test phase. This means the network is not able to generalize and fails in classifying never seen data. To limit this inconvenient and make the network more efficient, a regularization process can be useful. Below are presented

some regularization measures used in this thesis.

### Data augmentation

A deep network model can generalize better if a data augmentation is applied. Data augmentation allows to perform the training procedure using additional data synthesized by applying transformations (i.e., rotations, flips and random translations) to those already available. This makes it possible to vary the entire dataset and to avoid the network to learn certain unwanted variations due to particular cases.

### Dropout layers

Dropout layers [23] are always associated to functional layers, like convolutional or fully connected layers. Each dropout layer carries a dropout probability, specified by the user, that determines the frequency with which neurons of the corresponding functional layer can be turned 'on/off' during a training iteration. For instance, a dropout probability of 0.5 means half of neurons randomly turned off and not contributing either to forward nor to backward pass of the current iteration. However, a neuron turned off at iteration $k$ retains its previous set of incoming and outgoing weights that will be used at the iteration $k+1$ if it will be turned on again.
This mechanism is applied during the training phase to decrease the variance and overfitting in the network, while in the test phase all neurons are turned on and contribute to the classification.
This approach is equivalent to train a different deep network at each iteration and so more classifiers. As shown for ensemble classifiers, the resulting performance is improved respect to any of individual classifier. Thus, the same can holds for deep networks trained introducing dropout layers.

### Batch normalization layer

Batch normalization (BN) [22] layer is usually introduced between a liner layer and an activation layer. If the training is performed with a BN, a mini-batch of the dataset is forward passed through the network. First, the output of each linear layer is normalized by removing the mean value and dividing by the standard deviation of the output over

the mini-batch. This normalization is performed for each feature map separately over the mini-batch. Second, the BN layer re-scales and re-translates the normalized features according to a scale and a shift parameters which are learned during the training. This procedure ensures that only half of the feature responses lies above the activation threshold of the following non-linear activation layer. Thus, half of the feature responses always leads to a gradient update and situations in which all activation units in the activation layer or none of them are activated are avoided.

### 3.2.4 Optimization

To carry out an optimization procedure means to minimize an objective function, which corresponds to the loss function in deep neural networks context. The majority of algorithms used to train a deep neural network reduce the loss at every training iteration by following the negative gradients direction until convergence. Such methods are called gradient descent algorithms.

#### Gradient descent

This is the most used optimization technique for training deep neural networks. It consists on the computation of the objective function gradient followed by the parameters update in its opposite direction, with a step determined by the learning rate $\eta$. This process is known as Gradient descent (GD) and there are three main variants, presented below, which differ for the size of the dataset used in the GD algorithm.

#### Batch gradient descent

Batch gradient descent (BGD) performs the parameter update exploiting the entire training set. Consequently, the BGD output consists in a monotonically decreasing loss function. However, since usually a training set contains millions images, computing the gradient of the objective function using the entire training set for a single parameter update is computationally expensive and memory might not succeed in handle it. Hence, this solution is not really recommended.

### Stochastic gradient descent

Contrary to BGD, Stochastic gradient descent (SGD) performs the parameter update exploiting a single sample of the training set. This approach certainly allows to train huge training set without memory issues, but leads to a very fluctuating output loss.

### Mini-batch gradient descent

Mini-batch gradient descent (MBGD) is a good compromise between BGD and SGD methods. The parameter update is in this case performed using a subset of the whole training set, called mini-batch. This option avoids the issues related to the previous methods, resulting in less fluctuations of the output loss and in no memory problems.

### Adaptive methods

MBGD seems to be a really good solution, but it is not immune to some limitations. The choice of a proper learning rate $\eta$ is problem-dependent, and requires an accurate analysis of the optimization process. In addition, MBGD does not provide an automatic learning rate adjustment during the training process.
Consequently, several adaptive methods have been provided to handle MBGD issues. Some of them are presented below.

- Momentum

  This method [24] adds a fraction of the previous parameter update to the current one. The consequence of the momentum term is that parameters with gradients pointing in the same direction are updated with higher values, while those pointing towards opposite directions are updated with smaller values. This leads to a faster convergence.

- Nesterov accelerated gradient

  Nesterov accelerated gradient (NAG) [25] updates the current parameters based on an approximation of their future position. This approach tries to overcome the issue of skipping local minima as the momentum-based algorithms do.

- Adagrad

  Adaptive gradient (Adagrad) [26] extends the NAG adapting the updates to each

parameter accordingly to their importance. In particular, this method promotes larger updates for less frequently occurring features and smaller updates for more frequent features. Adagrad does not require a manually tuning of the learning rate, which changes for each parameter.

- RMSProp

  It might happen that Adagrad results in very small learning rates making the network to stop learning. RMSProp [27] overcomes this issue using a running average of the squared gradient recursively. In particular, the average squared gradient for a current step is determined by the average at the previous step and the current gradient.

- Adam

  Adaptive momentum estimation (Adam) [28] computes adaptive learning rates for each parameter too. It stores exponentially decaying averages of past squared gradients and past gradients.

## 3.3 Useful techniques for parameter tuning

### 3.3.1 K-fold cross validation

This technique allows to train and validate a classifier using different partitions of the entire dataset. The main goal is to obtain a model able to generalize while classifying accurately also new unannotated data. In k-fold cross validation the original dataset is randomly partitioned into $k$ subsets of equal size. One subset is then used as validation set, while the remaining $k - 1$ subsets will constitute the training set. This partition and training process is repeated $k$ times always changing the subsets used for training and validation. At the end of the entire process, average and standard deviation of the obtained results are computed to produce a global estimation.
The big advantage of k-fold cross validation is that all the dataset samples are used for both training and validation, and each of them contributes only once to the validation set. However, the choice of k requires more trials and is not a trivial task.

### 3.3.2 Grid search

The grid search algorithm can be used to tune the hyperparameters of a chosen classifier. This technique consists in selecting a subset of hyperparameters and train the classifier with all their possible combinations. For certain parameters the user should impose some bounds and a discretization before running the grid search. A performance metric, such as cross-validation, must guide the algorithm. To give a working example, in this thesis the grid search has been used to tune $C$ [1] and $\gamma$ [2] parameters characterizing the RBF kernel used in the SVM classifier. According to this technique, the SVM classifier was trained with each pair of $C, \gamma$ values, giving in output a grid containing the performances achieved for each tested parameter combination.
In this way it was possible to select the couple of parameters allowing to get the best model performances.

[1] this parameter determines the hardness of the margin and so the tolerance allowed for mis-classified samples. the higher it is, the lower is the tolerance. High values of C can lead to high performances during the training phase but also to a high risk of overfitting

[2] this parameter determines how far the influence of a single sample can extend. low gamma values mean 'far', high mean 'close'

# Chapter 4

# Method

In this chapter, will be presented the implemented pipeline to deal with an *in situ* sequencing dataset. The steps composing the pipeline are shown in figure 4.1 and each of them will be discussed in detail in the following paragraphs. Note that, for the final steps, multiple options have been provided. However, at each level of the workflow only one block must be set to get to the end.

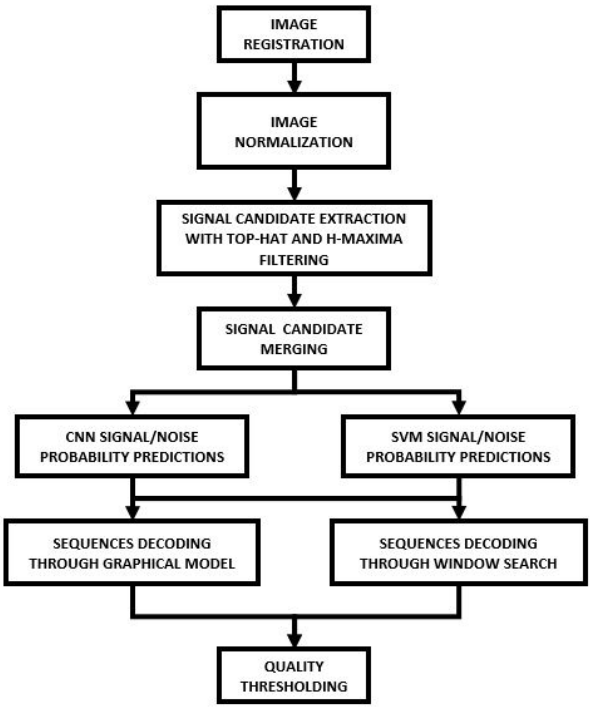



**Figure 4.1:** pipeline

## 4.1 Image registration

The registration step aims to align images to compensate misalignments among successive hybridization cycles. This procedure takes place involving two steps:

1. maximum intensity projection (MIP) is applied to general stain and nuclei channels of all the hybridization cycles to obtain the fixed reference images. Hence, for each hybridization cycle, the four colour channels are then aligned to the relative MIP;

2. the MIP of the first hybridization cycle is taken as fixed reference image and all the MIPs of the following hybridization cycles are aligned to it. From these last registrations, the relative transformation matrix is retrieved and multiplied for each colour channel for each hybridization cycle, providing the final aligned dataset.

If in the dataset no general stain channel information was acquired, then they will be replaced by obtaining a MIP through the combination of the five available channels for each hybridization cycle. Hence, in this particular case, the registration procedure will consist only in the second step. The alignment procedure is the same both in 2D and 3D datasets.

## 4.2 Signal candidate detection

In 3D datasets, it may happen for deeper z levels to be out of focus and so to be noisy and poor or even empty of information. Thus, a limit on z depth can be set to not consider these levels in the next analysis steps.

### 4.2.1 Normalization

In order to be able to compare intensity values among different channels, a normalization step is needed. Thus, images intensity values are scaled between the estimated background value (i.e. mode of the image) and the brightest fluorescent signal values (i.e. $99^{th}$ percentile of intensity values) as following:

$$I\_norm_{ij} = \frac{I_{ij} - mode(I_{ij})}{P99(I_{ij}) - mode(I_{ij})} \tag{4.1}$$

Where $I_{ij}$ is the registered image of the channel $j$ at the hybridization cycle $i$, and $P99(I_{ij})$ is the $99^{th}$ percentile.

### 4.2.2 Signal candidate extraction

A Top-Hat filtering is applied to enhance bright spots and attenuate background contribution. A disk and a ball as structuring element are used for 2D and 3D datasets respectively. Afterwards, an h-maxima transformation [29] is applied to consider as signal candidates only those local maxima whose dynamic is higher than a given $h$ value. This procedure is performed with a 4-pixels connectivity for 2D datasets and a 3D stack of 4-pixels connectivity for 3D datasets to separate more the signals among them. However, some very bright signals may present saturated intensity values and, at this step, this would result in regions of local maxima (i.e. region of several connected pixels). Thus, in these cases medoids coordinates are considered as signal candidate positions.
All the fixed parameters, such as the structuring element radius $r$ and the $h$ value, are user-defined. In general, the radius $r$ is set in such a way that it is equal to the average radius of the signals. The $h$ value, instead, is maintained low to ensure picking up also the weakest signals, even if this means retrieve some noise (i.e. false detections).
At the end of this step, for each considered image, a mask containing white pixels corresponding to the detected candidate positions is obtained.

## 4.3 Signal merging

Because of photobleaching and broad emission spectra, fluorescent signals can appear in more than one channel at the same time. Thus, at each hybridization cycle, candidates located at the same position are associated among them. In this way multiple detections related to the same mRNA molecule are filtered out. This procedure consists in three sequential steps:

1. candidate coordinates are extracted from the general stain mask and taken as reference;

2. for each couple of reference coordinates it is checked, in each colour channel mask and for each sequencing cycle, if there is a candidate whose coordinates are located in a 3x3 px window centred in the reference coordinates;

3. if any candidate is found, its intensity value is retrieved from the respective normalized image. In particular, in case of multiple findings, the retrieved candidate is the one having the lower Euclidean distance from the reference coordinates. On the other hand, if no candidate is found, then the intensity is extracted from the relative normalized image at the reference coordinates.

Once this procedure is over, for each set of associated candidates, is selected as final candidate the one with the maximum fluorescent intensity.

## 4.4 Signal candidate prediction

In this step each candidate is provided with its probability of being signal and noise. For this purpose, two different classifiers have been implemented: a CNN and an SVM.

### 4.4.1 Predictions with a CNN

The choice of a powerful classifier such as a CNN to deal with this task was made because judging if a fluorescent spot is a real signal or noise is often tricky also for a trained eye. Indeed, signals have a blurry appearance without any clear border due to low signal-to-noise ratio caused by the diffraction limit of the microscope. Thus, it is not easy to decide which features are the most important and the most meaningful to achieve a correct classification, and a self-learned approach might be more reliable than another based on handcrafted features.

#### Architecture

The built CNN architecture (Figure 4.2) is composed by a convolutional layer with 200 filters of 5x5 convolutions, a fully connected layer of 128 neurons and a softmax layer with 2 output classes. In addition, 2 dropout layers were placed before and after the fully connected layer to avoid overfitting. The convolutional layer takes as input an array of 5x5

px windows, each one centred on a signal candidate location and containing the intensity values extracted from the relative normalized image. In particular, for 3D cases, the 2D windows are retrieved in the same way but at each z level where the maximum is found. The training procedure was carried out with a k-fold cross-validation with k=3 and data

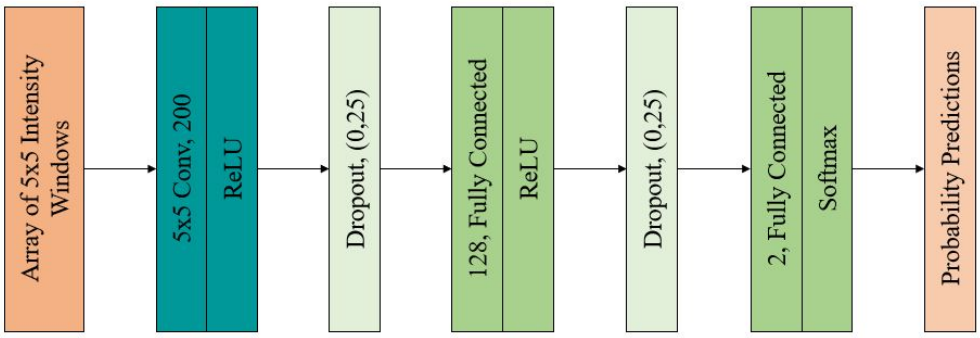



**Figure 4.2:** CNN architecture

augmentation executed by applying to training input signals horizontal and vertical flips. The best model was finally chosen depending on the highest validation accuracy. The details concerning training and test phases will be discussed in chapter 5.

#### Fine tuning

Despite good performances obtained during both training and test phases, it is important to pay attention when using an implemented CNN to classify a different dataset from the one used for the training. Indeed, it may happen that the CNN shows a lower accuracy when tested on another annotated dataset. This leak of accuracy is mainly due to different experimental parameters and biological sources of the samples composing the dataset regarding to the one used to train the CNN. Hence, a fine-tuning procedure is needed to compensate for those differences and being able to use a built CNN.

### 4.4.2 Predictions with an SVM

This classifier was built mainly to assess if a complex classifier, such as the CNN, was really needed or if an easier classifier based on handcrafted features could be sufficient to fulfill the task. To be able to compare the SVM and CNN performances, SVM was

trained with the same annotated dataset used for CNN, data augmentation included. In particular, the feature vectors given as inputs were created simply flattening the 5x5 px windows on 1x25 px vectors. To build the SVM, a RBF kernel was chosen, and its $C$ and $\gamma$ parameters were tuned through multiple grid searches carried out with five values of $C$ and $\gamma$ a time. Finally, as metric a k-fold cross validation with k=5 was used.

## 4.5 Decoding

In this final step the sequences are resolved by combining all signal candidates and probabilities predictions and two ways were exploited to achieve the decoding. The first approach is a more elaborated method based on a probabilistic graphical model. The second approach, instead, is a simpler and faster solution in which each sequence is resolved by searching for candidates within a window of fixed size.
The graphical model method was used as comparison to assess the second decoding approach, which is the only decoding method implemented in this thesis. Hence, the graphical model technique will be briefly presented because not covered by the scope of this thesis. Further details about it can be found in [6].

### 4.5.1 Graphical Model

This decoding method is based on the use of a graphical model that associates signal candidates of different hybridization cycles and fluorescent channels if they have a high probability to belong to the same RNA molecule.
Signal candidates are encoded as boolean random variables and connected through edges if they are located within a distance lower than a given threshold. Each boolean random variable and each edge are characterized by energies, or observations. Energies of boolean random variables, representing a signal candidate, are inversely proportional to the probability of being signal and noise for that specific candidate. Instead, energies of edges are inversely proportional to an affinity function based on the distance and difference between the intensity values of a pair of signal candidates. Solving the graph by minimizing the total energy will select the best set of signal candidates that decode a sequence by drawing the most probable connections within them.

In order to assure output solutions that encode valid representation of real sequences, the graphical model is injected with the following constrains:

- the formation of the paths representing possible decoded sequences must be topologically coherent to the logical structure of the data (i.e. there is distance constraint in paths formation);
- a sequencing result is accepted if its length is equal to the number of hybridization cycles;
- candidates composing a sequence must belong each one to a different hybridization cycle;
- a sequence must be composed all by true signal candidates;
- each candidate can contribute only to one sequence.

The choice of the right paths representing the decoded sequences is highly conditioned by the previously stated candidate predictions.

### 4.5.2 Window search

This method was developed for two main reasons: first, assessing if a complex and computationally demanding approach like the graphical model was really needed, second, if an easier solution could work in easier dataset cases (i.e. with lower signal density and no clustered signals).
Also in this procedure, sequences are resolved depending on signal candidates locations and predictions. The coordinates of all the candidates belonging to the first cycle are taken as reference to drive the window within the search and the decoding is carried out. The window size, which is user-defined, determines the region around the reference candidate in which the candidates of the following cycles must be searched to be part of the current sequence.
The search is carried out by centring the window on a reference candidate at a time. Three different cases might occur and will be handled during the sequence decoding:

1. if for each hybridization cycle only one candidate located within the window is found, then the sequence can be easily built by associating the found candidates among them accordingly to the hybridization cycle they belong to;

2. if there is any hybridization cycle in which no candidates located within the window are found, then the sequence is discarded because incomplete;

3. if there is any hybridization cycle in which more than one candidate is found, candidates' predictions are taken into account to decide which one should contribute to the sequence. The candidate with the highest probability of being signal (i.e. over 50%) is retrieved to be part of the current sequence, while the others are discarded. In this way a sequence with the correct length is obtained, like in the first case.

Each time a sequence having length equal to the number of hybridization cycles is built (cases 1 and 3 presented above), the candidates composing the sequence will not be considered for the following searches. This precaution allows to avoid multiple associations of the same candidates to more than one reference.
Another issue, in case 2, consists in the risk of associating a reference candidate with low probability of being signal to a candidate with a high one, making it unavailable for an eventual better association. Thus, to handle this risk, the reference candidates are ordered according to their probability of being signal before the beginning of the entire procedure.

## 4.6 Quality metric

Once sequences have been decoded, it is important to provide each of them with a measure indicating how much those results can be trusted.
This metric, defined as quality $Q$, is for the graphical model equal to the inverse of the total energy needed to form a sequence. The total energy is low if the sequence is composed by candidates stated as true signals by the classifier, consequently the quality related to the sequence will be high.
However, this same metric could not be used for the window search approach because no energy parameter is computed in this method. Hence, a study of other possible quality

metrics was carried out. The best metric was chosen assessing which one among all the tested options better separated expected sequences(i.e. targeted barcodes) from unexpected sequences (i.e. all non-targeted sequences) distributions. This means maximizing the number of expected sequences when the false positive rate is zero.
The tested quality metrics are all based on the probability predictions (*probabilities*) of the signal candidates composing a sequence provided by the classifier, and are listed below:

1. $Q = mean(probabilities)$
2. $Q = \min(probabilities)$
3. $Q = \frac{\min(probabilities)}{\sum(probabilities)}$
4. $Q = \sum(probabilities)$
5. $Q = n^{\circ}signals \times \sum(probabilities)$

In particular, in (5) $n^{\circ}signals$ is the number of candidates composing a sequence presenting a probability prediction equal or higher than 50%.

# Chapter 5

# Results and Discussion

In this chapter will be presented and discussed the quantitative results obtained evaluating all the combinations of the new proposed pipeline respect to the SoA pipeline. First of all, will be presented the datasets used to train and test the classifiers and the dataset used to evaluate the pipeline. Second, will be given detailed information concerning parameter setup and optimization relatively to training and testing phases of the classifiers and to the tested quality metrics. The final part will be dedicated to present and discuss the results obtained through the pipeline.

## 5.1 Datasets

For classifiers training and testing phases and for the evaluation procedure three different dataset were used respectively:

- the dataset 1 was taken from an *in situ* sequencing experiment of a schizophrenic mouse brain coronal section. This dataset was composed by 30 fluorescent images acquired over 5 sequencing cycles for 6 channels. Image size was $22635 \times 29861$ px (fig 5.1);
- the dataset 2 was taken from an *in situ* sequencing experiment of a healthy mouse brain coronal section. As the dataset 1, this dataset was composed by 30 fluorescent images of size $18976 \times 31677$ px each (fig 5.2);
- the dataset 3 was an already published sequencing dataset of a co-culture of murine

and human cells [5]. In this case, the general stain channel was available only for the first sequencing cycle, hence dataset 3 was composed by 21 fluorescent images acquired for the remaining 5 channels over 4 sequencing cycles. The image size was of $1445 \times 1097$ px. In this 2D dataset, a four-base-long sequence of $\beta$-actin transcript ($ACTB$) mRNA was sequenced. This sequence is identical for human and mouse cells except for a single nucleotide polymorphism (SNP) on the second base, which allows to differentiate the nature of the mRNA. This dataset was acquired in 3D at different focal depths then combined into a single MIP directly by the proprietary software of the microscope (fig 5.3).

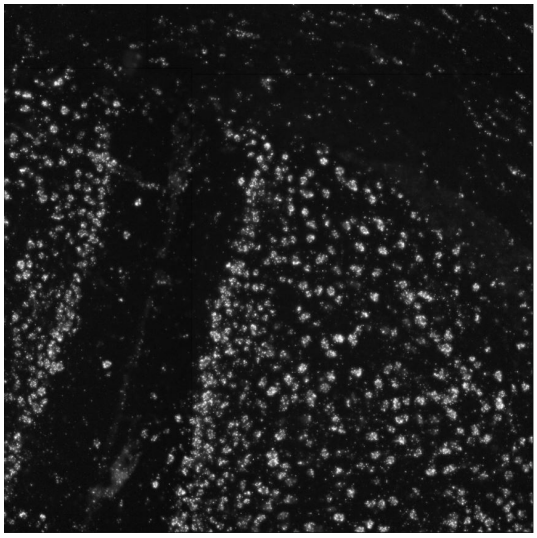


**Figure 5.1:** General stain tile of schizophrenic mouse dataset of the first hybridization cycle

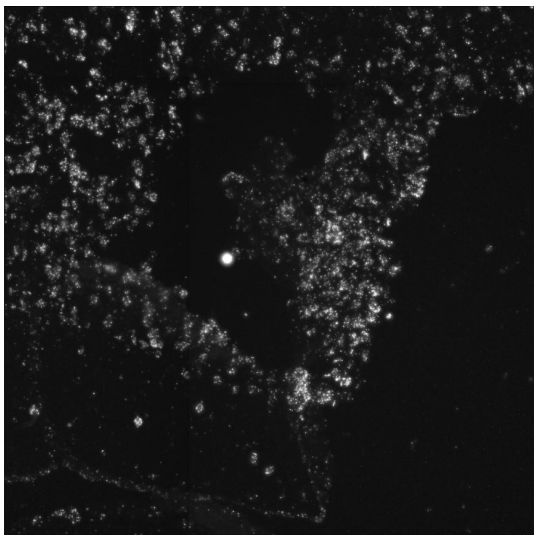


**Figure 5.2:** General stain tile of healthy mouse dataset of the first hybridization cycle

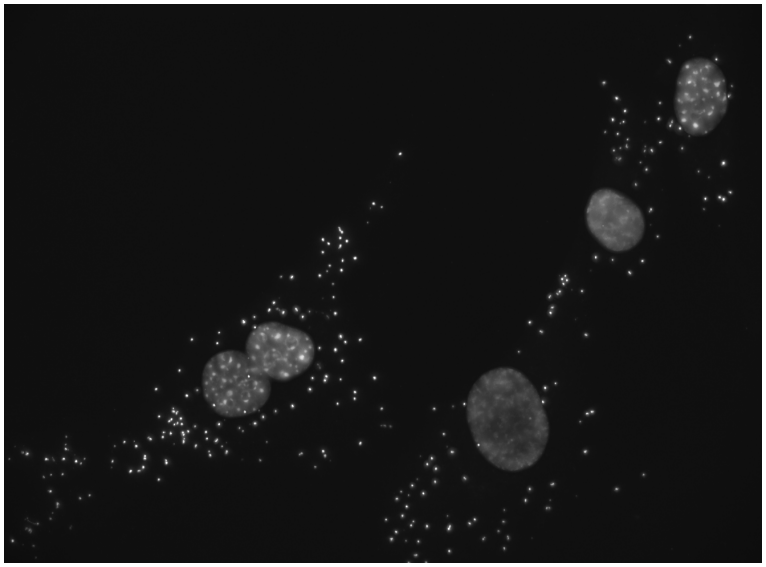


**Figure 5.3:** Co-colture of human and murine cells MIP of the first hybridization cycle general stain and nuclei channels

## 5.2 Parameter setup and optimization

### 5.2.1 Training and testing

To train the CNN, 13878 signals were manually annotated on dataset 1. To test it, other 1037 signals were manually annotated on dataset 2. The dataset prepared for the training was divided into training set and validation set through a stratified k-fold cross-validation with k=3 and then augmented by applying to its signals horizontal and vertical flips. Once ended the training procedure, among the three models obtained, the one with the highest validation accuracy was chosen. The best model accuracy was 83% in the training phase and 72.42% in the test phase. For the evaluation phase, a fine-tuning procedure

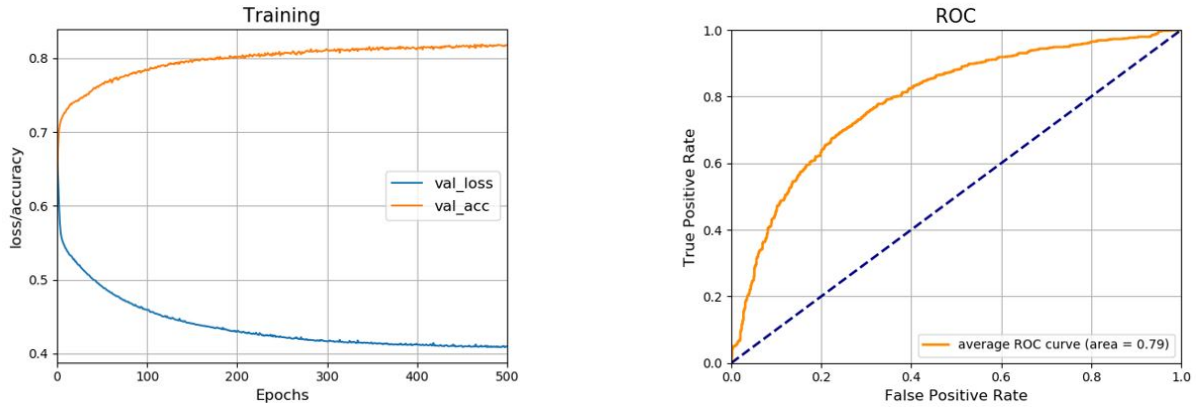



**Figure 5.4:** CNN training and test performances

was performed on the CNN with 200 manually annotated signals on dataset 3. The SVM was trained on the annotated dataset 1 and dataset 3 and tested on the annotated dataset 2. After a grid search parameter optimization, the best performances were obtained by setting $C$ at $10^6$ and $\gamma$ at 0.01, with an accuracy of 79.59% during the training and a 74,25% during the test.

### 5.2.2 Quality metric

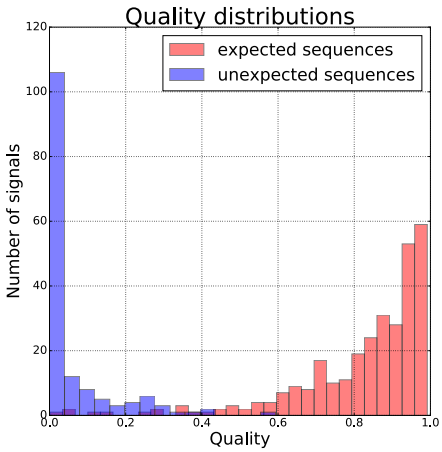
All the proposed quality metrics were tested on dataset 3. The dataset was processed through the implemented pipeline, choosing the fine-tuned CNN as classifier and the window search as decoding approach. The true positive and false positive distributions obtained testing the proposed quality metrics $Q$ are shown in fig 5.5.
It can be noticed that all the tested quality metrics are based on the probability predictions of the candidates composing a sequence. The best identified quality metric is represented in figure 5.5e.
Indeed, in this quality metric, also the number of signals in a sequence (i.e. number of candidates with prediction equal to/higher than 50%) is taken into account and used as an amplification term. This term allows to promote sequences with high content of signal candidates respect to those mainly or totally composed by noise candidates.

## 5.3 Evaluation

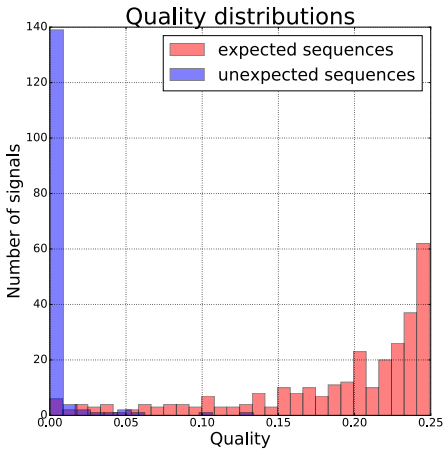
To validate the method, the pipeline was tested on the *in situ* sequencing dataset 3. The results were quantitatively assessed in terms of recall respect to the SoA image analysis pipeline at equal FP rate as

$$\frac{TP_{new} - TP_{SoA}}{TP_{SoA}} \times 100 \tag{5.1}$$

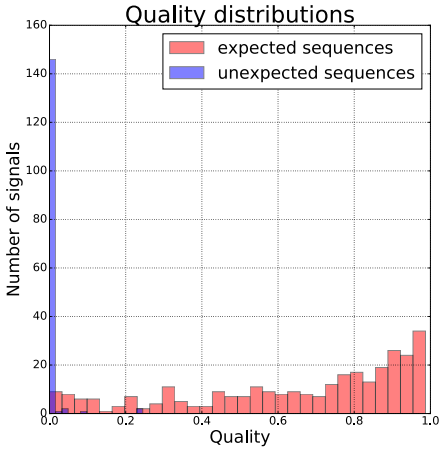
where $TP_{new}$ are the true detections of the new pipeline and $TP_{SoA}$ the true detections of the SoA pipeline, and in terms of precision by qualitatively evaluating the spatial coherence of the detected sequences with the biological information. The analysis was performed by setting the $h$ value at 0.1, the structuring element radius $r$ at 5 for the Top-Hat and the window size for window search sequencing at 3x3 px. Resolved sequences of



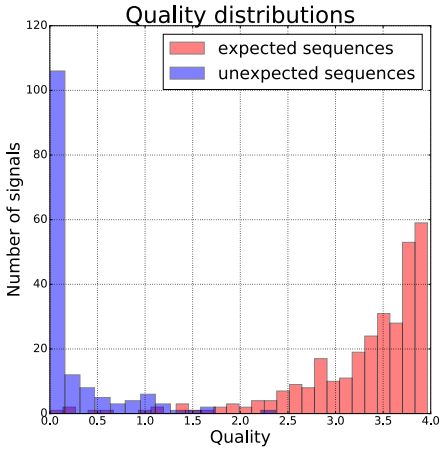

**(a)** $\texttt{Q} = \texttt{mean(probabilities)}$



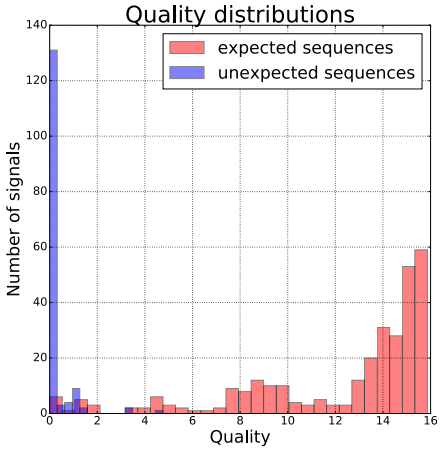

**(b)** $\texttt{Q} = \texttt{min(probabilities)}$




**(c)** $\texttt{Q} = \frac{\texttt{min(probabilities)}}{\sum\texttt{(probabilities)}}$




**(d)** $\texttt{Q} = \sum\texttt{(probabilities)}$




**(e)** $\texttt{Q} = \texttt{n°signals} \times \sum\texttt{(probabilities)}$

**Figure 5.5:** tested quality metric distributions

the targeted gene (sequences $AGGC$ and $AAGC$) were considered as *true positive* (TP) decoded sequences, while all the others were considered as *false positive*(FP) results. *Homopolymers* sequences (sequences composed by a single letter such as $TTTT$, $GGGG$, $CCCC$, $AAAA$) were excluded from FP count, being a particular kind of false detections arising from biological structures auto-fluorescence and thus never used in designed *in situ* sequencing experiments. In table 5.1 are presented the results obtained through the SoA pipeline and through all the different combinations of the final steps provided for the new proposed pipeline.

| Classifier | | Sequencing method | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| CNN | SVM | Graphical Model | Window Search | TP new pipeline no $Q_{th}$ | FP new pipeline no $Q_{th}$ | TP new pipeline $Q_{th}$ | TP SoA pipeline no $Q_{th}$ | TP SoA pipeline $Q_{th}$ | Recall increase/ decrease |
| X | | X | | 304 | 122 | 270 | 235 | 213 | **+27%** |
| X | | | X | 304 | 150 | 279 | 235 | 213 | **+31%** |
| | X | X | | 303 | 125 | 263 | 235 | 213 | **+24%** |
| | X | | X | 304 | 150 | 170 | 235 | 213 | **-20%** |

**Table 5.1:** Comparison between SoA pipeline and each combination of the new pipeline

It can be immediately noticed that, thanks to the preprocessing part of the new pipeline, both graphical model and window search approaches manage to decode, previous $Q_{th}$ selection, 304 sequences while the SoA pipeline 235. In figure 5.6 is represented the comparison between the qualitative evaluation of the new approach using the graphical model with the CNN combination and the SoA pipeline. It is evident that clustered signals, so more dense regions, are resolved with higher accuracy by the new approach than the SoA method. This happens because the SoA pipeline carried out a watershed undersegmentation in the effort to maintain high sensitivity. Thus, clustered signals in dense regions were merged together into a single object. In the new approach, instead, signals were generously included, even if this meant picking up some noise too. Indeed, noise issue is handled in the later stages of the pipeline by the the signal classifier.

However, to trust the decoded results it is important to filter out sequences with a low-quality measure. Those sequences should mainly consist in the unexpected ones, that are originated from noise or false detections. Hence, a quality threshold $Q_{th}$ was set for each tested combination of the new approach and for the SoA approach. Columns of *'TP new pipeline no $Q_{th}$'*, *'FP new pipeline no $Q_{th}$'* and *'TP SoA pipeline no $Q_{th}$'* indicate

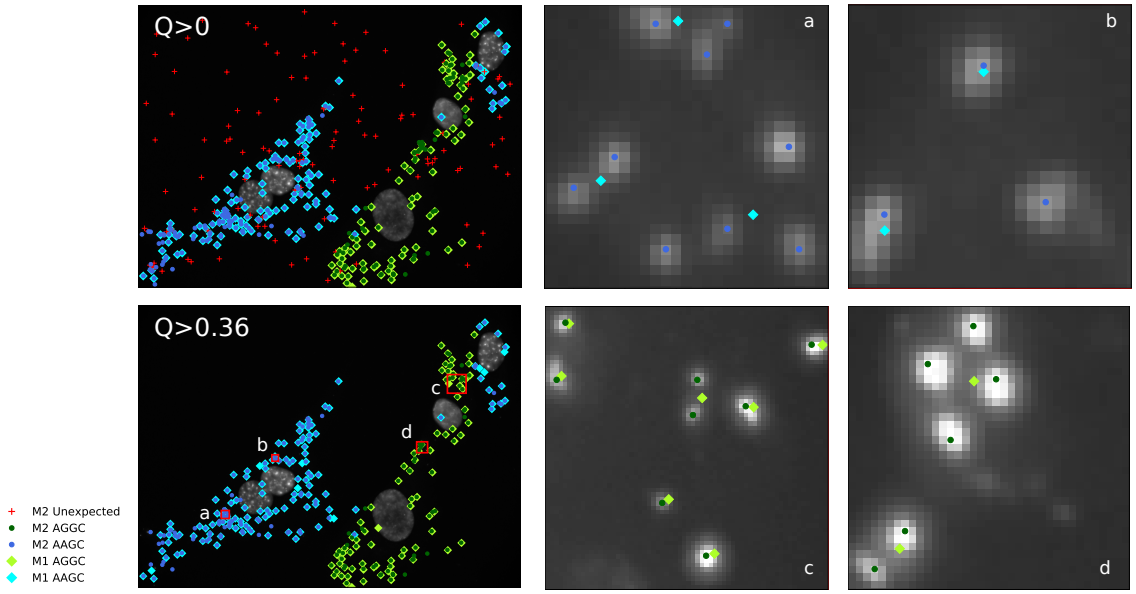
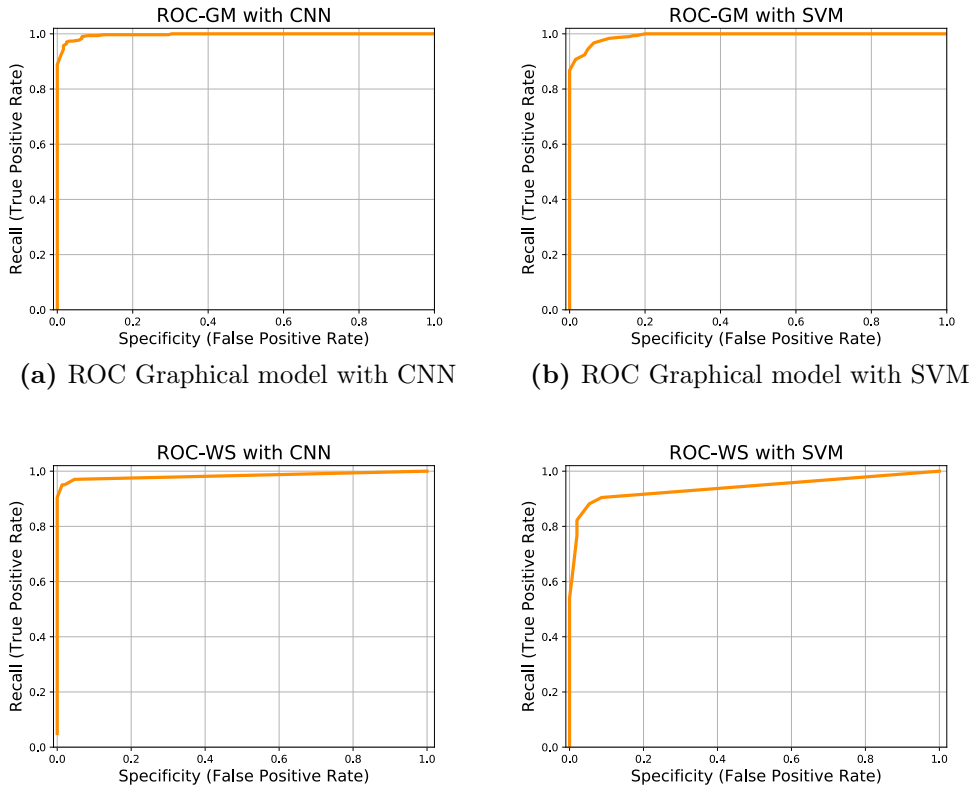



**Figure 5.6:** Markers overlap on the general stain image of the two targeted barcodes (*AGGC*, *AAGC*) and unexpected barcodes, decoded by the SoA pipeline *M1* [2] and the proposed method with the graphical model and the CNN combination *M2*. The top-left image shows sequences decoded by *M2* without any quality filtering. The bottom-left image shows barcodes with a quality metric higher than a given threshold whose value has been set to filter out unexpected sequences. The right side of the figure shows four different regions at different resolution levels where overlapping markers of decoded barcode sequences with high quality are shown in detail. *M2* often resolve signals that were merged by *M1*.

expected and unexpected sequences before setting $Q_{th}$ (Note that for the SoA pipeline no FP was detected, this was still a consequence of the watershed undersegmentation). Moreover, by setting each $Q_{th}$ high enough to exclude all FPs, it was also possible to compare the various combinations of the new pipeline among them. Those $Q_{th}$ were fixed looking at the relative receiver operating curve (ROC) (Figure 5.7).

After quality thresholding, it can be immediately noticed that using the CNN instead of the SVM as classifier allows to retain more *TP*. Even if both classifiers were trained, tested and fine-tuned with exactly the same datasets, there is a huge difference between their performances. Applying the window search in combination with the SVM classifier means loosing 61% of the sequences decoded with the CNN. For the graphical model the difference between the usage of SVM or CNN as classifier is less dramatic but still evident (3%).

This proves that the performances obtained for each classifier during the test and training phase cannot be blindly trusted. Indeed, despite the leak in test accuracy of the CNN respect to the SVM (2%), the SVM performs considerably worse in predicting candidates probabilities, consequently impacting on the final accuracy. Moreover, it is evident that




**(a)** ROC Graphical model with CNN

**(b)** ROC Graphical model with SVM

**(c)** ROC Window search with CNN

**(d)** ROC Window search with SVM

**Figure 5.7:** ROC of the different combinations of the new proposed approach used to set the relatives $Q_{th}$ and discard unexpected sequences

the task of classifying a signal as true signal or noise is not so easy as it seems. The nature of the signals, as already discussed in 4.4.1, makes a classifier based on handcrafted features less reliable than one based on self-learned features. However, this huge difference in performances can also be due to an inappropriate choice of the feature set for the SVM and to the small number of annotated signals on dataset 3 with respect to the number of annotated signals on dataset 1 used for the training. Indeed, maybe a deeper research on the features to be chosen would lead to a different conclusion, as well as a properly balanced training set in term of number of signals with different characteristics. However, it is clear that identifying a meaningful set of more suitable features requires lot of trials and maybe a feature selection procedure too. The choice of the right classifier is crucial because its probability predictions highly condition the outcomes of both the decoding procedures and of the quality metric values, and consequently the fulfillment of the final goal of the whole pipeline. Thus, from the evaluated results, using a CNN instead of an

SVM as classifier seems to be the best choice in term of final accuracy.

According to the outcomes, the combination of the window search with the CNN leads to the highest increase in term of recall respect to the SoA pipeline. However, as mentioned above, the graphical model has a lower variability in term of performances than the window search if the CNN or the SVM is used as classifier. This means that the graphical model approach is more independent then the window search from the classifier probability predictions.

However, this huge difference is not only due to the classifier and to the decoding approach chosen but also to the selected quality metric. Indeed, the quality metric applied for the window search approach is strictly dependent on classifier probability predictions, since it gives a higher quality value to those sequences whose candidates have been recognized as true signals. This implies the amplification of the classifier decisions, either they are wrong or correct. Thus, if the classifier is reliable then more decoded sequences will be over the $Q_{th}$ and expected and unexpected sequences distributions will be more separated. Vice versa, if the classifier is not reliable, then expected and unexpected sequences distributions will be less separated and lot of true detections will be under the $Q_{th}$.

Finally, some observations about the two decoding approaches should be done. The window search is less flexible and reliable than the graphical model, because the decisions to be made in different sequencing cases are fixed and predefined. For the graphical model, instead, there are no preset choices to be made accordingly to the case, but only few rules indicating to the model how to decide. Given those issues, it is evident that the window search approach cannot be recommended to handle complex decoding cases (i.e. lot of densely clustered signals) nor if the classifier does not perform well. However, with the window search there is a big gain concerning the computational time, because no model is built and the search can end much more quickly than the graphical model which, on the contrary, is computationally highly demanding trying to solve an NP-hard problem for each cluster of signals.

In conclusion, all the tested options of the new approach lead to decode more sequences previous quality thresholding than the SoA pipeline. The issues related to the SoA pipeline are greatly solved, because signal candidates are generously included in the first processing steps while the decision of which ones contribute to expected sequences is delayed to the

end. Furthermore, local misalignments of the candidates along the sequencing cycles are handled both in signal merging and in the final base-calling, and all the quality metrics proposed for the new approach depend on probability predictions based not only on intensity values but also on additional features. Recall is greatly improved at maintained sensitivity and this gain in precision is really meaningful for all the following spatial statistics of tissue heterogeneity analyzed with in situ sequencing approach.

However, among all the proposed options the best combination identified is the graphical model jointly used with the CNN. Indeed, the evaluated results show that this approach can be applied with less variability and more reliability to any dataset.

# Chapter 6

# Conclusions and Future Works

In order to overcome the limitation of the SoA pipeline for *in situ* sequencing gene expression decoding, a new image analysis pipeline has been developed in this thesis. The implemented pipeline generously includes signal candidates in the first processing steps, hence delaying the choice of which fluorescent spots contribute to the researched sequences. Indeed, after a registration step to align channels and cycles among them, signal candidates are extracted through a top-hat and a h-maxima transformation, filtering out those candidates having a local contrast lower than a given $h$ value. Therefore, at each sequencing cycle, candidates are related among them depending on their positions and, for each position, the candidate having the highest intensity value is retrieved. Once the final signal candidates have been collected, they are given as inputs to a classifier which provides for each of them a probability prediction describing how similar they are, compared to a true signal judged by visual inspection. In particular, two classifiers have been tested to fulfill this task: a convolutional neural network and a support vector machine. The sequence decoding procedure has finally been carried out through two different approaches called graphical model [6] and window search respectively. Both sequence decoding approaches resolve sequences by combining intensity, probability predictions and spatial distance. The first approach is a more elaborated and flexible method based on a probabilistic graphical model and was not implemented in this thesis. The second approach is a simpler and faster solution in which each sequence is resolved by searching for candidates within a window of fixed size. A study on possible quality metrics has been carried out to identify which one allowed to obtain the highest signal recall. All the

possible combination of the implemented pipeline have been assessed and compared to the SoA pipeline in term of signal recall and spatial coherence. Results highlighted all the combinations of the new proposed pipeline manage to decode 304 sequences, instead of 235 of the SoA pipeline, previous quality thresholding. With quality thresholding, the window-based search shows an opposite behaviour if the SVM or the CNN is used as classifier, with a consistent decrease in signal recall of 20% in the first case and an improvement of 31% in the second case. On the other hand, the graphical model sequence decoding method achieves improvements in signal recall with both classifiers (23% with SVM and 27% with CNN). According to the results, it can be deduced that a method involving a classifier based on self-learned features is more reliable than one based on handcrafted features to fulfill the intended task. About the sequence decoding approach, the highest improvement in signal recall respect to the SoA pipeline is achieved with the window search. However, it is worth to notice that the graphical model approach presents a lower variability in recall if the SVM or CNN classifier is used. This means that this method and its related quality metric are less dependent on the classifier signal probability predictions, hence they are more flexible than the window-based search combined with the quality metric chosen among the tested ones. In conclusion, the best combination identified among the four proposed ones is therefore the graphical model with the CNN. All the issues related to the SoA pipeline are greatly solved with the proposed pipeline, and recall has been highly improved at maintained sensitivity and spatial coherence. However further improvements of the implemented pipeline are still possible in several steps. First of all, an automatic parameter setting should be provided for the $h$ value used to carry out the h-maxima transformation accordingly to the SNR characterizing each fluorescent channel. This shrewdness is useful because the number of maxima detected can vary considerably depending from the amount of background noise present in each image. Image normalization can also be improved with an automatic parameter setting. In the implemented normalization, signal intensity values are represented through the $99^{th}$ percentile, while the mode should represent the background component below the signals. However, sometimes only a small portion of the image is occupied by signals, hence the remaining empty zones will only be filled by the background component. Therefore, in this case, the background component would represent most of the regions without information, moving

the intensity values representing signals above the $99^{th}$ percentile of intensities. Since, the percentile value depends on signals amount and intensity values,it is really variable from one channel to another. Consequently the components of the normalization step should be made more independent from the context of application to avoid obtaining intensity values not accurately scaled and too different results when merging the processed tiles. Lastly, a further study on SVM classifier can be carried out. Results highlighted the overwhelming performances of the CNN respect the SVM, but some margins of improvement are still possible for this classifier. Choosing more proper features and training the SVM taking care of the right balance in the amount of the data with different characteristics, might indeed change the drawn conclusions about the reliability in using this classifier to fulfill the intended task.

# Bibliography


[1] Ke, R., Mignardi, M., Hauling, T. and Nilsson, M., 2016. *Fourth Generation of Next-Generation Sequencing Technologies: Promise and Consequences*. Human mutation, 37(12), pp.1363-1367.

[2] Pacureanu, A., Ke, R., Mignardi, M., Nilsson, M. and Wählby, C., 2014, April. *Image based in situ sequencing for RNA analysis in tissue*. In Biomedical Imaging (ISBI), 2014 IEEE 11th International Symposium on (pp. 286-289). IEEE.

[3] Levsky, J.M. and Singer, R.H., 2003. *Gene expression and the myth of the average cell.* Trends in cell biology, 13(1), pp.4-6.

[4] Strell, C., Hilscher, M.M., Laxman, N., Svedlund, J., Wu, C., Yokota, C. and Nilsson, M., 2018. *Placing RNA in context and space–methods for spatially resolved transcriptomics*. The FEBS journal.

[5] Ke, R., Mignardi, M., Pacureanu, A., Svedlund, J., Botling, J., Wählby, C. and Nilsson, M., 2013. *In situ sequencing for RNA analysis in preserved tissue and cells.* Nature methods, 10(9), p.857.

[6] Partel, G., Milli, G. and Wählby, C., 2018. *Improving Recall of In Situ Sequencing by Self-Learned Features and a Graphical Model.* arXiv preprint arXiv:1802.08894.

[7] Murphy, D.B., 2002. *Fundamentals of light microscopy and electronic imaging.* John Wiley & Sons.

[8] Sadanandan, S.K., 2017. *Deep Neural Networks and Image Analysis for Quantitative Microscopy* (Doctoral dissertation, Acta Universitatis Upsaliensis).

[9] Ishaq, O., 2016. *Image Analysis and Deep Learning for Applications in Microscopy* (Doctoral dissertation, Acta Universitatis Upsaliensis).

[10] Lambert, T.J. and Waters, J.C., 2014. *Assessing camera performance for quantitative microscopy.* In Methods in cell biology (Vol. 123, pp. 35-53). Academic Press.

[11] Jonkman, J., Brown, C.M. and Cole, R.W., 2014. *Quantitative confocal microscopy: beyond a pretty picture.* In Methods in cell biology (Vol. 123, pp. 113-134). Academic Press.

[12] Vonesch, C., Aguet, F., Vonesch, J.L. and Unser, M., 2006. *The colored revolution of bioimaging.* IEEE signal processing magazine, 23(3), pp.20-31.

[13] Femino, A.M., Fogarty, K., Lifshitz, L.M., Carrington, W. and Singer, R.H., 2003. [13] *Visualization of single molecules of mRNAin Situ.* In Methods in enzymology (Vol. 361, pp. 245-304). Academic Press.

[14] Levsky, J.M., Shenoy, S.M., Pezo, R.C. and Singer, R.H., 2002. *Single-cell gene expression profiling.* Science, 297(5582), pp.836-840.

[15] Lubeck, E., Coskun, A.F., Zhiyentayev, T., Ahmad, M. and Cai, L., 2014. *Single-cell in situ RNA profiling by sequential hybridization.* Nature methods, 11(4), p.360.

[16] Shah, S., Lubeck, E., Zhou, W. and Cai, L., 2016. *In situ transcription profiling of single cells reveals spatial organization of cells in the mouse hippocampus.* Neuron, 92(2), pp.342-357.

[17] Chen, K.H., Boettiger, A.N., Moffitt, J.R., Wang, S. and Zhuang, X., 2015. *Spatially resolved, highly multiplexed RNA profiling in single cells.* Science, 348(6233), p.aaa6090.

[18] Lee, J.H., Daugharthy, E.R., Scheiman, J., Kalhor, R., Yang, J.L., Ferrante, T.C., Terry, R., Jeanty, S.S., Li, C., Amamoto, R. and Peters, D.T., 2014. *Highly multiplexed subcellular RNA sequencing in situ.* Science, 343(6177), pp.1360-1363.

[19] Sonka, M., Hlavac, V. and Boyle, R., 2014. *Image processing, analysis, and machine vision.* Cengage Learning.

[20] Nair, V. and Hinton, G.E., 2010. *Rectified linear units improve restricted boltzmann machines*. In Proceedings of the 27th international conference on machine learning (ICML-10) (pp. 807-814).

[21] Goodfellow, I., Bengio, Y., Courville, A. and Bengio, Y., 2016. *Deep learning* (Vol. 1). Cambridge: MIT press.

[22] Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

[23] Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R.R., 2012. *Improving neural networks by preventing co-adaptation of feature detectors*. arXiv preprint arXiv:1207.0580.

[24] Qian, N., 1999. *On the momentum term in gradient descent learning algorithms*. Neural networks, 12(1), pp.145-151.

[25] Nesterov, Y.E. (1983). *A Method for Solving the Convex Programming Problem with Convergence Rate O(1/k2)*. Soviet Mathematics Doklady, 27, 372-376.

[26] Bengio, Y., Boulanger-Lewandowski, N. and Pascanu, R., 2012. *Advances in optimizing recurrent networks*. arXiv preprint arXiv:1212.0901.

[27] Tieleman, T. and Hinton, G., 2012. *Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural networks for machine learning, 4(2), pp.26-31.

[28] Kingma, D.P. and Ba, J., 2014. *Adam: A method for stochastic optimization*. arXiv preprint arXiv:1412.6980.

[29] Soille, P., 2013. *Morphological image analysis: principles and applications*. Springer Science & Business Media.