

Classification of human epithelial cell's staining patterns

Sebastijan Dumancic

Thesis submitted for the degree of
Master of Science in Artificial
Intelligence

Thesis supervisor:
Prof. dr. Hendrik Blockeel

Mentor:
Antoine Adam

© Copyright KU Leuven

Without written permission of the thesis supervisor and the author it is forbidden to reproduce or adapt in any form or by any means any part of this publication. Requests for obtaining the right to reproduce or utilize parts of this publication should be addressed to the Departement Computerwetenschappen, Celestijnenlaan 200A bus 2402, B-3001 Heverlee, +32-16-327700 or by email info@cs.kuleuven.be.

A written permission of the thesis supervisor is also required to use the methods, products, schematics and programs described in this work for industrial or commercial use, and for submitting this publication in scientific contests.

Preface

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Sebastijan Dumancic

Contents

Preface	i
Abstract	iv
List of Figures and Tables	v
List of Abbreviations and Symbols	vii
1 Introduction	1
1.1 Motivational Scenario	2
1.2 Detecting the autoimmune diseases	2
1.3 Problem statement	3
2 Background	5
2.1 Understanding the domain	5
2.2 Machine learning	8
3 Related work	11
3.1 Cell segmentation	11
3.2 Intesity level classification	12
3.3 Staining pattern classification	12
4 Cell segmentation	15
4.1 Background segmentation	15
4.2 Positioning a prior	23
4.3 Precise contours with Morphological snakes	25
5 Fluorescence intensity classification	31
5.1 Classifying intensity	31
6 Describing cells	35
6.1 Interesting features	35
6.2 Deep learning	36
6.3 Deep belief networks	39
6.4 Evaluation	42
7 Rule mining	45
7.1 Decision Trees	45
7.2 Inductive logic programming	46
7.3 Results and induced rules	49
8 Conclusion	53

A Induces rules	57
Bibliography	61

Abstract

The abstract environment contains a more extensive overview of the work. But it should be limited to one page.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

List of Figures and Tables

List of Figures

1.1	Cell examples	3
2.1	Illustration of antibody structure	6
3.1	Bad segmentation example	11
4.1	Example of an image histogram	17
4.2	Approximation of an image histogram	22
4.3	A calculation of a threshold	22
4.4	Hough transformation with all detected circles (a) and after removing background circles (b)	25
4.5	Oversegmented results	30
5.1	Potential image that could obtain bad estimation	33
5.2	Results of intensity classification	34
6.1	An illustration of the Restricted Boltzmann machines	40
6.2	Symbolic feature mapping	43
A.1	Rules induced by RIPPER	57
A.2	Rules induced by Decision tree	58
A.3	Rules induced by FOIL	59

List of Tables

4.1	Number of detected and overlapping cells	24
4.2	Number of overlapping cells after snakes	30
6.1	Description of cell types	37
6.2	Identified visual patterns - positive intensity	38
6.3	Identified visual patterns - intermediate intensity	38
7.1	Precision of pattern classification	52

7.2 Recall of pattern classification	52
--	----

List of Abbreviations and Symbols

Abbreviations

LoG	Laplacian-of-Gaussian
MSE	Mean Square error
PSNR	Peak Signal-to-Noise ratio

Symbols

42	“The Answer to the Ultimate Question of Life, the Universe, and Everything” according to [1]
c	Speed of light
E	Energy
m	Mass
π	The number pi



Introduction

In the last couple of decades, computers have found a number of applications in biology and medicine. We may even say they have become an essential tool in revealing the questions of life. A significant role in those problems was played by machine learning, a branch of artificial intelligence concerned with the construction of models capable of learning from data . Probably the most inspiring example comes from the field of bioinformatics where scientists have used the methods from statistics and artificial intelligence to sequence the human genome for the first time. If that was one of the first results, then, what can we expect from the future?

Besides the genomic data which is represented as a sequence of nucleotides, a great amount of biological data can be acquired by different imaging methods such as microscopy, PET or CT imaging and others. That is where the medical imaging and bioimage analysis fields come from. The field of bioimage analysis studies the biological problems by examining an image, or image sequence of a process of interest, while the medical image analysis is concerned with developing of methods that will help in a medical diagnosis process based on imaging.

The above mentioned field of medical image analysis overlaps with a field of computer aided diagnosis (CAD) which collects a broader range of methods used as an assistance to the doctors. An application area of this Thesis would fit the best in that field. This thesis will take a direction in a specific task of assistance to an autoimmune disease diagnosis, with a special emphasis to human interpretable models. During the recent few years, this specific problem has been solved very efficiently. If the problem has been solved, why is this thesis taking another look at it?

The goal of the Thesis is to provide a CAD method capable of making a decision based on a microscopic image of HEp-2 cell in a human interpretable way. To demonstrate the importance of interpretable model in CAD systems, I consider the following scenario.

1.1 Motivational Scenario

Imagine a following scenario. *Peter* is a doctor, an immunologist. Being an immunologist, his job is to find out which autoimmune disease a patient has. As that is an extremely hard task, Jules uses computer assistance - an artificial intelligence program named *Hal*. *Hal*'s role is to confirm *Peter*'s diagnosis, or to provide an additional insight if the decision is hard to make.

We are interested in the situation where *Peter* and *Hal* have made a conflicted decision. There are two cases representing the insight *Hal* can provide. Each case reflects *Hal*'s interior structure : a **black-box** model¹ or an **interpretable** one. If *Hal* is the black-box model, it is not much of an assistance, as *Hal* can't elaborate it's decision. We can only agree we disagree. In this case, *Hal* is not much of a help.

On the other hand, if *Hal* is an interpretable model, it can elaborate it's decision and help *Peter*. If a wrong decision was made by *Hal*, *Peter* can observe it's model parameters, correct the wrong one(s), query again and see if new result supports it's decision. If a wrong decision was made by *Peter*, *Peter* can observe *Hal*'s model, find the parameters he might have overseen and correct his diagnosis.

The example clearly demonstrates one thing – the importance of interpretable models for CAD systems. Computers have proven their ability of inferring from a large amount of data, usually outperforming humans, but in specific situations, a good and accurate model is not enough. We also need to interpret results if we are going to use them.

1.2 Detecting the autoimmune diseases

As mentioned already, this Thesis will address the specific topic of autoimmune diseases classification. The human immune system creates antibodies to fight against infections whereas antinuclear antibodies affect healthy tissues. The Antinuclear Autoantibodies test (ANA) is widely used to determine whether the immune system is developing antibodies or not. Indirect immunofluorescence (IIF) with HEp-2 cells is the recommended method to diagnose the presence of antinuclear autoantibodies in patient serum.

IIF method consists of four consecutive steps:

- cell segmentation
- intensity level classification
- mitosis detection

¹By black-box model we assume a model for which we can only observe it's output, not a decision process

- staining pattern classification

The final step usually classifies cell images into one of following patterns: homogeneous, speckled, nucleolar, cytoplasmic, centromere (see figure 1.1). Some variations may be introduced in different datasets due to large number of possible patterns. Those staining patterns further have a clinical associations with a specific autoimmune disease such as Scleroderma, malignant tumor, Lupus nephritis and others.

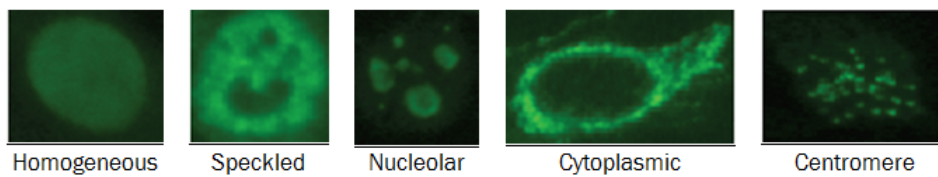


FIGURE 1.1: HEp-2 cell patterns

Although IIF possesses qualities such as high sensitivity and a large number of antigens that can be detected, it suffers from numerous shortcomings. The most important ones are liability to subjectivity and time and labour consuming.

In order to avoid any kind of subjectivity, there is a great need for standardization and formalization of the mentioned procedure. Addressing this problem calls for CAD techniques which combines methods from machine learning and image analysis.

1.3 Problem statement

The content of the thesis proposes solutions for three major steps in the IFF process.

First of all, in order to find out a pattern, cells should be isolated from an image acquired by IFF. As it is going to be explained (see chapter 3), although the problem of cell segmentation has been studied for more than 50 years, segmenting HEp-2 cells still suffers from certain problems, mostly due to different green fluorescent protein absorption across the cells. This thesis proposes a method based on a region growing algorithm that demonstrates an encouraging result for overcoming mentioned problems.

Once we have isolated the cells, the next step is to determine the fluorescent intensity of an image. The fluorescent intensity level describes a subjective measure of *clarity* of an image. This parameter is treated as a feature later in the process, but due to a sake of procedure, this step is developed separately.

1. INTRODUCTION

The final step presents a novel approach to the staining pattern classification. The current state of the art solution solves the problem very successfully, but acts like black-box solutions not providing any explanation for the decision. This thesis tends to develop a method based on the human interpretable representation and reasoning. As IF-THEN rules are the most natural way of representing human knowledge, the thesis will follow a rule mining approach, such as Inductive Logic Programming (ILP).

Background

2.1 Understanding the domain

2.1.1 Immune system and antibodies

The immune system is the central part of the human body responsible for protection against infections. In order to function properly, the immune system has to detect a wide range of threats, and at the same time distinguish them from healthy tissue. The main weapon the immune system can use are antibodies.

The antibody is a protein complex produced by *B cells*¹ that initiates an immune response against a target antigen². Their primal role is to recognize the unique part of the foreign target and protect the body from infections. The basic organization of the antibody includes two functional domains that, together, resemble the letter Y (Figure 2.1, left). The *Fab* part makes up the arms of the Y, and it contains the antigen-binding site - the region responsible for antigen binding. The *Fc* part comprises the tail of the Y and effects other cells, proteins and antibodies.

This unique structure allows detection of antigens, in direct or indirect manner. By direct detection we assume detection using a single fluorophore-labeled antibody, and by indirect detection we assume detection through binding of a fluorophore-labeled secondary antibody raised against the *Fc* part of an unlabeled primary antibody (as illustrated in Figure 2.1, right). This system is versatile and cost-effective because few labeled antibodies are required to detect many possible primary antibodies.

¹ a subgroup of white blood cells, a viral part of the immune system

²Foreign substance that, when introduced into the body, is capable of stimulating an immune response

2. BACKGROUND

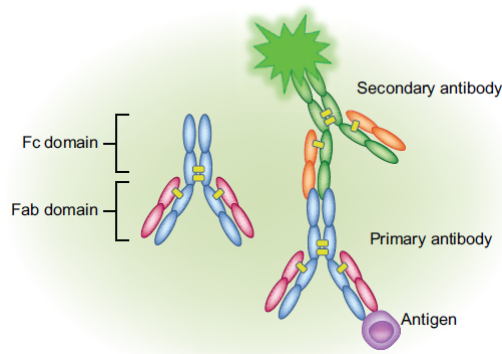


FIGURE 2.1: Illustration of antibody structure (from [7])

The immune system can sometimes suffer from different disorders. A disorder of special importance to this Thesis is *autoimmunity*. Autoimmunity results in the disability of the immune system to recognize an organism's healthy tissue, and therefore attacks normal tissues as if they were foreign organisms. In the case of autoimmunity, antibodies are called antinuclear antibodies. We can observe those antibodies by using indirect immunofluorescence.

2.1.2 Indirect Immunofluorescence

As it was already mentioned, the indirect detection is the main focus of the thesis. Indirect immunofluorescence is a diagnostic methodology based on image analysis that reveals the presence of autoimmune diseases by searching for antibodies in the patient serum. Indirect immunofluorescence is a two-step technique, in which a primary, unlabeled antibody binds to the target, after which a fluorophore-labeled³ second antibody is used to detect the first antibody (figure 2.1, right). Indirect immunofluorescence is more sensitive than a direct one because more than one secondary antibody can bind to each primary antibody, which amplifies the fluorescence signal.

As a result of its effectiveness, there has been a growing demand for diagnostic tests for systemic autoimmune diseases. Unfortunately, IIF still remains a subjective method that depends too heavily on the experience and expertise of the physician. The main reasons causing the problems are:

- the lack of quantitative information supplied to physicians
- varieties of reading systems and optics

³fluorophore-labeling is a method to color the antibodies so they can be observed under the microscope

- the photo-bleaching effect caused by a light source irradiating the cells over a short period of time
- the low reproducibility of the diagnostic protocol.

2.1.3 Putting it all together

The focus of this thesis is on the Antinuclear antibodies test (ANA), which plays the main role in the serological⁴ diagnosis of autoimmune disease. ANAs are directed against a variety of antigens and can be detected in patient serum through laboratory tests. IIF uses the human epithelial (HEp-2) substrate, which bonds with serum antibodies forming a molecular complex. This complex then reacts with human immunoglobulin⁵ and becomes visible under a fluorescence microscope which reveals the antigen-antibody reaction.

The procedure of ANA starts with fluorescence intensity classification, a segmentation step is not a part of ANA procedure. The Center for Disease Control and Prevention in Atlanta, USA have published guidelines [10] for scoring the intensity. The score ranges from 0 to 4+ as follows:

- 4+ : brilliant green (maximal fluorescence)
- 3+ : less brilliant green fluorescence
- 2+ : defined pattern but diminished fluorescence
- 1+ : very subdued fluorescence
- 0 : negative.

Although the guidelines provide very detailed instructions, in [14] Rigon et al. analyzed the variability between a set of physician's fluorescence intensity classifications. Their work has shown a big variance of classifications made by physicians on the same dataset, so they suggested to classify the fluorescence intensity into three classes, namely negative, intermediate and positive. This work follows the protocol.

The final step consists of staining pattern recognition. As shown in figure 1.1, there are several patterns that may be observed. [3] and [13] provide a description of all staining patterns which is a valuable input taking into consideration a human interpretable perspective of this step. A summary is presented here:

- **Centromere:** characterized by several discrete speckles (~ 40 - 60) distributed throughout the interphase⁶ nuclei and characteristically found in the condensed nuclear chromatin during mitosis as a bar of closely associated speckles.

⁴Further explanation

⁵Immunoglobulin is a specific type of antibody created by plasma cells

⁶The interphase is the nonmitotic phase of the cell cycle in which the cell spends the majority of its time and performs the majority of its purposes

2. BACKGROUND

- **Nucleolar:** characterized by clustered large granules in the nucleoli of interphase cells which tend towards homogeneity, with less than six granules per cell.
- **Homogeneous:** characterized by a diffuse staining of the interphase nuclei and staining of the chromatin of mitotic cells.
- **Fine Speckled:** characterized by a fine granular nuclear staining of the interphase cell nuclei.
- **Coarse Speckled:** characterized by a coarse granular nuclear staining of the interphase cell nuclei.
- **Cytoplasmatic:** characterized by a highly irregular shape and large granule in the nucleoli

2.2 Machine learning

The main tool for solving such tasks is *machine learning*. The goal of machine learning is to build *intelligent* systems that improve their performance with experience. Machine learning systems typically learn concepts from examples or learn skills, by observation or by interacting with the environment. For example, a machine learning system can predict which web page you want to visit, filter your spam emails, or assist doctors with diagnosing patients based on their medical records.

Machine learning algorithms work by learning *models* of the world. These models represent a function of the concept being learned, which depends of the system's inputs. A model can be learned for one of two purposes. First, the model can be learned to predict its environment and take actions according to its prediction. Second, the model can assist humans in understanding the environment, where model can be interpreted and understood by humans.

Models can be learned in different settings. First, in *supervised* learning settings, models can access both input and output of a system they try to imitate. In *unsupervised* settings, only inputs are known so task is to identify regularities in input data. *Reinforcement* learning, on the other hand, represents the most different setting, where model is inferred continuously taking actions and receiving *prize* for good actions and *punishment* for bad actions.

Based on a way how these models are inferred, they can be divided in two categories. *Probabilistic* models are built so they maximize the *likelihood* of examples observed. Likelihood of data could be interpreted as a measure of how well the

parameters of a model describe the observed data. More formally, if \mathbf{x}_i is one instance from data set \mathcal{D} , the likelihood \mathcal{L} of model with parameters θ can be calculated as

$$\mathcal{L}(\mathcal{D}|\theta) = \prod_{i=1}^N p(\mathbf{x}_i|\theta). \quad (2.1)$$

In the second category are *discriminative* models. These models treat each data example as a point in n -dimensional space, where n is a number of *features*, an information units, describing the concept. These models learn concepts by finding a *hyperplane* that separates the concepts in feature space. The hyperplane is chosen to minimize a *misclassification* error defined as

$$\text{error rate} = \frac{\text{\#correctly classified examples}}{\text{\#examples}}. \quad (2.2)$$

Related work

3.1 Cell segmentation

Several studies have been proposed to classify autoantibody fluorescence patterns by using an automatic thresholding method, i.e. Otsu's method, to segment the cells. The thresholding method can choose the threshold to minimize the intraclass variance of the black and white pixels automatically. Due to the variety of ANA patterns, Otsu's algorithm always failed to segment cells of speckled and nucleolar patterns, such as cases of a very blurry image of low intensity. Figure 3.1 shows the over-segmentation results by using Otsu's algorithm. Additional challenge for the segmentation here is to separate overlapping cells which are quite common in the process.

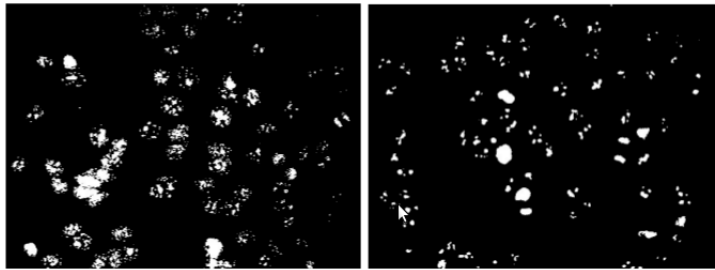


FIGURE 3.1: Segmentation results of the Otsu method (from [6])

In [6], Huang et al. present an adaptive edged-based segmentation method for automatically detecting outlines of fluorescence cells in IIF images. Their approach is based on specific properties of the images regarding the pattern class. They have divided the images in two groups : sparse region and mass region cells. The mass region cells are those ones which have a *compact* appearance, that look like a smooth object, while the sparse region cells are those ones for which we can detect multiple

object in a cell. The approach trains a classifier to classify each image in the groups and applies different segmentation procedure for each group. In the case of the mass region cells, the cells are segmented using Otsu segmentation method, while in the case of the sparse region cell segmentation is performed by an edge detection. Their approach resulted in better segmentation results, but approximately 10% of the cells remained undetected.

In [5], same authors further improve their method by incorporating watershed segmentation. The second approach also includes segmentation in two stages, depending on defined criteria. After a segmentation step performed by the watershed, the approach merges parts located relatively close and eliminates parts not large enough to represent a cell. If the retrieved number of regions doesn't satisfy the defined criteria, the segmentation step is performed again with different parameter settings determined by Otsu's thresholding.

All forementioned approaches report similar shortcomings : approximately 10% of cells remained undetected and the inability to separate overlapping cells. The focus of the segmentation part of the Thesis will be on overcoming those problems.

3.2 Intesity level classification

The following step, the intensity level classification, hasn't attracted a lot of scientific research, but has demonstrated remarkable results so far.

In [15], authors propose a system based on *Multi-Layer Perceptrons* and a *Radial Basis Network* for the intensity classification step. That system, which makes use of features inspired from medical practice, shows error rates up to 1%, but it uses a reject option and it does not cast a result in about 50% of cases. In [16] the authors further refine their system. They train three experts, one specialized for each class, with a different set of features. They threat the classifiers similar to the *one-vs-all* approach, so the final decision is made by a classifier most certain in it's decision. The authors report a success rate of 92,6% accuracy.

3.3 Staining pattern classification

As this problem was emphasized on the *International Conference on Pattern Recognition 2012* as a contest, this step has been well researched and several very successful methods have been proposed. In [3], Foggia et al. provides a detailed overview of the methods submitted for the contest. The three most successful ones are presented here.

In [8], Kuan presents a method based on four texture descriptors: a rotation invariant form of local binary patterns (LBP) with multi-scale analysis, discrete cosine transformation, the mean values and standard variances of 2-D Gabor wavelets, and some global appearance based statistical features. A multiclass SVM was trained on each class of the four feature sets. The SVMs are then integrated in one classifier by using the AdaBoost.M1 algorithm.

In [12], Nosaka presents a similar approach on an extension of LBP, namely CoALBP [11]. The advantage of this method is that the method can observe not only local LBP, but also the spatial relations among adjacent LBP. The classifier is a linear SVM trained on an extended dataset including the rotated patterns of the original images.

Xianfeng et al. proposed a system based on MR8 method [17] to extract statistical intensity features. The method calculates filter responses locally on the image, and then trains a global texton dictionary using K -means clustering. In that way, each image is represented by the frequency histogram of textons. The decision is made by a k -NN classifier.

Although there are many more papers in existence covering this problem, they are not presented here due to different, and not so rigorous, evaluation. Most of the early work was done on private datasets not available to public, which makes them not comparable to the new research results. Other papers with a more recent date do not follow the evaluation procedure, so it is very hard to compare their efficiency with those ones in the overview.

More recently, a new overview of the method was presented by Agrawal et al. in [2]. The committee of *ICPR'13* has released a new, much bigger dataset for the same problem. The authors experimented with the most commonly used features in the previous contest - statistical features, histograms of oriented gradients, shape and size descriptors and texture descriptors. They have chosen the most typical representatives of classifiers, namely Naive Bayes, k -NN, SVM and Random forest. The SVM with Law's textural representation significantly outperformed other classifiers and feature representations.



Cell segmentation

Finding the cells in images is the first and crucial step in the procedure. The problems encountered in the cell segmentation are discussed in Chapter 3. All of the problems occur due to coloring procedure that is time dependent, so cells expose different properties over an image. Because of that, part of cells remain undetected and overlapping. The challenge here is to find a method that can overcome different illumination of objects. Proposed solution deals with mentioned problems in separate steps.

The proposed solution is based on an observation that, although cells exhibit different properties across image due to different illumination levels, the background is uniform over a whole image and exhibit constant properties. Following the observations, the method first segments the background to find locations of the cells and then uses second segmentation step to refine the borders of cells.

4.1 Background segmentation

The first step of background segmentation is intended to overcome the problem of undetected cells. A desirable property of segmentation algorithm for this task is the capability of capturing global properties of the background region across an image. One such algorithm is *Region growing*. Region growing is a simple method that extends a region based on a similarity in intensity levels - the intensity of each candidate pixel is compared with a mean intensity of a current region. If a difference is less than a given threshold, the pixel is added to region.

The algorithm is summarized in algorithm 1. It starts with a single point, adds its neighbors to candidate region and compares their intensity to the mean intensity of the region. If a candidate point satisfies the criteria, it is added to the region and its neighbors become candidate points. The process is repeated while there are

4. CELL SEGMENTATION

candidate points to be tested.

Algorithm 1 Region Growing

```
1: function REGIONGROWING(starting point, criteria)
2:   candidates  $\leftarrow$  {neighborhood(starting point)}
3:   region  $\leftarrow$  {starting point}
4:   currentpoint  $\leftarrow$  starting point
5:   while no new candidates do
6:     for each candidate of current point do
7:       if candidate satisfies criteria then
8:         add it to region
9:         update parameters of region
10:      end if
11:    end for
12:    currentpoint  $\leftarrow$  next point from border
13:  end while
14: end function
```

The bottleneck of the method is the evaluation of candidate pixels. If target regions are big, as in this case, the method could be very slow. To overcome this bottle neck, the method is extended with a pre-filling step in which a part of an image is assigned to belong to the background by default. As it can be noticed from images, the majority of image is a dark background which can be observed in an image histogram as the highest peak. To speed up the segmentation, every pixel with intensity lower or equal to the highest peak is automatically assigned to the background.

The issue here is the estimation of a threshold for comparison. It is a parameter that depends on a distribution of intensities over an image. In a sense, this parameter models a variance in intensity of the background region. Motivated by a variance fitting, the proposed solution is based on an assumption that an image histogram should consist of two regions - one modeling the background and a second one modeling cells, as illustrated in image 4.1. The goal now is to find those two regions in the histogram. One way to accomplish that is to approximate the histogram with a mixture of Gaussian functions. This approach is taken here.

4.1.1 Gaussian mixture model

Mixture density is a linear combination of K probabilistic density functions:

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k p(\mathbf{x}|\boldsymbol{\theta}_k) \quad (4.1)$$

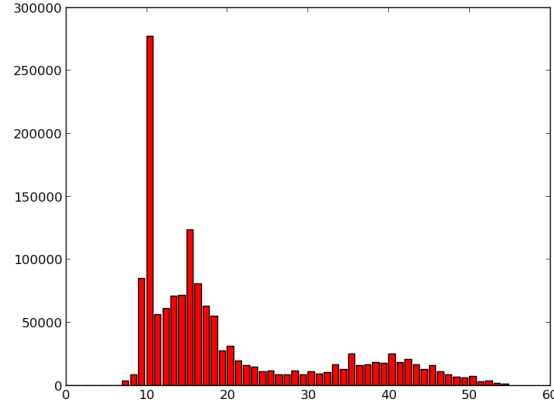


FIGURE 4.1: Example of an image histogram

where $p(\mathbf{x}|\theta_k)$ represents mixture components with their parameters θ_k . In this case, these mixture components are Gaussian functions $\mathcal{N}(\mu_k, \Sigma_k)$. Parameters π_k are mixture coefficients that satisfy $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^K \pi_k = 1$. As they satisfy given criteria, these coefficient can be treated as prior probability of a component k . Equation 4.1 can be then rewritten as :

$$p(\mathbf{x}) = \sum_{k=1}^K P(\mathcal{G}_k) p(\mathbf{x}|\mathcal{G}_k), \quad (4.2)$$

where $P(\mathcal{G}) = \pi_k$ and $p(\mathbf{x}|\theta_k) = p(\mathbf{x}|\mathcal{G}_k)$. The task now is to determine the parameters

$$\theta = \{P(\mathcal{G}_k), \theta_k\}_{k=1}^K, \quad (4.3)$$

or more precisely for the Gaussian function

$$\theta = \{P(\mathcal{G}_k), \mu_k, \Sigma_k\}_{k=1}^K. \quad (4.4)$$

An efficient algorithm to determine those values is the Expectation Maximization algorithm.

4.1.2 Expectation Maximization Algorithm

The goal of the Expectation Maximization algorithm is to find parameters θ that maximize the log-likelihood

$$\mathcal{L}(\theta|\mathcal{D}) = \ln \prod_{i=1}^N p(\mathbf{x}^{(i)}) = \ln \prod_{i=1}^N \sum_{k=1}^K p(\mathbf{x}^{(i)}|\theta_k) = \sum_{i=1}^N \ln \sum_{k=1}^K p(\mathbf{x}^{(i)}|\theta_k). \quad (4.5)$$

Unfortunately, there is no closed-form solution for this formulation. Model $p(\mathbf{X}|\theta)$ is extended with latent variable \mathbf{Z} which determines to which cluster belongs every x_i . The density function is now described with $p(\mathbf{X}, \mathbf{Z}|\theta)$. Marginal density $p(\mathbf{X}|\theta)$, which is density of interest, can always be reconstructed from a joint probability by marginalization :

$$p(\mathbf{X}|\theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta).$$

The log-likelihood that has to be maximized is now

$$\ln \mathcal{L}(\theta|\mathbf{X}) = \ln p(\mathbf{X}|\theta) = \ln \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta). \quad (4.6)$$

As values of the latent variable \mathbf{Z} are not yet known, the log-likelihood cannot still be used directly. Instead, the expectation of the log-likelihood $\mathbb{E}[\ln \mathcal{L}(\theta|\mathbf{X}, \mathbf{Z})]$ is used. The main idea behind the EM-algorithm is to iteratively adjust parameters θ to maximize the expectation.

The maximization of $\mathbb{E}[\ln \mathcal{L}(\theta|\mathbf{X}, \mathbf{Z})]$ is now done by switching between two steps – E-step and M-step. E-step (*expectation step*) calculates the expectation of the log-likelihood with respect to the current values of the parameters $\theta^{(t)}$. That expectation $\mathcal{Q}(\theta|\theta^{(t)})$ we can calculate as

$$\begin{aligned} \mathcal{Q}(\theta|\theta^{(t)}) &= \mathbb{E}_{\mathbf{Z}|\mathbf{X}, \theta^{(t)}} [\ln \mathcal{L}(\theta|\mathbf{X}, \mathbf{Z})] \\ &= \mathbb{E}_{\mathbf{Z}|\mathbf{X}, \theta^{(t)}} [\ln p(\mathbf{X}, \mathbf{Z}|\theta)] \\ &= \sum_{\mathbf{Z}} P(\mathbf{Z}|\mathbf{X}, \theta^{(t)}) \ln p(\mathbf{X}, \mathbf{Z}|\theta). \end{aligned} \quad (4.7)$$

The expectation is now expressed on variable \mathbf{Z} with fixed values of \mathbf{X} and θ . Probability $P(\mathbf{Z}|\mathbf{X}, \theta^{(t)})$ is the *a posteriori* probability of the latent variable \mathbf{Z} with fixed parameters which can be evaluated using Bayes rule.

Only thing left is to optimize the parameters θ . M-step (*maximization step*) chooses new parameters $\theta^{(t+1)}$ by maximizing the expression

$$\theta^{(t+1)} = \arg \max_{\theta} \mathcal{Q}(\theta|\theta^{(t)}). \quad (4.8)$$

EM-algorithm is briefly summarized in Algorithm 4.1.2. Starting with the initial parameters $\theta^{(0)}$, it iteratively switches between E and M step until converged. The convergence is guaranteed as algorithm maximizes the expectation in every iteration. On the other hand, found solution might not be the global optimum.

Algorithm 2 Expectation-maximization algorithm

```

1: function EM
2:   Initialize parameters  $\theta^{(0)}$ 
3:    $t \leftarrow 0$ 
4:   repeat
5:     E-step: calculate  $P(\mathbf{Z}|\mathbf{X}, \theta^{(t)})$ 
6:     M-step:  $\theta^{(t+1)} \leftarrow \arg \max_{\theta} \mathcal{Q}(\theta|\theta^{(t)})$ 
       where  $\mathcal{Q}(\theta|\theta^{(t)}) = \sum_{\mathbf{Z}} P(\mathbf{Z}|\mathbf{X}, \theta^{(t)}) \ln p(\mathbf{X}, \mathbf{Z}|\theta)$ 
7:      $t \leftarrow t + 1$ 
8:   until converged
9: end function

```

4.1.3 EM algorithm for mixture models

The EM-algorithm for mixture models works as follows. Let $\mathbf{z} = (z_1, \dots, z_K)$ be a latent variable vector indicating a cluster to which an example belong to. $z_k = 1$ if an example belongs to a cluster \mathcal{G}_k , $z_k = 0$ otherwise. Each cluster is assigned with a prior probability

$$P(z_k = 1) = \pi_k,$$

such that $\sum_k \pi_k = 1$. Distribution over the latent variable \mathbf{z} hence can be expressed as

$$P(\mathbf{z}) = \prod_{k=1}^K \pi_k^{z_k}. \quad (4.9)$$

To evaluate the probability $p(\mathbf{x}, \mathbf{z}|\theta)$ the factor $p(\mathbf{x}|\mathbf{z}, \theta)$ is missing. It can be expressed as

$$p(\mathbf{x}|\mathbf{z}, \theta) = \prod_{k=1}^K p(\mathbf{x}|\theta_k)^{z_k}. \quad (4.10)$$

The joint probability $p(\mathbf{x}, \mathbf{z}|\theta)$ can now be expressed as

$$p(\mathbf{x}, \mathbf{z}|\theta) = P(\mathbf{z})p(\mathbf{x}|\mathbf{z}, \theta) = \prod_{k=1}^K \pi_k^{z_k} \prod_{k=1}^K p(\mathbf{x}|\theta_k)^{z_k}. \quad (4.11)$$

Using the model obtained in 4.11, the log-likelihood $\ln \mathcal{L}(\theta|\mathcal{D}, \mathcal{Z})$ can be expressed. Take $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ as a set of examples and $\mathcal{Z} = \{\mathbf{z}^{(i)}\}_{i=1}^N$ as a set of latent

variables describing the relationship between examples and model. Log-likelihood of the model 4.11 is

$$\begin{aligned}\ln \mathcal{L}(\boldsymbol{\theta}|\mathcal{D}, \mathcal{Z}) &= \ln \prod_{i=1}^N p(\mathbf{x}^{(i)}, \mathbf{z}^{(i)}|\boldsymbol{\theta}) = \ln \prod_{i=1}^N \prod_{k=1}^K \pi_k^{z_k^{(i)}} p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k)^{z_k^{(i)}} \\ &= \sum_{i=1}^N \sum_{k=1}^K z_k^{(i)} \left(\ln \pi_k + \ln p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k) \right).\end{aligned}\quad (4.12)$$

E-step

In the E-step of the algorithm, the expectation $\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ is calculated:

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) &= \mathbb{E}_{\mathcal{Z}|\mathcal{D}, \boldsymbol{\theta}^{(t)}} [\ln \mathcal{L}(\boldsymbol{\theta}|\mathcal{D}, \mathcal{Z})] \\ &= \mathbb{E}_{\mathcal{Z}|\mathcal{D}, \boldsymbol{\theta}^{(t)}} \left[\sum_{i=1}^N \sum_{k=1}^K z_k^{(i)} \left(\ln \pi_k + \ln p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k) \right) \right] \\ &= \sum_{i=1}^N \sum_{k=1}^K \mathbb{E} \left[z_k^{(i)} | \mathcal{D}, \boldsymbol{\theta}^{(i)} \right] \left(\ln \pi_k + \ln p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k) \right).\end{aligned}\quad (4.13)$$

As $z_k^{(i)}$ is a Bernoulli variable (as only one component of $\mathbf{z}^{(i)}$ can be 1) which depends only on one example from \mathcal{D} , $\mathbf{x}^{(i)}$, it holds

$$\mathbb{E} \left[z_k^{(i)} | \mathcal{D}, \boldsymbol{\theta}^{(t)} \right] = \mathbb{E} \left[z_k^{(i)} | \mathbf{x}^{(i)}, \boldsymbol{\theta}^{(i)} \right] = P(z_k^{(i)} = 1 | \mathbf{x}^{(i)}, \boldsymbol{\theta}^{(i)}). \quad (4.14)$$

The expectation of the latent variable equals to its *a posteriori* probability and it can be evaluated by applying the Bayes rule:

$$\begin{aligned}P(z_k^{(i)} = 1 | \mathbf{x}^{(i)}, \boldsymbol{\theta}^{(i)}) &= \frac{p(\mathbf{x}^{(i)} | z_k^{(i)} = 1, \boldsymbol{\theta}^{(i)}) P(z_k^{(i)} = 1)}{\sum_{j=1}^K p(\mathbf{x}^{(i)} | z_j^{(i)} = 1, \boldsymbol{\theta}^{(i)}) P(z_j^{(i)} = 1)} \\ &= \frac{p(\mathbf{x}^{(i)} | z_k^{(i)} = 1, \boldsymbol{\theta}^{(i)}) \pi_k^t}{\sum_{j=1}^K p(\mathbf{x}^{(i)} | z_j^{(i)} = 1, \boldsymbol{\theta}^{(i)}) \pi_j^t} \equiv h_k^{(i)}.\end{aligned}\quad (4.15)$$

Hence, the expectation of the log-likelihood equals

$$\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \sum_{i=1}^N \sum_{k=1}^K h_k^{(i)} \ln \pi_k + \sum_{i=1}^N \sum_{k=1}^K h_k^{(i)} \ln p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k). \quad (4.16)$$

M-step

M-step maximizes the log-likelihood obtained in 4.16. The solution is found analytically by solving $\nabla_{\theta} \mathcal{Q}(\theta|\theta^{(t)})$. To obtain the mixture components, $\pi_k^{(t+1)}$, the equation is $\nabla_{\pi_k} \mathcal{Q}(\theta|\theta^{(t)})$. As the second term in 4.16 does not depend on π_k , so it can be ignored. Starting from

$$\nabla_{\pi_k} \left(\sum_{i=1}^N \sum_{k=1}^K h_k^{(i)} \ln \pi_k + \lambda \left(\sum_k \pi_k - 1 \right) \right) = 0, \quad (4.17)$$

where *Langrange multiplier* is used to satisfy the condition $\sum_{k=1}^K \pi_k = 1$, it can be easily obtained

$$\pi_k^{(t+1)} = \frac{1}{N} \sum_{i=1}^N h_k^{(i)}. \quad (4.18)$$

To find parameters of each component, $\theta_k^{(i)}$, the equation $\nabla_{\theta_k} \mathcal{Q}(\theta|\theta^{(t)}) = 0$ has to be solved. Now first term in the equation 4.16 does not depend on θ_k so it can be ignored

$$\nabla_{\theta_k} \sum_{i=1}^N \sum_{k=1}^K h_k^{(i)} \ln p(\mathbf{x}^{(i)}|\theta_k). \quad (4.19)$$

Substituting $p(\mathbf{x}^{(i)}|\theta_k)$ with a multidimensional Gaussian distribution and deriving with regards to μ_k and σ_k^2 new parameters are calculated

$$\mu_k^{(t+1)} = \frac{\sum_i h_k^{(i)} x^{(i)}}{\sum_i h_k^{(i)}} \quad (4.20)$$

$$(\sigma^2)_k^{(i)} = \frac{\sum_i h_k^{(i)} (x^{(i)} - \mu_k^{(t+1)})(x^{(i)} - \mu_k^{(t+1)})^T}{\sum_i h_k^{(i)}}. \quad (4.21)$$

4.1.4 Threshold estimation

Once histogram has been approximated with two Gaussian functions, obtained functions are used to find a threshold. Such approximation is illustrated in figure 4.1.4. A Gaussian with a lower mean is assumed to model the background. By examining its variance, the threshold is estimated. The threshold is now defined as a distance from the lower mean to a point where two Gaussian are equally probable. Additional restriction that has to be satisfied is that this point should be placed

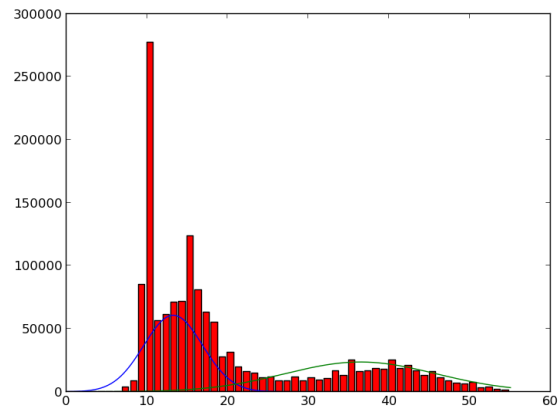


FIGURE 4.2: Approximation of an image histogram

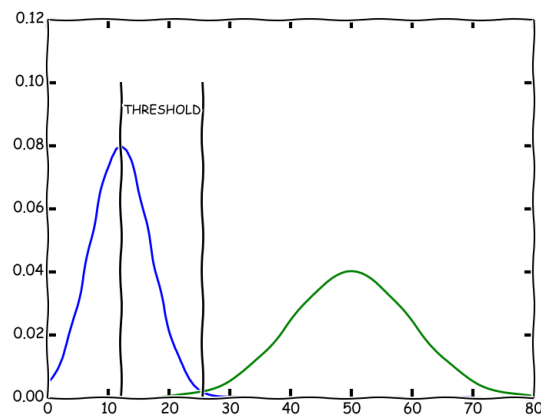


FIGURE 4.3: A calculation of a threshold

between the means of Gaussians. The principle used for calculating the threshold is illustrated in figure 4.1.4.

The point of using the background segmentation step is to overcome a problem of undetected cells that expose a low intensity. To evaluate this step, the optimal threshold for each image in dataset was determined manually. The evaluation was performed in two ways - on object and pixel level. First, the number of detected cells was used as the number of cells in each image is known in advance. Second, segmented region were compared to the ground truth. As a measure of similarity *precision* is used - number of pixels in segmented regions that is contained in the ground truth segmentation divided by the number of pixels in the ground truth segmentation. This measure should indicate a similarity of obtained segmentation to a manually segmented cells.

Table 4.1 summarizes the object level results. The results show that the Region growing approach usually detects more objects than contained in the ground truth segmentation. Those extra object detected by the proposed approach are actually partial cells that were removed from the ground truth segmentation. The presence of those partial cells demonstrates the advantages and sensitivity of region growing, but also raises a question should those cells be eliminated from further process or used as a *fully observed* cells.

4.2 Positioning a prior

Once the background has been found, we have a rough estimate of cell positions. Still, a lot of cells overlap. In order to split them, we will make use of circularity properties of the cells. If circles of cells could be detected, or at least circular parts of cells, they might be split. To detect circles Hough transformation is used.

4.2.1 Hough transformation for circles

Hough transform is a general voting procedure used in Computer vision. The outline of the method is given in algorithm 3. It discretize a parameter space and assigns every bin a number of votes proportional to the number of edges in an image that could be generated with that specific bin parameters. In this case, each bin represents one candidate circle in an image with equation

$$\mathcal{H}(\hat{x}, \hat{y}, r) = (\hat{x} - x)^2 + (\hat{y} - y)^2 = r^2, \quad (4.22)$$

where \hat{x} and \hat{y} represent the origin of a circle and r its radius. Now every edge point in an image *votes* for every bin that could have generated it, under assumption that the edge is a part of a circle.

After voting, every local maximum is selected as a circle. Additionally, every vote could be weighted proportionally to its magnitude. Note that not all local

TABLE 4.1: Number of detected and overlapping cells

image ID	objects	detected ob- jects	overlapping objects
01	64	69	35
02	52	63	13
03	93	99	38
04	73	73	28
05	52	52	16
06	77	77	36
07	62	62	28
08	60	60	27
09	52	55	16
10	36	42	13
11	49	56	16
12	57	60	49
13	52	57	25
14	67	78	33
15	68	76	58
16	44	48	16
17	20	20	6
18	43	53	15
19	70	77	4
20	49	49	11
21	66	70	9
22	120	127	76
23	53	56	15
24	75	85	45
25	24	24	14
26	47	47	47
27	44	44	40
28	13	13	6

Algorithm 3 Hough Transform

```

1: function HOUGH TRANSFORMATION
2:   initialize accumulator  $\mathcal{H}$  to all zeros
3:   for each edge in image do
4:     increment every cell  $\mathcal{H}(x, y, r)$  which could be the center of a circle
5:   end for
6:   search for local maxima cells of  $\mathcal{H}$ 
7: end function

```

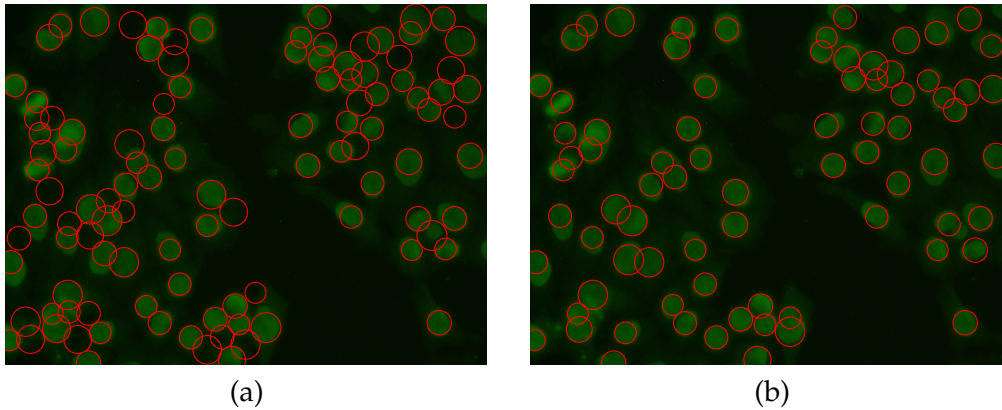


FIGURE 4.4: Hough transformation with all detected circles (a) and after removing background circles (b)

maxima are actual circles. Although quite straightforward, the method can easily become unfeasible. Searching for all radius's is obviously unfeasible for large images, so any restriction on circle size range is most certainly helpful. To help the search, radius is limited to a range of minimal to the largest radius found in the data set.

Another problem with Hough transformation is its sensitivity to noise. If we assume there is an uniform noise in an image, it is easy to conclude that might imply false circles over an image. Methods to overcome that problem usually include better discretization of parameters or smoothing in the accumulator by incrementing neighboring bins, but for fully automated application this is not suitable solution. Instead, information about background extracted in the previous step could be used. As we can obtain background quite successfully, every circle with a center in the background region can be removed. so as every other circle which has similar properties as the background circles. As a criteria here, a mean intensity in the green channel is used - every circle with a mean intensity lower than the highest mean intensity found in the background is removed. Figure 4.4 illustrates that selection. Now each circle serves as a seed point for precise segmentation of contours and splitting overlapping cells.

4.3 Precise contours with Morphological snakes

Morphological snakes or active contour methods are one very popular computer vision method. This method employs a energy optimization approach where it tries to minimize an energy functional induced by a curve. The minimum is usually find by a steepest descent approach.

As the images can be quite noisy, a specific version of morphological snakes called *Active contour without edges* (ACWE) is used. The ACWE doesn't need well defined borders and it is less sensitive to the initial configuration and model param-

eters. Due to a noisy nature of images used in this process, the ACWE has a potential to successfully deal with the noise. The Active contour methods try to adjust the starting contour to a local minimum by iteratively solving a time-dependent partial differential equations (PDE). However, solving the PDE is computationally costly and often has a stability issues.

The most efficient approaches avoid to solve the PDE directly. The key idea that allows the simplification of the problem is that the curve is implicitly represented by the set of image points neighboring the target area. The evolution of curve is then achieved by elimination and introduction of points in the target set. Any geometric property can be approximated with local operations on the set of points. Those methods are based on mathematical morphology and make use of morphological operators such as *erosion* and *dilation* to approximate the PDEs. For the equivalence of differentiation and morphological operators see [9].

4.3.1 Background

This section summarizes the most important parts of curve evolution with PDEs and morphological operators.

Contour evolution with PDEs

Differential operators are key to contour evolution with PDEs – the change of contour is given by a differential operator.

Let \mathcal{C} be a parametrized 2D curve over time so that $\mathcal{C} : \mathbb{R}^+ \times [0, 1] \rightarrow \mathbb{R}^2 : (t, p) \rightarrow \mathcal{C}(t, p)$. A differential operator \mathcal{L} defines the curve evolution with PDE $\mathcal{C}_t = \mathcal{L}(\mathcal{C})$. The result of evolution depends on the differential operator. There are many forms in which it can be written but one popular is $\mathcal{L} = \mathcal{F} \cdot \mathcal{N}$, where \mathcal{N} is the normal to the curve and \mathcal{F} is a scalar field which determines the speed of evolution.

A particularly interesting form is the one in which \mathcal{L} is defined with $\mathcal{F} \in \{-1, 1\}$. Such operator evolves the curve along its normal direction with constant speed. If \mathcal{F} takes the form of the Euclidean curvature of \mathcal{C} , then the problem is convex so it will converge to the optimal solution.

With the explicit representation of curve certain problems occur. It is not easy to deal with topological changes and re-parameterization of the curve. An alternative to the explicit representation is an implicit one. Let $u : \mathbb{R}^+ \times \mathbb{R}^2 \rightarrow \mathbb{R}$ an implicit representation of \mathcal{C} such that $\mathcal{C}(t) = \{(x, y); u(t, (x, y)) = 0\}$. When $\mathcal{C}_t = \mathcal{F} \cdot \mathcal{N}$, the curvature evolution of any function $u(x, y)$ which embeds the curve is $\frac{\partial u}{\partial t} = \mathcal{F}|\nabla u|$.

With that in mind, the PDEs that are of interest to the curvature evolution are

$$\frac{\partial u}{\partial t} = \pm |\nabla u| \tag{4.23}$$

$$\frac{\partial u}{\partial t} = \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) \cdot |\nabla u| \quad (4.24)$$

where $\mathcal{F} = \pm 1$ in 4.23 and $\mathcal{F} = \mathcal{K}$ for 4.24. \mathcal{K} denotes the Euclidean curvature.

Morphological operators

Morphological operators are monotone operators that are translation- and contrast-invariant. Every morphological operator T can be represented in a *sup* – *inf* representation of form

$$(T_h u)(\mathbf{x}) = \sup_{B \in \mathcal{B}} \inf_{\mathbf{y} \in \mathbf{x} + hB} u(\mathbf{y}) \quad (4.25)$$

or a dual *inf* – *sup* form

$$(T_h u)(\mathbf{x}) = \inf_{B \in \mathcal{B}} \sup_{\mathbf{y} \in \mathbf{x} + hB} u(\mathbf{y}), \quad (4.26)$$

where \mathcal{B} is a set of structuring elements that uniquely defines the operator and h is the size of operator.

The most common morphological operators are dilation and erosion. A dilation D_h with radius h of function u is defined as

$$D_h u(\mathbf{x}) = \sup_{\mathbf{y} \in hB(0,1)} u(\mathbf{x} + \mathbf{y}) \quad (4.27)$$

while the erosion has a similar form

$$E_h u(\mathbf{x}) = \inf_{\mathbf{y} \in hB(0,1)} u(\mathbf{x} + \mathbf{y}), \quad (4.28)$$

where $B(0,1)$ is a ball of radius 1 centered at 0 and h is the scaling factor.

Curvature operator

A curvature operator works as follows. Let SI_h and IS_h be respectively *sup* – *inf* and *inf* – *sup* morphological operators with a base $\mathcal{B}^2 = \{[-1, 1]_\theta \subset \mathbb{R}^2 : \theta \in [0, \pi)\}$. The base is made of all segments of length 2 centered at the origin. It can be proved that the successive application of a *mean operator*, for a very small h , is equivalent to the curvature flow of PDE 4.24.

The mean operator is now defined as a composition of two morphological operators

$$T_{h/2}^2 \circ T_{h/2}^1 \approx \frac{T_h^2 u + T_h^1 u}{2}. \quad (4.29)$$

4.3.2 Morphological snakes

It can be proven that above mentioned morphological operators - dilation, erosion and the curvature flow operator - behave infinitesimally like PDEs 4.23 and 4.24. In that context, given the curvature evolution function which includes both terms 4.23 and 4.24, those operators may be used to evolve the contour by approximating the solution to the PDEs.

The ACWE uses an energy functional for image segmentation which takes into account the content of the interior and exterior regions of the curve. The function of a curve \mathcal{C} used in the ACWE looks like

$$\begin{aligned} F(c_1, c_2, \mathcal{C}) = & \mu \cdot \text{length}(\mathcal{C}) + v \cdot \text{area}(\text{inside}(\mathcal{C})) \\ & + \lambda_1 \int_{\text{inside}(\mathcal{C})} \|I(\mathbf{x}) - c_1\| d\mathbf{x} + \lambda_2 \int_{\text{outside}(\mathcal{C})} \|I(\mathbf{x}) - c_2\| d\mathbf{x} \end{aligned} \quad (4.30)$$

where I represent the image and the non-negative parameters $\mu, v, \lambda_1, \lambda_2$ controls the strength of factors. The solution is now found as

$$\min_{c_1, c_2, \mathcal{C}} F(c_1, c_2, \mathcal{C}), \quad (4.31)$$

which is a bit challenging problem. The problem could be relaxed by calculating the parameters given a fixed contour. The values c_1 and c_2 that minimize F are given as the mean of the values of I inside and outside of the contour

$$c_1(\mathcal{C}) = \frac{\int_{\text{inside}(\mathcal{C})} I(\mathbf{x}) d\mathbf{x}}{\int_{\text{inside}(\mathcal{C})} d\mathbf{x}} \quad (4.32)$$

$$c_2(\mathcal{C}) = \frac{\int_{\text{outside}(\mathcal{C})} I(\mathbf{x}) d\mathbf{x}}{\int_{\text{outside}(\mathcal{C})} d\mathbf{x}}. \quad (4.33)$$

The evolution of the contour is now given with the Euler-Lagrange equation of functional 4.30

$$\frac{\partial u}{\partial t} = |\nabla u| \left(\mu \cdot \text{div} \left(\frac{\nabla u}{|\nabla u|} \right) - v - \lambda_1(I - c_1) - \lambda_2(I - c_2) \right). \quad (4.34)$$

This equation specifies the direction contour should evolve to minimize the functional F in a steepest descent manner. The factor

$$|\nabla u| \cdot v \quad (4.35)$$

is called the *balloon force* and it determines the strength of fragments in the contour : when this term is strong the fragment of the contour is far from the target region and it reduces as the fragment approaches the target region. This factor is approximated with the dilation and erosion operators and it represent the approximation of the PDE 4.23.

The factor

$$|\nabla u| \cdot \mu \cdot \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) \quad (4.36)$$

is called the *smoothing force* which represent a weighted version of the mean curvature divergence and corresponds to the PDE 4.24. The μ controls the strength of smoothing operator at each point. This factor is approximated with the mean curvature operator.

The remaining of the equation determines should a point be included in or excluded from the current region, or stay where it is. If $\lambda_1(I - c_1) < \lambda_2(I - c_2)$ at \mathbf{x} , then \mathbf{x} should be included in the current region, or excluded if the inequality is reversed. If $\lambda_1(I - c_1) = \lambda_2(I - c_2)$ \mathbf{x} should stay where it is.

Finally, the ACWE algorithm is given with these three successive steps

$$u^{n+\frac{1}{3}}(\mathbf{x}) = \begin{cases} (D_h u^n)(\mathbf{x}) & \text{if } v < 0 \\ (E_h u^n)(\mathbf{x}) & \text{if } v > 0 \\ u^n(\mathbf{x}) & \text{otherwise} \end{cases} \quad (4.37)$$

$$u^{n+\frac{2}{3}}(\mathbf{x}) = \begin{cases} 1 & \text{if } |\nabla u^{n+\frac{1}{3}}|(\lambda_1(I - c_1)^2 < \lambda_2(I - c_2)^2)(\mathbf{x}) < 0 \\ 0 & \text{if } |\nabla u^{n+\frac{1}{3}}|(\lambda_1(I - c_1)^2 < \lambda_2(I - c_2)^2)(\mathbf{x}) > 0 \\ u^{n+\frac{1}{3}} & \text{otherwise} \end{cases} \quad (4.38)$$

$$u^{n+1}(\mathbf{x}) = \left((SI_h \circ IS_h)^\mu u^{n+\frac{2}{3}} \right)(\mathbf{x}). \quad (4.39)$$

Only question that remains to be answered is how to set the initial curve. As the initial curve the circles detected in the previous step are used.

4. CELL SEGMENTATION

TABLE 4.2: Number of overlapping cells after snakes

image ID	cells	overlapping cells - be- fore	overlapping cells - after
01	64	35	16
02	52	13	2
03	93	38	5
04	73	28	7
05	52	16	2
06	77	36	10
07	62	28	6
08	60	27	5
09	55	16	10
10	42	13	5
11	49	16	6
12	60	49	8
13	52	25	6
14	67	33	6

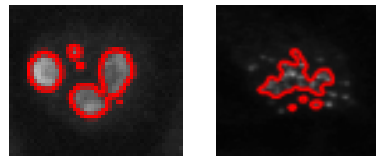


FIGURE 4.5: Oversegmented results

Table 4.2 summarizes the results obtained with the morphological snakes. This method reduced the number of overlapping cells significantly. Still, a problem occurs with the *nucleolar* and *centromere* patterns which have bright organelles. These patterns are usually oversegmented – the output of the segmentation are organelles, not a whole cell (Figure ??). Although those organelles help to split the overlapping cells, they are not the target objects. How to effectively overcome this problem is a problem for future research, but for now, a *convex envelope* of segmentation results were used.

Fluorescence intensity classification

The fluorescence intensity is a parameter that describes the *clarity* of cells in image. It is a subjective parameter which, unfortunately, doesn't have a strong theoretical explanation. The fluorescence intensity is described with three values - positive, intermediate and negative. Positive value defines images in which cells are perfectly separated from background, while negative value defines images in which cells in which cells cannot be identified. The intermediate value covers images that are not positive nor negative. This parameter is used as a symbolic feature later in the process, but for the sake of *standard procedure* it is estimated separately.

In [14] Rigon studied the variability of decision made by doctors and showed that it is hard to achieve a consensus about unique determination of the fluorescence intensity value. The lack of underlying model is making this problem hard to formulate. The intuition behind the suggested approach is an assumption that intensity level could be observed in the histogram of an image. The fluorescence intensity should correspond to the difference to a region describing the background and a region describing cells. Following the intuition, an image histogram is approximated with the Gaussian mixture model.

The idea is to approximate the histogram with a mixture of two Gaussian functions - one representing the background and second one to model the cells. The intuition is that images with positive intensity should have Gaussians with higher means and more further apart.

5.1 Classifying intensity

Once histogram has been approximated with two Gaussian functions, the estimated means and variances have taken as features for the classification. SVM with radial

basis functions as kernel function has been trained for the task. Evaluation is performed using a 10-fold cross validation.

5.1.1 Support Vector Machine

Support vector machine is a very popular machine learning technique. It is a representative of a more general class of *kernel methods*. The most interesting property of the support vector machine is that it tries to find the *optimal hyperplane* that separates classes. Consider a two-class case. The sample is $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N$ where $y^{(i)} = 1$ if $\mathbf{x}^{(i)} \in \mathcal{C}_1$ and $y^{(i)} = -1$ if $\mathbf{x}^{(i)} \in \mathcal{C}_2$. We want to find a margin \mathbf{w} so that

$$\mathbf{w}^T \mathbf{x}^{(i)} + w_0 \geq +1 \quad \text{for } y^{(i)} = 1$$

$$\mathbf{w}^T \mathbf{x}^{(i)} + w_0 \leq -1 \quad \text{for } y^{(i)} = -1$$

or simplified

$$y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + w_0) \geq 1.$$

The support vector machine extends the *basic* hyperplane requirement - setting the instances on the right side of the hyperplane - by trying to maximize the margin - the distance from the hyperplane to the instances closest to it. So, the *optimal separating hyperplane* is the one that maximizes the margin.

The distance of $\mathbf{x}^{(i)}$ to the hyperplane equals to

$$\frac{|\mathbf{w}^T \mathbf{x}^{(i)} - w_0|}{\|\mathbf{w}\|}. \quad (5.1)$$

Finding the maximal margin can be expressed as

$$\frac{y^{(i)}(\mathbf{w}^T \mathbf{x} - w_0)}{\|\mathbf{w}\|} \geq \rho, \forall i, \quad (5.2)$$

i.e., we want the margin to be at least ρ . As there are infinitely many solutions that can be obtained by scaling \mathbf{w} , we fix the $\rho\|\mathbf{w}\| = 1$ and minimize $\|\mathbf{w}\|$ to maximize the margin. The problem can now be written in a form of quadratic optimization problem

$$\begin{aligned} & \min_{\mathbf{w}, w_0} \quad \frac{1}{2} \|\mathbf{w}\|^2 \\ & \text{subject to} \quad y^{(i)} (\mathbf{w}^T \mathbf{x} + w_0) \geq 1. \end{aligned}$$

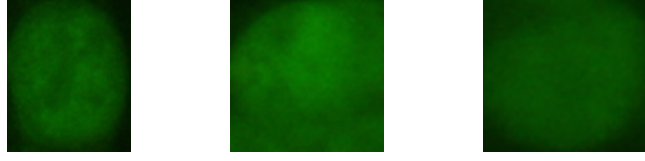


FIGURE 5.1: Potential image that could obtain bad estimation

Now there will be instances that are $\frac{1}{\|\mathbf{w}\|}$ away from the hyperplane and the total margin equals to $\frac{2}{\|\mathbf{w}\|}$.

If the problem is not linearly separable, one trick is to map the problem to a new space, with more dimension than the original space, by using nonlinear basis functions. It is generally the case that higher dimensional representation is easier to separate. One such mapping function is the *radial basis function*.

5.1.2 Radial basis functions

Radial basis functions are an instance of a *local representation*, where for a given input, only a few factors are *active*. It is in a way a partition of a space so that *locally tuned* partitions are selective to only certain inputs.

With the concept of local partitioning we need to define a measure of similarity between an input $\mathbf{x}^{(i)}$ and the local clusters μ^1, \dots, μ^n . The radial basis functions are defined as

$$\mathbf{r}(\mathbf{x}^{(i)}, \mu^k) = \exp\left(-\frac{\|\mathbf{x}^{(i)} - \mu^k\|}{2\sigma_k^2}\right), \quad (5.3)$$

that is, it uses the Euclidean distance as a measure of similarity and Gaussian function as a response function. The response function express a property of having a maximum where $\mathbf{x}^{(i)} = \mu^k$ and decreasing as they get less similar.

5.1.3 Results

One issue that might occur is a bad estimation of the background or cell body as some cells takes a very small portion of an image or, on the other size, takes almost a whole image when segmented as shown in image 5.1. To see how this influences the problem, histograms are approximated with Gaussian functions in two ways. First, the histogram is as in Chapter 4, without any restrictions. Second, the background and cells part of an image are separated and each part is approximated with a Gaussian individually.

Table 5.2 summarizes the results obtained with separation of background and cell regions (a) and without (b). The accuracy of proposed solution is 96.21 % and

FIGURE 5.2: Results of intensity classification

Pred		True	
		pos	int
Pred	pos	804	31
	int	25	595

a)

Pred		True	
		pos	int
Pred	pos	800	36
	int	29	590

b)

96.08 % for separation and without it, respectively. As the results for both cases don't show any significant difference, the stability of method is promising. The best performance reported so far is 92,6 % although the authors have used a private data set that is not available online, so it is hard to make a comparison. So far, the results of this data set are not yet reported.

The evaluation is performed on cell level - the intensity level is estimated on cell level. In practice, the intensity level is assigned on image level. Each cell in the image is then assigned with a level of the image. Averaging the classification over an image, a 100 % accuracy is achieved.

Describing cells

Once cells have been segmented and their fluorescence intensities classified, there are assigned with features that describe a human perception of the cells' properties. The interesting properties are summarized in the following section.

This step maps *pixel features* to *symbolic features* used to describe the cells. Those symbolic features are learned by *Deep belief network*.

6.1 Interesting features

In [3], Foggia et al. summarizes the description of interesting properties for each type of cells. Table 6.1 provides the description of every cell type. The problem with such descriptions is that they are quite unstructured and sometimes ubiquitous. For example, when speaking about organelles contained in a cell's body, they refer to dark organelles as *granules* and bright organelles as *speckles*.

In order to develop a method that will generate such descriptions automatically, those description should be structured first. Tables 6.2 and 6.3 gives a description of each pattern in a more structured way. The appearance of patterns depends on the fluorescence intensity of an image. Several important visual patterns were identified.

The first property describes the shape of a cell. It can take two possible values - *circular* and *irregular*. The motivation for this feature is the observation that the irregular shape is very characteristic for the cytoplasmatic class, while all other classes have a circular shape.

The property *speckles* denotes are there any objects present in the cell or not.

Incorporating the knowledge about the mitotic cells found on the same image can be quite useful. The medical literature suggested that the mitotic cells, although

there is no a general consensus about the importance of the mitotic cells, might be informative for discriminating between certain classes. The mitotic cells can appear in four different patterns - *neutral* with no specific characteristics, *bright center*, *bright center* with sparkles or *dark center*.

The following property describes the appearance of organelles in a cell's body, if any present. This property is very discriminatory towards nucleolar and centromere which have bright organelles, speckled which contain dark organelles while others don't have any organelles.

The number of organelles is also an informative property. The centromere class has significantly larger number of organelles contained in a body. The centromere, nucleolar and speckled class relatively small number of organelles expressed in the body, while the homogeneous and cytoplasmatic class don't have any organelles.

For certain classes, the texture could be quite informative. Although most of the classes have a relatively smooth texture, some classes have some very distinguishing properties. On images with the positive fluorescence intensity, the cytoplasmatic class has characteristic blob. Also, the coarse speckled class has a very specific texture that looks like a *sea surface*.

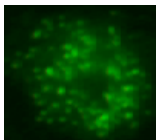
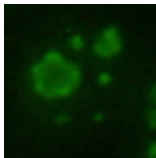
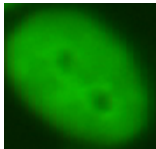
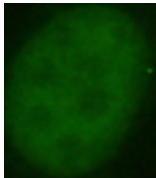
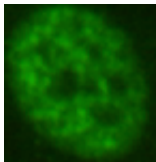
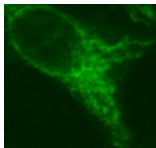
6.2 Deep learning

Deep learning is a relatively new approach in machine learning which is often referred as *Representation learning*. It is built on *Artificial neural networks* and imitates human brain in representing the data. The motivation for learning representation is quite clear – the form in which data is represented is important. The success of our classifier depends on the quality of data used for training.

The key concept used in Deep learning is a *hierarchical representation of data*. Deep learning aims at learning feature hierarchies with features from higher levels of the hierarchies formed by composing a lower level features. Automatically learning features at multiple levels of abstraction allow a system to learn complex functions representing the data, without depending completely on human-crafted features. That is particularly important because humans often do not know how to express higher-level abstraction in terms of raw sensory input. The ability to automatically learn powerful features will become increasingly important as the amount of data and range of applications to machine learning methods continues to grow.

Consider a computer vision framework, for example. Raw input corresponds to raw pixels of an image. The idea of Deep learning is to gradually *abstract* the pixels to higher level features. Raw pixels are further combined to form edges, edges are then abstracted to specific forms and so on. The focus of deep architecture learning

TABLE 6.1: Description of cell types

pattern type	example	description
centromere		characterized by several discrete speckles ($\approx 40 - 60$) distributed throughout the interphase nuclei and characteristically found in the condensed nuclear chromatin during mitosis as a bar of closely associated speckles.
nucleolar		characterized by clustered large granules in the nucleoli of interphase cells which tend towards homogeneity, with less than six granules per cell.
homogeneous		characterized by a diffuse staining of the interphase nuclei and staining of the chromatin of mitotic cells.
fine speckled		characterized by a fine granular nuclear staining of the interphase cell nuclei
coarse speckled		characterized by a coarse granular nuclear staining of the interphase cell nuclei
cytoplasmatic		characterized by a specific shape and large granule

6. DESCRIBING CELLS

TABLE 6.2: Identified visual patterns - positive intensity

pattern type	shape	mitotic cell	organelle type	organelle count	texture	speckles
centromere	circular	X	bright on dark	lots	sparkly	yes
nucleolar	circular	X	bright on dark	few	smooth	yes
cytoplasmatic	irregular	X	dark on bright	none	blob (positive)	no
homogeneous	circular	X	neutral	none	smooth	no
fine speckled	circular	X	neutral	none	smooth	no
coarse speckled	circular	X	dark on bright	few	sparkly	yes

TABLE 6.3: Identified visual patterns - intermediate intensity

pattern type	shape	mitotic cell	organelle type	organelle count	texture	speckles
centromere	circular	X	bright on dark	lots	sparkly	yes
nucleolar	circular	X	bright on dark	few	smooth	yes
cytoplasmatic	irregular	X	neutral	none	smooth	no
homogeneous	circular	X	neutral	none	smooth	no
fine speckled	circular	X	dark on bright	none	smooth	no
coarse speckled	circular	X	dark on bright	few	sparkly	yes

is to automatically discover such abstractions, from the lowest level features to the highest level concepts. Ideally, we would like learning algorithms that enable this discovery with as little human effort as possible.

Depth of architecture refers to the number of levels of composition of non-linear operations in the function learned. Most of current machine learning algorithm correspond to *shallow architectures* while the brain is organized in a *deep architectures*, with multiple levels of abstraction.

Another important thing to notice is that the information is *distributed*: in the brain analogy, the information is not localized in a particular neuron but distributed across many. Each level of abstraction found in the brain consists of the *activation* (neural excitation) of a small subset of a large number of features that are, in general, not mutually exclusive. In addition to being distributed, it appears that the brain uses a representation that is *sparse*: only a around 1-4% of the neurons are active together at a given time.

6.3 Deep belief networks

Deep belief network is a specific instance of deep learning approach. It can be seen as a multi-layer generative model where each layer consists of multiple nodes, similar to a neural network. First layer, often referred as *visible layer*, represents the raw input data, while every higher-level layer is referred as a *hidden layer*. The main difference between deep belief networks and neural networks is the *unsupervised* pre-training phase of a deep belief network.

To train the network in unsupervised way, the network is represented using the *Restricted Boltzmann machines*.

6.3.1 Restricted Boltzmann Machines

The Boltzmann machines are an instance of *energy-based* models, in which a scalar energy is associated to each configuration of variables. Learning in such settings corresponds to modifying the energy function so that it reflects certain properties. Energy-based probabilistic models define a probability distribution over a variable through an energy function

$$P(\mathbf{x}) = \frac{e^{-\text{Energy}(\mathbf{x})}}{Z}. \quad (6.1)$$

The normalization factor Z is called the *partitioning function* by analogy with physical systems

$$Z = \sum_{\mathbf{x}} e^{-\text{Energy}(\mathbf{x})} \quad (6.2)$$

with a sum running over the input space. The energy function is usually formulated as a sum of term f_i

$$\text{Energy}(\mathbf{x}) = \sum_i f_i(\mathbf{x}), \quad (6.3)$$

where each f_i evaluates some properties of an input vector \mathbf{x} .

In a case of hidden variables, probability is expressed as

$$P(\mathbf{x}, \mathbf{h}) = \frac{e^{-\text{Energy}(\mathbf{x}, \mathbf{h})}}{Z}, \quad (6.4)$$

where probability of the observed part can be calculated by marginalization

$$P(\mathbf{x}) = \sum_{\mathbf{h}} \frac{e^{-\text{Energy}(\mathbf{x}, \mathbf{h})}}{Z}. \quad (6.5)$$

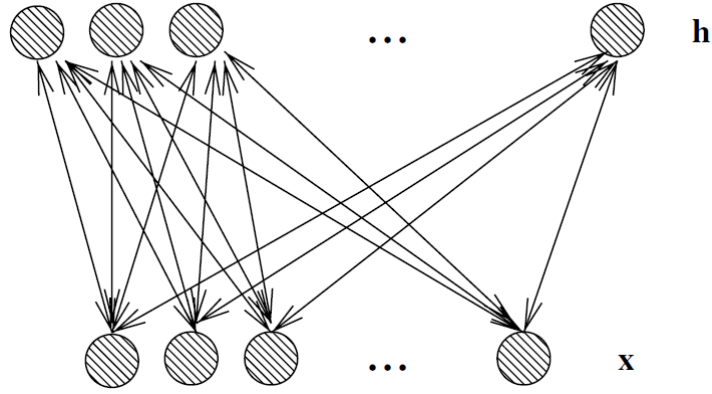


FIGURE 6.1: An illustration of the Restricted Boltzmann machines

Conditional probability distributions can be efficiently represented in the energy-based framework

$$P(y|\mathbf{x}) = \frac{e^{-\text{Energy}(\mathbf{x},y)}}{\sum_y e^{-\text{Energy}(\mathbf{x},y)}}. \quad (6.6)$$

The Boltzmann machine defines the energy functional as a second-order polynomial

$$\text{Energy}(\mathbf{x}, \mathbf{h}) = -\mathbf{b}^T \mathbf{x} - \mathbf{c}^T \mathbf{h} - \mathbf{h}^T \mathbf{W} \mathbf{x} - \mathbf{x}^T \mathbf{U} \mathbf{x} - \mathbf{h}^T \mathbf{V} \mathbf{h} \quad (6.7)$$

where \mathbf{x} represents a raw data, \mathbf{h} response of hidden units, \mathbf{b}_i and \mathbf{c}_i are the offsets associated with a single element from \mathbf{x} or \mathbf{h} and W_{ij} , U_{ij} and V_{ij} are weights associated with each pair of units.

The *restriction* in the Restricted Boltzmann machines comes from connections between layer. In the Restricted Boltzmann machine there are no connections within a layer, only connections between successive layer. The output of each layer serves as the input for the successive layer. This restriction simplifies the equation 6.7 and eliminates the factors U and V . The equation 6.7 is now

$$\text{Energy}(\mathbf{x}, \mathbf{h}) = -\mathbf{b}^T \mathbf{x} - \mathbf{c}^T \mathbf{h} - \mathbf{h}^T \mathbf{W} \mathbf{x} \quad (6.8)$$

and has a property that is shares parameterization with individual layers of a Deep belief networks, which makes them a perfect building blocks. The restriction is illustrated in figure 6.1. Another advantage of the restriction is that it makes calculations of $P(\mathbf{x}|\mathbf{h})$ and $P(\mathbf{h}|\mathbf{x})$ tractable because they factorize.

6.3.2 Training the Restricted Boltzmann machines

Training the Restricted Boltzmann machine is usually done by gradient descent. It turns out that standard gradient of the log-likelihood, although in very simple form, is very hard to estimate. The derivative of the log-likelihood parameterized with the weights \mathbf{w} equals

$$\frac{\partial p(\mathbf{x})}{\partial \mathbf{w}} = \sum_n \log \left[\frac{1}{Z} e^{-\text{Energy}(\mathbf{x})} \right] \quad (6.9)$$

$$= N \left[\left\langle \frac{\partial \text{Energy}(\mathbf{x})}{\partial \mathbf{w}} \right\rangle_{P(\mathbf{x})} - \left\langle \frac{\partial \text{Energy}(\mathbf{x})}{\partial \mathbf{w}} \right\rangle_n \right] \quad (6.10)$$

$$= \langle \mathbf{x}\mathbf{h} \rangle_{data} - \langle \mathbf{x}\mathbf{h} \rangle_{model} \quad (6.11)$$

where $\langle \rangle$ marks the expectation under the distribution specified with the subscript. The obtained results lead to a very simple rule for updating weights

$$\Delta w_{ij} \propto \epsilon (\langle x_i h_j \rangle_{data} - \langle x_i h_j \rangle_{model}). \quad (6.12)$$

Estimating $\langle x_i h_j \rangle_{data}$ is relatively simple as there are no connections between hidden units in a RBM. Given a random input data \mathbf{x} the binary state h_j of each hidden unit is set to 1 with probability

$$p(h_j|\mathbf{x}) = \sigma \left(b_j + \sum_i x_i w_{ij} \right) \quad (6.13)$$

where σ stands for sigmoid function. $x_i h_j$ is then unbiased sample. Because there is no connections between visible units in a RBM, it is also easy to get an unbiased sample of the state of a visible unit, given a hidden vector

$$p(x_i = 1|\mathbf{h}) = \sigma \left(a_i + \sum_j h_j w_{ij} \right). \quad (6.14)$$

Getting an unbiased sample of $\langle x_i h_j \rangle_{model}$ is much more difficult. An efficient method for estimating that distribution is given in [4]. This method starts by assigning the training data to visible layer units. Then, the states of the hidden units are evaluated using equation 6.13. Once states for the hidden units have been chosen, the input data is *reconstructed* by setting each x_i to 1 with probability given by equation 6.14. The change in weights is then given by

$$\Delta w_{ij} = \epsilon (\langle x_i h_j \rangle_{data} - \langle x_i h_j \rangle_{model}). \quad (6.15)$$

The methods is summarized in Algorithm 4. This assignment could be repeated several times to obtain a better approximation, but just one assignment works surprisingly well. The network is trained the same way layer by layer.

Although this approach works fairly well in practice, it is actually a quite crude approximation of the log-probability. It is more closely an approximation of the *Contrastive Divergence* - the difference between two Kullback-Lieber divergences.

Real valued units

So far, it has been assumed that all units are binary. That has been done for the sake of simplicity. Binary case is easily generalized to a real valued units. The main change is the adjustment of the energy term. For the real values visible units, a Gaussian visible units are popular choice

$$\text{Energy}_{G1}(\mathbf{x}, \mathbf{h}) = \sum_{i \in \text{visible}} \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j \in \text{hidden}} b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} h_j w_j \quad (6.16)$$

where σ_i is the standard deviation of the Gaussian noise for visible unit i . For real valued units in hidden layers, the energy functional looks like

$$\text{Energy}_{G2}(\mathbf{x}, \mathbf{h}) = \sum_{i \in \text{visible}} \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j \in \text{hidden}} \frac{(h_j - b_j)^2}{2\sigma_j^2} - \sum_{i,j} \frac{v_i}{\sigma_i} \frac{h_j}{\sigma_j} w_j. \quad (6.17)$$

6.3.3 Supervised fine tuning

6.4 Evaluation

For each of above mentioned features, except shape and information about the mitotic cells, a Deep belief networks is trained. For shape classification, the segmentation masks are used and classified with SVM as described in Chapter 5. The results are summarized in tables 6.2.

Obtained results are quite optimistic. Recognition rate of each feature is more than 90 % accurate. This demonstrates a great potential of deep learning methods for generating symbolic representations, for a specific problems like this one. A bit problematic cases seems to be blob texture recognition, detection of a large number of organelles and detection of dark bodies. By analyzing the misclassified cells, it has been observed that a great majority of misclassified cells come from a low intensity images which was expected. The most of those images represent the edge cases where it was difficult to assign the true class.

Algorithm 4 Training RBM

```

1: function TRAINRBM( $\mathbf{x}_1, \epsilon, W, \mathbf{b}, \mathbf{c}$ )
2:   repeat
3:     for all hidden units  $i$  do
4:       compute  $p(h_{1i} = 1 | \mathbf{x}_1) = \sigma(c_i + \sum_j W_{ij}x_{1j})$ 
5:       sample  $h_{1i} \in \{0, 1\}$  for  $p(h_i | \mathbf{x}_1)$ 
6:     end for
7:     for all visible units  $j$  do
8:       compute  $p(x_{2j} | \mathbf{h}_1) = \sigma(b_j + \sum_i W_{ij}h_{1i})$ 
9:       sample  $x_{2j} \in \{0, 1\}$  from  $p(x_{2j} | \mathbf{h}_1)$ 
10:    end for
11:    for all hidden units  $i$  do
12:      compute  $p(h_{2i} = 1 | \mathbf{x}_2) = \sigma(c_i + \sum_j W_{ij}x_{2j})$ 
13:    end for
14:     $W \leftarrow W + \epsilon (\mathbf{h}_1 \mathbf{x}_1^T - p(\mathbf{h}_2 = 1 | \mathbf{x}_2) \mathbf{x}_2^T)$ 
15:     $\mathbf{b} \leftarrow \mathbf{b} + \epsilon (\mathbf{x}_1 - \mathbf{x}_2)$ 
16:     $\mathbf{c} \leftarrow \mathbf{c} + \epsilon (\mathbf{h}_1 - p(\mathbf{h}_2 = 1 | \mathbf{x}_2))$ 
17:  until converged
18: end function

```

FIGURE 6.2: Symbolic feature mapping

		True					True		
		bright	neutral	dark			rough	smooth	blob
Pred	bright	322	2	12	Pred	rough	176	13	4
	neutral	15	243	23		smooth	42	547	12
	dark	7	5	191		blob	0	0	26
a) Organelle Type					b) Texture				
		True					True		
		lots	few	none			spec	hom	
Pred	lots	178	6	9	Pred	spec	539	33	
	few	19	226	13		hom	46	202	
	none	20	22	327					
c) number of objects					d) speckles				

Rule mining

IF-THEN rules are the most natural way of expressing knowledge. They are easy to understand and quite expressive. Because of their nature to represent knowledge, mining knowledge in such form is of interest to this project.

The symbolic representation of cells obtained in the previous step is here used as an input features. Two popular approached for mining such rules in supervised way are *Decision trees* and *Inductive Logic Programming*.

7.1 Decision Trees

Decision trees are one popular instance of machine learning algorithms due to their *simplicity* and *interpretability*. It approximates discrete functions and it is very robust to a noisy data which makes it suitable for this application. It used the standard *tree* structure to represent and compress the data where each *node* represent a test on attribute and each *branch* stands for values of the tested attribute.

Decision tree performs *greedy search* of a search space and choose an attribute to split data on. The chosen attribute then becomes the node in the tree and values it can take become branches. As a measure of *quality* of an attribute, *information gain* is used.

Information gain compares the *entropy* of the dataset before and after splitting. For a K class case, entropy of data set \mathcal{D} is defined as

$$\text{Entropy}(\mathcal{D}) = \sum_{k=1}^K p(C_k) \log_2 p(C_k) \quad (7.1)$$

where $p(C_k)$ stands for a fraction of examples in data set \mathcal{D} that belong to a class C_k . Informally, it measures the *impurity* of data. Having entropy defined, information gain achieved by splitting on attribute A is defined as

$$\text{InformationGain}(\mathcal{D}, A) = \text{Entropy}(\mathcal{D}) - \sum_{v \in \text{values}(A)} \frac{|\mathcal{D}_v|}{|\mathcal{D}|} \text{Entropy}(\mathcal{D}_v) \quad (7.2)$$

where $\mathcal{D}_v = \{x \in \mathcal{D} | A(x) = v\}$ represents a subset of the original data set \mathcal{D} with attribute A equal to v . Informally, as entropy measures the *impurity* of data, information gain measures how much of *impurity* has been lost by splitting the data. The partitioning is until there are not attributes to split on or data contains only examples for the same class. Decision trees are summarized in Algorithm 5.

Algorithm 5 Decision Tree learning

```

1: function DECISIONTREE( $\mathcal{D}, \mathbf{X}$ )
2:    $T \leftarrow$  new tree
3:   if all instances in  $\mathcal{D}$  have same class  $c$  then
4:     Label( $T$ ) =  $c$ 
5:     return  $T$ 
6:   end if
7:   if  $\mathbf{X} = \emptyset$  then
8:     Label( $T$ ) = most common class in  $\mathcal{D}$ 
9:     return  $T$ 
10:  end if
11:   $X \leftarrow$  attribute with highest information gain
12:  Label( $T$ ) =  $X$ 
13:  for each  $x$  in  $\mathbf{X}$  do
14:     $\mathcal{D}_x \leftarrow$  examples from  $\mathcal{D}$  with  $X = x$ 
15:    if  $\mathcal{D}_x$  is empty then
16:      let  $T_x$  be a new tree
17:      Label( $T_x$ )  $\leftarrow$  most common class in  $\mathcal{D}_x$ 
18:    else
19:       $T_x \leftarrow$  DecisionTree( $\mathcal{D}_x, \mathbf{X} - \{X\}$ )
20:    end if
21:    add a branch from  $T$  to  $T_x$ 
22:  end for
23:  return  $T$ 
24: end function

```

7.2 Inductive logic programming

Inductive logic programming (ILP) is an intersection between Machine learning and Logic programming where logic representation is used to induce the knowledge.

Inductive logic programming is concerned with finding a hypothesis \mathcal{H} from a set of positive and negative examples \mathcal{P} and \mathcal{N} . It is required that the hypothesis \mathcal{H} covers all positive examples in \mathcal{P} and none of the negative examples in \mathcal{N} .

ILP programs incorporate *background knowledge* to induce rules from positive examples. It is convenient to view the background knowledge \mathcal{B} as a logic program that is provided to the ILP system and fixed during the learning process. Under the presence of background knowledge, the hypothesis \mathcal{H} together with the background theory \mathcal{B} should cover all positive and none of the negative examples.

Many ILP systems have been developed over the years and three representative systems are described here.

7.2.1 FOIL

FOIL algorithm is an instance of *Sequential covering* algorithms. Sequential covering employs the *divide-and-conquer* principle and tries to learn one rule at time. Every induce rule should cover as many positive examples as possible, and as less negative examples as possible.

FOIL is summarized in Algorithm 6. FOIL performs general-to-specific search where each candidate clause to a current rule is in one of the following forms:

- $\mathcal{Q}(v_1, \dots, v_n)$ where \mathcal{Q} is a predicate from the set of all predicates and v_i are either variables already present in the rule or new variables
- $\text{Equal}(x_i, x_j)$ where x_i and x_j are variables already present in the rule
- the negation of either of above mentioned forms.

Another important step in FOIL is the evaluation of candidate clauses. Let R' be the rule created by adding candidate clause L to rule R . Performance measure $\text{FoilGain}(L, R)$ is defined as

$$\text{FoilGain}(L, R) = t \left(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right), \quad (7.3)$$

where p_0 stands for the number of positive bindings of rule R , n_0 is the number of negative bindings of R , while p_1 and n_1 represent the number of positive and negative bindings of R' .

7.2.2 RIPPER

The RIPPER algorithm is another instance of the sequential covering algorithms. Compared to FOIL, it introduces two additional steps - rule pruning and rule optimization. The algorithm is summarized in Algorithm 7.

Algorithm 6 FirstOrderInductiveLearner

```
1: function FOIL(examples)
2:    $Pos \leftarrow$  a set of positive examples
3:    $Neg \leftarrow$  a set of negative examples
4:    $LearnedRules \leftarrow \{\}$ 
5:   while  $Pos$  do
6:      $NewRule \leftarrow \text{LearnRule}()$ 
7:      $LearnedRules \leftarrow LearnedRules + NewRule$ 
8:      $Pos \leftarrow Pos - \{\text{covered positive examples}\}$ 
9:   end while
10:  return  $LearnedRules$ 
11: end function

12: function LEARNRULE
13:    $NewRule \leftarrow$  the empty rule that predict the target attribute
14:    $NewRuleNeg \leftarrow Neg$ 
15:   while  $NewRuleNeg$  do
16:      $CandidateLiterals \leftarrow$  generate candidate literals for  $NewRule$ 
17:      $BestLiteral \leftarrow \arg \max_{L \in CandidateLiterals} FoilGain(L, NewRule)$ 
18:     add  $BestLiteral$  to precondition of  $NewRule$ 
19:      $NewRuleNeg \leftarrow$  subset of  $NewRuleNeg$  covered by  $NewRule$ 
20:   end while
21: end function
```

For multi-class problems, it orders the classes according to increasing class prevalence - fraction of instances that belong to a particular class. It learns the rule set for the smallest class first and treats other classes as negative. It then repeats the process taking next smallest class as positive class. RIPPER also uses *FoilGain* measure as the performance measure.

The pruning step is done by using *Incremental Reduced Error Pruning*. Data set is split into training and validation set. The rule is then grown at the training set. The validation set is used to prune the rules. For each rule, performance measure v is calculated with a precondition removed. The performance measure v is defined as

$$v(pruneRule, Pos, Neg) = \frac{p - n}{p + n}, \quad (7.4)$$

where p is the number of positive examples in the validation set covered by pruned rule and n is the number of negative examples in the validation set covered by pruned rule.

In the optimization step, for each rule two new rules are generated. One is completely build from scratch while second one is generated by adding new conjunctions to the existing rule. As RIPPER performs the greed search, the rule built

from scratch might not be the same as the starting one. In that way, three separate rule sets are generated and compared. Resulting rule set is chosen to minimize the *Minimum Description Length* (MDL). MDL uses the assumption that any *regularity* in data can be used to *compress* the data. Given the hypothesis $\mathcal{H}^{(0)}, \mathcal{H}^{(1)}, \dots, \mathcal{H}^{(n)}$, which in this case are the rule sets, the best hypothesis is the one which minimizes the distance

$$L(\mathcal{H}) + L(\mathcal{D}|\mathcal{H}), \quad (7.5)$$

where $L(\mathcal{H})$ is the length, in bits, of the description of the hypothesis and $L(\mathcal{D}|\mathcal{H})$ is the length, in bits, of the description of data when encoded with the help of the hypothesis.

7.2.3 Aleph

Aleph (**A** Learning Engine for Proposing Hypotheses) is another commonly used rule induction system. The outline of the approach is summarized in Algorithm 8.

It starts with selection of a example to generalize. When the examples is selected, the most specific clause is built. Constructed clause have to entail the selected example and it is usually a *bottom clause* - a definite clause with many literals.

Aleph then performs search to generalize the bottom clause. Generalization is done by searching for a subset of the literals in the bottom clause that score the best. The search is performed by a *branch-and-bound* search. At this moment Aleph tries to find the best generalization of the current bottom clause, although it does not produce all generalizations. When the best clause is found, it is added to current theory and all example covered by the clause are removed.

The generalization of clause is summarized in function GENERALIZE in Algorithm 8. It starts by selecting a clause from *active set*. Clauses are selected in a way that clauses with fewer literals are chosen first. The selected clause is then *branched* by adding one literal at time. For each child, cost and lower bound are calculated. Lower bounds represent the lower cost that can be obtained at the node (that represent clause) and the sub-tree below it.

7.3 Results and induced rules

Above described methods are used to extract rules that describe the patterns. Classifiers are evaluated with a 10-fold cross validation. Tables 7.1 and 7.2 summarize the precision and recall of each classifier. Results are represented class-wise together with the number of rules found.

When analyzing the performance of classifier, both accuracy and the number of induced rules should be considered. The results suggest that Decision trees

Algorithm 7 RIPPER

```
1: function RIPPER(examples, predicates)
2:   LearnedRules  $\leftarrow \{\}$ 
3:   order classes in ascending order according to their prevalence
4:   CurrentClass  $\leftarrow$  smallest class
5:   Pos, Neg  $\leftarrow$  positive and negative examples of CurrentClass
6:   while Pos do
7:     split data set into training and validation set
8:     NewRule  $\leftarrow$  LearnRule(training set)
9:     NewRule  $\leftarrow$  PruneRule(validation set, NewRule)
10:    LearnedRules  $\leftarrow$  LearnedRules + NewRule
11:    remove examples covered by NewRule
12:    CurrentClass  $\leftarrow$  smallest class after removing examples
13:    Pos, Neg  $\leftarrow$  positive and negative examples of CurrentClass
14:  end while
15:  Optimize(LearnedRules)
16:  return LearnedRules
17: end function

18: function LEARNRULE(Pos, Neg)
19:   Rule  $\leftarrow \{\}$ 
20:   while there are Neg covered do
21:     add conjunct if it improves FoilGain
22:   end while
23:   return Rule
24: end function

25: function PRUNE(Pos, NewRule)
26:   repeat
27:     Predicate  $\leftarrow$  select a precondition from Rule
28:      $v(\text{pruneRule}, \text{Pos}, \text{Neg}) = \frac{p-n}{p+n}$ 
29:     delete Predicate if it improves v
30:   until no deletion improves v
31: end function

32: function OPTIMIZE(Rules)
33:   for each rule r in Rules do
34:     replacement rule r*  $\leftarrow$  build new rule
35:     revised rule r'  $\leftarrow$  add conjuncts to extend r
36:     compare the rule set for r against the rules set for r* and r'
37:     choose the rule set that minimizes the MDL
38:   end for
39: end function
```

Algorithm 8 Aleph

```
1: function ALEPH(examples)
2:    $Pos \leftarrow$  positive examples
3:    $Neg \leftarrow$  negative examples
4:    $theory \leftarrow \{\}$ 
5:   while  $Pos$  do
6:      $example \leftarrow$  select example
7:      $clause \leftarrow$  BuildMostSpecificClause( $example$ )
8:      $rule \leftarrow$  Generalize( $clause$ )
9:      $theory \leftarrow theory + rule$ 
10:    Remove redundant
11:  end while
12: end function

13: function GENERALIZE(examples,clause)
14:    $activeSet \leftarrow \{0\}$ 
15:    $Cost \leftarrow \infty$ 
16:    $CurrentBest \leftarrow$  select random
17:   while  $activeSet$  contains clauses do
18:     remove first node  $k$  from  $activeSet$ 
19:     generate children of  $k$ , compute costs  $Cost_i$  and lower bounds on costs  $L_i$ 
20:     for each child do
21:       if  $L_i > C$  then
22:         prune child
23:       else
24:         if child is complete solution AND  $Cost_i < C$  then
25:            $Cost \leftarrow Cost_i$ 
26:            $CurrentBest \leftarrow$  child
27:           prune clauses in  $activeSet$  with  $L_i$  more than  $Cost_i$ 
28:         end if
29:         add child to  $activeSet$ 
30:       end if
31:     end for
32:   end while
33: end function
```

TABLE 7.1: Precision of pattern classification

method	homogeneous	nuleolar	centromere	cytoplasmatic	fine speckled	coarse speckled	rules
RIPPER	93.48 %	94.65 %	98.24 %	96.33 %	83.33 %	92.78 %	10
DTREE	93.48 %	97.10 %	100 %	96.46 %	87.00 %	96.86 %	25

TABLE 7.2: Recall of pattern classification

method	homogeneous	nuleolar	centromere	cytoplasmatic	fine speckled	coarse speckled	rules
RIPPER	93.48 %	94.65 %	98.24 %	96.33 %	83.33 %	92.78 %	10
DTREE	93.48 %	97.10 %	100 %	96.46 %	87.00 %	96.86 %	25

and RIPPER perform the best. Overall accuracy of decision tree and RIPPER are 95.26 % and 94.50 % respectively. Decision tree achieves the highest accuracy, but produces significantly many more rules - 25 compared to 10 induced by RIPPER. Such a difference suggests overfitting. Induced rules are shown in Appendix A. It can be observed that RIPPER not only produces less rules, but those rules are also much simpler than ones induced by decision tree. Rules induced by RIPPER usually have one or two clauses in the body, while rules induced by decision tree usually have three or four clauses in the body. Also, it can be seen that 9 out of 25 rules induced covers less than 10 examples in the dataset. The state-of-the-art performance reported so far is 94.9 % [?]. The obtained results are quite comparable to that paper, but also provide explainable results.

Taking both performance and simplicity into account, the rules induced by RIPPER seems to perform the best. Their simplicity makes them more applicable to the real life scenarios. They also correspond more to the intuition about the patterns. The induced rules mostly extract the most distinguishing property of a certain pattern. For example, *centromere* class is identified by a large number of objects in the cell body, *cytoplasmatic* by the irregular shape and *coarse speckled* by rough texture. All of these properties are characteristic for specific patterns and don't appear for other types. The drawback of this approach is that the rules should be always applied *sequentially*.

As suggested in literature, the most difficult classes to classify are *fine speckled* and *homogeneous*. It is very hard to distinguish between those two classes because they look very similar, especially in low intensity images. The literature indicated that mitotic cells are highly informative for for this task. It appears that RIPPER and Decision trees are able to capture that relations, while ALEPH and FOIL are not.



Conclusion

This thesis proposes a solution for a computer aided assistance in one commonly used diagnostics in autoimmune diseases. The solution closely follows the procedure medical experts suggests. It covers the segmentation task, the fluorescence intensity level classification and staining pattern classification. The strongest contribution of this thesis is the interpretable approach to prediction models that help medical experts.

The work on segmentation part was motivated by two problems found in previous approaches. Due to a coloring procedure of fluorescence imaging, cells exhibit different properties which makes them hard to detect, especially in noisy images. By finding background instead of cells directly that problem was successfully solved with great improvement. Still, a lot of cells overlaps. By using the information about background, morphological snakes were introduced. Although not always capable of separating overlapping cells, the method demonstrated very promising result. The problematic case were the patterns with bright organelles where those organelles were segmented.

The cells were then characterized with *visual concepts* describing their appearance. Deep belief networks were used to map image, or pixel features to a *symbolic* ones. Deep approaches to machine learning demonstrated a great potential in symbolic feature recognition. Surprisingly, convolutional components seem not to improve the recognition, but a full potential of this approach is yet to be investigated.

The symbolic representation learned in the previous step are then used to mine rules that describe the patterns. Four commonly used rule mining algorithms were compared and proven to perform accurately as state-of-the-art methods, but offer explanations about their decisions. I believe such approach is needed when computers assist in making diagnosis, based on images or other diagnostic test.

Future work

In order to make a fully functional system that supports this process, there are still problems that need to be solved. The segmentation task still suffers from certain problem, such as *overexposed* organelles in a cell's body which end oversegmented. Introducing a model based segmentation method, where movement of the contour is restrained by model learned from data, might be a way to improve the segmentation of cells.

There are two actions that have not be mentioned in this thesis, but are of high interest to his task. Mitotic cells were taken as a ground truth so far, but detecting and classifying them is very important task for the diagnostic procedure. Together with mitotic cells, images taken by fluorescence imaging often have artifacts that don't care any information. Both mitotic cells and artifacts are very rare objects in images which is the main difficulty of this task. Approaching this problem from an unbalanced dataset or outlier detection perspective could be a good start.

Symbolic feature learning achieved by deep belief network has proven a great potential. As deep learning demonstrates an enormous growth in research, experimenting with different deep approaches to improve symbolic representation learning leaves a lot of space for future work.

Appendices

Induces rules

In this appendix, the distributions of symbolic features are analyzed. The histogram of values for each symbolic features is plotted with regard to class.

FIGURE A.1: Rules induced by RIPPER

- 1: IF *number_of_obj* = *lots*
THEN *centromere*
- 2: IF *mitotic_cells* = *bright_middle*
THEN *homogeneous*
- 3: IF *organelle_type* = *bright_on_dark*
THEN *nucleolar*
- 4: IF *texture* = *rough*
THEN *coarse speckled*
- 5: IF *shape* = *circular* AND *number_of_obj* = *few*
THEN *fine speckled*
- 6: IF *shape* = *irregular*
THEN *cytoplasmatic*
- 7: IF *mitotic_cells* = *dark spot* AND *organelle_type* = *neutral*
THEN *fine speckled*
- 8: IF *intensity* = *intermediate* AND *mitotic_cells* = *neutral*
THEN *nucleolar*
- 9: IF *mitotic_cells* = *neutral* AND *speckled* = *homogeneous*
THEN *fine speckled*
- 10: **else** *cytoplasmatic*

FIGURE A.2: Rules induced by Decision tree

- 1: IF mitotic_cells = *bright middle* AND number_of_obj = *few*
THEN *homogeneous*
- 2: IF mitotic_cells = *bright middle* AND number_of_obj = *lots*
THEN *centromere*
- 3: IF mitotic_cells = *bright middle* AND number_of_obj = *none*
THEN *homogeneous*
- 4: IF mitotic_cells = *bright middle sparkle*
THEN *centromere*
- 5: IF mitotic_cells = *dark spot* AND texture = *blob*
THEN *cytoplasmatic*
- 6: IF mitotic_cells = *dark spot* AND texture = *rough* AND organelle_type = *bright_on_dark*
THEN *coarse speckled*
- 7: IF mitotic_cells = *dark spot* AND texture = *rough* AND organelle_type = *dark_on_bright* AND speckles = *homogeneous*
THEN *fine speckled*
- 8: IF mitotic_cells = *dark spot* AND texture = *rough* AND organelle_type = *dark_on_bright* AND speckles = *speckled*
THEN *coarse speckled*
- 9: IF mitotic_cells = *dark spot* AND texture = *rough* AND organelle_type = *neutral*
THEN *coarse speckled*
- 10: IF mitotic_cells = *dark spot* AND texture = *smooth* AND organelle_type = *bright_on_dark* AND intensity = *intermediate*
THEN *fine speckled*
- 11: IF mitotic_cells = *dark spot* AND texture = *smooth* AND organelle_type = *bright_on_dark* AND intensity = *positive*
THEN *nucleolar*
- 12: IF mitotic_cells = *dark spot* AND texture = *smooth* AND organelle_type = *dark_on_bright* AND number_of_obj = *few*
THEN *fine speckled*
- 13: IF mitotic_cells = *dark spot* AND texture = *smooth* AND organelle_type = *dark_on_bright* AND number_of_obj = *lots*
THEN *fine speckled*
- 14: IF mitotic_cells = *dark spot* AND texture = *smooth* AND organelle_type = *dark_on_bright* AND number_of_obj = *none*
THEN *cytoplasmatic*
- 15: IF mitotic_cells = *dark spot* AND texture = *smooth* AND organelle_type = *neutral*
THEN *fine speckled*
- 16: IF mitotic_cells = *neutral* AND organelle_type = *bright_on_dark*
THEN *nucleolar*
- 17: IF mitotic_cells = *neutral* AND organelle_type = *dark_on_bright* AND texture = *rough* AND intensity = *intermediate*
THEN *coarse speckled*
- 18: IF mitotic_cells = *neutral* AND organelle_type = *dark_on_bright* AND texture = *rough* AND intensity = *positive*
THEN *fine speckled*
- 19: IF mitotic_cells = *neutral* AND organelle_type = *dark_on_bright* AND texture = *smooth*
THEN *fine speckled*
- 20: IF mitotic_cells = *neutral* AND organelle_type = *neutral* AND number_of_obj = *few*
THEN *nucleolar*
- 21: IF mitotic_cells = *neutral* AND organelle_type = *neutral* AND number_of_obj = *lots*
THEN *coarse speckled*
- 22: IF mitotic_cells = *neutral* AND organelle_type = *neutral* AND number_of_obj = *none* AND intensity = *intermediate*
THEN *nucleolar*
- 23: IF mitotic_cells = *neutral* AND organelle_type = *neutral* AND number_of_obj = *none* AND intensity = *positive*
THEN *fine speckled*
- 24: IF mitotic_cells = *unknown* AND number_of_obj = *few*
THEN *nucleolar*
- 25: IF mitotic_cells = *unknown* AND number_of_obj = *none*
THEN *cytoplasmatic*

FIGURE A.3: Rules induced by FOIL

- 1: IF *organelle_type = bright on dark* AND *number_of_obj = lots*
THEN *centromere*
- 2: IF *number_of_obj = lots* AND *mitotic_cells = bright middle*
THEN *centromere*
- 3: IF *intensity = intermediate* AND *speckles = speckled* AND *texture = rough*
THEN *coarse speckled*
- 4: IF *texture = rough* AND *organelle_type = neutral* AND *mitotc_cells = dark spot*
THEN *coarse speckled*
- 5: IF *intensity = positive* AND *texture = rough* AND *organelle_type = dark on bright* AND *number_of_obj = few* AND *mitotic_cells = dark spot*
THEN *coarse speckled*
- 6: IF *shape = irregular*
THEN *cytoplasmatic*
- 7: IF *texture = blob*
THEN *cytoplasmatic*

Bibliography

- [1] D. Adams. *The Hitchhiker's Guide to the Galaxy*. Del Rey (reprint), 1995. ISBN-13: 978-0345391803.
- [2] P. Agrawal, M. Vatsa, and R. Singh. Hep-2 cell image classification: A comparative analysis. In G. Wu, D. Zhang, D. Shen, P. Yan, K. Suzuki, and F. Wang, editors, *Machine Learning in Medical Imaging*, volume 8184 of *Lecture Notes in Computer Science*, pages 195–202. Springer International Publishing, 2013.
- [3] P. Foggia, G. Percannella, P. Soda, and M. Vento. Benchmarking hep-2 cells classification methods. *IEEE Trans. Med. Imaging*, 32:1878–1889, 2013.
- [4] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, Aug. 2002.
- [5] Y.-L. Huang, C.-W. Chung, T.-Y. Hsieh, and Y.-L. Jao. Outline detection for the hep-2 cell in indirect immunofluorescence images using watershed segmentation. In *Sensor Networks, Ubiquitous and Trustworthy Computing*, 2008. *SUTC '08. IEEE International Conference on*, pages 423–427, 2008.
- [6] Y.-L. Huang, Y.-L. Jao, T.-Y. Hsieh, and C.-W. Chung. Adaptive automatic segmentation of hep-2 cells in indirect immunofluorescence images. In *Proceedings of the 2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (Sutic 2008)*, *SUTC '08*, pages 418–422. IEEE Computer Society, 2008.
- [7] D. C. Ian D Odell. *Immunofluorescence techniques*, 2013.
- [8] K. Li, J. Yin, Z. Lu, X. Kong, R. Zhang, and W. Liu. Multiclass boosting svm using different texture features in hep-2 cell staining pattern classification. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 170–173, 2012.
- [9] P. Marquez-Neila, L. Baumela, and L. Alvarez. A morphological approach to curvature-based evolution of curves and surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(1):2–17, 2014.
- [10] R. Nakamura. *Quality Assurance for the Indirect Immunofluorescence Test for Autoantibodies to Nuclear Antigen (IF-ANA): Approved Guideline (1996)*. NCCLS document. NCCLC, 1996.

- [11] R. Nosaka, Y. Ohkawa, and K. Fukui. Feature extraction based on co-occurrence of adjacent local binary patterns. In *Proceedings of the 5th Pacific Rim Conference on Advances in Image and Video Technology - Volume Part II*, PSIVT'11, pages 82–91. Springer-Verlag, 2012.
- [12] R. Nosaka, C. H. Suryanto, and K. Fukui. Rotation invariant co-occurrence among adjacent lbps. In *Proceedings of the 11th International Conference on Computer Vision - Volume Part I*, ACCV'12, pages 15–25. Springer-Verlag, 2013.
- [13] P. Perner, H. Perner, and B. Mueller. Mining knowledge for hep-2 cell image classification. *Artificial Intelligence in Medicine*, 26:161–173, 2002.
- [14] A. Rigon, P. Soda, D. Zennaro, G. Iannello, and A. Afeltra. Indirect immunofluorescence in autoimmune diseases: Assessment of digital images for diagnostic purpose. *Cytometry Part B-clinical Cytometry*, 72B:472–477, 2007.
- [15] P. Soda and G. Iannello. A multi-expert system to classify fluorescent intensity in antinuclear autoantibodies testing. In *Computer-Based Medical Systems, 2006. CBMS 2006. 19th IEEE International Symposium on*, pages 219–224, 2006.
- [16] P. Soda, G. Iannello, and M. Vento. A multiple expert system for classifying fluorescent intensity in antinuclear autoantibodies analysis. *Pattern Anal. Appl.*, 12(3):215–226, Sept. 2009.
- [17] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *Int. J. Comput. Vision*, 62(1-2):61–81, Apr. 2005.