POLITECNICO DI MILANO
Corso di Laurea Specialistica in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

Mitosis detection in histological images. Algorithms based on machine learning and their performance compared to humans.

Relatore: Prof. Vincenzo Caglioti
Correlatore: Ing. Alessandro Giusti

Tesi di Laurea di:
Claudio G. Caccia, matricola 751302

Anno Accademico 2012-20013

a Elena, Giovanna e Leonardo

Abstract

Acknowledgements

....

Contents

List of Figures

List of Tables

Glossary

AI Artificial Intelligence. 15

BR Bloom and Richardson Grading System. 3, 6, 26

CAD Computer Aided Diagnosis. 8

CNN Convolutional Neural Network. 24, 25, 34, 41

CoC Convention over Configuration. 46

CV Computer Vision. 8, 9, 12, 17, 18, 38

DNN Deep Neural Network. 25

DT Decision Tree. 42–44, 56

FN false negative. 17, 52

FP false positive. 17, 52

GGMM Gamma Gaussian Mixture Model. 24

GLCM Gray-level Co-occurrence Matrix. 10

GLEM Gray-level Entropy Matrix. 10

GLRM Gray-level Run-length Matrix. 10

GT Ground Truth. 26, 41

HE Hematoxylin and Eosin. 5, 26

HPF High Power Fields. 6

HSV Hue Saturation Value. 37

LBP Local Binary Patterns. 10–12, 38, 39

ML Machine Learning. 15, 16, 20, 22, 24, 72

MRI Magnetic Resonance Imaging. 12

NGS Nottingham Grading System. 3

NN Neural Network. 17

PCA Principal Component Analysis. 72, 73

PR Pattern Recognition. 15, 20

RBF Radial Basis Function. 42, 72

RF Random Forest. 17, 42–44, 55–57, 60, 61, 63, 65–68, 72, 73, 98

RGB Red-Green-Blue. 36, 37

ROC Receiver Operating Characteristic. 29, 30, 57, 60–62, 64, 65

ROI Region of Interest. 8, 20, 33

RoR Ruby on Rails. 46

SVD Singular Value Decomposition. 73

SVM Support Vector Machine. 17, 24, 25, 41, 42, 55–57, 60, 63, 64, 66–68, 71–73, 98

TN true negative. 17, 52, 60, 62

TP true positive. 17, 52, 60, 62

VAR Rotation Invariant Variance Measure. 11, 12, 39

WT Wavelet Transform. 12

Chapter 1

Introduction

"Λέγειν τὰ προγενόμενα, γινώσκειν τὰ παρεόντα, προλέγειν τὰ ἐσόμενα: μελετᾶν ταῦτα. Ἄσκεῖν περὶ τὰ νοσήματα δύο, ὠφελεῖν ἢ μὴ βλάπτειν. Ἡ τέχνη διὰ τριῶν, τὸ νόσημα καὶ ὁ νοσέων καὶ ὁ ἰητρός: ὁ ἰητρός ὑπηρέτης τῆς τέχνης, ὑπεναντιοῦσθαι τῷ νοσήματι τὸν νοσέοντα μετὰ τοῦ ἰητροῦ.
(The physician must be able to tell the antecedents, know the present, and foretell the future: must mediate these things, and have two special objects in view with regard to disease, to do good or to do no harm. The art consists in three things: the disease, the patient, and the physician. The physician is the servant of the art, and the patient must combat the disease along with the physician.)"

῾Ιποκράτης(Hippocrates, Epid. 1.2.11)

First part topics

- Detection problems in Computer Vision and in particular in biomedical imaging
- Relation between detection and classification
- Mitosis Detection as a component in breast cancer assessment
- Machine Learning used to automate the mitotic count task
- The validation problem:
  - from clinical point of view
  - from ML point of view

Second part topics

- General overview of the work: automatic Mitosis Detection in breast cancer histological images and comparison of the performances between humans and algorithms.
  - some literature
  - specificity of this work
  - achievements
  - research directions

Third part topics

- Structure of the work
  - Section 1: state of the art...
  - Section 2: approach to the problem and model
  - Section 3: design of a mitosis detection algorithm
  - Section 4: design of a user study
  - Section 5: experimental results
  - Section 6: Conclusions
  - Appendixes: implementation details

Chapter 2

State of the art

"Rem tene, verba sequentur"
(Know the subject, the words will follow)

Marcius Porcius Cato Censorius

2.1 Background

Breast cancer classification divides breast cancer into categories according to different schemes[1], each serving a different purpose. The purpose of classification is to select the best treatment[27].
Within the last decade, histological grading has become widely accepted as a powerful indicator of prognosis in breast cancer. The grading depends on the microscopic similarity of breast cancer cells to normal breast tissue, and classifies the cancer as well differentiated (low grade), moderately differentiated (intermediate grade), and poorly differentiated (high grade), reflecting progressively less normal appearing cells that have a worsening prognosis. Although grading is fundamentally based on how biopsied, cultured cells behave, in practice the grading of a given cancer is derived by assessing the cellular appearance of the tumor.
The Nottingham Grading System (NGS) (also called Elston-Ellis) is a modification [24] of the Bloom and Richardson Grading System (BR)[9, 28]. NGS is judged more reproducible and is the recommended grading method [1].

NGS grades breast carcinomas by adding up scores for:

[1] `http://www.breastpathology.info/`

- tubule formation,
- nuclear pleomorphism,
- mitotic count

each of which is given 1 to 3 points. The scores for each of these three criteria and then added together to give an overall final score and corresponding grade as follows [19]:

3-5 **Grade 1 tumor** (well-differentiated). Best prognosis.

6-7 **Grade 2 tumor** (moderately-differentiated). Medium prognosis.

8-9 **Grade 3 tumor** (poorly-differentiated). Worst prognosis.

Lower grade tumors, with a more favorable prognosis, can be treated less aggressively, and have a better survival rate.

Mitotic activity (see 7 for some details) is one of the strongest prognosticators for invasive breast carcinoma. It is expressed as the number of mitotic figures per tissue area. Early detection plays an important role in reducing cancer mortality. The current procedure for breast cancer grading is manually performed by pathologists, for both nuclear pleomorphism [22] and mitotic count. Breast tissue samples of patients are taken and examined under microscopes. Pathologists grade the tissue samples based on the deviation of the cell structures from normal tissues. A pathologist may have to examine a great amount of slides [73]. This process can be time consuming and subjective (see 3.4.1).
In the following subsection we give a short overview of the mitosis count procedure[3].

2.1.1 Tissue preparation

After tumor excision is performed, the excised material is sent for analysis in a pathology lab. The tissue preparation process starts with making smaller cuts of the material that are then fixed in formalin and (after processing) embedded in paraffin.
Using a high precision cutting instrument (microtome), thin sections are cut from the paraffin block, which are then put on glass slides. The final stage of the tissue preparation process is the staining of the sections with stains that

highlight specific structures of the tissue so they are better visible under a microscope. The standard staining protocol uses the Hematoxylin and Eosin (HE) stains. The hematoxylin dyes the nuclei a dark purple color and the eosin dyes other structures (cytoplasm, stroma, etc.) a pink color.

2.1.2 Digital Pathology

Recent years have brought the trend of digitization of histological slides. Digital slide scanners 2.1, in combination with digital slide viewers, aim to provide the experience of viewing a digital slide on a computer monitor in a manner analogous to viewing it under a microscope, but with all the added benefits of the digital format (ease of annotation, image analysis, collaborative viewing etc.). The output of the digital slide scanners are multi-layered images, stored in a format that enables fast zooming and panning. Depending on the area of the tissue that is present on the slide and the magnification and resolution at which the slide is scanned, the lowest layer of the digital slide can be up to several tens of thousands of pixels in width or height. Currently, digital slides are mainly used for research, education and remote consultation purposes. Their use for routine diagnosis and prognosis is not yet common [42]. Availability of automatic image analysis algorithms that can aid pathologists in their work can be a major incentive for acceptance of digital slides in the routine pathology lab workflow.

Figure 2.1: Aperio ScanScope XT scanner

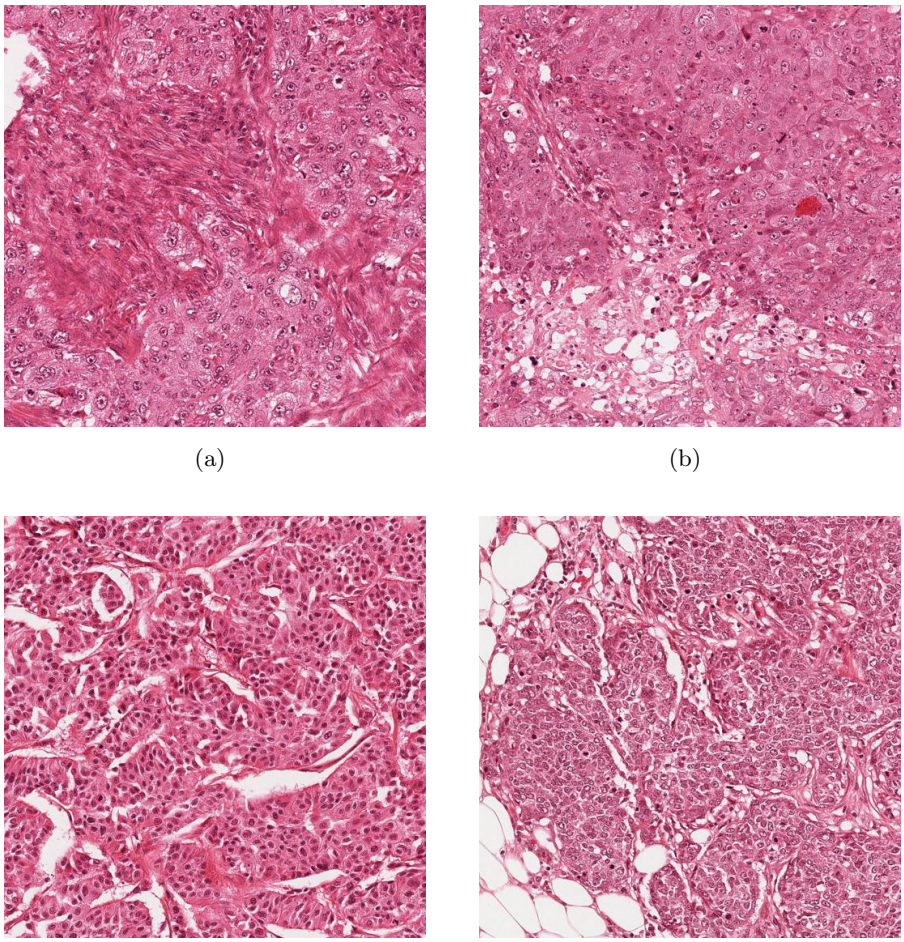
2.1.3 Mitosis Counting

Mitotic activity is one of the strongest prognosticators for invasive breast carcinoma and it is expressed as the number of mitotic figures per tissue

area. As part of the BR grading system, mitotic activity is routinely assessed in pathology labs across the world. In addition, the mitotic activity can be used as a prognosticator independently of the BR grading system. Typically, the pathologist receives a panel of slides for each case that is to be graded. He or she then proceeds to select one slide where the histological grading will be performed. The mitosis counting is performed in 8-10 consecutive microscope High Power Fields (HPF) [41]. A HPF has a size of $512 \times 512 \mu m^2$ (i.e. an area of 0.262 mm^2), which is the equivalent of a microscope field diameter of $0.58mm$. The standard guidelines are to select an area that encompasses the most invasive part of the tumor, at the periphery and with highest cellularity. Depending on the number of figures counted, a mitotic activity score is assigned. Cases with 7 or fewer mitotic figures present are assigned score 1 (best prognosis). Cases with more than 12 mitotic figures are assigned score 3 (worst prognosis). The intermediate cases are assigned score 2.

2.1.4 Challenges in Mitosis Detection

Because of the aberrant chromosomal makeup of many tumors (aneusomy, polysomy, translocations, amplifications, deletions), the appearance of mitotic figures in the images can significantly differ from the textbook examples of a splitting nucleus[49]. In addition, imperfections of the tissue preparation process result in tissue appearance variability, which can present a challenge also for an automated mitosis detection system.
Most commonly, mitotic figures are exhibited as hyperchromatic objects. In addition, they have absence of a clear nuclear membrane, "hairy"protrusions around the edges and basophilia instead of eosinophilia of the surrounding cytoplasm. However, these are more guidelines than hard rules, and the bulk of the training of pathologists is done by looking as specific examples of mitotic figures. One of the main challenges in spotting mitotic figures is that other objects such as apoptotic nuclei can have similar appearance, making it difficult even for trained experts to make a distinction [83]. Lymphocytes, compressed nuclei, "junk"particles and other artifact form the tissue preparation process, can also have hyperchromatic appearance. The images in Figures 2.2, 2.4 and 2.5 try to give an idea of the difficulty of the task.



(a) (b)

(c) (d)

Figure 2.2: Examples of digital histological images

2.2 Mitosis Detection and Computer Vision

The task of automatic mitosis detection involves topics in varius fields of research, in particular

- Image Analysis
- Machine Learning

We consider a framework in which, in the whole image, some candidates are detected and the classified as mitosis or non-mitosis.
In this chapter we give an overview of the main aspects concerning *image analysis* and in the following one (2.3) we analyze the *machine learning* elements.

2.2.1 Overview of Medical Imaging

Over the past decade, dramatic increases in computational power and improvement in image analysis algorithms have allowed the development of powerful computer-assisted analytical approaches to radiological and histopathological data[30]. Digitized tissue histopathology has now become amenable to the application of computerized image analysis and machine learning techniques. Analogous to the role of Computer Aided Diagnosis (CAD) algorithms in medical imaging to complement the opinion of a radiologist, CAD algorithms have begun to be developed for disease detection, diagnosis, and prognosis prediction to complement the opinion of the pathologist[80].

2.2.2 Software Tools

The imaging modalities rely heavily on computational approaches. In fact, in many cases the computational technology is just as important as the optics, not just for the digital capture that all systems now use but in many cases also for visualizing and properly interpreting the data. The article in [23] reviews each computational step that biologists encounter when dealing with digital images and the overall status of available software for bioimage informatics. It is worth highlighting the existence of open-source software tools like *Fiji* [77] and *ImageJ* [78], which supply some basic features for *object detection* and *feature extraction*[75].

2.2.3 Features

The concept of feature is used to denote a piece of information which is relevant for solving a computational task[65]. A feature is defined as an "interesting"part of an image, and features are used as a starting point for many Computer Vision (CV) algorithms. They can be the result of a general *neighborhood operation*[45] applied to the image, or specific structures in the image itself. Types of image features include:

- Edges
- Corners
- Blobs or Regions of Interest (ROIs)
- Ridges or elongated objects (i.e. blood vessels in medical images)

Other examples of features are related to motion in image sequences, to shapes defined in terms of curves or boundaries between different image regions, or to properties of such a region[35].

The feature concept is very general and the choice of features in a particular CV system may be highly dependent on the specific problem to be considered.

2.2.4 Feature Detectors

Many algorithms have been developed to detect specific features, and a complete overview of them is beyond the scope of this work. Some of the most famous ones, like *Canny edge detector*[14], *Harris edge and corner detector*, or SUSAN [82] are available in most widely used commercial and open-source Computer Vision software packages (i.e. MATLAB Image Processing Toolbox[2] or OpenCV[3]).

Features are sometimes extracted over several scalings. One of these methods is *Scale-invariant feature transform*; in this algorithm, various scales of the image are analyzed to extract features[55] (the underlying theory can be found in [52]).

2.2.5 Image Segmentation

Segmentation is the process of partitioning a digital image into multiple segments (sets of pixels) in order to simplify or change the representation of an image into something that is more meaningful and easier to analyze[50]. Image segmentation is typically used to locate objects and boundaries (i.e. features) in images. Such a process assigns a label to every pixel in an image so that pixels with the same label share certain visual characteristics[83].

2.2.6 Texture Algorithms

An image texture is a set of metrics designed to quantify the perceived texture of an image. Image texture gives information about the spatial arrangement of color or intensities in an image or in selected region of it[25]. Image textures are used in *segmentation*(see 2.2.5), or *classification* of images (see

[2] `http://www.mathworks.com/products/image/index.html`
[3] `http://opencv.org/`

2.3). To address the issue of texture analysis, the so called "statistical approach"is more widely used as it is easier to compute. This approach sees an image texture as a quantitative measure of the arrangement of intensities in a region.

Co-occurrence Matrix

Co-occurrence matrix captures numerical features of a texture using spatial relations of similar gray tones. Numerical features computed from the co-occurrence matrix can be used to represent, compare, and classify textures[38, 95]. The following are a subset of standard features derivable from a normalized co-occurrence matrix, as described in [34]:

$$\text{Contrast} = \sum_{n=0}^{N_g-1} n^2 \left\{ \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p[i,j] \right\}, \text{ where} |i-j| = n \quad (2.1)$$

$$\text{Correlation} = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} (i,j) \cdot p[i,j] - \mu_x \mu_y}{\sigma_x \sigma_y} \quad (2.2)$$

$$\text{Entropy} = -\sum_i \sum_j p[i,j] \cdot \log(p[i,j]) \quad (2.3)$$

Where:

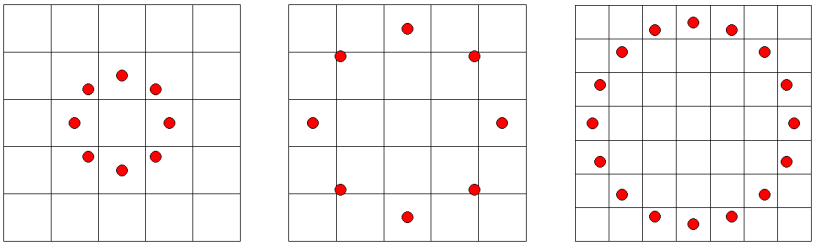
- N_g is the number of gray levels in the quantized image,
- $p[i,j]$ is the $(i,j)th$ entry in a normalized gray-tone spatial dependence matrix,
- $\mu_x, \sigma_x, \mu_y, \sigma_y$ are the mean and the standard deviation of respectively $p_x = \sum_{j=1}^{N_g} p(i,j)$ and $p_y = \sum_{i=1}^{N_g} p(i,j)$.

Various algorithms use texture feature like Gray-level Co-occurrence Matrix (GLCM) [69], Gray-level Run-length Matrix (GLRM) [58] or Gray-level Entropy Matrix (GLEM) for image classification, also in medical [12] and biological imaging [99].

Local Binary Patterns

Local Binary Patterns (LBP) is another type of feature used for classification in Computer Vision. LBP is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel with the value of the center pixel and considers the result as a binary

number. The distance and the number of neighbors can be selected, as shown in Figure 2.3[67]. The notation (P, R) is used for pixel neighborhoods which means P sampling points on a circle of radius of R.



(a) (8,1) (b) (8,2) (c) (16,3)

Figure 2.3: Examples LBP neighbors and distances

The computation of the LBP code of a pixel of coordinates (x_c, y_c) is given by:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c) \cdot 2^p \quad \text{where } s(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

This operator used jointly with a simple local contrast measure provided very good performance in unsupervised texture segmentation. Another extension to the original operator is the definition of so called uniform patterns, which can be used to reduce the length of the feature vector and implement a simple rotation-invariant descriptor. This extension was inspired by the fact that some binary patterns occur more commonly in texture images than others. A LBP is called uniform if the binary pattern contains at most two bitwise transitions from 0 to 1 or vice versa when the bit pattern is traversed circularly.
. In the computation of the LBP labels, uniform patterns are used so that there is a separate label for each uniform pattern and all the non-uniform patterns are labeled with a single label. For example, when using $(8, R)$ neighborhood, there are a total of 256 patterns, 58 of which are uniform, which yields in 59 different labels.

The uniform and rotation invariant LBP can be further enhanced by combining it with a Rotation Invariant Variance Measure (VAR) operator, with the same parameters (P, R), that characterizes the contrast of local image texture[66]. Both operators are also computationally attractive, as they can be realized with a few operations in a small neighborhood and a lookup table. The VAR operator is described by the following relations:

$$VAR_{(P,R)} = \frac{1}{P}\sum_{p=0}^{P-1}(g_p - \mu)^2 \quad \text{where } \mu = \sum_{p=0}^{P-1} g_p^2 \tag{2.5}$$

LBP(P, R) and VAR(P, R) are complementary and a feature set made by the combination of the two is expected to be a very powerful rotation invariant measure of local image texture. It is also possible to use joint feature sets composed by operators with different neighborhood.

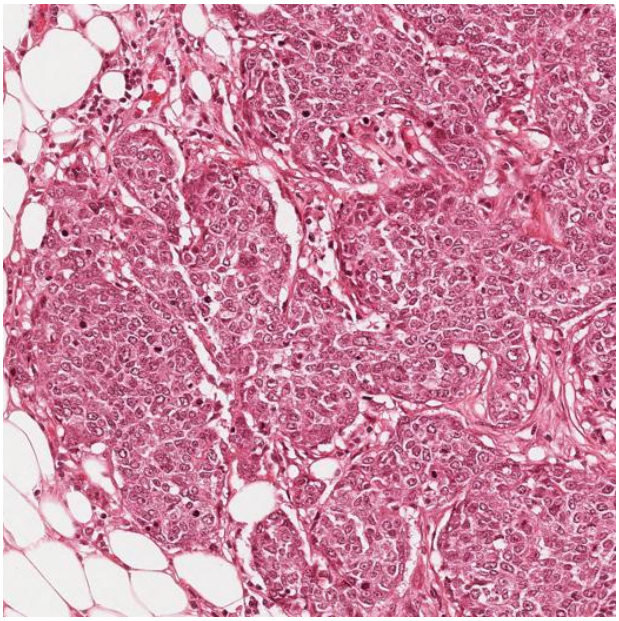
Wavelets

The Wavelet Transform (WT) is having greater importance medicine and biology. The main uses of the WT concern the analysis of one-dimensional physiological signals obtained by electrocardiography (ECG) and electroencephalography (EEG), including evoked response potentials[91]. A survey of recent wavelet developments in medical imaging can be found in [90]. These include biomedical image processing algorithms (e.g., noise reduction, image enhancement, and detection) and image reconstruction and acquisition schemes (tomography, and Magnetic Resonance Imaging (MRI)).

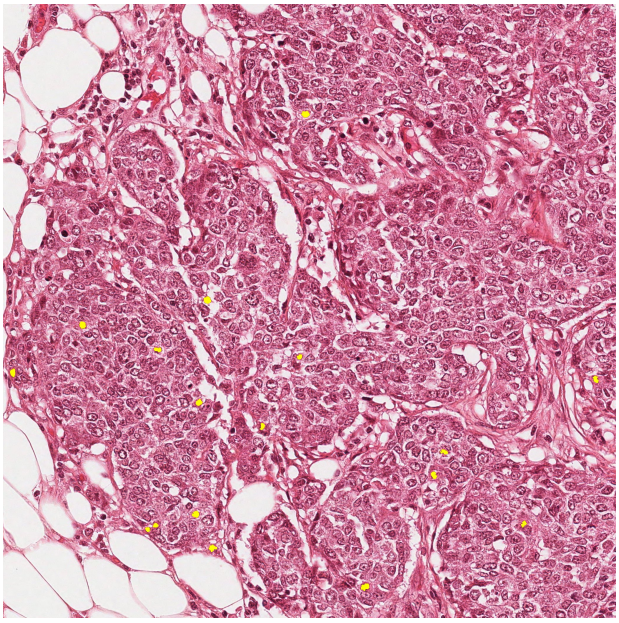
2.2.7 Object detection and recognition

Object detection is a Computer Vision technology that deals with detecting instances of semantic objects of a certain class (such as humans, traffic signs, mitotic cells) in digital images Humans recognize a multitude of objects in images with little effort, despite the fact that the image of the objects may be in different orientation, or in different size/scale. Objects can even be recognized when they are partially obstructed from view. This task is still a challenge for CV systems and represents the connection between Image Analysis topics and Machine Learning. Viola and Jones proposed a well known object detection framework [94, 93], which involves the sums of image pixels within rectangular areas, using the so-called Haar-like features, a name that resembles the Haar wavelet adopted in [70]. The technique generates a large amount of features and uses the boosting algorithm *AdaBoost* to reduce the over-complete set, by selecting the best features and training classifiers that use them. The evaluation of the classifiers generated in the learning phase can be quick, but generally not enough to be run in real-time. For this reason, the classifiers are arranged in a cascade in order of complexity, where each susequent classifier is trained only on those selected samples which pass through the preceding classifiers. If at any stage in the

cascade a classifier rejects a sample, no further processing is performed. The cascade therefore has the form of a degenerate tree.

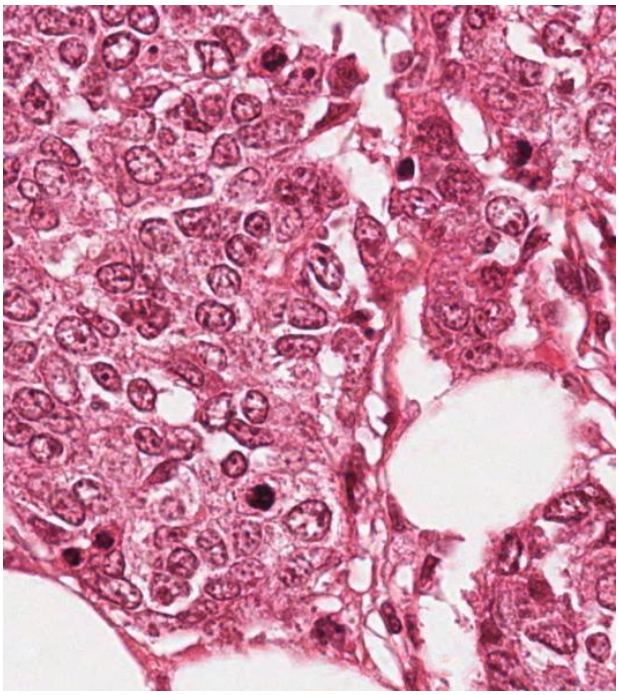


(a) source image

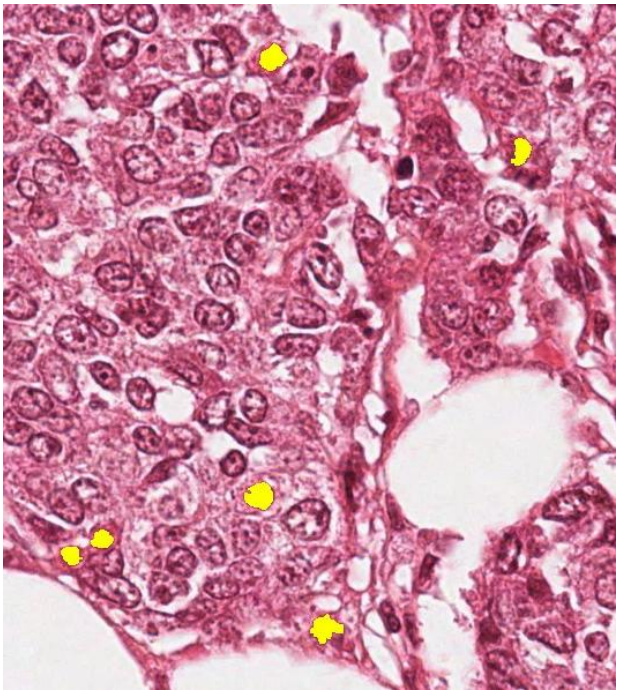


(b) mitoses

Figure 2.4: Example of image with highlighted mitoses (yellow)



(a) source image (zoom)



(b) mitoses (zoom)

Figure 2.5: Example of image with highlighted mitoses (yellow) detail of Figure 2.4

2.3 Machine Learning

Machine Learning (ML), a branch of Artificial Intelligence (AI), deals with the ability to define and to build systems that can learn from data. The core of ML deals with the representation of data and their generalization. Representation deals with the way the system describes the data. Generalization deals with the ability of the system to perform on unseen data samples. In Machine Learning, the observations are often known as *instances*, the explanatory variables are termed *features* (grouped into a *feature vector*), and the possible categories to be predicted are *classes*.

ML algorithms can be divided into different types:

- **Supervised Learning** generates a function that maps inputs to desired outputs usually called *labels*, because they are often provided by human experts classifying the training examples.

- **Unsupervised learning** models a set of inputs. It can also be referred to as *data mining* and knowledge discovery. Here, labels are not known during training.

- **Semi-supervised learning** combines both labeled and unlabeled examples to generate an appropriate function or classifier.

- **Reinforcement learning** learns how to act given an observation of the world. Every action has some impact in the environment, and the environment provides feedback in the form of rewards that guides the learning algorithm.

There exists a great variety of ML algorithms, and a detailed review is beyond the scope of this work[4].
We focous, in our analysis, on *Pattern Recognition* and in particular on *Supervised Learning* methods.

2.3.1 Pattern Recognition

Pattern Recognition (PR) is the assignment of a label to a given input value [8, 89]. In its most general form, PR involves:

- **Classification** is the problem of identifying to which of a set of categories a new observation belongs, on the basis of a training set of data containing instances whose category membership is known,

[4] A list of ML algorithms can be found in `http://en.wikipedia.org/wiki/List_of_machine_learning_algorithms`

- **Regression** is a technique for estimating the relationships among variables, assigning a real-valued output to each input,
- **Sequence labeling** refers to the assignment of a categorical label to each member of a sequence of observed values, in particular by making choices which depend on the one made for nearby elements (e.g. speech tagging)
- **Parsing** is the process of analyzing a string of symbols according to the rules of a formal grammar.

2.3.2 Classification

Among the different types of learning methods and pattern recognition techniques we focus our attention on *classification* which, in general ML terminology, is an instance of *supervised learning*.
The formal definition of a supervised classification problem can be stated as follows: an unknown function g maps the input instances $x \in X$ to the output labels $y \in Y$:

$$g : X \to Y \tag{2.6}$$

Equation 2.6 represents the *ground truth*(GT).
The *training set*

$$T = (x_1, y_1), \ldots, (x_n, y_n) \tag{2.7}$$

is assumed to represent the mapping of g in an accurate way. The classifier then tries to build a function $h : X \to Y$ that approximates as closely as possible the correct mapping. The measure of the performance (see 3.4 for details) is generally done on a separate set of data (the *test set*) whose labels are known but whose data are not used during the learning phase[53].

A common subclass of classification is *probabilistic classification*. Algorithms of this type involve statistical tools to define the best class for a given instance[71]. Probabilistic algorithms output a probability that the instance is a member of each of the possible classes. The best class is normally then selected as the one with the highest probability. Classification can be also divided into two separate problems - *binary classification* and *multi-class classification*. In binary classification, only two classes are involved, whereas multi-class classification considers the problem of assigning an object to one of several classes. Since many classification methods have been developed specifically for binary classification, multi-class classification often requires the combined use of multiple binary classifiers.

2.3.3 Binary Classification

Binary classification is the task of classifying the members of a given set of objects into two groups on the basis of whether they have some property or not[79]. Medical testing is a typical binary classification task (i.e. to determine if a patient has certain disease or not). In traditional statistical hypothesis testing, the tester starts with a null hypothesis and an alternative hypothesis, performs an experiment, and then decides whether to reject the null hypothesis in favor of the alternative. Hypothesis testing is therefore a binary classification of the hypothesis under study [62]. A *positive* result is one which rejects the null hypothesis. Rejecting the null hypothesis when it is actually true - a False positive (FP) - is a **type I error**; on the other hand, when the null hypothesis is false results in a True positive (TP). A *negative* result is one which does not reject the null hypothesis. Accepting the null hypothesis when it is actually false - a False negative (FN) - is a **type II error**; on the other hand, when the null hypothesis is true results in a True negative (TN). How the number of TP, FP, TN and FN can be used to assess the performances of a classification algorithm is treated in Section 3.4.

2.3.4 Binary Classifiers

An algorithm that implements a classification, is defined a **classifier**. The term also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category (i.e. *class*). A great amount of algorithms has been developed for classification purposes, in particular for CV tasks [56]. Some methods suitable for learning binary classifiers include[96]:

- Naive Bayes classifiers
- Bayesian networks [98]
- Decision trees [5]
- Random Forests (RFs) [36]
- Support Vector Machines (SVMs) [40]
- Hidden Markov models
- Neural Networks (NNs) [74]

In our work we focused on two types of classifiers: *Support Vector Machines* and *Random Forests* which are widely used in CV classification problems (e.g. [83] and [10]).

2.3.5 Software Tools

Classification tasks can be accomplished by a large amount of software tools. Here we mention the ones that we consider to be the most relevant ones.
Weka [26, 31] is a **FLOSS** general purpose data mining software tool developed by the Waikato University [5] which allows to implement a great variety of classifiers [96]. It also has an interface with **R** [6] [39].
MATLAB can perform classification task by means of some of its toolboxes (i.e. Bioinformatics [7] and Statistics [8]).

[5] http://www.cs.waikato.ac.nz/ml/weka/
[6] http://cran.r-project.org/
[7] http://www.mathworks.com/products/bioinfo/
[8] http://www.mathworks.com/products/statistics/

Chapter 3

Problem Definition

"πάντες ἄνθρωποι τοῦ εἰδέναι ὀρέγονται φύσει"
(All men naturally desire knowledge)

Ἀριστοτέλης(Aristotle, Met. 1.980a)

The aim of our work is to analyze the performances of mitosis detection algorithms compared to humans trying to classify the same images. To acheve this goal, we selected a subset of the publicly-available *MITOS dataset* made for the *ICPR 2012 Contest* on `Mitosis Detection in Breast Cancer Histological Images`[1]. Then we run the following activities:

- collected the performances of the top-scoring algorithms developed for the ICPR 2012 Mitosis Detection Contest (focusing on the dataset),
- applied some classifiers to the dataset,
- implemented a web-based test for humans,
- analyzed the performances of the algorithms compared to the results achieved by humans.

The main definitions for the problem in exam are the subject of this chapter.

3.1 Framework

The purpose of automating the mitosis detection problem requires the definition of a framework that involves Computer Vision and Machine Learning

[1] `http://ipal.cnrs.fr/ICPR2012/`

aspects. ML is growing in importance for biology-related tasks [88]. In general, PR (see 2.3.1) is the computational approach used to analyze datasets of images [81].

3.1.1 Detection

The analysis of digital images requires identifying ROIs or *candidates* within the images. Once a region is isolated, a digital image allows many types of measurements and statistics to be collected, as well as the number of objects and their distribution. This region selection can be done manually by drawing boxes or free-hand regions using an interactive tool [87], or automatically using computer algorithms known as segmentation algorithms [54]. The input to the algorithm may be an entire image, a sub-image region identified with segmentation algorithms, or simply image samples in the form of rectangular tiles.

3.1.2 From Detection to Classification

PR then requires training a computer to classify groups of images (i.e. a subset of images with manually detected mitoses). The machine can learn on its own what aspects of the images represent natural experimental variation and are therefore irrelevant, and what aspects are important for distinguishing the groups of control images (i.e. the testing set) from each other (see 3.2). This ability to select different image measurements allows the use of a great variety of image description algorithms, potentially making the collection of algorithms very general. The benefits of subdividing images into ROIs involve:

- reduce the number of pixels to consider
- bias the algorithm to process objects of interest rather than background
- center or align objects

A further step consists in the extraction of image content descriptors (*image features*), which are values that describe the image content numerically. These values can reflect various texture parameters of the image, the statistical distribution of pixel intensities, edges, etc. While the dimensionality of the raw pixels can be high, the number of image features ranges between a dozen to a few hundred. Each feature value describes a specific image characteristic. Then, the image features are used to draw conclusions about

the data. The feature set is then used to infer rules for combining them in a classifier. These two steps constitute the training stage in PR, where the goal is to correctly classify the training images. The trained classifier is then tested on control images that were excluded from the training stage. This cross-validation is important to establish the classifier's ability to identify new images, ensuring that it is not restricted to recognizing images it was trained with.

3.1.3 Performances

If the performance of the algorithm are not satisfying (see 3.4 for details), the algorithm can be trained again on a different set of features, until the detection capabilities reach the desired values (if feasible). Finally, the results of image classification need to be interpreted by the researcher in an experimental context to reach a biological conclusion. Figure 3.1 shows the steps described above.

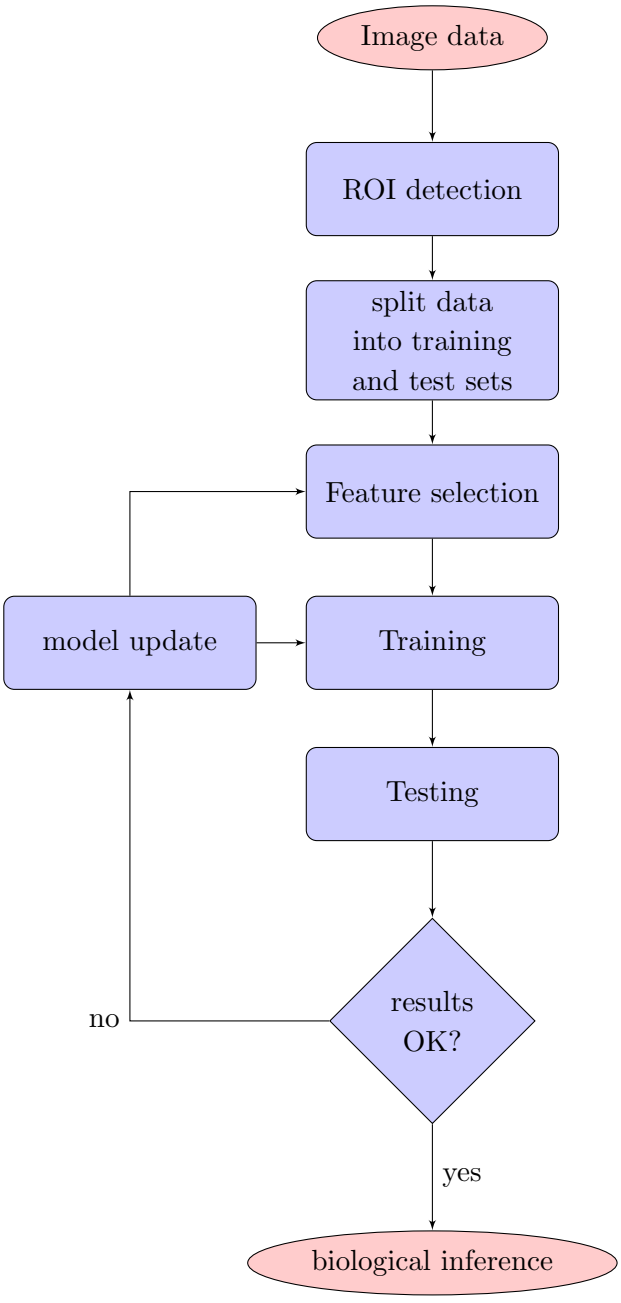


Figure 3.1: Flowchart of Detection Algorithm

3.2 Definition of Classification

In ML the idea of *classification* refers to the problem of identifying to which of a set of categories (named *classes*) a new observation belongs, on the basis

of a training set of data containing instances whose category membership is known. In case of mitosis detection, the elements of a classification are basically the following:

- the *input* to the classification problem is a set of *features* computed on each of the *candidates* selected in a preparatory phase. Each set of features composing a candidate is known as *instance*. Each instance is *labeled* with the *class* which it belongs to.

- the *classes* are simply two: `mitosis` (which we call *class 1* or *positive*) or `non-mitosis` (which we call *class 0* or *negative*) making it a case of binary classification.

- the *output* of the classification can be a *hard* classification: the output of the classifier is simply *0* or *1*, corresponding to mitosis or non-mitosis respectively. On the other hand the classification can be *soft*: the output of the classifier is a real number c:

$$0 \leq c \leq 1 \tag{3.1}$$

  a subsequent phase of analysis consists in selecting the best threshold so that:

$$class = \begin{cases} 0, & \text{if } c < threshold. \\ 1, & \text{otherwise.} \end{cases} \tag{3.2}$$

  the selection of the threshold is made in function of the measured performances of the classification algorithm (see 3.4).

3.3 Review of Algorithms solving the mitosis detection problem

Different to other pattern recognition tasks, mitotic cells essentially are irregular shape objects. As a result, there is no simple or unique way of extracting the features of mitotic cells and then many different classifiers can be made.

Here we briefly review the main algorithms found in literature that solve the mitosis detection task.

- The method proposed in [92] consists of two main components: candidate extraction and candidate classification. Candidate objects are extracted by image segmentation with the Chan-Vese level set method [63]. A statistical classifier is trained with a number of features that describe the size, shape, color and texture of the candidate objects.

- The approach in [49] uses, after a phase of automatic segmentation of the image, a Gamma Gaussian Mixture Model (GGMM) to classify the candidates: the GGMM is a parametric technique for estimating probability density function. In this context, it is formulated as a function of pixel intensities.

- The work in [41] also proposes a two phases approach: the detection candidates points are selected by using an algorithm named eXclusive Independent Component Analysis (XICA), which gives two sets of training patterns: positive and negative patterns (positive and negative basis set). Then a sparse representation method [97] is used to classify the candidates.

- Also the approach in [44] has two phases. In the first stage, the detection of candidate mitosis is performed. The input RGB images are transformed into blue-ratio images. A Laplacian of Gaussian (LoG), thresholding and morphological operations on blue-ratio images is then executed to generate candidate mitosis regions. Then, the candidate regions are selected using morphological rules; the center point of each region is used as seed point for mitosis. In the second stage, co-occurrence features, run-length features and SIFT features are computed for each candidate patch. Finally a classification is performed to put the candidate patch either in the mitosis class or in the non-mitosis class. Three different classifiers have been evaluated: decision tree, linear kernel SVM and non-linear kernel SVM.

- The article in [57] uses a simple rule that extracts blobs representing nuclei of possible mitotic figures to establish a set of candidates. ML is applied in three phases. One phase applies a support vector regression which remaps the color palette of the original image to normalized values. The next phase is a Convolutional Neural Network (CNN), applied at each extracted blob. The CNN contributes a generate a feature vector, which also contains many other measurements regarding the shape, color, mass, and texture of the blob and its neighborhood. In the final phase, a SVM uses the feature vector to classify the area around the blob as a mitotic figure or not.

The last two works that we mention here are particularly interesting because they work on the same dataset that we used, the *MITOS Dataset* (see Chapter 4 for details).

- The approach in [43] works on z-stack focus planes for detection of mitosis candidates. Then candidates are detected using thresholding and morphological operations on selected band and focus plane. A multi-spectral features vector is computed for detected candidates having intensity and texture features across all bands of multi-spectral images. In addition, using segmented regions of detected candidates, morphological features are also computed. A feature selection algorithm is employed on this features vector in order to save the computation cost, to discard any redundancy in the data, and to improve classification accuracy. Classification is achieved using Bayesian, Decision Tree , Neural Network as well as linear and non-linear SVM classifiers.

- The approach in [16] is procedurally simpler than other methods, as no candidate selection is performed.A supervised Deep Neural Network (DNN) as a powerful pixel classifier. The DNN is a type of CNN. It directly operates on raw RGB data sampled from a square patch of the source image, centered on the pixel itself. The DNN is trained to differentiate patches with a mitotic nucleus close to the center from all other windows. Mitosis in unseen images are detected by applying the classifier on a sliding window, and post-processing its outputs with simple techniques. Because the DNN operates on raw pixel values, no human input is needed.

In our work, we also used, as a reference, the performances other top-scoring algorithms developed for the *MITOS Dataset*, whose main features will be described in a special issue of the *Journal of Pathology Informatics*[2] expected for June 2013.

3.4 Performance and Benchmarking

In order to set up a correct and valid comparison among mitosis detection algorithms, a consistent definition of *performance* plays a fundamental role. The general appearance of a mitosis results in the fact that automatically detecting mitoses is very challenging, and in fact even the agreement between pathologists is not perfect.

[2] `http://www.jpathinformatics.org/`

3.4.1 Pathologists' Agreement

The work in [57] deeply analyzes the agreement among pathologists examining the same HE images. The BR grading system is widely recognized as the one giving the most stable definitions, and its grades are widely used to select treatments. Nevertheless, the level of agreement is shown to be far from perfect.

The level of agreement may be reported in Cohen's Kappa (κ) [17] whose range is $0 \leq \kappa \leq 1$, with ***1*** corresponding to perfect agreement, and ***0*** in the case of probabilistically independent decisions.

The value of κ can be divided in ranges:

- **0-0.2** is often considered as *slight* agreement,
- **0.2**–**0.4** as *fair*,
- **0.4**–**0.6** as *moderate*,
- **0.6**–**0.8** as *good*,
- **0.8**–**1** as almost *perfect*.

Most studies show that value of κ generally varies from *fair* to *moderate* (e.g. the study in [61] reports a value of $\kappa = 0.5$).
The low level of agreement among pathologists is an issue also for algorithms' benchmarking, as it can be difficult to establish a definite Ground Truth (GT) (i.e. the process of gathering the proper objective data for the test). Nonetheless, the images of the *MITOS Dataset* have been annotated by only one pathologist: the algorithms of the *2012 ICPR Contest* and our work based their GT on that.

3.4.2 Benchmarking

Benchmarking of different algorithms and comparison with human performance play a key role in a detection framework, it is so of great importance the definition of *performance*.
Given a GT, the *Confusion Matrix* (or Error Matrix [84]), is so defined: each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. The name originates from the fact that it makes it easy to see if the system is confusing two classes (i.e. mislabeling one as another). The elements of the matrix are:

- **TP**: *True Positive*, a sample labeled as true is predicted as true,
- **TN**: *True Negative*, a sample labeled as false is predicted as false,
- **FP**: *False Positive*, a sample labeled as false is predicted as true (i.e. false alarm, or *Type I* error),
- **FN**: *False Negative*, a sample labeled as true is predicted as false (i.e. miss, or *Type II* error),

Table 3.1: Confusion Matrix

	predicted *Positive*	predicted *Negative*
Actual *Positive*	True Positive (TP)	False Negative (FN)
Actual *Negative*	False Positive (FP)	True Negative (TN)

The data in Table 3.1 represent the minimum required data to assess the performance of a classifier (human or automatic). Starting from this, some other measurements can be done.
The data in the table can be assembled to define some performance indicators.

Accuracy

The accuracy of a test represents the degree of closeness of prediction to the actual value, and it is measured as:

$$\text{Accuracy} \quad ACC = \frac{TP + TN}{P + N} \tag{3.3}$$

$$\text{where} \quad P = TP + FN \text{ and } N = TN + FP \tag{3.4}$$

Precision, Recall, F-Score

A first set of measures that can be done on the data of the confusion matrix are: *precision*,also named Positive Predictive Value (PPV), *recall*, or True Positive Rate (TPR), and *F-Score* [29]. They are defined as follows:

$$\text{Precision} \quad p = \frac{TP}{TP+FP} \tag{3.5}$$

$$\text{Recall} \quad r = \frac{TP}{TP+FN} \tag{3.6}$$

Both precision and recall have a natural interpretation in terms of probability. Precision may be defined as the probability that an instance has class **1**, given that it is classified as **1**, while the recall is the probability that a class **1** object is classified:

$$p = P(\text{label} = true \mid \text{class} = true) \tag{3.7}$$

$$r = P(\text{class} = true \mid \text{label} = true) \tag{3.8}$$

The weighted (with parameter β) harmonic average of *precision* and *recall* leads to the *F-Score*[60]:

$$\text{F-Score} \quad F_\beta = (1+\beta^2) \cdot \frac{pr}{r+\beta^2 p} = \frac{(1+\beta^2)TP}{(1+\beta^2)TP + \beta^2 FN + FP} \tag{3.9}$$

F_1-Score is most widely used as a measure of the accuracy of the classifier. It can be interpreted as a weighted average of the precision and recall: an F_1-Score reaches its best value at **1** and worst score at **0**.

Specificity, Sensitivity

Sensitivity and *Specificity* are often used in clinical tests as a measure of the ability of the test to confirm or refute the presence of a disease[51]. Ideally a test correctly identifies all patients with the disease, and similarly correctly identifies all patients who are disease free. In other words, a perfect test is never positive in a patient who is disease free and is never negative in a patient who is in fact diseased.
The sensitivity of a clinical test refers to the ability of the test to correctly identify those patients with the disease:

$$Sensitivity = \frac{TP}{TP+FN} \tag{3.10}$$

It can be noted that the definition of sensitivity is the same as the definition of recall. A high sensitivity is clearly important where the test is used to identify a serious but treatable disease.

The specificity, or True Negative Rate (TNR), of a clinical test refers to the ability of the test to correctly identify those patients without the disease:

$$Specificity = \frac{TN}{TN + FP} \tag{3.11}$$

High specificity results in few patients who are disease free being told of the possibility that they have the disease and are then subject to further investigation or treatments. Also the following relation holds:

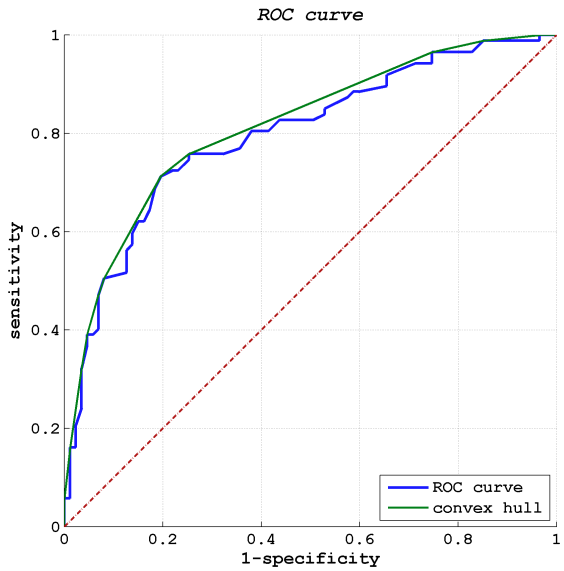
$$Specificity = 1 - FPR \tag{3.12}$$

$$\text{where } FPR = \frac{FP}{FP + TN} \tag{3.13}$$

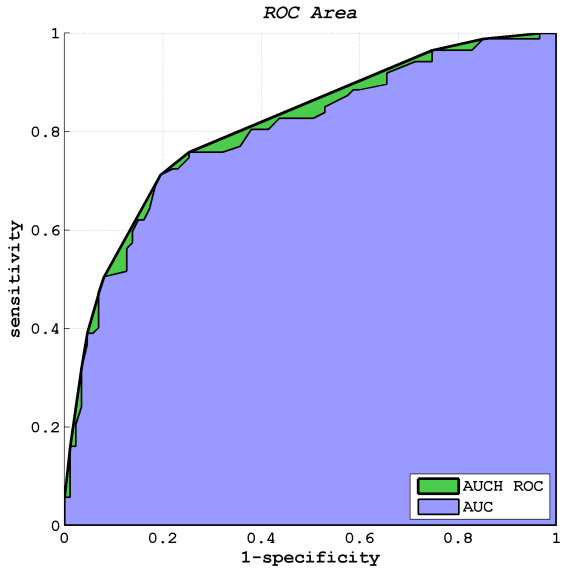
Receiver Operating Characteristic (ROC)

As mentioned in 3.2, the classifier or diagnosis result can be a real value (continuous output), in this case the boundary between the two classes of the binary classifier must be determined by a threshold value. A Receiver Operating Characteristic (ROC) space is defined by FPR and TPR as x and y axes respectively, which depicts relative trade-offs between true positive (benefits) and false positive (costs)[100]. Since TPR is equivalent with sensitivity and FPR is equal to 1 − specificity, the ROC graph is sometimes called the sensitivity vs (1 − specificity) plot[20]. Each prediction result or instance of a confusion matrix represents one point in the ROC space (see Figure 3.2(a)). The best possible prediction method would yield a point in the upper left corner or coordinate (0,1) of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). The (0,1) point is also called a perfect classification. A completely random guess would give a point along a diagonal line from the left bottom to the top right corners.
The diagonal divides the ROC space. Points above the diagonal represent good classification results (better than random), points below the line poor results (worse than random).

The ROC is used to generate summary statistics. One of the often used is the area under the ROC curve, or AUC (Area Under Curve)[13, 32] (see Figure 3.2(b)): AUC is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. The AUC can be related to other summary statistics like the *Gini coefficient* [21] and the *Mann-Withney U* [59]. Another common measure



(a) ROC curve



(b) AUC

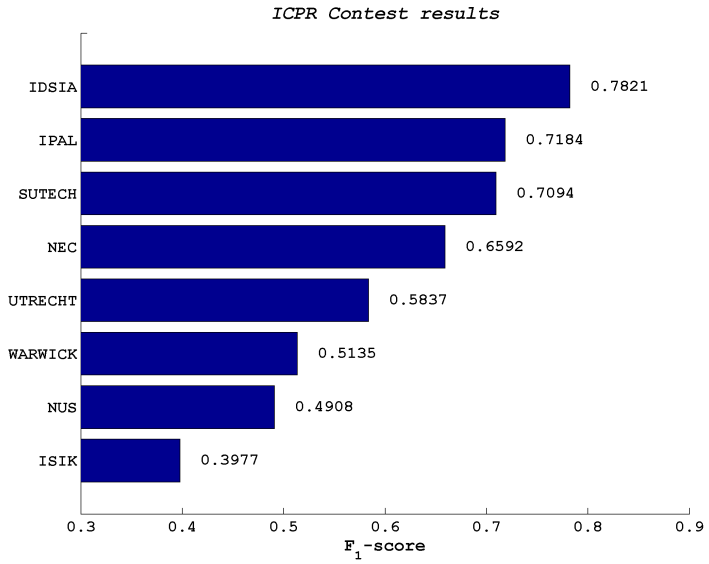
Figure 3.2: Example of ROC curves

related to the ROC curve is known as the Area Under the ROC Convex Hull (*AUCH ROC*, in Figure 3.2(b)), which computes the area unter the convex hull of the ROC curve, as it can be shown that any point on the line

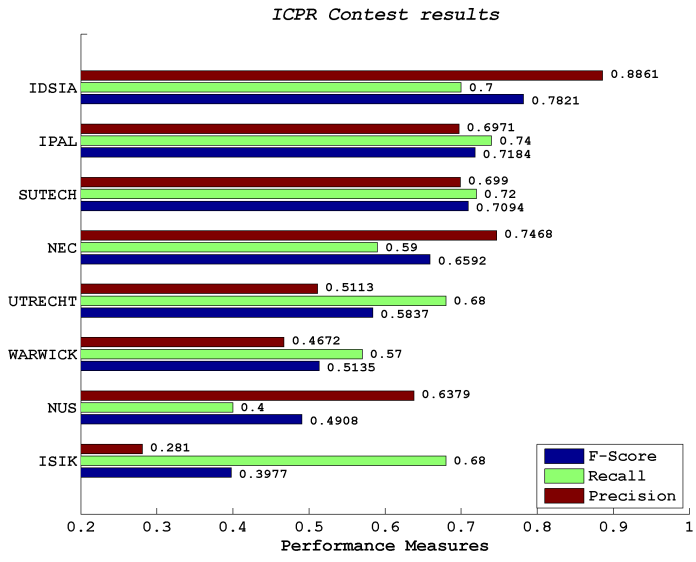
segment between two prediction results can be achieved by randomly using one or other system with probabilities proportional to the relative length of the opposite component of the segment.

3.4.3 Performances of Algorithms on MITOS Dataset

We report here the performances of the best-scoring algorithms that participated to the ICPR2012 Contest, as shown on the contest website (Figure 3.3). The principal metric adopted to compare algorithms is the F_1-Score (Figure 3.3(a)) , but also precision and recall are shown 3.3(b).



(a) F-Score



(b) metrics

Figure 3.3: Performances of best algorithms in ICPR 2012 contest

Chapter 4

Design of a Mitosis Detection algorithm

"Ab uno
disces omnis""
(Learn everything from one)

Publius Vergilius Maro (Aeneis II, 65-66)

We developed an algorithm to perform mitosis-detection as a part of our work, with the aim to compare its results with humans facing the same task.

4.1 Dataset

We used the public MITOS dataset [2]. The dataset is composed by a total of 50 2084×2084 pixel images covering an area of 512×512 μm each, acquired with an APERIO XT scanner (see Figure 2.1). A unique split is defined by the dataset authors, with 35 images used for training and 15 for evaluation. The dataset contains a total of about 300 mitosis, which were annotated by an expert pathologist. The performance of the algorithms participating to the *2012 ICPR mitosis detection contest* are shown in Section 3.4.3.
With reference to Figure 3.1, we focused on on the classification subproblem, with the ROIs given as an input. The input is given in form of an image patch with size 100×100 pixel: such size completely contains the image of the cell. The task is to map each patch to one of two classes:

C1: the image contains a mitosis at its center,

C0: the image does not contain a mitosis anywhere.

There are no samples in which a mitosis is visible off-center.

4.1.1 Image Candidates

For the ***C1*** class, all the 216 mitosis available in the 35 training images are chosen as training samples, and all 87 mitosis in the evaluation images are chosen as evaluation samples.
We enforced an even distribution of the two classes classes both in training and in evaluation sets, and therefore selected 216 ***C0*** samples for training, and 87 ***C0*** samples for evaluation; the resulting training set contained 432 samples.
Millions of different ***C0*** samples may be randomly chosen from the original training and evaluation images: an overwhelming majority of such samples would not contain any nucleus and be non-informative for training and trivial for evaluation. Limiting the choice to non-mitotic nuclei — which greatly outnumber mitotic ones — would not solve the problem, since most of such nuclei look very similar to each other and are trivially identified as non-mitotic. Only a small subset of non-mitotic nuclei — as well as other structures and artifacts — pose an actual challenge, both for humans and for algorithms.
In order to select such objects as ***C0*** samples, we used the output produced by a simple CNN-based mitosis detector, similar to the one outlined in [16] for selecting useful training samples. The detector, built at IDSIA, was trained on few images in the training set, then applied on the whole dataset. Because the detector was simple and trained on a small amount of data, it performed poorly and detected a lot of false positives. ***C0*** samples have been randomly chosen among the outputs of such detector which are farther than 50 pixels from the centroid of any mitosis; this ensures that no actual mitosis is visible in the corresponding image patch. The resulting samples do in fact resemble mitosis, are informative in the training set, and appear non-trivial in the evaluation set. Finally, 10 ***C0*** samples in the evaluation set are substituted with 5 random false positives obtained from each of the two best performing algorithms (IDSIA and IPAL). These last 10 samples are particularly useful to compare humans to algorithms, in fact allowed us to better observe how test subjects behave on the algorithms' false positives, which are rare in the evaluation set because algorithms were tuned to solve a problem with very low prevalence of mitotic samples.

4.1.2 Extended Dataset

We extended our dataset by rotating and mirroring each image patch (see Figure 4.1). We used the extended dataset only for the detection algorithm, so that we could analyze the effect of different features, which can be explicitly dependent on orientation or not, on the global performance of the classifier.
In case of extended dataset, the classification of a single image patch becomes the average of the classifications obtained on the 8 samples.

$$c_i = \frac{\sum_{j=1}^{8} c_{ij}}{8} \tag{4.1}$$

Where c_i represents the classification of image patch i, and c_{ij} represents the classification of variation j of image patch i .

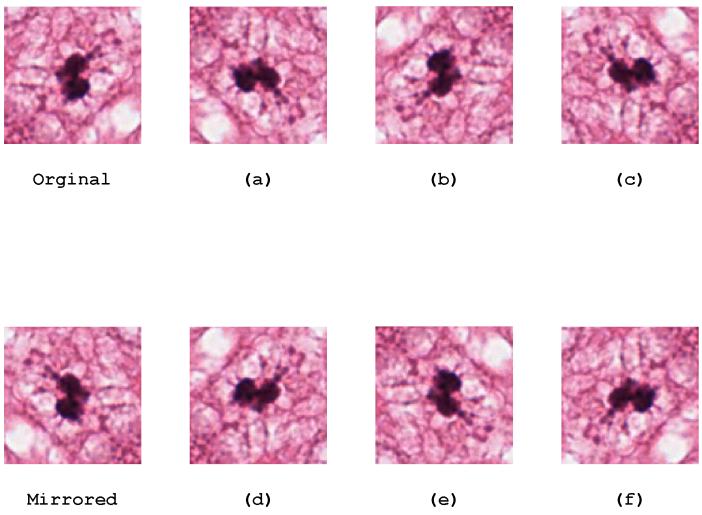


Figure 4.1: Extended dataset
(a),(b),(c): $\pi/2$ clockwise rotations, (d),(e),(f): mirror and $\pi/2$ clockwise rotations.

4.2 Features Extraction

Each image patch can be represented as a 100×100×3 matrix, where the $(i, j, :)$ triplet represents the RGB value of point with coordinates (i, j) in the

image. Each value is in the range 0 to 255. Starting from these (raw) data we extracted some features by which we trained and tested our classifiers.

4.2.1 Simple Features

The simplest features that can be computed involve the average and the standard deviation of the Red-Green-Blue (RGB) values of the image patch. They can be computed on all the data or can be maintained separated for each RGB component. In the first case, average and standard deviation each give one value every instance:

$$m = \frac{1}{100 \cdot 100 \cdot 3} \left(\sum_{i=1}^{100} \sum_{j=1}^{100} \sum_{k=1}^{3} i_{ijk} \right) \tag{4.2}$$

$$\sigma = \sqrt{\frac{1}{100 \cdot 100 \cdot 3} \left(\sum_{i=1}^{100} \sum_{j=1}^{100} \sum_{k=1}^{3} (i_{ijk} - m)^2 \right)} \tag{4.3}$$

Otherwise, average and standard deviation produce a vector of three components:

$$\overline{M} = \begin{bmatrix} \frac{1}{100 \cdot 100} \left(\sum_{i=1}^{100} \sum_{j=1}^{100} i_{ij1} \right) \\ \frac{1}{100 \cdot 100} \left(\sum_{i=1}^{100} \sum_{j=1}^{100} i_{ij2} \right) \\ \frac{1}{100 \cdot 100} \left(\sum_{i=1}^{100} \sum_{j=1}^{100} i_{ij3} \right) \end{bmatrix} \tag{4.4}$$

$$\overline{S} = \begin{bmatrix} \sqrt{\frac{1}{100 \cdot 100} \left(\sum_{i=1}^{100} \sum_{j=1}^{100} (i_{ij1} - M(1))^2 \right)} \\ \sqrt{\frac{1}{100 \cdot 100} \left(\sum_{i=1}^{100} \sum_{j=1}^{100} (i_{ij2} - M(2))^2 \right)} \\ \sqrt{\frac{1}{100 \cdot 100} \left(\sum_{i=1}^{100} \sum_{j=1}^{100} (i_{ij3} - M(3))^2 \right)} \end{bmatrix} \tag{4.5}$$

Another simple set of features is represented by the *median* of each RGB value. The median is defined as the numerical value separating the higher half of the data sample, from the lower half and can be found by arranging all the data from lowest value to highest value and picking the middle one, or the mean of the two middle values, in case of even data. Each of the features above are independent of the orientation of the image.

4.2.2 Color Histograms and Intensities

A color histogram is a representation of the distribution of colors in an image, i.e. the number of pixels that have colors in each of a fixed list of color ranges [85], that span the image's color space. The color histogram can be built for any kind of color space, although the term is more often used for three-dimensional spaces like RGB or Hue Saturation Value (HSV). A histogram of an image is produced first by discretization of the colors in the image into a number of bins, and counting the number of image pixels in each bin. We built the RGB color histogram for each image patch, using 16 bins for each channel. The feature vector is so composed of 48 elements. Also this feature is orientation independent.

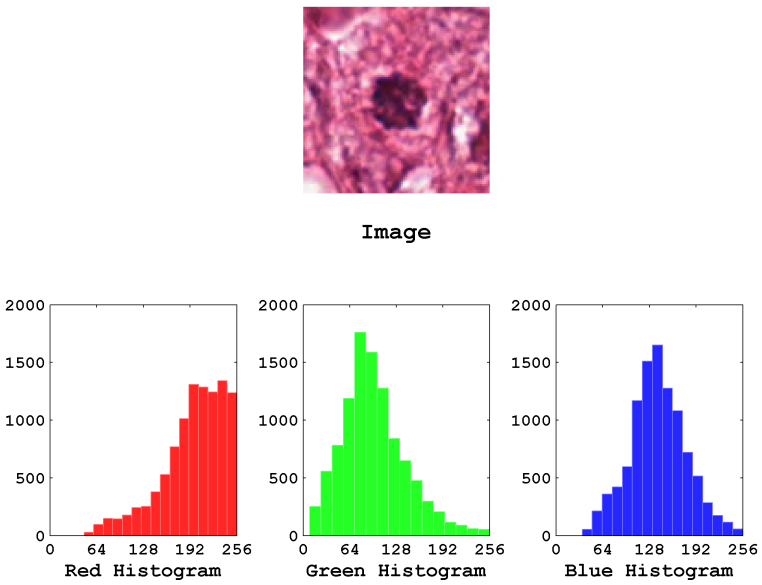


Figure 4.2: Color Histograms of sample image

It is generally possible to transform a color image into a gray-scale one. One typical transformation algorithm, applied pixel by pixel, is the following:

$$pix_{gray} = 0.2989 \cdot pix_{red} + 0.5870 \cdot pix_{green} + 0.1140 \cdot pix_{blue} \tag{4.6}$$

On the resulting monochromatic image, it is possible to compute an *intensity histogram*.
We preferred to compute a slightly different feature: the average intensity in

the 25 central regions of the image. We first computed the gray-scale image according to Equation 4.6, then we selected the central part of the image and divided it in a grid of 5×5 elements. We finally computed the mean intensity for each element. Figure 4.3 illustrates the procedure.

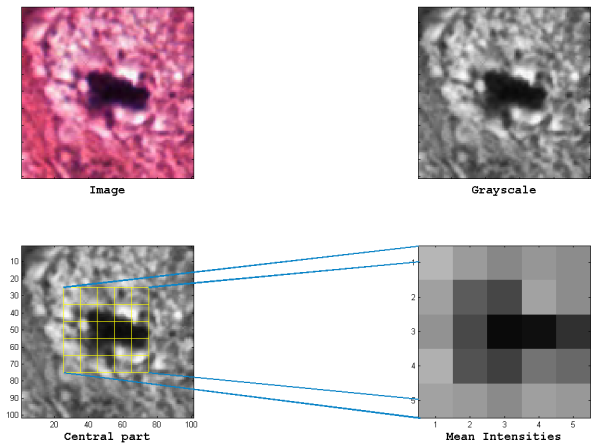


Figure 4.3: Mean gray-scale intensity of central part of image patch

The resulting feature vector is composed of 25 values, corresponding to the intensities, ordered columnwise. This type of feature is orientation dependent.

4.2.3 Texture Features

Texture features are widely used in different CV tasks, as pointed out in Section 2.2.6. We focused on the features described in [67] and [66], based on Local Binary Patterns (LBP). The general idea of LBP is described on page 10. The LBP features considered here are labeled $\text{LBP}_{P,R}$, where P is the number of neighbors considered and R is the distance from the pixel. The two main characteristics of the LBPs considered are:

- *uniformity*: which is a fundamental property of local image texture. It refers to the uniform appearance of the local binary pattern, that is, there is a limited number of transitions or discontinuities in the circular presentation of the pattern. The most frequent uniform binary patterns correspond to primitive "microfeatures", such as edges, corners, and spots; hence, they can be regarded as feature detectors that are triggered by the best matching pattern.

- *rotation invariance*: which takes into account if a spatial pattern is affected by rotation or not.

Three different types of features can be built, on the basis of Equation 2.4:

1. $\text{LBP}^{u2}_{P,R}$: uniform feature,
2. $\text{LBP}^{ri}_{P,R}$: rotation invariant feature,
3. $\text{LBP}^{riu2}_{P,R}$: uniform and rotation invariant feature,

In particular we used:

$$\text{LBP}^{\text{type}}_{8,R} \quad \text{where} \begin{cases} \text{type} & \in \{ri, u2, riu2\} \\ R & \in \{1, 2, 3\} \end{cases} \tag{4.7}$$

while building the feature vector, we used the *type* parameter in a mutually exclusive way, i.e. we did not concatenate LBPs of different types. On the other hand, we built feature vectors with various combitations of radii. The following equations show the three different mutually exclusive texture feature sets that we considered.

$$\overline{L} = \left[\text{LBP}^{riu2}_{8,1}, \text{LBP}^{riu2}_{8,2}, \text{LBP}^{riu2}_{8,3}\right] \tag{4.8}$$

$$\overline{U} = \left[\text{LBP}^{u2}_{8,1}, \text{LBP}^{u2}_{8,2}, \text{LBP}^{u2}_{8,3}\right] \tag{4.9}$$

$$\overline{R} = \left[\text{LBP}^{ri}_{8,1}, \text{LBP}^{ri}_{8,2}, \text{LBP}^{ri}_{8,3}\right] \tag{4.10}$$

Finally, we considered the VAR operator, as described in Equation 2.5. As, from early tests, a single VAR value for the entire image patch proved to be non-significant, we decided to follow an approach similar to the one described for the intensity histogram (see Figure 4.3) and evaluated the mean value of a grid of samples in the central region of the image. Figure 4.4 shows a sample of $VAR(8,1)$ computation. Please note that the gray-scale mapping of the VAR(8,1) figure has been adjusted to be visible with full gray-scale range.

The resulting feature vector is composed of 36 values, corresponding to the average VAR(8,1) in each element of the grid, ordered columnwise. This type of feature is orientation dependent.
The Matlab code implemented to build the feature vectors is listed in .1

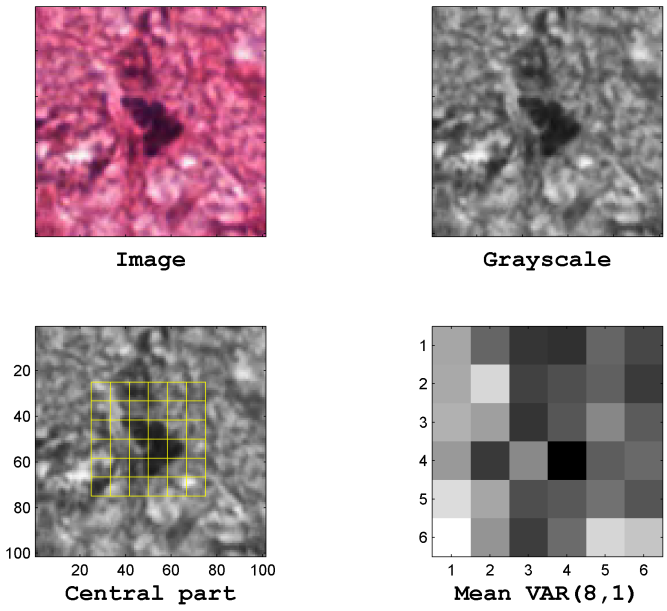


Figure 4.4: Example of VAR(8,1) feature

4.3 Classifiers

Once defined the set of feature to be considered, it is possible to build a matrix whose lines represent an *instance* (i.e. an image patch) and whose columns represent a *feature* (or a component of it): Equation 4.11 represents such matrix.

$$M_{feats} = \begin{array}{c} \\ 1 \\ \vdots \\ \vdots \\ n_i \end{array} \overset{\overset{features}{\longrightarrow}}{\begin{array}{cccccccc} 1 & \cdots & & \cdots & & \cdots & & n_{fc} \\ c_{111} & c_{112} & \cdots & c_{1k1} & \cdots & c_{1kn_k} & \cdots & c_{1n_f1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ c_{n_i11} & c_{n_i12} & \cdots & c_{n_ij1} & \cdots & c_{n_ijn_j} & \cdots & c_{n_in_f1} \\ & \underbrace{\qquad}_{feat_1} & & \underbrace{\qquad\qquad}_{feat_j} & & & & \underbrace{\qquad}_{feat_{n_f}} \end{array}} \tag{4.11}$$

Where n_f is the total number of features and n_i is the total number of instances. Each feature can be made of more than one component (e.g. $feat_1$ and $feat_j$ in the example). For this reason, the total number of columns in the matrix (n_{fc}) is given by the sum of all the feature components. So, each element of the matrix c_{ijk} is the k^{th} component of the j^{th} feature in

the i^{th} instance. The matrix representing the eavluation set is built in the same way.
A vector represents the class which every instance belongs to. Equation 4.12 describes such vector:

$$V_{class} = \begin{array}{c} \\ 1 \\ \vdots \\ \\ \vdots \\ n_i \\ {\scriptstyle instances} \downarrow \end{array} \begin{pmatrix} {\scriptstyle class} \\ \overbrace{e_1} \\ \vdots \\ e_i \\ \vdots \\ e_{n_i} \end{pmatrix} \tag{4.12}$$

where e_i belongs to one of the two classes. In some implementations of binary classifiers it is required that $e_i \in \{-1, 1\}\ \forall i = 1, \cdots, n_i$, otherwise $e_i \in \{0, 1\}\ \forall i = 1, \cdots, n_i$. The vector representing the GT of the evaluation set is built in the same way.
Having a matrix representing the training feature set, a matrix representing the evaluation (i.e. testing) feature set and two vectors including the GT classification of each image patch, it is possible to run a classifier that tries to get insights form the feature set in order to classify the evaluation set.

In our work we focused on two types of classifiers:

- *Support Vector Machines*, which are widely used in computer vision classification problems, in particular in biomedical imaging ([83, 86, 46, 15], see also Section 3.3),
- *Random Forests*, which is a relatively new ensemble approach that can also be thought of as a form of nearest neighbor predictor ([11, 37, 10]).

We also mention CNN, as it played a relevant role in the definition of our dataset (see Section 4.1.1).

4.3.1 Support Vector Machines

We used the Matlab implementation of the *libSVM* described in [40]. SVMs are a popular classification technique. The goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data given only the test data attributes. Given a training set of instance-label pairs $(\mathbf{x_i}, y_i)$, $i = 1, \cdots, l$, where $\mathbf{x_i} \in \mathbb{R}^n$ and $y \in \{-1, 1\}^l$.
The SVM requires the solution of the following optimization problem:

$$\min_{\mathbf{w},b,\xi} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i$$

Subject to: (4.13)

$$y_i\left(\mathbf{w}^T\phi(\mathbf{x_i}) + b\right) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$

The training vectors $\mathbf{x_i}$ are mapped into a higher dimensional space (maybe infinite), by the function ϕ. SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. $C > 0$ is the penalty parameter of the error term. The function

$$K(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i})^T\phi(\mathbf{x_j}) \tag{4.14}$$

is called the *kernel function*. Many kernel functions have been defined, the most common are:

- *linear*: $K(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i}^T\mathbf{x_j}$,
- *polynomial*: $K(\mathbf{x_i}, \mathbf{x_j}) = \left(\gamma\mathbf{x_i}^T\mathbf{x_j} + r\right)^d, \gamma > 0$,
- *Radial Basis Function (RBF)*: $K(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\gamma\|\mathbf{x_i} - \mathbf{x_j}\|^2\right), \gamma > 0$,
- *sigmoid*: $K(\mathbf{x_i}, \mathbf{x_j}) = \tanh\left(\mathbf{x_i}^T\mathbf{x_j} + r\right)$.

Where γ, d and r are kernel parameters [64].
In our work we focused on RBFs and sigmoid kernels, which are used in most cases. In SVMs the *support vectors* are the training instances that concur to define the separating hyperplane in the kernel space. The image of Figure 4.5 gives a linear representation of a SVM.

4.3.2 Random Forests

Decision Trees (DTs) are attractive classifiers due to their high execution speed and simplicity. However, trees often suffer from performance loss, in terms of generalization accuracy on unseen data when the complexity of the problem grows [36].
Random Forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest [11]. So, RF can be viewed

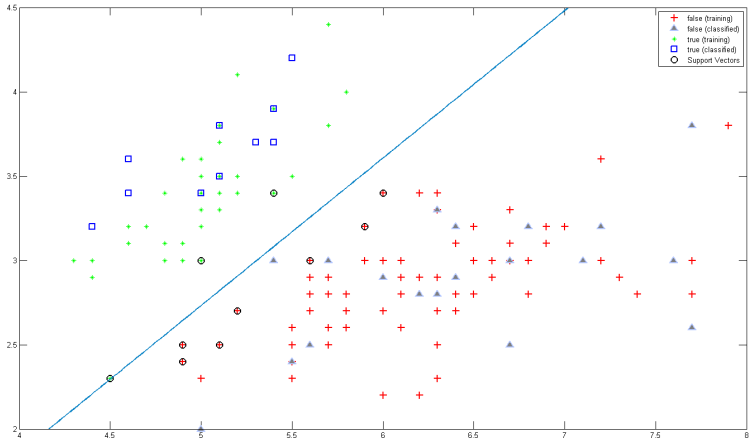


Figure 4.5: Representation of a SVM

as an ensemble approach that can also be thought of as a form of nearest neighbor predictor. Ensembles are a divide-and-conquer approach used to improve performance. The main principle behind ensemble methods is that a group of "weak learners"can be combined together to form a "strong learner". Each classifier, individually, is a weak learner, while all the classifiers taken together are a strong learner. An example of DT is shown in Figure 4.6.

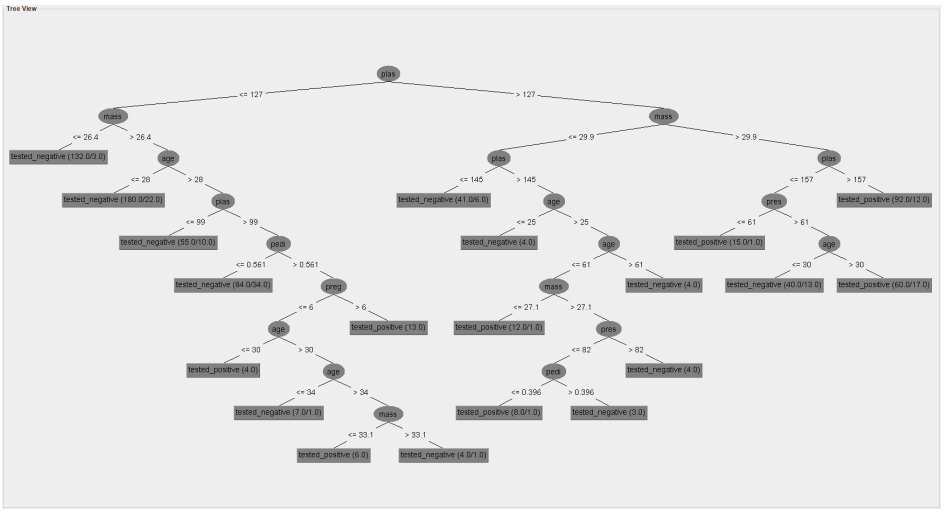


Figure 4.6: Example of a Decision Tree

A RF is composed by a number of trees **T**. For some number m, m fea-

tures are randomly selected from the feature vector. The subset of variables is used to train a DT.
According to Breiman implementation of RFs, m should be that $\ll$ than the number of features.
We adopted the convention in [11] that $m \leq \log_2 F+1$ and used an ensemble of 500 trees.
Running a RF,when a new input is entered into the system (a test sample), it is run down all of the trees, each of which classifies it in a "hard"way (see 3.2): in a sense, each tree gives a vote for the current sample. The result is the average of all of the terminal nodes that are reached, giving a final *soft* classification. RFs are generally quite fast, robust classifiers, and are also used in image classification [10].
The Matlab code implemented to classify data is listed in .2.

4.4 Classification Process

Once a classifier is trained on the training set, it can be used to classify unseen data (i.e., the evaluation or testing set). The classifier function is applied to each instance of the *evaluation* feature set, which is built as the matrix described in Equation 4.11.
The output of the classifier is a vector like the one described in Equation 4.12, unless that, generally, the classification process gives a *soft* classification (see Section 3.2), which means that $-1 \leq e_i \leq 1 \; \forall i = 1, \cdots, n_i$, or $0 \leq e_i \leq 1$, depending on the definition of the classes.
The performance parameters are computed as a function of a *classification threshold*, as described in Section 3.4.2.

Chapter 5

Design of a User Study

> *"πάντων χρημάτων μέτρον', ἄνθρωπον εἶναι, 'τῶν μὲν ὄντων ὡς ἔστι, τῶν δὲ μὴ ὄντων ὡς οὐκ ἔστιν.'"*
> (man is "the measure of all things, of the existence of the things that are and the non-existence of the things that are not.")
>
> Πλάτων(Plato, Theaet. 152a)

We built a web interface to collect data originated from mitosis classification performed by humans.

5.1 Test Design

The problem of detecting mitosis can be cast as a problem of classifying image patches. In fact, most detection algorithms are based on classifiers which map an image patch to the probability that a mitosis appears at its center; once such classifier is known, the detection problem is solved by applying it on a sliding window over the input image, or to a set of candidate patches identified in a previous step. The classification task can be presented to an user through a very simple and immediate interaction mechanism: in fact, a single decision is required for each patch. In contrast, detection would require a more complicated interaction with users. For this reason, we focused on the classification subproblem. For a given sample, input is given in form of an image patch with size 100×100 pixel: such size completely contains the image of the cell, and most algorithms generally use data from a smaller window. The task proposed to the used the same as the one tackled in automatic classifcation (see Chapter 4), that is to map each patch to one of two classes:

C1: the image contains a mitosis at its center,

C0: the image does not contain a mitosis anywhere.

with no samples containing a mitosis visible off-center.

5.1.1 Dataset

The dataset is the same as the one described in Section 4.1.

5.1.2 Programming Framework

The user interface has been built in Ruby on Rails (RoR)[1], which is an open source web application framework which runs on the *Ruby* programming language [18], and allows to develop complete, dynamic pages without too much overhead [6]. RoR makes an extensive use of the concept of Convention over Configuration (CoC) which is a software design paradigm which seeks to decrease the number of decisions that developers need to make, gaining simplicity and standardization. In fact the directory structure of a RoR project is auto-generated and standardized, and also class names are conventionally mapped to identically named database tables and the fields to its columns.
The application built for this project has been depolyed on the *Heroku application platform*[2] and its online implementation is reachable at the following `url`: `http://mitosis-detection.herokuapp.com/`.

5.2 User Interface

In this section we describe the user interface built to present samples to the user and to collect the classification data.

5.2.1 Introduction

In the first stage the user receives some information about the purpose of the test and is required to give some simple information (see Figure 5.1),

[1] `http://rubyonrails.org/` [33]
[2] `https://www.heroku.com/`

summarized in:

- her/his experience, in particular:
  1. the user doesn't work in biology and new to such a problem,
  2. he is a biologist,
  3. he is an histologist, and so has direct experience in the task.
- her/his color ability, that is:
  1. he has normal color ability,
  2. he is colorblind.
- the user can optionally give a *nickname* that is recorded with the data.

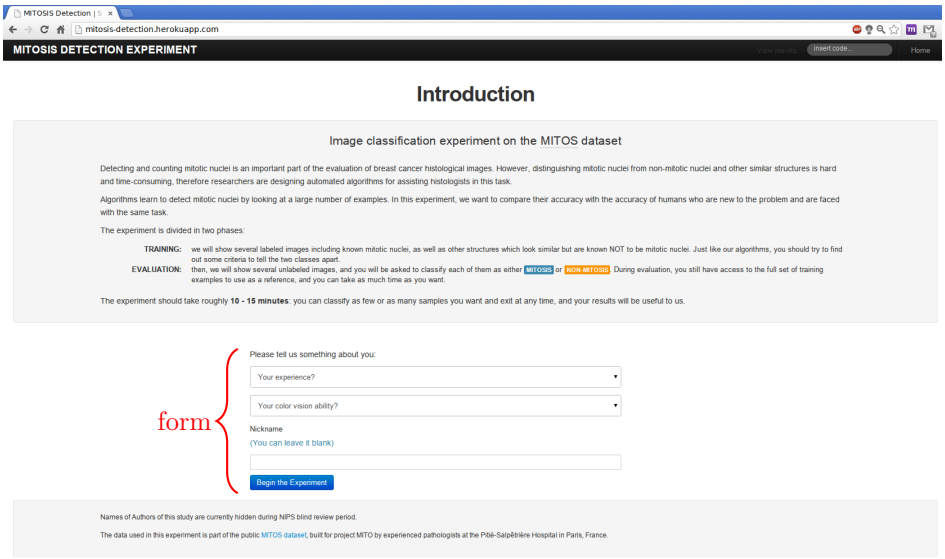


Figure 5.1: Intro page

5.2.2 Training

Once the user decides to participate, a new *detection* entity is created and linked to a unique alphanumeric string that can be used to retrieve the summary of the performance. The first step of the classification process is the *training* phase, during which the subject is shown 216 labeled **C1** samples, 216 labeled **C0** samples, and instructed to study them and devise some differentiating criteria (see Figure 5.2). The dataset is composed of

the same images as the one used for automatic classification (see Section 4.1).

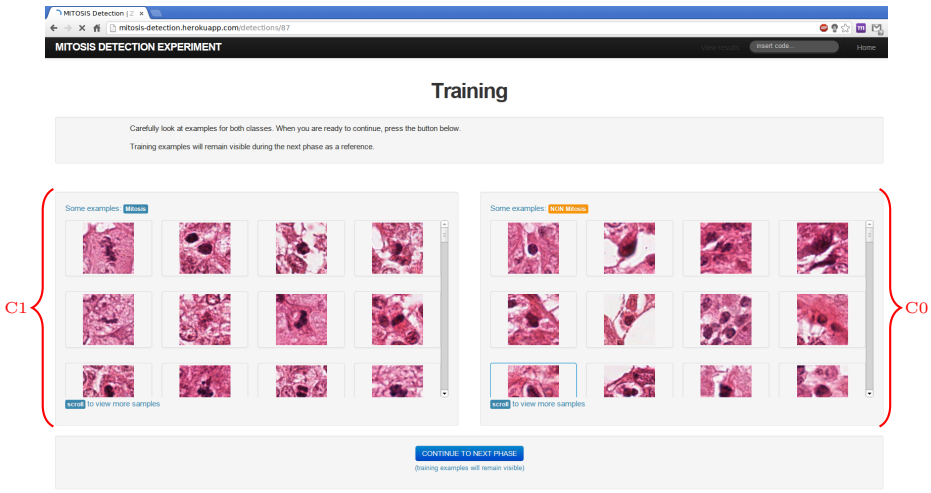


Figure 5.2: Training page

5.2.3 Evaluation

During evaluation, the subject is presented with one evaluation sample at a time (randomly chosen among unseen ones), and asked to provide a classification as one of:

- *definitely mitosis*: $p(C1) = 1.0$,
- *probably mitosis*: $p(C1) = 0.75$,
- *probably non-mitosis*: $p(C1) = 0.25$,
- *definitely non-mitosis*: $p(C1) = 0.0$,

During the evaluation phase, the whole training set remains visible for reference (see Figure 5.3). The number of classification options has been chosen so that the user is led to make a commitment over the type of current image: towards *C0* or towards *C1*.

A number of design decisions are taken in order to balance the trade-off between test fairness and subject engagement. Most importantly, the user is given immediate feedback as to whether the last decision was correct or wrong (see Figure 5.4): on one hand, this encourages continuous learning while the evaluation is taken and makes users much more willing to improve

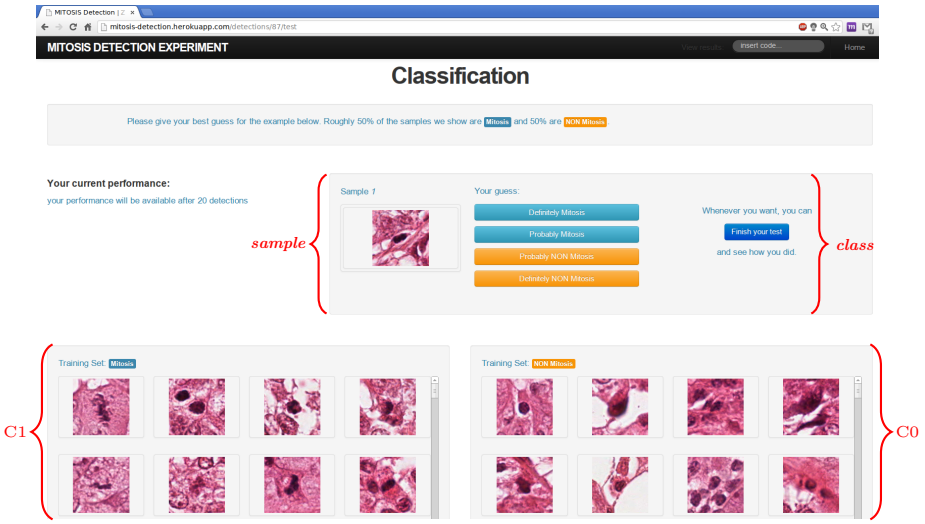


Figure 5.3: Evaluation page

and fine-tune their strategies; on the other hand, subjects can count on a growing training set, which gives them an unfair advantage over algorithms.

In addition, subjects are allowed to finish the test at any time, the ones who reach a minimum of 20 classifications are shown their current average accuracy, as shown in Figure 5.5.

5.2.4 Comments

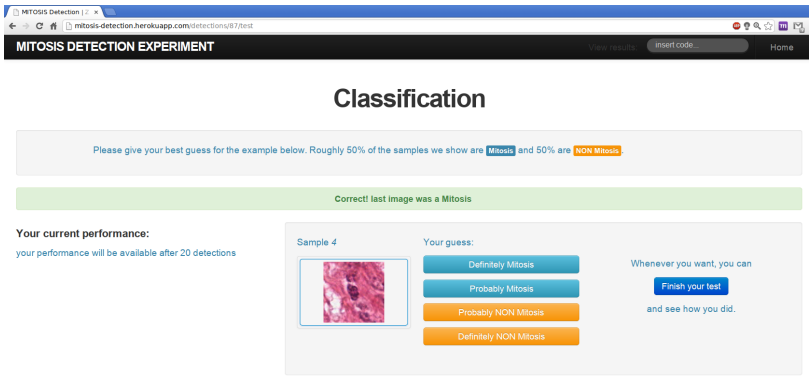
After concluding the classifications, the subject can write his opinions about the classification criteria that he devised during the process (see Figure 5.6).

5.2.5 Performances

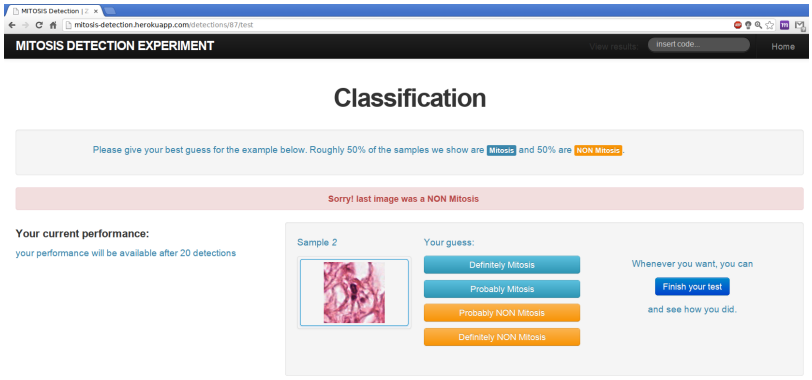
Finally, the user can review his overall performance, viewing his *confusion matrix*, his *accuracy*, *sensitivity* and *specificity* (as described in Section 3.4). The results page is shown in Figure 5.7.

5.3 Data collection

In a not directly reachable web-page, it is possible to view and download all the data collected by the site. A table, shown in Figure 5.8, reports the main information of all the concluded classifications.



(a) positive Feedback



(b) negative Feedback

Figure 5.4: Examples of classification feedback

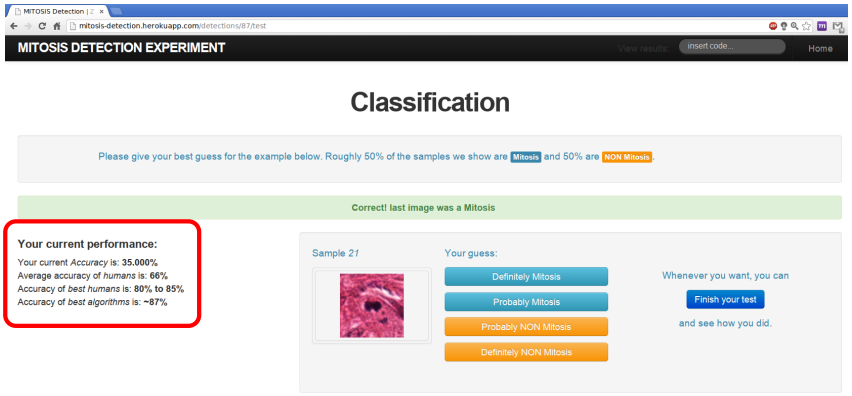


Figure 5.5: Current performance

It is possible to download (see Figure 5.9) two `.csv` files. One (`images.csv`) summarizes the dataset, for each image patch it gives:

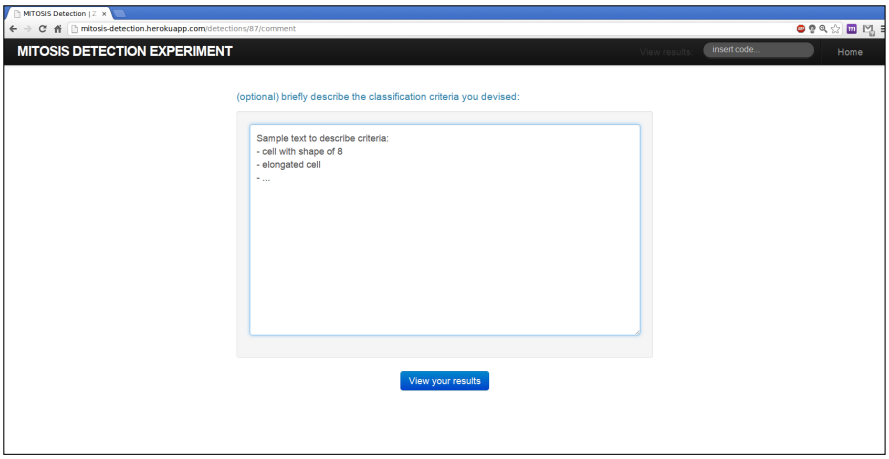


Figure 5.6: Comment page

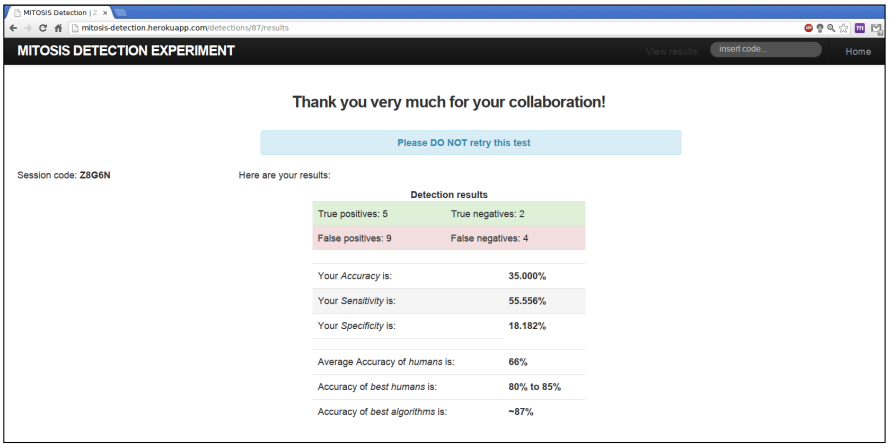


Figure 5.7: user results page

- *id*: a unique number identifying each image,
- *image*: the name of the image from which the patch has been taken,
- *coordinates*: the (x, y) coordinates of the center in the image,
- *type*: if the image is **C1** or **C0**.

The other file (`users.csv`) summarizes the classifications. Each detection starts with a line beginning with the keyword `USER`. The first line of a detection reports the information concerning user and detection:

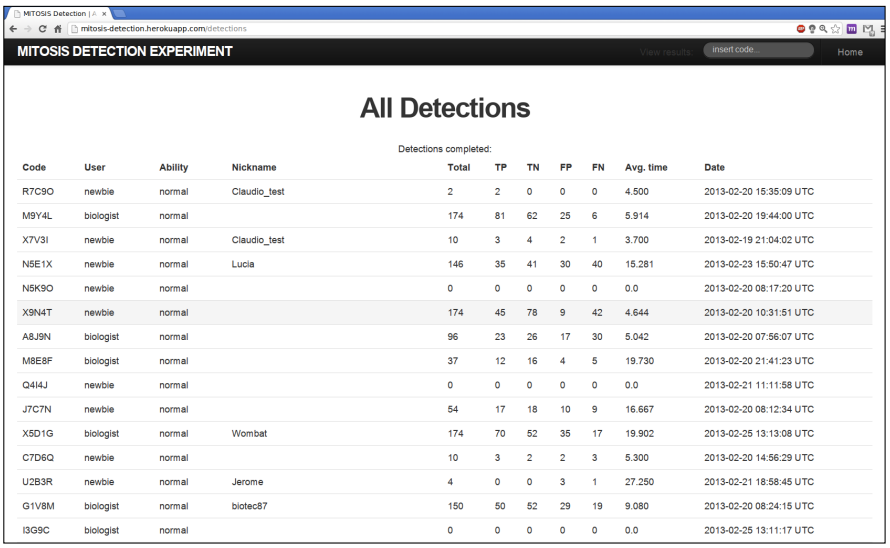
- nickname, user type and color ability of the user,
- timestamp of the detection,

- number of detections: TPs, TNs, FPs, FNs,
- *ID* of the detection.

The line concerning the classified images reports:

- *id*: the unique number identifying the patch,
- *image*: the name of the image from which the patch has been taken,
- *coordinates*: the (x, y) coordinates of the center in the image,
- *type*: if the image is **C1** or **C0**.
- *classification*: how the user classified the image: $\{0.0, 0.25, 0.75, 1.0\}$,
- *time*: how many seconds took the user to decide.

Finally it is possible to view all the comments left by the users (see Figure 5.10).



Code	User	Ability	Nickname	Total	TP	TN	FP	FN	Avg. time	Date
R7C9O	newbie	normal	Claudio_test	2	2	0	0	0	4.500	2013-02-20 15:35:09 UTC
M9Y4L	biologist	normal		174	81	62	25	6	5.914	2013-02-20 19:44:00 UTC
X7V3I	newbie	normal	Claudio_test	10	3	4	2	1	3.700	2013-02-19 21:04:02 UTC
N5E1X	newbie	normal	Lucia	146	35	41	30	40	15.281	2013-02-23 15:50:47 UTC
N5K9O	newbie	normal		0	0	0	0	0	0.0	2013-02-20 08:17:20 UTC
X9N4T	newbie	normal		174	45	78	9	42	4.644	2013-02-20 10:31:51 UTC
A8J9N	biologist	normal		96	23	26	17	30	5.042	2013-02-20 07:56:07 UTC
M8E8F	biologist	normal		37	12	16	4	5	19.730	2013-02-20 21:41:23 UTC
Q4I4J	newbie	normal		0	0	0	0	0	0.0	2013-02-21 11:11:58 UTC
J7C7N	newbie	normal		54	17	18	10	9	16.667	2013-02-20 08:12:34 UTC
X5D1G	biologist	normal	Wombat	174	70	52	35	17	19.902	2013-02-25 13:13:08 UTC
C7D6Q	newbie	normal		10	3	2	2	3	5.300	2013-02-20 14:56:29 UTC
U2B3R	newbie	normal	Jerome	4	0	0	3	1	27.250	2013-02-21 18:58:45 UTC
G1V8M	biologist	normal	biotec87	150	50	52	29	19	9.080	2013-02-20 08:24:15 UTC
I3G9C	biologist	normal		0	0	0	0	0	0.0	2013-02-25 13:11:17 UTC

Figure 5.8: Overall results page

5.4 Source Code

Some extracts of the source code of the project can be found in Appendix .3. The entire project source code is maintained at `https://github.com/Ccaccia73/tydes`.

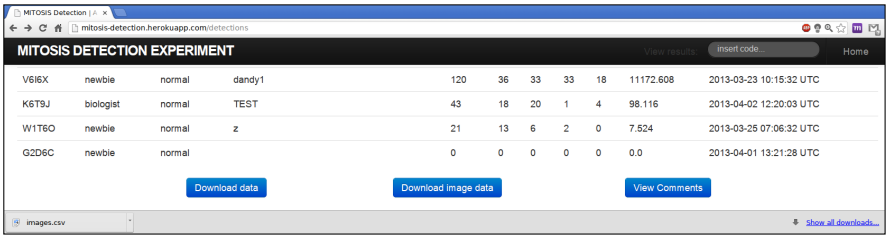


Figure 5.9: Download buttons

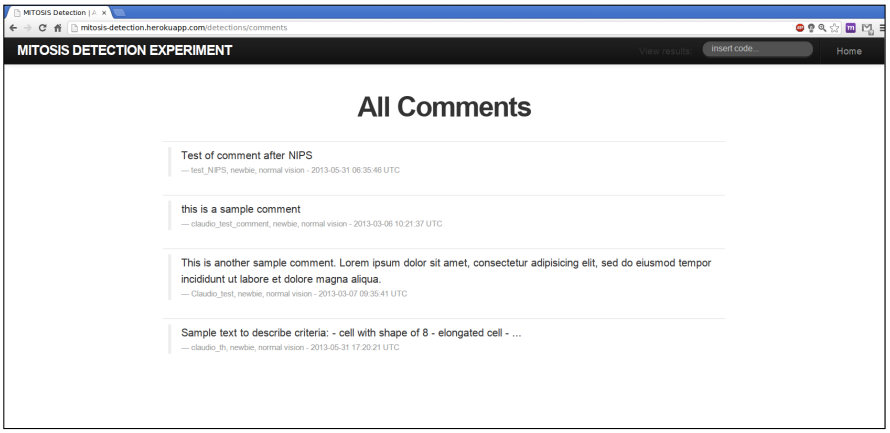


Figure 5.10: User comments

Chapter 6

Experimental Results

> *"Duo enim sunt modi cognoscendi, scilicet per argumentum et experimentum. Argumentum concludit et facit nos concedere conclusionem, sed non certificat neque removet dubitationem ut quiescat animus in intuitu veritatis, nisi eam inveniat via experientiae."*
> (There are two modes of acquiring knowledge, reasoning and experience. Reasoning guides us to a sound conclusion, but does not remove doubt from the mind until confirmed by experience.)
>
> Roger Bacon (Opus Majus part VI, ch. I)

In this section we describe the experiments made and the performance obtained from some classifiers built on the feature set described in Section 4.2, furthermore we compare those results with the performances of users who classified the images using the website described in Chapter 5.

6.1 Experimental setup

Here we describe the set of experiments that we run on the dataset. Each experiment has been executed classifying data with a Random Forest (RF) classifier and a Support Vector Machine (SVM).
We adopted some conventions to describe a specific feature-set in a short way. As described in Section 4.3, a feature matrix is composed of features placed side by side, so a complete set can be represented by a string whose characters correspond to a specific feature. We used the following nomenclature:

- M: mean value per color (Section 4.2.1),
- S: standard deviation per color (Section 4.2.1),

- d: median per color (Section 4.2.1),
- H: color histograms (Section 4.2.2),
- i: mean intensities (Section 4.2.2),
- L: LBP^{riu2} with radii 1-2-3, 8 neighbors (Section 4.2.3),
- R: LBP^{ri} with radii 1-2-3, 8 neighbors (Section 4.2.3),
- U: LBP^{u2} with radii 1-2-3, 8 neighbors (Section 4.2.3),
- V: VAR, pixel variance (Section 4.2.3).

Thus a feature set can be described by the string `MSHLV`, meaning that the above corresponding features have been concatenated. We remind here that features L, R and U are used in a mutually exclusive way.

6.2 Experiments

We run different experiments to analyze the performances of the selected classifiers and to determine which feature set is most suitable to classify our data. In this section we describe the experiments made and, for each one of them, we report the most significant results.

6.2.1 Normalization

Te first aspect that we took into account regarded *normalization* of data. Each feature in the *feature vector* can range among different values. The procedure described in the guide in [40] claims that the scaling of data is very important in order to obtain a good classification with SVMs [48]. On the other hand, RFs, as composed of DTs, should not be influenced by scaling. The general idea is to rescale the data so that each new feature z has: $\mu = 0, \sigma = 1$, using the following relations:

$$\mu = \tfrac{1}{N} \sum_{i=1}^{N} x_i \tag{6.1}$$

$$\sigma = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^{N} (x_i - \mu)^2} \tag{6.2}$$

$$z_i = \frac{x_i - \mu}{\sigma} \tag{6.3}$$

We run some test with different sets of features.
The Matlab code implemented to run *experiment 1* is listed in .3

6.2.2 Normalization: Experimental Results

We tried to classify different feature sets, starting with simple ones.

Features: `MSi`

With a simple set of features the main results are the following:

Classifier	AUC	accuracy	precision	F_1-Score	sensitivity	specificity
SVM std	0.79	74.14%	81.82%	0.71	86.21%	62.07%
SVM norm	0.74	71.26%	70.79%	0.72	70.11%	72.41%
RF std	0.80	75.86%	77.78%	0.75	79.31%	72.41%
RF norm	0.80	75.86%	78.48%	0.75	80.46%	71.26%

Table 6.1: `MSi` results

The data in Table 6.1 show that, with normalization, the RF classifier remains almost unchanged, which is to be expected, while, the SVM classifier worsens its performance. With these simple features no advantages have been obtained. The ROC curve of the classifiers is shown in Figure 6.1.

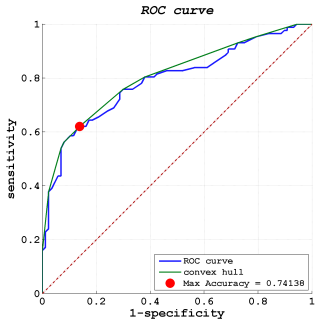
Features: `MSiHLV`

The results of applying normalization to data are different when much more features are involved. For example, in a `MSiHLV` feature-set (but same results have been found for example for `MSiHUV`), the SVM classifier is unable to find a proper classification with standard features; on the other hand, the results are interesting when normalization is applied. Similarly to the previous case (Table 6.1), the RF classifier is slightly influenced by normalization.

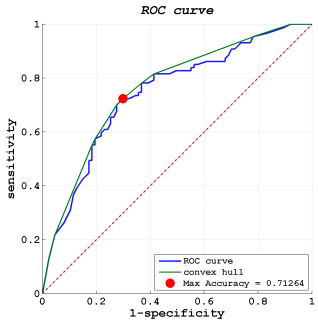
Classifier	AUC	accuracy	precision	F_1-Score	sensitivity	specificity
SVM norm	0.87	79.89%	74.07%	0.82	67.82%	91.95%
RF std	0.89	81.03%	79.35%	0.82	78.16%	83.91%
RF norm	0.90	81.61%	77.23%	0.83	73.56%	89.66%

Table 6.2: `MSiHLV` results

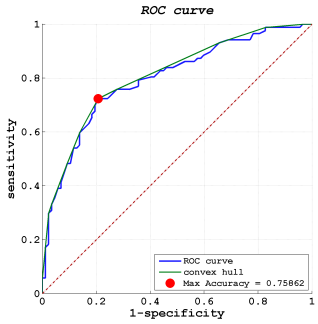
Table 6.2 shows the results. The ROC curve of the classifiers is shown in Figure 6.2.



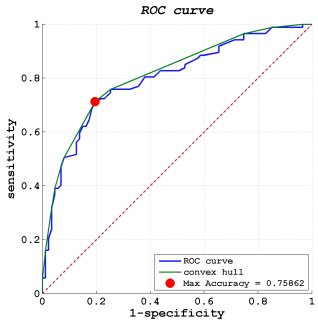
(a) SVM std - ROC



(b) SVM norm - ROC



(c) RF std - ROC



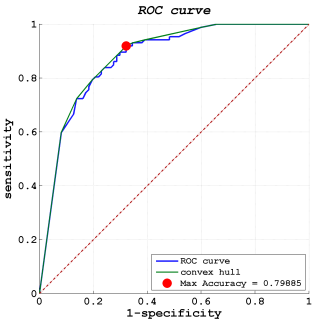
(d) RF std - ROC

Figure 6.1: ROC curves for MSi features classification

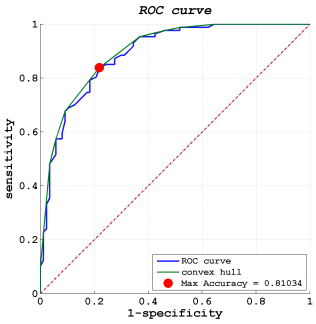
As normalization is generally considered a good practice and, with or experiments, we found advantages when we applied it to our data, we considered only a normalized dataset in all the following experiments.

6.2.3 Extended Dataset

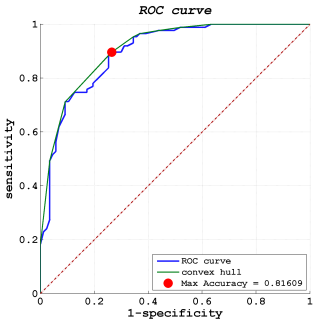
In this experiment we analyzed the effect of considering the extended dataset, as described in Section 4.1.2. Rotated and mirrored images should provide some information for all the features that are orientation dependent, in all the other cases, the added elements are just replicates of already present instances. Also in this case we run test with different sets of features.
The Matlab code implemented to run *experiment 2* is listed in .3



(a) SVM norm - ROC



(b) RF std - ROC



(c) RF std - ROC

Figure 6.2: ROC curves for `MSiHLV` features classification

6.2.4 Extended Dataset: Experimental Results

We considered three different ways of extending our dataset, resulting in four different experiments:

- no dataset is extended (abbreviated with *default*),
- the *train* dataset is extended (abbreviated with *ext-T*),
- the *evaluation* dataset is extended (abbreviated with *ext-E*),
- both dataset are extended (abbreviated with *ext-A*).

We expect some advantages in extending the dataset when orientation-dependent features are involved. On the other hand, growing the dataset too much could bring in much more noise than useful information, resulting in a worse performance of the classifier. When the evaluation dataset is extended, the classification value of an image is the average of the classification of the derived images (see Section 4.1.2). We report here the most significant experiments.

Features: `MSiHU` - classifier: SVM

We applied our SVM classifier to the feature-set coded `MSiHU`, please note that feature `U` is orientation dependent. The results are the following:

Classifier	AUC	accuracy	precision	F_1-Score	sensitivity	specificity
SVM def.	0.86	79.89%	80.23%	0.80	80.46%	79.31%
SVM ext-T	0.87	81.03%	80.00%	0.81	79.31%	82.76%
SVM ext-E	0.87	81.61%	85.71%	0.80	87.36%	75.86%
SVM ext-A	0.88	81.61%	80.22%	0.82	79.31%	83.91%

Table 6.3: `MSiHU` results (SVM)

More in detail, the number of classified images at optimal threshold is the following:

Classifier	TP	FN	TN	FP
SVM def.	69	18	70	17
SVM ext-T	72	15	69	18
SVM ext-E	66	21	76	11
SVM ext-A	73	14	69	18

Table 6.4: `MSiHU` classified images(SVM)

Looking at Table 6.4, the number of TPs shows an interesting trend. Extending the *train* dataset improves the classification performance of TPs, worsening the number of TNs of just a unit. It seems that an extended dataset brings some more information. When the only *evaluation* dataset is extended, the number of TPs lowers considerably, meaning that the training set lacks some information to classify mitoses. On the other hand, the number of TNs is at top. The overall best performance is found when all the datasets are extended. The ROC curves of this classification is shown in Figure 6.3.

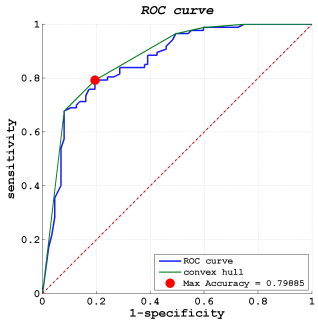
Features: `MSiHU` - classifier: RF

We classified the same dataset using our RF classifier, with the following results:

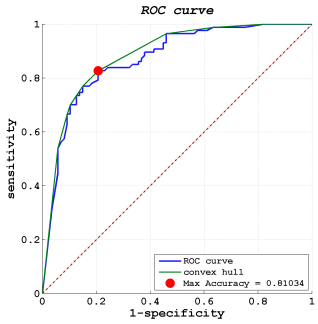
Classifier	AUC	accuracy	precision	F_1-Score	sensitivity	specificity
RF def.	0.85	78.74%	81.25%	0.78	82.76%	74.71%
RF ext-T	0.86	79.31%	76.29%	0.80	73.56%	85.06%
RF ext-E	0.86	78.16%	74.75%	0.80	71.26%	85.06%
RF ext-A	0.86	77.59%	74.00%	0.79	70.11%	85.06%

Table 6.5: `MSiHU` results (RF)

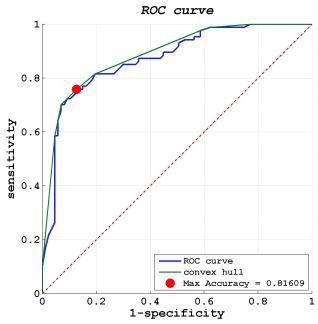
More in detail, the number of classified images at optimal threshold is the following:



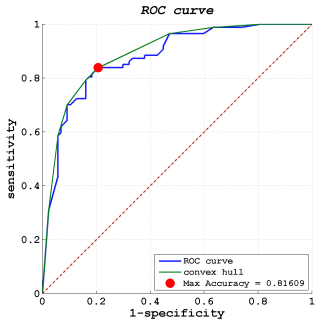
(a) SVM default - ROC



(b) SVM ext-T - ROC



(c) SVM ext-E - ROC



(d) SVM ext-A - ROC

Figure 6.3: ROC curves for MSiHU features - SVM classification

Classifier	TP	FN	TN	FP
RF def.	65	22	72	15
RF ext-T	74	13	64	23
RF ext-E	74	13	62	25
RF ext-A	74	13	61	26

Table 6.6: MSiHU classified images (RF)

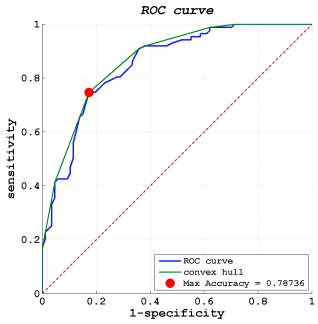
In this classification the extended dataset brings an improvement of the detection of mitoses, while worsens the detection of non-mitoses. The ROC curves of this classification is shown in Figure 6.4.

Features: MSiHR - classifier: RF

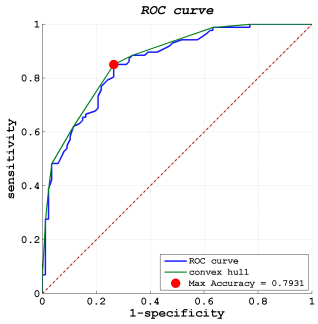
We applied our RF classifier to the feature-set coded MSiHR, which is an orientation independent dataset. The results are the following:

More in detail, the number of classified images at optimal threshold is the following:

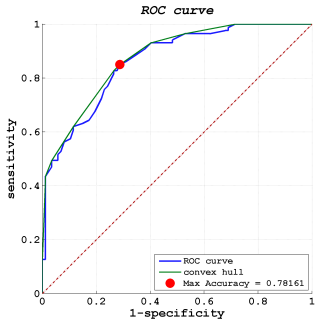
Looking at Table 6.8, the most interesting trend concerns the number of



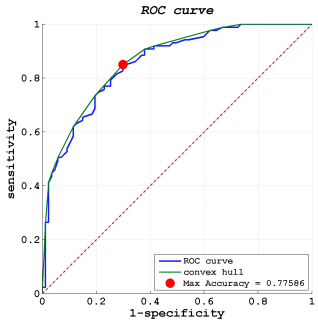
(a) RF default - ROC



(b) RF ext-T - ROC



(c) RF ext-E - ROC



(d) RF ext-A - ROC

Figure 6.4: ROC curves for MSiHU features - RF classification

Classifier	AUC	accuracy	precision	F_1-Score	sensitivity	specificity
RF def.	0.89	82.18%	84.15%	0.82	85.06%	79.31%
RF ext-T	0.90	81.61%	83.95%	0.81	85.06%	78.16%
RF ext-E	0.89	81.61%	85.71%	0.80	87.36%	75.86%
RF ext-A	0.90	81.03%	85.53%	0.80	87.36%	74.71%

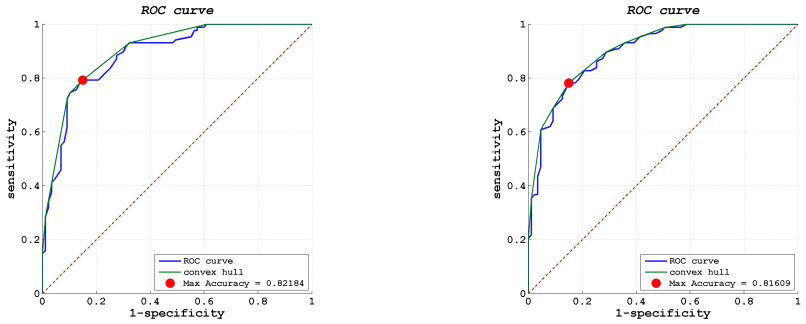
Table 6.7: MSiHR results (RF)

Classifier	TP	FN	TN	FP
RF def.	69	18	74	13
RF ext-T	68	19	74	13
RF ext-E	66	21	76	11
RF ext-A	65	22	76	11

Table 6.8: MSiHR classified images (RF)

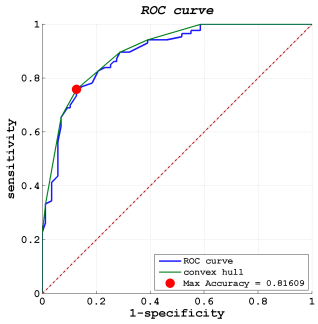
TPs. In fact, using an extended dataset with no rotation dependent features brings no advantages, instead reduces the number of detected mitoses. In a sense, there is more noise than useful information. On the other hand, it is always visible the fact that TNs are better detected with extended datasets. The ROC curves of this classification is shown in Figure 6.5.

In the following examples we considered extended datasets, unless ex-

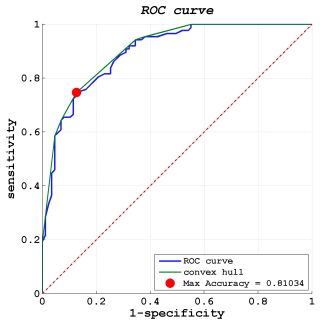


(a) RF default - ROC

(b) RF ext-T - ROC



(c) RF ext-E - ROC



(d) RF ext-A - ROC

Figure 6.5: ROC curves for MSiHR features - RF classification

plicitly specified.

6.2.5 Best Feature Combinations

In this experiment we looked for the best combination of features, so we considered all the features described in 4.2. Having n features, maybe multi-component, they can be combined in $2^n - 1$ ways. As the texture features (see Section 4.2.3, in particular Equation 4.8) are mutually exclusive, we run three different experiments, one for each texture feature set.
The Matlab code implemented to run *experiment 3* is listed in .3

6.2.6 Best Feature Combinations: Experimental Results

Having run all possible combinations of features, we report here the best performance found, divided for RF and SVM classifiers.

Best Performances - classifier: SVM

The four feature sets which gave best results, when classified with SVM are described in Table 6.9. The detailed number of classified images, at the optimal classification threshold is shown in Table 6.10.

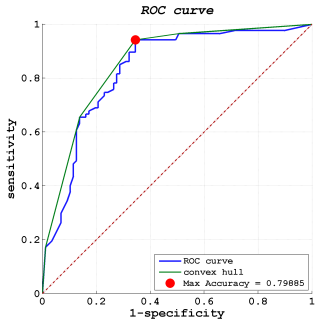
Classifier	AUC	accuracy	precision	F_1-Score	sensitivity	specificity
SVM - H	0.83	79.89%	73.21%	0.82	65.52%	94.25%
SVM - MSiVH	0.85	78.74%	87.88%	0.76	90.80%	66.67%
SVM - SiU	0.88	84.48%	81.91%	0.85	80.46%	88.51%
SVM - SiVHU	0.88	80.46%	77.32%	0.82	74.71%	86.21%

Table 6.9: Best SVM results

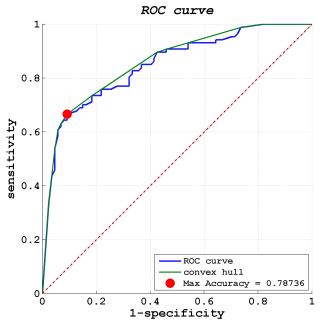
Classifier	TP	FN	TN	FP
SVM - H	82	5	57	30
SVM - MSiVH	58	29	79	8
SVM - SiU	77	10	70	17
SVM - SiVHU	75	12	65	22

Table 6.10: Best SVM results - classified images

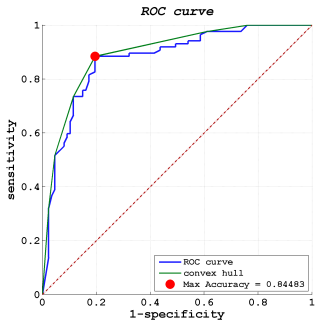
The ROC curves of these classifications are shown in Figure 6.6.



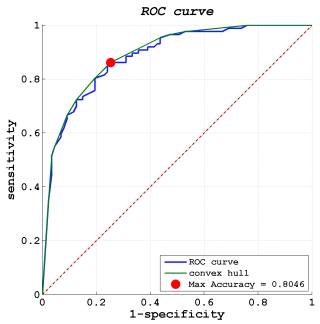
(a) SVM H - ROC



(b) SVM MSiVH - ROC



(c) SVM SiU - ROC



(d) SVM SiVHU - ROC

Figure 6.6: ROC curves for best feature-set - SVM classification

Best Performances - classifier: RF

The four feature sets which gave best results, when classified with RF are described in Table 6.11. The detailed number of classified images, at the optimal classification threshold is shown in Table 6.12.

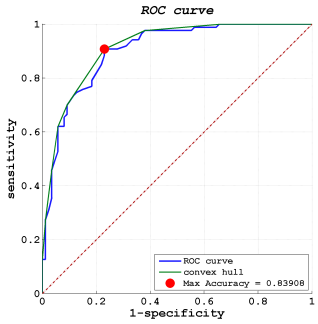
Classifier	AUC	accuracy	precision	F_1-Score	sensitivity	specificity
RF - iVHL	0.90	83.91%	79.80%	0.85	77.01%	90.80%
RF - MSHL	0.89	81.03%	89.71%	0.79	91.95%	70.11%
RF - MSiVHR	0.91	83.91%	81.05%	0.85	79.31%	88.51%
RF - SHL	0.89	80.46%	73.04%	0.83	64.37%	96.55%

Table 6.11: Best RF results

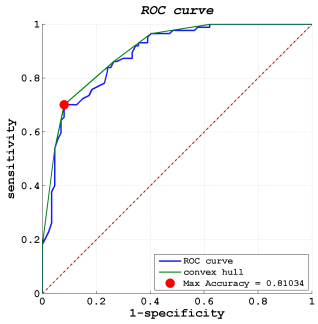
Classifier	TP	FN	TN	FP
RF - iVHL	79	8	67	20
RF - MSHL	61	26	80	7
RF - MSiVHR	77	10	69	18
RF - SHL	84	3	56	31

Table 6.12: Best RF results - classified images

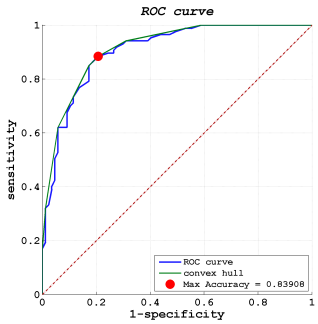
The ROC curves of these classifications are shown in Figure 6.7.



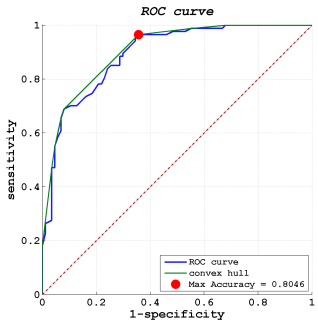
(a) RF iVHL - ROC



(b) RF MSHL - ROC



(c) RF MSiVHR - ROC

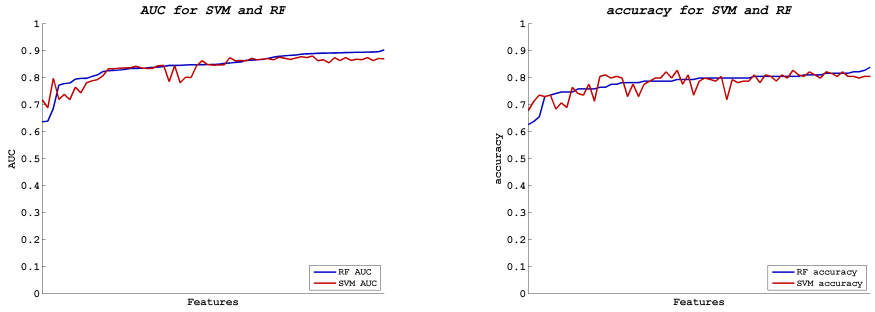


(d) RF SHL - ROC

Figure 6.7: ROC curves for best feature-set - RF classification

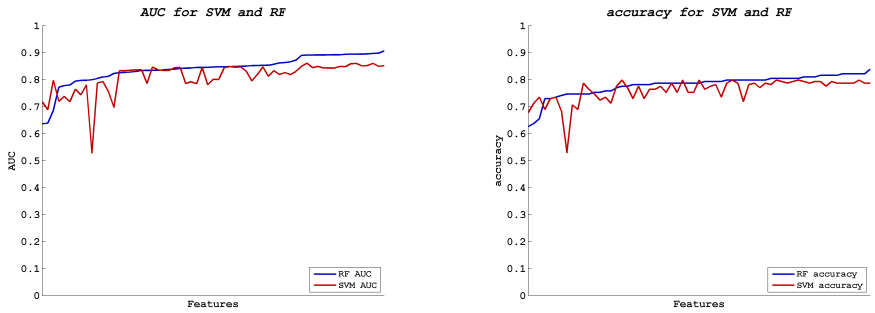
Comparison between classifiers

Having classified our evaluation set with a considerable number of different features, we could try to answer the question whether one of the two considered classifiers outperforms the other. We used, as a metric for our analysis, the *AUC* and the *accuracy*, and considered the three different complete feature sets: `MSiVHU`, `MSiVHR` and `MSiVHL`. Having tried all the possible combinations, the sorted the results obtained by the RF classifier in ascending order and plotted the results obtained by the SVM classifier in the same sequence.
The results are showed on Figures 6.8, 6.9 and 6.10.



(a) MSiVHL - AUC (b) MSiVHL - accuracy

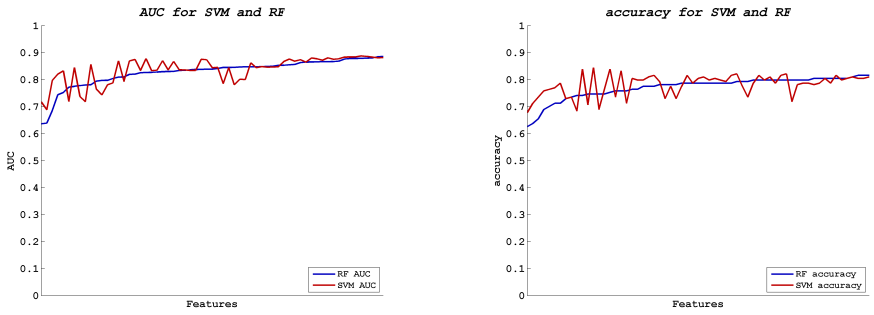
Figure 6.8: Features MSiVHL - overall performances



(a) MSiVHR - AUC (b) MSiVHR - accuracy

Figure 6.9: Features MSiVHR - overall performances

While it is generally true that the mean performance of the RF classifier is better (in terms of *AUC* and *accuracy*) than the one given by the SVM classifier, it is not possible to say that RF outperforms SVM. There are many cases in which the SVM performance turns out to be better, in particular



(a) MSiVHU - AUC

(b) MSiVHU - accuracy

Figure 6.10: Features MSiVHU - overall performances

when the LBP^{u}2 feature is involved (see Figure 6.10). Nevertheless, the best RF performance is better than SVM's.

6.2.7 Dataset Dimension

In this experiment we analyzed the effect of the size of the training dataset on the classification performance, in term of selected instances. To achieve this goal, we repeatedly selected random subsets of the extended dataset and applied our classifier. To avoid the dependence on the specific selected subset, we run many different experiments with randomly chosen subsets with the same size and then we averaged the results.
The Matlab code implemented to run *experiment 4* is listed in .3

6.2.8 Dataset Dimension: Experimental Results

Considering fractions of the *training* dataset ranging 1% →100%, we performed the classification on the whole *evaluation* set having trained the classifier o a randomly selected subset. In order to reduce the risk of biases due to a specific subset, we made different trials, selecting at each time the training dataset. We used the following empirical rule to decide the number of trials in function of the subset size:

$$\frac{\text{subset-size}}{\text{trial}} \cdot (\sharp \text{ of trials}) \approx 3 \tag{6.4}$$

Equation 6.4 brings to the number of trials illustrated in Figure 6.11.

We run experiments on different sets of features that performed well in previous tests (not necessarily the best ones): `iVHL`, `MiVHU` and `MSVHR`.

We initially used used both of our classifiers. However, it emerged that, with some combination of instances, the SVM was unable to find a proper solution. For this reason we preferred to focus on the performances of the RF classifier, which contextually turned out to be more robust.
As usual we considered, as a metric of performance, AUC and *accuracy*.

Classifier: RF - Features `iVHL`

The results are shown in Figure 6.12, where the continuous line represents the average of the performance, and the $*$represent the single classification result.

It is apparent that, on one side, the performance grows with the subset size, and on the other side the variance of the performance reduces. Once the 20% of the dataset size is reached, the average performance remains steady, but it is necessary to reach about the 50% of the dataset to have small variability of the data.

Classifier: RF - Features `MiVHU`

Figure 6.13, shows the results with `MiVHU` feature set.

The results are similar to the ones shown in Figure 6.13. In this case, at

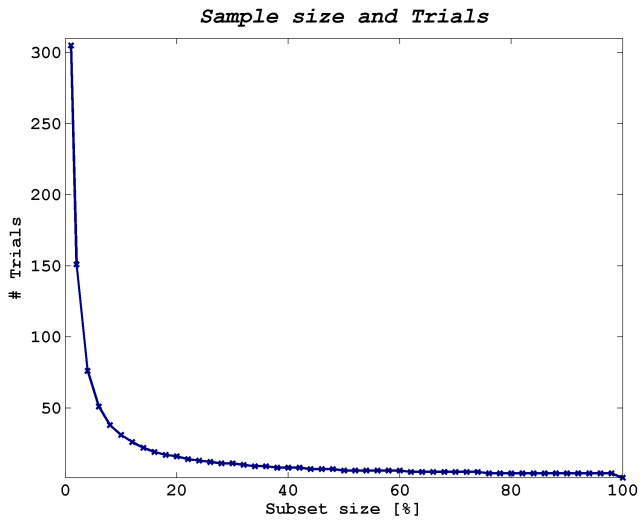
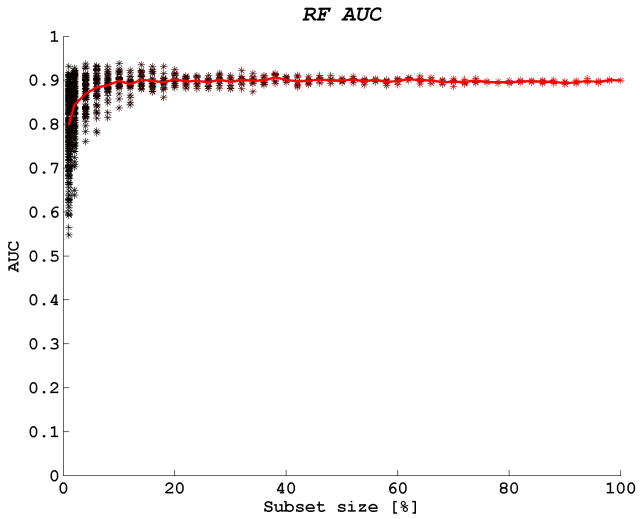
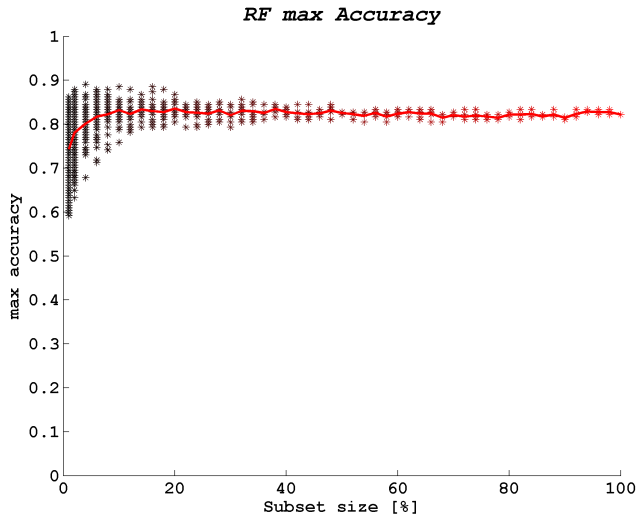


Figure 6.11: Subset size and number of trials



(a) RF iVHL - AUC



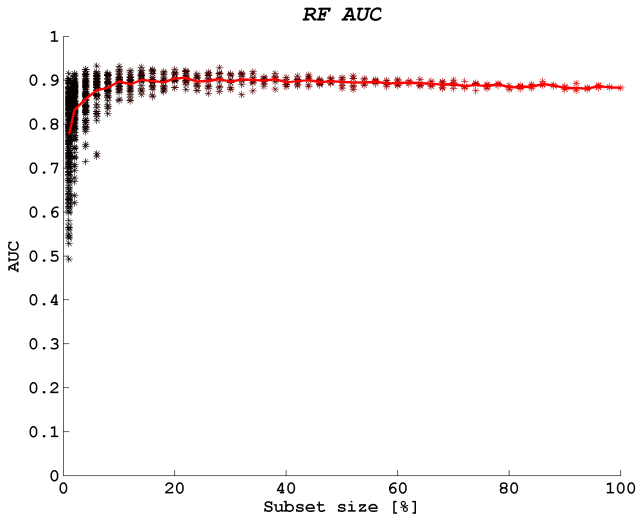
(b) RF iVHL - accuracy

Figure 6.12: Features iVHL - sample size

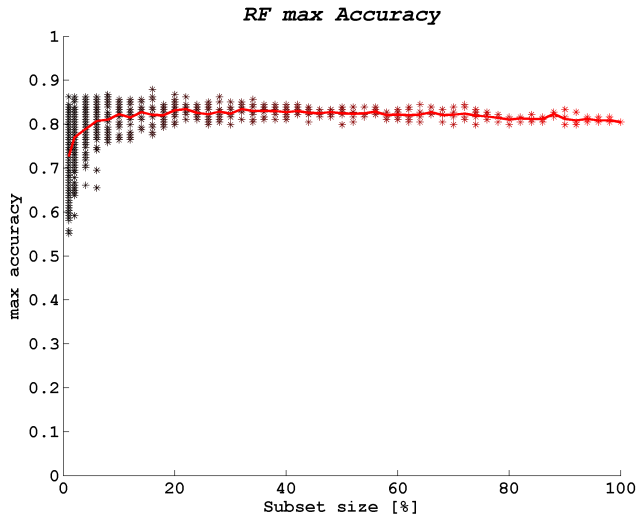
20% of the dataset size a maximum of the AUC performance is reached, even if the negative slope of subsequent data is negligible (see Figure 6.13(a)).

Classifier: RF - Features `MSVHR`

Figure 6.14, shows the results with `MSVHR` feature set.



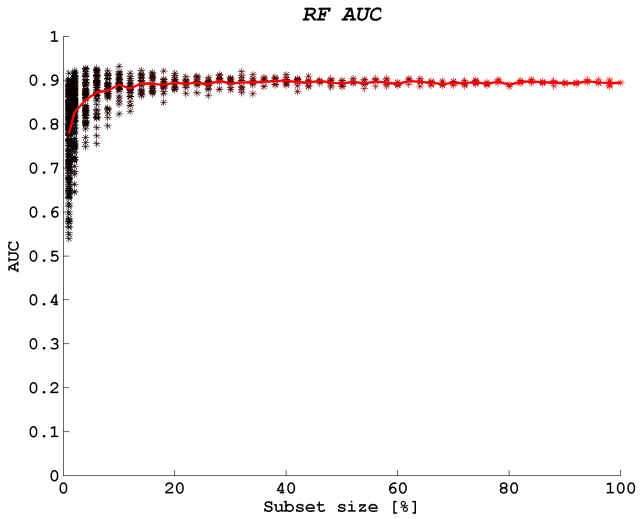
(a) RF MiVHU - AUC



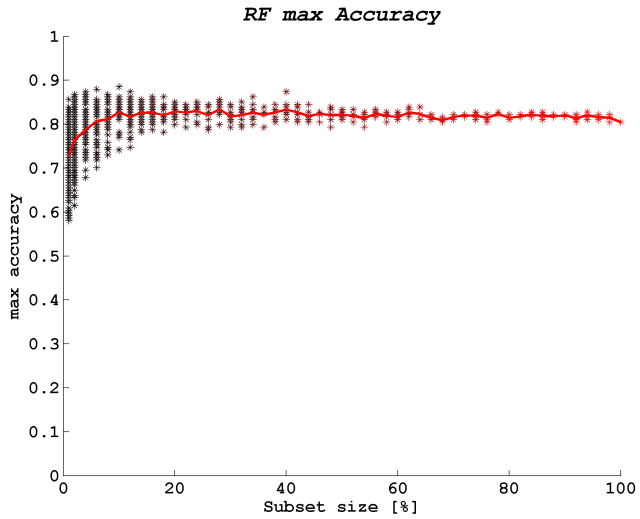
(b) RF MiVHU - accuracy

Figure 6.13: Features MiVHU - sample size

The results are similar to the previous ones: in this case the maximum AUC performance is reached even at lower subset dimension, and then a steady state is maintained, while a negligible variability is reached at about 40% of the dataset (see Figure 6.14(a)). In this experiment the variability of *accuracy* appears to be accentuated (see Figure 6.14(b)).



(a) RF MSVHR - AUC



(b) RF MSVHR - accuracy

Figure 6.14: Features MSVHR - sample size

6.2.9 SVM parameters

In the experiments above we noted that SVMs are more sensible to the selected features and to the size of the dataset, meaning that, in some cases, SVM performs poorly on data. So we decided to analyze in a deeper way if some of the configuration options of the classifier (i.e. kernel type [Section 4.3.1], degree in kernel function, etc.) could improve the performance.

The Matlab code implemented to run *experiment 5* is listed in .3

6.2.10 SVM parameters: Experimental results

We tried different configuration parameters for the SVM classifier on the feature set `MSiHLV`, of which SVM performed poorly: $AUC = 0.531$ and $accuracy = 0.58$.
We changed the allowed parameters in the *libSVM* implementation that we used (see Section 4.3.1):

- *Kernel type*: RBFs (default) or sigmoidal,
- *degree*: from 3 (default) up to 7.

As we didn't notice any particular benefit in modifying the parameters above (i.e. the performances remained identical), in the following experiments we continued using the default parameters. This experiments confirmed that, for our classification problem, RFs turned out to be more robust.

6.2.11 Principal Component Analysis

Even when considering only simple features, the feature space can reach high dimensions (i.e. a lot of components). It is common experience, in many classification problems, that when the dimensionality increases, the volume of the space increases so fast that the meaningful data become sparse in a substrate of noise.
This situation is often referred to as the *curse of dimensionality* [7]. In ML problems that involve learning from a finite number of data samples in a high-dimensional feature space, with each feature having a number of possible values, an enormous amount of training data are required to ensure that there are several samples with each combination of values. With a fixed number of training samples, the predictive power reduces as the dimensionality increases, and this is known as the *Hughes effect* [68].
Principal Component Analysis (PCA) is a way to reduce the dimensionality of the dataset [47]. PCA is a mathematical procedure that uses an orthogonal transformation (SVD transformation) to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called *principal components*. This transformation is defined in such a way that the components are sorted in descending magnitude of *explained*

variance: that is, the first principal component has the largest possible variance (accounts for as much of the variability in the data as possible), and each succeeding component, has the highest variance possible under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding components. PCA is sensitive to the relative scaling of the original variables, and so we applied it on normalized data [76]. In our experiments we selected some datasets with high dimension feature spaces, applied PCA and classified the resulting components adding one component at a time, with the aim to observe the classification performances in relation to the number of components and the percentage of explained variance.
The Matlab code implemented to run *experiment 6* is listed in .3

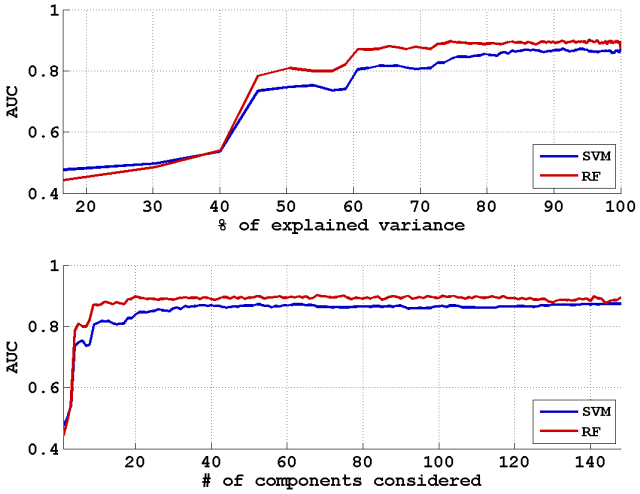
6.2.12 PCA: Experimental results

The Matlab function `pca` returns the principal component coefficients for a data matrix whose rows correspond to observations and columns correspond to variables (i.e. the training feature matrix). It returns a coefficient matrix. Each column of that matrix contains coefficients for one principal component, and the columns are in descending order of component variance. By default, `pca` uses the Singular Value Decomposition (SVD) algorithm. It also returns the percentage of the total variance explained by each principal component.
For our analysis we selected two sets of features with many components: `MSidHLV` (with 148 elements) and `MSidHUV` (with 292 elements). As in previous experiments, we used AUC and *accuracy* as measures of performance.

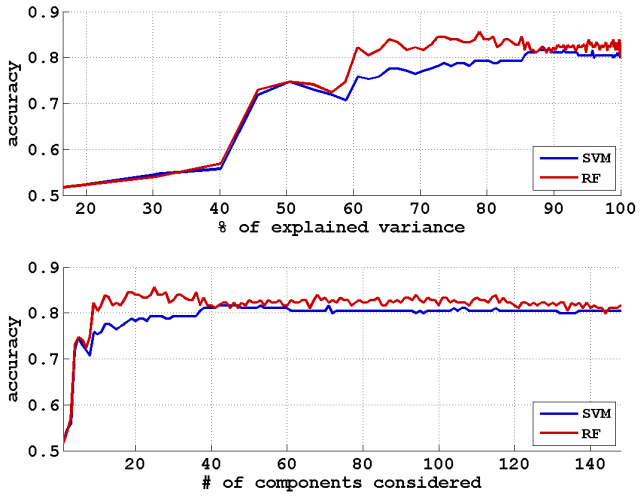
PCA: `MSidHLV`

Figure 6.15 shows the results of PCA applied to `MSidHLV` feature set. We plotted the results as function of percentage of explained variance and number of components considered.

The results show a generally better performance of RF than SVM. In terms of AUC (see Figure 6.15(a)), the RF classification reaches a maximum at about 74% of explained variance (20 components), and then it remains almost stable with small fluctuations due to noise. The SVM classification shows similar behavior.
In terms of *accuracy* (see Figure 6.15(b)) the RF classification reaches a maximum at about 79% of explained variance (27 components) and then



(a) PCA MSidHLV - AUC



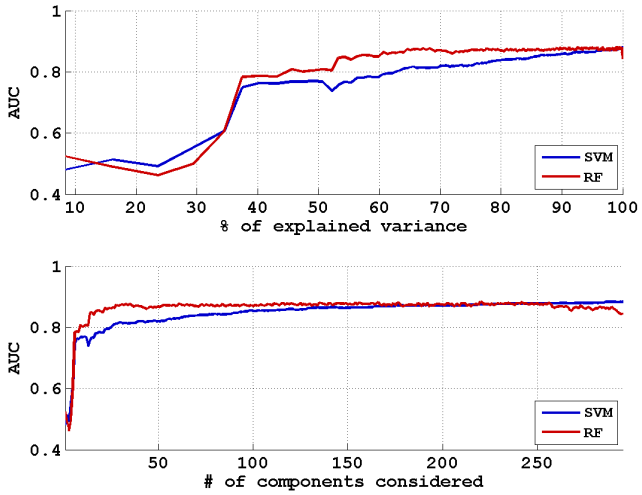
(b) PCA MSidHLV - accuracy

Figure 6.15: Features MSidHLV - Principal Component Analysis

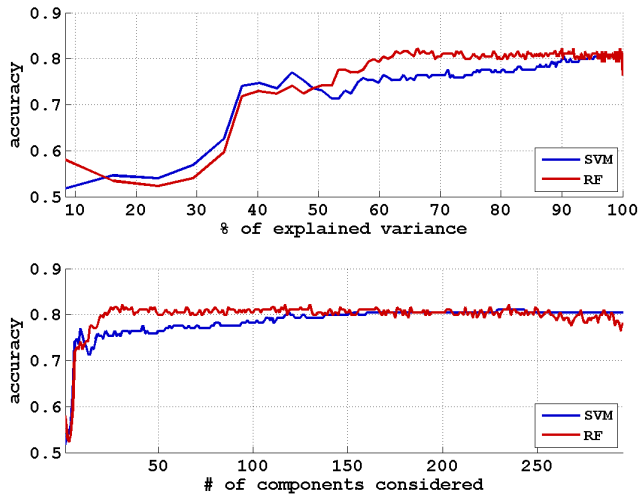
starts decreasing moderately. SVM shows a maximum at about 80% (43 components): the overall behavior appears to be steadier.

PCA: `MSidHUV`

Figure 6.16 shows the results of PCA applied to `MSidHUV` feature set. Also in this case, we plotted the results as function of percentage of explained variance and number of components considered.



(a) PCA MSidHUV - AUC



(b) PCA MSidULV - accuracy

Figure 6.16: Features MSidHUV - Principal Component Analysis

The results show a generally better performance of RF than SVM, even if, at high number of components (> 250), SVM still find information useful

fpr classification and improves performance, while RF starts decreasing. It appears that, with a high number of components, RF becomes more sensible and the performance becomes more noisy. In terms of AUC (see Figure 6.16(a)), the RF classification reaches a maximum at about 69% of explained variance (33 components), and then it decreases with fluctuations due to noise induced by further components. The SVM classification appears more stable and contiunes increasing.
In terms of *accuracy* (see Figure 6.16(b)) the RF classification reaches a maximum at about 66% of explained variance (26 components) and then starts decreasing. SVM shows a local maximum at about 45% (just 12 components) and then has an important dagradation of performance up to about 54% of explained variance. Then it starts increasing again up to the end.

6.2.13 Size of the Image Patch

In all the experiments above, we considered the size of each image patch, on which to compute features, to be 100×100 pixels. We wanted to analyze if there is a smaller sub-image that includes all the meaningful information, while the boundary contains essentially background (i.e. noise for the purposes of classification), while an excessively small patch would not allow to classify data.
So we run experiments considering, form time to time, bigger portions of the images and analyzed the classification performances.
The Matlab code implemented to run *experiment 7* is listed in .3

6.2.14 Image Size: Experimental Results

We tried different image sizes, starting from 10px up to 100px, with a step of 10px. We calculated different feature sets on the images and finally we performed classifications. Even if it appeared to be quite hard to find a "best image size", we could draw some considerations about the relation among image size and some of the features selected.
Here we report the most interesting ones, focusing in particular on the following features:

- Color Histograms (H),
- LBP^{riu2} (L),
- VAR (V).

In this case we used AUC as a measure of performance.

Image Size: SVM Classifier

By ordering the SVM classification results in ascending order for cumulate AUC along feature size, we could plot the behavior as reported in Figure 6.17 and in Figure 6.18

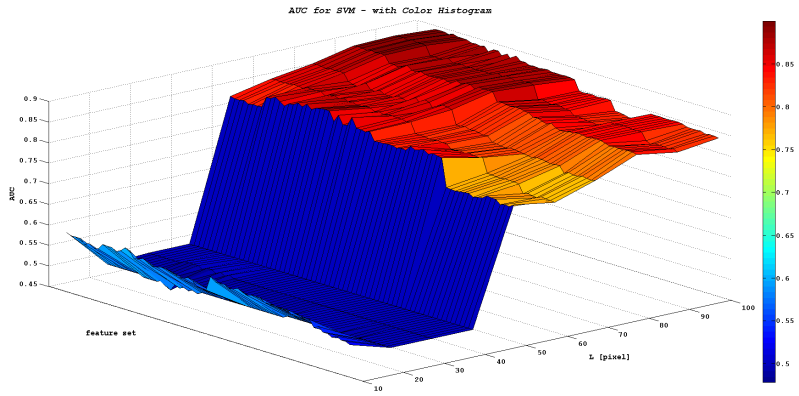


Figure 6.17: Image size and sets with 'H' feature - SVM classifier

Ordering the results we found that **all** the feature sets with the `H` elements have the same behavior. Figure 6.17 shows that SVM is unable to classify any dataset containing the `H` feature up to image size of 40px. Please note that an AUC of 0.5 means "random classification". After 50px the performance increases abruptly to high values.

Among the other feature set (i.e. the ones not containing `H`), it emerged that the ones containing the `LV` combination have the best performance. Figure 6.18 shows that those sets have a maximum generally at about 40px. With higher image size the performance slightly decreases.

Image Size: RF Classifier

We performed the same operations on the results coming from our RF classifier and found similar results. In this case, ordering in ascending order the result for mean performance over image size, we found that **all** the feature set containing the `HLV` components showed good results and also performed in the same way (see Figure 6.19).

Figure 6.19, similarly to Figure 6.18, shows a general maximum at about 40px.

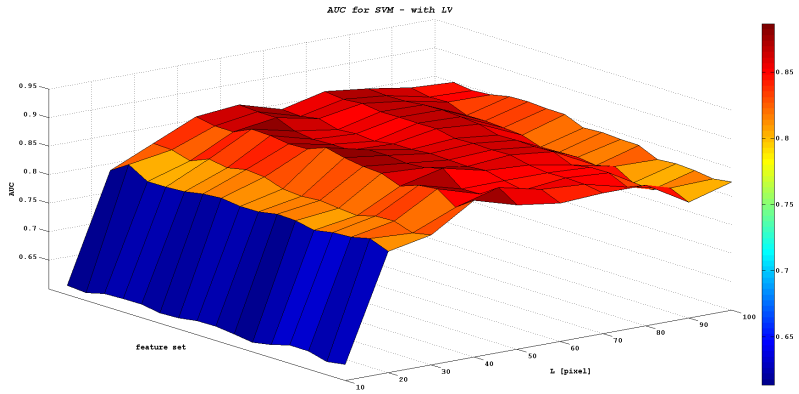


Figure 6.18: Image size and sets with 'LV' features - SVM classifier

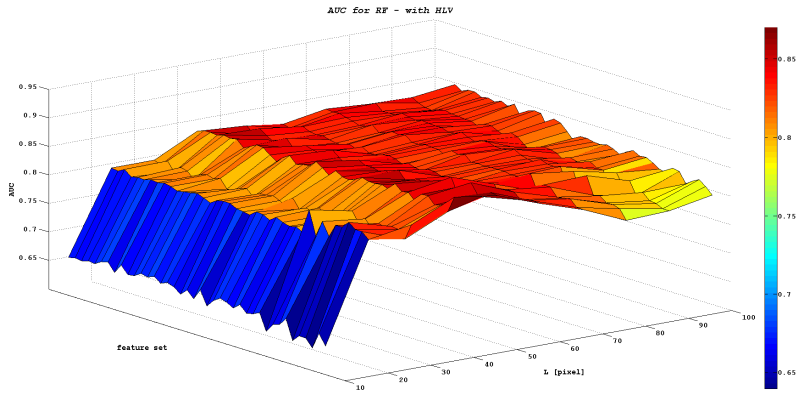


Figure 6.19: Image size and sets with 'HLV' features - RF classifier

6.3 Accuracy of Humans

6.4 Comparison

(rif. paper)

Chapter 7

Conclusions

"Quote 7"

Author 7

Bibliography

[1] National Comprehensive Cancer Network (NCCN) guidelines Breast Cancer Version 2.2011. `http://www.nccn.org/professionals/physician_gls/pdf/breast.pdf`, 2011.

[2] The icpr 2012 mitosis detection challenge, and mitos dataset. http://ipal.cnrs.fr/ICPR2012/, 2012.

[3] Assessment of mitosis detection algorithms. `http://amida13.isi.uu.nl`, 2013.

[4] Alberts, B., Johnson, A., and Lewis, J. e. a. *Molecular Biology of the Cell*. Garland Science, 2007.

[5] Amit, Y., and Geman, D. Shape quantization and recognition with randomized trees. *Neural computation 9*, 7 (1997), 1545–1588.

[6] Bachle, M., and Kirchberg, P. Ruby on rails. *Software, IEEE 24*, 6 (2007), 105–108.

[7] Bellman, R. Dynamic programming. 2003.

[8] Bishop, C. M., et al. *Pattern recognition and machine learning*, vol. 1. springer New York, 2006.

[9] Bloom, H., and Richardson, W. Histological grading and prognosis in breast cancer: a study of 1409 cases of which 359 have been followed for 15 years. *British Journal of Cancer 11*, 3 (1957), 359.

[10] Bosch, A., Zisserman, A., and Muoz, X. Image classification using random forests and ferns. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on* (2007), pp. 1–8.

[11] Breiman, L. Random forests. *Machine Learning 45*, 1 (2001), 5–32.

[12] Bro-Nielsen, M. Rigid registration of ct, mr and cryosection images using a glcm framework. In *CVRMed-MRCAS'97* (1997), Springer, pp. 171–180.

[13] Brown, C. D., and Davis, H. T. Receiver operating characteristics curves and related decision measures: A tutorial. *Chemometrics and Intelligent Laboratory Systems 80*, 1 (2006), 24 – 38.

[14] Canny, J. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on PAMI-8*, 6 (1986), 679–698.

[15] Chapelle, O., Haffner, P., and Vapnik, V. Support vector machines for histogram-based image classification. *Neural Networks, IEEE Transactions on 10*, 5 (1999), 1055–1064.

[16] Ciresan, D., Giusti, A., Gambardella, L., and Schmidhuber, J. Mitosis detection in breast cancer histology images with deep neural networks. *Journal of Pathology Informatics special issue* (2013), to appear.

[17] Cohen, J., et al. A coefficient of agreement for nominal scales. *Educational and psychological measurement 20*, 1 (1960), 37–46.

[18] Collingbourne, H. *The book of Ruby*. No Starch Press, 2011.

[19] Damjanov, I., and Fan, F. *Cancer grading manual*. Springer Science+ Business Media, 2007, ch. 11, pp. 75 – 81.

[20] Davis, J., and Goadrich, M. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning* (New York, NY, USA, 2006), ICML '06, ACM, pp. 233–240.

[21] Dorfman, R. A formula for the gini coefficient. *The Review of Economics and Statistics 61*, 1 (1979), pp. 146–149.

[22] Dunne, B., and Going, J. Scoring nuclear pleomorphism in breast cancer. *Histopathology 39*, 3 (2001), 259–265.

[23] Eliceiri, K. W., Berthold, M. R., Goldberg, I. G., Ibáñez, L., Manjunath, B., Martone, M. E., Murphy, R. F., Peng, H., Plant, A. L., Roysam, B., et al. Biological imaging software tools. *Nature methods 9*, 7 (2012), 697–710.

[24] Elston, C., and Ellis, I. Pathological prognostic factors in breast cancer. i. the value of histological grade in breast cancer: experience from a large study with long-term follow-up. *Histopathology 19*, 5 (1991), 403–410.

[25] Forsyth, D. A., and Ponce, J. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002, ch. 10.

[26] Frank, E., Hall, M., Trigg, L., Holmes, G., and Witten, I. H. Data mining in bioinformatics using weka. *Bioinformatics 20*, 15 (2004), 2479–2481.

[27] Genestie, C. Mammary pathology. `http://ipal.cnrs.fr/doc/projects/MammaryPathology_CatherineGenestie_2011.pdf`, 2011.

[28] Genestie, C., Zafrani, B., Asselain, B., Fourquet, A., Rozan, S., Validire, P., Vincent-Salomon, A., Sastre-Garau, X., et al. Comparison of the prognostic value of scarff-bloom-richardson and nottingham histological grades in a series of 825 cases of breast cancer: major importance of the mitotic count as a component of both grading systems. *Anticancer research 18*, 1B (1998), 571.

[29] Goutte, C., and Gaussier, E. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *Advances in Information Retrieval*, D. Losada and J. Fernández-Luna, Eds., vol. 3408 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 345–359.

[30] Guican, M., Boucheron, L., Can, A., Madabhushi, A., Rajpoot, N., and Yener, B. Histopathological image analysis: A review. *Biomedical Engineering, IEEE Reviews in 2* (2009), 147–171.

[31] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter 11*, 1 (2009), 10–18.

[32] Hanley, J. A., McNeil, B. J., et al. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology 148*, 3 (1983), 839–843.

[33] Hansson, D. H., et al. Ruby on rails. *Website. Projektseite: http://www. rubyonrails. org* (2009).

[34] Haralick, R. M., Shanmugam, K., and Dinstein, I. H. Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, 6 (1973), 610–621.

[35] Hartley, R. I., and Zisserman, A. *Multiple View Geometry in Computer Vision*, second ed. Cambridge University Press, ISBN: 0521540518, 2004.

[36] Ho, T. K. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on* (1995), vol. 1, pp. 278–282 vol.1.

[37] Ho, T. K. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 20*, 8 (1998), 832–844.

[38] Honeycutt, C. E., and Plotnick, R. Image analysis techniques and gray-level co-occurrence matrices (glcm) for calculating bioturbation indices and characterizing biogenic sedimentary structures. *Computers & Geosciences 34*, 11 (2008), 1461–1472.

[39] Hornik, K., Buchta, C., and Zeileis, A. Open-source machine learning: R meets weka. *Computational Statistics 24*, 2 (2009), 225–232.

[40] Hsu, C.-W., Chang, C.-C., and Lin, C.-J. A practical guide to support vector classification. Tech. rep., Department of Computer Science, National Taiwan University, Taipei 106, Taiwan, 2010.

[41] Huang, C.-H., and Lee, H.-K. Automated mitosis detection based on exclusive independent component analysis. In *Pattern Recognition (ICPR), 2012 21st International Conference on* (2012), pp. 1856–1859.

[42] Huang, C.-H., Veillard, A., Roux, L., Loménie, N., and Racoceanu, D. Time-efficient sparse analysis of histopathological whole slide images. *Computerized Medical Imaging and Graphics 35*, 7 - 8 (2011), 579 – 591. ¡ce:title¿Whole Slide Image Process¡/ce:title¿.

[43] Irshad, H., Gouaillard, A., Roux, L., and Racoceanu, D. Multispectral spatial characterization: Application to mitosis detection in breast cancer histopathology. *arXiv preprint arXiv:1304.4041* (2013).

[44] Irshad, H., Jalali, S., Roux, L., Racoceanu, D., Hwee, L. J., Le Naour, G., and Capron, F. Automated mitosis detection using texture, sift features and hmax biologically inspired approach.

[45] Jähne, B., and Haussecker, H. *Computer vision and applications: a guide for students and practitioners*. Academic Press, 2000.

[46] Joachims, T. *Text categorization with support vector machines: Learning with many relevant features*. Springer, 1998.

[47] Jolliffe, I. *Principal component analysis*. Wiley Online Library, 2005.

[48] Juszczak, P., Tax, D., and Duin, R. Feature scaling in support vector data description. In *Proc. ASCI* (2002), Citeseer, pp. 95–102.

[49] Khan, A., El-Daly, H., and Rajpoot, N. A gamma-gaussian mixture model for detection of mitotic cells in breast cancer histopathology images. In *Pattern Recognition (ICPR), 2012 21st International Conference on* (2012), pp. 149–152.

[50] Khan, A., Simmons, E., El-Daly, H., and Rajpoot, N. Hymap: A hybrid magnitude-phase approach to unsupervised segmentation of tumor areas in breast cancer histology images. *Journal of Pathology Informatics 4*, 2 (2013), 1.

[51] Lalkhen, A. G., and McCluskey, A. Clinical tests: sensitivity and specificity. *Continuing Education in Anaesthesia, Critical Care & Pain 8*, 6 (2008), 221–223.

[52] Lindeberg, T. *Scale-Space*. Wiley Online Library, 2008.

[53] Liu, J., Sun, J., and Wang, S. Pattern recognition: An overview. *IJCSNS International Journal of Computer Science and Network Security 6*, 6 (2006), 57–61.

[54] Ljosa, V., and Carpenter, A. E. Introduction to the quantitative analysis of two-dimensional fluorescence microscopy images for cell-based screening. *PLoS computational biology 5*, 12 (2009), e1000603.

[55] Lowe, D. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on* (1999), vol. 2, pp. 1150–1157 vol.2.

[56] Lu, D., and Weng, Q. A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing 28*, 5 (2007), 823–870.

[57] Malon, C., Brachtel, E., Cosatto, E., Graf, H. P., Kurata, A., Kuroda, M., Meyer, J. S., Saito, A., Wu, S., and Yagi, Y. Mitotic figure recognition: agreement among pathologists and computerized detector. *Analytical Cellular Pathology (Amst) 35*, 2 (2012), 97–100.

[58] Manavalan, R., and Thangavel, K. Evluation of textural feature extraction from grlm for prostate cancer trus medical images. *International Journal of Computer Applications (0975-8887) Volume 36-No.12* (December 2011), pp.33 – 39.

[59] Mann, H., and Whitney, D. On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Stat. 18* (1947), 50–60.

[60] Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to information retrieval*, vol. 1. Cambridge University Press Cambridge, 2008.

[61] Meyer, J. S., Alvarez, C., Milikowski, C., Olson, N., Russo, I., Russo, J., Glass, A., Zehnbauer, B. A., Lister, K., and Parwaresch, R. Breast carcinoma malignancy grading by bloom–richardson system vs proliferation index: reproducibility of grade and advantages of proliferation index. *Modern pathology 18*, 8 (2005), 1067–1078.

[62] Mitchell, T. *Machine Learning.* Mc Graw Hill, 1997.

[63] Moelich, M. Tracking objects with the chan-vese algorithm. *UCLA CAM Report* (2003), 03–14.

[64] Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of Machine Learning.* The MIT Press, 2012.

[65] Nixon, M., and Aguado, A. S. *Feature extraction & image processing.* Academic Press, 2008.

[66] Ojala, T., Pietikäinen, M., and Mäenpää, T. A generalized local binary pattern operator for multiresolution gray scale and rotation invariant texture classification. In *In: Advances in Pattern Recognition, ICAPR 2001 Proceedings* (2001).

[67] Ojala, T., and Pietikäinen M & Mäenpää, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7)* (2002), 971 – 987.

[68] Oommen, T., Misra, D., Twarakavi, N., Prakash, A., Sahoo, B., and Bandopadhyay, S. An objective analysis of support vector machine based classification for remote sensing. *Mathematical Geosciences 40*, 4 (2008), 409–424.

[69] Paliwal, J., Jayas, D., Visen, N., and White, N. Quantification of variations in machine-vision-computed features of cereal grains. *Canadian Biosystems Engineering 47* (2005), 7–1.

[70] Papageorgiou, C. P., Oren, M., and Poggio, T. A general framework for object detection. In *Sixth International Conference on Computer Vision* (1998), IEEE, pp. 555–562.

[71] Rasmussen, C. E. *Gaussian processes for machine learning*. Citeseer, 2006.

[72] Raven, P. H., and Johnson, G. B. *Biology 9th edition*. Mc Graw Hill, 2010.

[73] Roux, L., Tutac, A., Loménie, N., Balensi, D., Racoceanu, D., Veillard, A., Leow, W.-K., Klossa, J., and Putti, T. A cognitive virtual microscopic framework for knowlege-based exploration of large microscopic images in breast cancer histopathology. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE* (2009), pp. 3697–3702.

[74] Russell, S. J., Norvig, P., Davis, E., Russell, S. J., and Russell, S. J. *Artificial intelligence: a modern approach*, vol. 2. Prentice hall Englewood Cliffs, 2010.

[75] Saeys, Y., Inza, I., and Larrañaga, P. A review of feature selection techniques in bioinformatics. *Bioinformatics 23*, 19 (2007), 2507–2517.

[76] Sanguansat, P., Ed. *Principal Component Analysis - Multidisciplinary Applications*. InTech, 2012.

[77] Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., Preibisch, S., Rueden, C.,

Saalfeld, S., Schmid, B., et al. Fiji: an open-source platform for biological-image analysis. *Nature methods 9*, 7 (2012), 676–682.

[78] Schneider, C. A., Rasband, W. S., and Eliceiri, K. W. Nih image to imagej: 25 years of image analysis. *Nat Methods 9*, 7 (2012), 671–675.

[79] Schölkopf, B., and Smola, A. J. *Learning with kernels: support vector machines, regularization, optimization and beyond*. the MIT Press, 2002.

[80] Sertel, O., Kong, J., Shimada, H., Catalyurek, U., Saltz, J. H., and Gurcan, M. N. Computer-aided prognosis of neuroblastoma on whole-slide images: Classification of stromal development. *Pattern Recognition 42*, 6 (2009), 1093–1103.

[81] Shamir, L., Delaney, J. D., Orlov, N., Eckley, D. M., and Goldberg, I. G. Pattern recognition software and techniques for biological image analysis. *PLoS computational biology 6*, 11 (2010), e1000974.

[82] Smith, S. M., and Brady, J. M. Susan - a new approach to low level image processing. *International Journal of Computer Vision 23*, 1 (1997), 45–78.

[83] Sommer, C., Fiaschi, L., Hamprecht, F. A., and Gerlich, D. W. Learning-based mitotic cell detection in histopathological images. In *Pattern Recognition (ICPR), 2012 21st International Conference on* (2012), IEEE, pp. 2306–2309.

[84] Stehman, S. V. Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment 62*, 1 (1997), 77–89.

[85] Stricker, M., and Swain, M. The capacity of color histogram indexing. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on* (1994), pp. 704–708.

[86] Suykens, J. A. K., and Vandewalle, J. Least squares support vector machine classifiers. *Neural Process. Lett. 9*, 3 (June 1999), 293–300.

[87] Swedlow, J. R., Goldberg, I. G., Eliceiri, K. W., et al. Bioimage informatics for experimental biology. *Annual review of biophysics 38* (2009), 327.

[88] Tarca, A. L., Carey, V. J., Chen, X.-w., Romero, R., and Drăghici, S. Machine learning and its applications to biology. *PLoS computational biology 3*, 6 (2007), e116.

[89] Theodoridis, S., and Koutroumbas, K. Pattern recognition. *Academic Press, Boston MA, USA* (2008).

[90] Unser, M., and Aldroubi, A. A review of wavelets in biomedical applications. *Proceedings of the IEEE 84*, 4 (April 1996), 626–638.

[91] Unser, M., and Blu, T. Wavelet theory demystified. *IEEE Transactions on Signal Processing 51*, 2 (February 2003), 470–483.

[92] Veta, M., van Diest, P. J., and Pluim, J. P. W. Detecting mitotic figures in breast cancer histopathology images. *Proc. SPIE 8676, Medical Imaging: Digital Pathology, 867607* (2013).

[93] Viola, P., and Jones, M. Rapid object detection using a boosted cascade of simple features. pp. 511–518.

[94] Viola, P., and Jones, M. Robust real-time object detection. In *International Journal of Computer Vision* (2001).

[95] Wang, B.-h., Wang, H.-j., and Qi, H.-n. Wood recognition based on grey-level co-occurrence matrix. In *Computer Application and System Modeling (ICCASM), 2010 International Conference on* (2010), vol. 1, IEEE, pp. V1–269.

[96] Witten, I. H., Frank, E., and Hall, M. A. *Data Mining: Practical Machine Learning Tools and Techniques*, 3 ed. Morgan Kaufmann, Amsterdam, 2011.

[97] Wright, J., Yang, A. Y., Ganesh, A., Sastry, S. S., and Ma, Y. Robust face recognition via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 31*, 2 (2009), 210–227.

[98] Yin, Z., Bise, R., Chen, M., and Kanade, T. Cell segmentation in microscopy imagery using a bag of local bayesian classifiers. In *Biomedical Imaging: From Nano to Macro, 2010 IEEE International Symposium on* (2010), pp. 125–128.

[99] Yogesan, K., Jørgensen, T., Albregtsen, F., Tveter, K., and Danielsen, H. Entropy-based texture analysis of chromatin structure in advanced prostate cancer. *Cytometry 24*, 3 (1996), 268–276.

[100] Zweig, M. H., and Campbell, G. Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine. *Clinical chemistry 39*, 4 (1993), 561–577.

Mitosis

Description of the Mitosis phases. [4, 72]

Mitosis is the process by which an eukaryotic cell separates the chromosomes in its cell nucleus into two identical sets, in two separate nuclei.

Source Code Listings

.1 Features extraction

Matlab code used to build feature vectors:

```
function [ train_features, train_classes, eval_features, eval_classes,
    train_features_norm, eval_features_norm ] = extractFeatures(extTrainDataSet,
    extEvalDataSet, selected_features, ext_train, ext_eval, normalize, save_data,
     filename )
%extractFeatures extract selected features from dataset
%
%   Parameters:
%
% extTrainDataSet: structure containing extended Train Dataset
% extEvalDataSet: structure containing extended Eval Dataset
%
% selected_features: string indicating type of features
% - m: mean
% - s: standard deviation
% - i: mean intensity in 25 central regions of the image
% - l: lbp riu2 radius 1, 8 neighbors
% - v: mean pixel variance
% - M: mean per color
% - S: std per color
% - d: median per color
% - H: color histograms (16 bins)
% - L: lbp riu2 radius 1-2-3, 8 neighbors concatenated
% - r: lbp ri radius 1, 8 neighbors
% - u: lbp u2 radius 1, 8 neighbors
% - V: pixel variance, 36 elements
% - R: lbp ri radii 1-2-3, 8 neighbors
% - U: lbp u2 radii 1-2-3, 8 neighbors
%
% ext_train: if to use extended trainset (4 rotations and 4 mirrorings)
%
% ext_eval: if to use extended evalset (4 rotations and 4 mirrorings)
%
% normalize: if to normalize on the basis of trainset
%
% save_data: if to save features and classes with name = filename
%       if name not provided filename = 'features.mat'
%
% Returns:
% features and classes for selected training and evaluation set
% if normalize alsto normalized features are returned

%load('../extTrainDataSet.mat');
%load('../extEvalDataSet.mat');


if ext_train
  max_train = 8;
else
  max_train = 1;
end

if ext_eval
  max_eval = 8;
```

```
else
  max_eval = 1;
end

neighbors = 8;

train_features = [];
train_classes = zeros(length(extTrainDataSet)*max_train,1);
eval_features = [];
eval_classes = zeros(length(extEvalDataSet)*max_eval,1);

% classes
for k1 = 1:length(extTrainDataSet)
  for k2=1:max_train
    train_classes(max_train*(k1-1) + k2) = extTrainDataSet(k1).class;
  end
end

for k1 = 1:length(extEvalDataSet)
  for k2=1:max_eval
    eval_classes(max_eval*(k1-1) + k2) = extEvalDataSet(k1).class;
  end
end


for j1 = 1:length(selected_features)
  switch (selected_features(j1))
    case 'm'
      disp('Feature <mean> selected')

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,1);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,1);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          tmp_train_feat(max_train*(k1-1) + k2) = mean(extTrainDataSet(k1).data{
    k2}(:));
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          tmp_eval_feat(max_eval*(k1-1) + k2) = mean(extEvalDataSet(k1).data{k2
    }(:));
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    case 's'
      disp('Feature <std> selected')

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,1);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,1);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          tmp_train_feat(max_train*(k1-1) + k2) = std(extTrainDataSet(k1).data{k2
    }(:));
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          tmp_eval_feat(max_eval*(k1-1) + k2) = std(extEvalDataSet(k1).data{k2
    }(:));
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    case 'i'
      disp('Feature <25 central intensities> selected')
```

```
      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,25);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,25);

      disp(['number of features: ',num2str(size(tmp_train_feat,2))]);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          imc = extTrainDataSet(k1).data{k2}(26:75,26:75,:);  %central part of
    image
          imcg = rgb2gray(imc);                % grayscale
          imcgs = imresize(imcg,[5 5]);           % resized 5x5
          tmp_train_feat(max_train*(k1-1) + k2,:) = imcgs(:)';
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          imc = extEvalDataSet(k1).data{k2}(26:75,26:75,:); %central part of
    image
          imcg = rgb2gray(imc);                % grayscale
          imcgs = imresize(imcg,[5 5]);           % resized 5x5
          tmp_eval_feat(max_eval*(k1-1) + k2,:) = imcgs(:)';
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    case 'l'
      disp('Feature <18 riu2 local binary pattern> selected')

      mapping=getmapping(neighbors,'riu2');

      nf = length(lbp(extEvalDataSet(1).data{1},1,neighbors,mapping,'nh'));

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,nf);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,nf);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          tmp_train_feat(max_train*(k1-1) + k2,:) = lbp(rgb2gray(extTrainDataSet(
    k1).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh');
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          tmp_eval_feat(max_eval*(k1-1) + k2,:) = lbp(rgb2gray(extEvalDataSet(k1)
    .data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh');
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    case 'v'
      disp('Feature <mean of pixel Variance> selected')

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,1);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,1);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          c1 = cont(rgb2gray(extTrainDataSet(k1).data{k2}(26:75,26:75,:)),4,16);
          tmp_train_feat(max_train*(k1-1) + k2) = mean(c1(:));
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          c1 = cont(cont(rgb2gray(extEvalDataSet(k1).data{k2}(26:75,26:75,:))
    ,4,16));
          tmp_eval_feat(max_eval*(k1-1) + k2) = mean(c1(:));
        end
      end
```

```
      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];
    case 'M'
      disp('Feature <mean per color> selected')

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,3);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,3);

      disp(['number of features: ',num2str(size(tmp_train_feat,2))]);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          for k3=1:3
            tmp_im = extTrainDataSet(k1).data{k2}(:,:,k3);
            tmp_train_feat(max_train*(k1-1) + k2,k3) = mean(tmp_im(:));
          end
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          for k3=1:3
            tmp_im = extEvalDataSet(k1).data{k2}(:,:,k3);
            tmp_eval_feat(max_eval*(k1-1) + k2,k3) = mean(tmp_im(:));
          end
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    case 'S'
      disp('Feature <std per color> selected')

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,3);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,3);

      disp(['number of features: ',num2str(size(tmp_train_feat,2))]);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          for k3=1:3
            tmp_im = extTrainDataSet(k1).data{k2}(:,:,k3);
            tmp_train_feat(max_train*(k1-1) + k2,k3) = std(tmp_im(:));
          end
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          for k3=1:3
            tmp_im = extEvalDataSet(k1).data{k2}(:,:,k3);
            tmp_eval_feat(max_eval*(k1-1) + k2,k3) = std(tmp_im(:));
          end
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];


    case 'd'
      disp('Feature <median per color> selected')

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,3);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,3);

      disp(['number of features: ',num2str(size(tmp_train_feat,2))]);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          for k3=1:3
            tmp_train_feat(max_train*(k1-1) + k2,k3) = median(median(
    extTrainDataSet(k1).data{k2}(:,:,k3),1),2);
          end
        end
```

```
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          for k3=1:3
            tmp_eval_feat(max_eval*(k1-1) + k2,k3) = median(median(extEvalDataSet
    (k1).data{k2}(:,:,k3),1),2);
          end
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    case 'H'
      disp('Feature <histograms of color distributions> selected')

      nbins = 16;

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,3*nbins);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,3*nbins);

      disp(['number of features: ',num2str(size(tmp_train_feat,2))]);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          for k3=1:3
            tmp_train_feat(max_train*(k1-1) + k2,nbins*(k3-1)+1:nbins*k3) =
    imhist(extTrainDataSet(k1).data{k2}(:,:,k3),nbins);
          end
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          for k3=1:3
            tmp_eval_feat(max_eval*(k1-1) + k2,nbins*(k3-1)+1:nbins*k3) = imhist(
    extEvalDataSet(k1).data{k2}(:,:,k3),nbins);
          end
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    case 'L'
      disp('Feature <18 riu2 local binary pattern> selected 3 different RADII
    1-2-3')

      mapping=getmapping(neighbors,'riu2');

      nf = length(lbp(rgb2gray(extEvalDataSet(1).data{1}(26:75,26:75,:)),1,
    neighbors,mapping,'nh'))*3;

      disp(['number of features: ',num2str(nf)]);

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,nf);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,nf);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          tmp_train_feat(max_train*(k1-1) + k2,:) = [lbp(rgb2gray(extTrainDataSet
    (k1).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh'), ...
                                lbp(rgb2gray(extTrainDataSet(k1).data{k2
    }(26:75,26:75,:)),2,neighbors,mapping,'nh'),...
                                lbp(rgb2gray(extTrainDataSet(k1).data{k2
    }(26:75,26:75,:)),3,neighbors,mapping,'nh')];
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          tmp_eval_feat(max_eval*(k1-1) + k2,:) = [lbp(rgb2gray(extEvalDataSet(k1
    ).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh'), ...
                                lbp(rgb2gray(extEvalDataSet(k1).data{k2
    }(26:75,26:75,:)),2,neighbors,mapping,'nh'), ...
```

```
                                lbp(rgb2gray(extEvalDataSet(k1).data{k2
}(26:75,26:75,:)),3,neighbors,mapping,'nh')];
    end
  end

  train_features = [train_features, tmp_train_feat];
  eval_features = [eval_features, tmp_eval_feat];

case 'r'
  disp('Feature <ri local binary pattern> selected')

  mapping=getmapping(neighbors,'ri');

  nf = length(lbp(rgb2gray(extTrainDataSet(k1).data{k2}(26:75,26:75,:)),1,
neighbors,mapping,'nh'));

  disp(['number of features: ',num2str(nf)]);

  tmp_train_feat = zeros(length(extTrainDataSet)*max_train,nf);
  tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,nf);

  for k1 = 1:length(extTrainDataSet)
    for k2=1:max_train
      tmp_train_feat(max_train*(k1-1) + k2,:) = lbp(rgb2gray(extTrainDataSet(
k1).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh');
    end
  end

  for k1 = 1:length(extEvalDataSet)
    for k2=1:max_eval
      tmp_eval_feat(max_eval*(k1-1) + k2,:) = lbp(rgb2gray(extEvalDataSet(k1)
.data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh');
    end
  end

  train_features = [train_features, tmp_train_feat];
  eval_features = [eval_features, tmp_eval_feat];

case 'u'
  disp('Feature <u2 local binary pattern> selected')

  mapping=getmapping(neighbors,'u2');

  nf = length(lbp(extEvalDataSet(1).data{1},1,neighbors,mapping,'nh'));

  disp(['number of features: ',num2str(nf)]);

  tmp_train_feat = zeros(length(extTrainDataSet)*max_train,nf);
  tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,nf);

  for k1 = 1:length(extTrainDataSet)
    for k2=1:max_train
      tmp_train_feat(max_train*(k1-1) + k2,:) = lbp(rgb2gray(extTrainDataSet(
k1).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh');
    end
  end

  for k1 = 1:length(extEvalDataSet)
    for k2=1:max_eval
      tmp_eval_feat(max_eval*(k1-1) + k2,:) = lbp(rgb2gray(extEvalDataSet(k1)
.data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh');
    end
  end

  train_features = [train_features, tmp_train_feat];
  eval_features = [eval_features, tmp_eval_feat];

case 'V'
  disp('Feature <pixel Variance - 36 means> selected')

  tmp_train_feat = zeros(length(extTrainDataSet)*max_train,36);
  tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,36);

  disp(['number of features: ',num2str(size(tmp_train_feat,2))]);

  for k1 = 1:length(extTrainDataSet)
```

```
        for k2=1:max_train
          tmp_feat01 = cont(rgb2gray(extTrainDataSet(k1).data{k2}(26:75,26:75,:))
    ,1,neighbors);
          tmp_feat02 = imresize(tmp_feat01,[6 6]);
          tmp_train_feat(max_train*(k1-1) + k2,:) = tmp_feat02(:)';
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          tmp_feat01 = cont(rgb2gray(extEvalDataSet(k1).data{k2}(26:75,26:75,:))
    ,1,neighbors);
          tmp_feat02 = imresize(tmp_feat01,[6 6]);
          tmp_eval_feat(max_eval*(k1-1) + k2,:) = tmp_feat02(:)';
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    case 'R'
      disp('Feature <ri local binary pattern> selected, 3 different RADII 1-2-3')

      mapping=getmapping(neighbors,'ri');

      nf = length(lbp(rgb2gray(extTrainDataSet(k1).data{k2}(26:75,26:75,:)),1,
    neighbors,mapping,'nh'))*3;

      disp(['number of features: ',num2str(nf)]);

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,nf);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,nf);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          tmp_train_feat(max_train*(k1-1) + k2,:) = [lbp(rgb2gray(extTrainDataSet
    (k1).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh'), ...
                                lbp(rgb2gray(extTrainDataSet(k1).data{k2
    }(26:75,26:75,:)),2,neighbors,mapping,'nh'), ...
                                lbp(rgb2gray(extTrainDataSet(k1).data{k2
    }(26:75,26:75,:)),3,neighbors,mapping,'nh')];
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          tmp_eval_feat(max_eval*(k1-1) + k2,:) = [lbp(rgb2gray(extEvalDataSet(k1
    ).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh'), ...
                               lbp(rgb2gray(extEvalDataSet(k1).data{k2
    }(26:75,26:75,:)),2,neighbors,mapping,'nh'), ...
                               lbp(rgb2gray(extEvalDataSet(k1).data{k2
    }(26:75,26:75,:)),3,neighbors,mapping,'nh')];
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    case 'U'
      disp('Feature <u2 local binary pattern> selected, 3 different RADII 1-2-3')

      mapping=getmapping(neighbors,'u2');

      nf = length(lbp(rgb2gray(extEvalDataSet(1).data{1}(26:75,26:75,:)),1,
    neighbors,mapping,'nh'))*3;

      disp(['number of features: ',num2str(nf)]);

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,nf);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,nf);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          tmp_train_feat(max_train*(k1-1) + k2,:) = [lbp(rgb2gray(extTrainDataSet
    (k1).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh'), ...
                                lbp(rgb2gray(extTrainDataSet(k1).data{k2
```

```
    }(26:75,26:75,:)),2,neighbors,mapping,'nh'), ...
                                lbp(rgb2gray(extTrainDataSet(k1).data{k2
    }(26:75,26:75,:)),3,neighbors,mapping,'nh')];
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          tmp_eval_feat(max_eval*(k1-1) + k2,:) = [lbp(rgb2gray(extEvalDataSet(k1
    ).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh'), ...
                                lbp(rgb2gray(extEvalDataSet(k1).data{k2
    }(26:75,26:75,:)),2,neighbors,mapping,'nh'), ...
                                lbp(rgb2gray(extEvalDataSet(k1).data{k2
    }(26:75,26:75,:)),3,neighbors,mapping,'nh')];
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    otherwise
      disp('Feature not recognized')
  end

end


if normalize
  [train_features_norm, m_t, std_t] = normalizeF(train_features);
  eval_features_norm = normalizeF(eval_features, m_t, std_t);
end

if save_data
  if exist('filename','var')
    if isempty(strfind(filename,'.mat'))
      filename = strcat(filename,'.mat');
    end
  else
    filename = 'features.mat';
  end

  if normalize
    save(filename,'selected_features','train_features','train_classes','
     eval_features','eval_classes','train_features_norm','eval_features_norm');
  else
    save(filename,'selected_features','train_features','train_classes','
     eval_features','eval_classes');
  end
end
```

Listing 1: extractFeatures.m

.2 Classification

Matlab code used to run SVMs and RFs classifiers:

```
function [ train_features, train_classes, eval_features, eval_classes,
     train_features_norm, eval_features_norm ] = extractFeatures(extTrainDataSet,
     extEvalDataSet, selected_features, ext_train, ext_eval, normalize, save_data,
      filename )
%extractFeatures extract selected features from dataset
%
%   Parameters:
%
% extTrainDataSet: structure containing extended Train Dataset
% extEvalDataSet: structure containing extended Eval Dataset
%
% selected_features: string indicating type of features
% - m: mean
% - s: standard deviation
% - i: mean intensity in 25 central regions of the image
```

```
% - l: lbp riu2 radius 1, 8 neighbors
% - v: mean pixel variance
% - M: mean per color
% - S: std per color
% - d: median per color
% - H: color histograms (16 bins)
% - L: lbp riu2 radius 1-2-3, 8 neighbors concatenated
% - r: lbp ri radius 1, 8 neighbors
% - u: lbp u2 radius 1, 8 neighbors
% - V: pixel variance, 36 elements
% - R: lbp ri radii 1-2-3, 8 neighbors
% - U: lbp u2 radii 1-2-3, 8 neighbors
%
% ext_train: if to use extended trainset (4 rotations and 4 mirrorings)
%
% ext_eval: if to use extended evalset (4 rotations and 4 mirrorings)
%
% normalize: if to normalize on the basis of trainset
%
% save_data: if to save features and classes with name = filename
%       if name not provided filename = 'features.mat'
%
% Returns:
% features and classes for selected training and evaluation set
% if normalize alsto normalized features are returned

%load('../extTrainDataSet.mat');
%load('../extEvalDataSet.mat');


if ext_train
  max_train = 8;
else
  max_train = 1;
end

if ext_eval
  max_eval = 8;
else
  max_eval = 1;
end

neighbors = 8;

train_features = [];
train_classes = zeros(length(extTrainDataSet)*max_train,1);
eval_features = [];
eval_classes = zeros(length(extEvalDataSet)*max_eval,1);

% classes
for k1 = 1:length(extTrainDataSet)
  for k2=1:max_train
    train_classes(max_train*(k1-1) + k2) = extTrainDataSet(k1).class;
  end
end

for k1 = 1:length(extEvalDataSet)
  for k2=1:max_eval
    eval_classes(max_eval*(k1-1) + k2) = extEvalDataSet(k1).class;
  end
end


for j1 = 1:length(selected_features)
  switch (selected_features(j1))
    case 'm'
      disp('Feature <mean> selected')

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,1);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,1);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          tmp_train_feat(max_train*(k1-1) + k2) = mean(extTrainDataSet(k1).data{
    k2}(:));
        end
```

```
    end

    for k1 = 1:length(extEvalDataSet)
      for k2=1:max_eval
        tmp_eval_feat(max_eval*(k1-1) + k2) = mean(extEvalDataSet(k1).data{k2
  }(:));
      end
    end

    train_features = [train_features, tmp_train_feat];
    eval_features = [eval_features, tmp_eval_feat];

  case 's'
    disp('Feature <std> selected')

    tmp_train_feat = zeros(length(extTrainDataSet)*max_train,1);
    tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,1);

    for k1 = 1:length(extTrainDataSet)
      for k2=1:max_train
        tmp_train_feat(max_train*(k1-1) + k2) = std(extTrainDataSet(k1).data{k2
  }(:));
      end
    end

    for k1 = 1:length(extEvalDataSet)
      for k2=1:max_eval
        tmp_eval_feat(max_eval*(k1-1) + k2) = std(extEvalDataSet(k1).data{k2
  }(:));
      end
    end

    train_features = [train_features, tmp_train_feat];
    eval_features = [eval_features, tmp_eval_feat];

  case 'i'
    disp('Feature <25 central intensities> selected')

    tmp_train_feat = zeros(length(extTrainDataSet)*max_train,25);
    tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,25);

    disp(['number of features: ',num2str(size(tmp_train_feat,2))]);

    for k1 = 1:length(extTrainDataSet)
      for k2=1:max_train
        imc = extTrainDataSet(k1).data{k2}(26:75,26:75,:);  %central part of
  image
        imcg = rgb2gray(imc);                  % grayscale
        imcgs = imresize(imcg,[5 5]);            % resized 5x5
        tmp_train_feat(max_train*(k1-1) + k2,:) = imcgs(:)';
      end
    end

    for k1 = 1:length(extEvalDataSet)
      for k2=1:max_eval
        imc = extEvalDataSet(k1).data{k2}(26:75,26:75,:); %central part of
  image
        imcg = rgb2gray(imc);                  % grayscale
        imcgs = imresize(imcg,[5 5]);            % resized 5x5
        tmp_eval_feat(max_eval*(k1-1) + k2,:) = imcgs(:)';
      end
    end

    train_features = [train_features, tmp_train_feat];
    eval_features = [eval_features, tmp_eval_feat];

  case 'l'
    disp('Feature <18 riu2 local binary pattern> selected')

    mapping=getmapping(neighbors,'riu2');

    nf = length(lbp(extEvalDataSet(1).data{1},1,neighbors,mapping,'nh'));

    tmp_train_feat = zeros(length(extTrainDataSet)*max_train,nf);
    tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,nf);
```

```
      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          tmp_train_feat(max_train*(k1-1) + k2,:) = lbp(rgb2gray(extTrainDataSet(
    k1).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh');
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          tmp_eval_feat(max_eval*(k1-1) + k2,:) = lbp(rgb2gray(extEvalDataSet(k1)
    .data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh');
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    case 'v'
      disp('Feature <mean of pixel Variance> selected')

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,1);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,1);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          c1 = cont(rgb2gray(extTrainDataSet(k1).data{k2}(26:75,26:75,:)),4,16);
          tmp_train_feat(max_train*(k1-1) + k2) = mean(c1(:));
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          c1 = cont(cont(rgb2gray(extEvalDataSet(k1).data{k2}(26:75,26:75,:))
    ,4,16));
          tmp_eval_feat(max_eval*(k1-1) + k2) = mean(c1(:));
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];
    case 'M'
      disp('Feature <mean per color> selected')

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,3);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,3);

      disp(['number of features: ',num2str(size(tmp_train_feat,2))]);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          for k3=1:3
            tmp_im = extTrainDataSet(k1).data{k2}(:,:,k3);
            tmp_train_feat(max_train*(k1-1) + k2,k3) = mean(tmp_im(:));
          end
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          for k3=1:3
            tmp_im = extEvalDataSet(k1).data{k2}(:,:,k3);
            tmp_eval_feat(max_eval*(k1-1) + k2,k3) = mean(tmp_im(:));
          end
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    case 'S'
      disp('Feature <std per color> selected')

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,3);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,3);

      disp(['number of features: ',num2str(size(tmp_train_feat,2))]);
```

```
      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          for k3=1:3
            tmp_im = extTrainDataSet(k1).data{k2}(:,:,k3);
            tmp_train_feat(max_train*(k1-1) + k2,k3) = std(tmp_im(:));
          end
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          for k3=1:3
            tmp_im = extEvalDataSet(k1).data{k2}(:,:,k3);
            tmp_eval_feat(max_eval*(k1-1) + k2,k3) = std(tmp_im(:));
          end
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];


    case 'd'
      disp('Feature <median per color> selected')

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,3);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,3);

      disp(['number of features: ',num2str(size(tmp_train_feat,2))]);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          for k3=1:3
            tmp_train_feat(max_train*(k1-1) + k2,k3) = median(median(
    extTrainDataSet(k1).data{k2}(:,:,k3),1),2);
          end
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          for k3=1:3
            tmp_eval_feat(max_eval*(k1-1) + k2,k3) = median(median(extEvalDataSet
    (k1).data{k2}(:,:,k3),1),2);
          end
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    case 'H'
      disp('Feature <histograms of color distributions> selected')

      nbins = 16;

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,3*nbins);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,3*nbins);

      disp(['number of features: ',num2str(size(tmp_train_feat,2))]);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          for k3=1:3
            tmp_train_feat(max_train*(k1-1) + k2,nbins*(k3-1)+1:nbins*k3) =
    imhist(extTrainDataSet(k1).data{k2}(:,:,k3),nbins);
          end
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          for k3=1:3
            tmp_eval_feat(max_eval*(k1-1) + k2,nbins*(k3-1)+1:nbins*k3) = imhist(
    extEvalDataSet(k1).data{k2}(:,:,k3),nbins);
```

```
        end
      end
    end

    train_features = [train_features, tmp_train_feat];
    eval_features = [eval_features, tmp_eval_feat];

  case 'L'
    disp('Feature <18 riu2 local binary pattern> selected 3 different RADII
  1-2-3')

    mapping=getmapping(neighbors,'riu2');

    nf = length(lbp(rgb2gray(extEvalDataSet(1).data{1}(26:75,26:75,:)),1,
  neighbors,mapping,'nh'))*3;

    disp(['number of features: ',num2str(nf)]);

    tmp_train_feat = zeros(length(extTrainDataSet)*max_train,nf);
    tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,nf);

    for k1 = 1:length(extTrainDataSet)
      for k2=1:max_train
        tmp_train_feat(max_train*(k1-1) + k2,:) = [lbp(rgb2gray(extTrainDataSet
  (k1).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh'), ...
                                lbp(rgb2gray(extTrainDataSet(k1).data{k2
  }(26:75,26:75,:)),2,neighbors,mapping,'nh'),...
                                lbp(rgb2gray(extTrainDataSet(k1).data{k2
  }(26:75,26:75,:)),3,neighbors,mapping,'nh')];
      end
    end

    for k1 = 1:length(extEvalDataSet)
      for k2=1:max_eval
        tmp_eval_feat(max_eval*(k1-1) + k2,:) = [lbp(rgb2gray(extEvalDataSet(k1
  ).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh'), ...
                               lbp(rgb2gray(extEvalDataSet(k1).data{k2
  }(26:75,26:75,:)),2,neighbors,mapping,'nh'), ...
                               lbp(rgb2gray(extEvalDataSet(k1).data{k2
  }(26:75,26:75,:)),3,neighbors,mapping,'nh')];
      end
    end

    train_features = [train_features, tmp_train_feat];
    eval_features = [eval_features, tmp_eval_feat];

  case 'r'
    disp('Feature <ri local binary pattern> selected')

    mapping=getmapping(neighbors,'ri');

    nf = length(lbp(rgb2gray(extTrainDataSet(k1).data{k2}(26:75,26:75,:)),1,
  neighbors,mapping,'nh'));

    disp(['number of features: ',num2str(nf)]);

    tmp_train_feat = zeros(length(extTrainDataSet)*max_train,nf);
    tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,nf);

    for k1 = 1:length(extTrainDataSet)
      for k2=1:max_train
        tmp_train_feat(max_train*(k1-1) + k2,:) = lbp(rgb2gray(extTrainDataSet(
  k1).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh');
      end
    end

    for k1 = 1:length(extEvalDataSet)
      for k2=1:max_eval
        tmp_eval_feat(max_eval*(k1-1) + k2,:) = lbp(rgb2gray(extEvalDataSet(k1)
  .data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh');
      end
    end

    train_features = [train_features, tmp_train_feat];
    eval_features = [eval_features, tmp_eval_feat];
```

```
    case 'u'
      disp('Feature <u2 local binary pattern> selected')

      mapping=getmapping(neighbors,'u2');

      nf = length(lbp(extEvalDataSet(1).data{1},1,neighbors,mapping,'nh'));

      disp(['number of features: ',num2str(nf)]);

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,nf);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,nf);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          tmp_train_feat(max_train*(k1-1) + k2,:) = lbp(rgb2gray(extTrainDataSet(
    k1).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh');
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          tmp_eval_feat(max_eval*(k1-1) + k2,:) = lbp(rgb2gray(extEvalDataSet(k1)
    .data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh');
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    case 'V'
      disp('Feature <pixel Variance - 36 means> selected')

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,36);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,36);

      disp(['number of features: ',num2str(size(tmp_train_feat,2))]);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          tmp_feat01 = cont(rgb2gray(extTrainDataSet(k1).data{k2}(26:75,26:75,:))
    ,1,neighbors);
          tmp_feat02 = imresize(tmp_feat01,[6 6]);
          tmp_train_feat(max_train*(k1-1) + k2,:) = tmp_feat02(:)';
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          tmp_feat01 = cont(rgb2gray(extEvalDataSet(k1).data{k2}(26:75,26:75,:))
    ,1,neighbors);
          tmp_feat02 = imresize(tmp_feat01,[6 6]);
          tmp_eval_feat(max_eval*(k1-1) + k2,:) = tmp_feat02(:)';
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    case 'R'
      disp('Feature <ri local binary pattern> selected, 3 different RADII 1-2-3')

      mapping=getmapping(neighbors,'ri');

      nf = length(lbp(rgb2gray(extTrainDataSet(k1).data{k2}(26:75,26:75,:)),1,
    neighbors,mapping,'nh'))*3;

      disp(['number of features: ',num2str(nf)]);

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,nf);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,nf);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          tmp_train_feat(max_train*(k1-1) + k2,:) = [lbp(rgb2gray(extTrainDataSet
    (k1).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh'), ...
                                    lbp(rgb2gray(extTrainDataSet(k1).data{k2
```

```
    }(26:75,26:75,:)),2,neighbors,mapping,'nh'), ...
                                lbp(rgb2gray(extTrainDataSet(k1).data{k2
    }(26:75,26:75,:)),3,neighbors,mapping,'nh')];
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          tmp_eval_feat(max_eval*(k1-1) + k2,:) = [lbp(rgb2gray(extEvalDataSet(k1
    ).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh'), ...
                                lbp(rgb2gray(extEvalDataSet(k1).data{k2
    }(26:75,26:75,:)),2,neighbors,mapping,'nh'), ...
                                lbp(rgb2gray(extEvalDataSet(k1).data{k2
    }(26:75,26:75,:)),3,neighbors,mapping,'nh')];
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    case 'U'
      disp('Feature <u2 local binary pattern> selected, 3 different RADII 1-2-3')

      mapping=getmapping(neighbors,'u2');

      nf = length(lbp(rgb2gray(extEvalDataSet(1).data{1}(26:75,26:75,:)),1,
    neighbors,mapping,'nh'))*3;

      disp(['number of features: ',num2str(nf)]);

      tmp_train_feat = zeros(length(extTrainDataSet)*max_train,nf);
      tmp_eval_feat = zeros(length(extEvalDataSet)*max_eval,nf);

      for k1 = 1:length(extTrainDataSet)
        for k2=1:max_train
          tmp_train_feat(max_train*(k1-1) + k2,:) = [lbp(rgb2gray(extTrainDataSet
    (k1).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh'), ...
                                lbp(rgb2gray(extTrainDataSet(k1).data{k2
    }(26:75,26:75,:)),2,neighbors,mapping,'nh'), ...
                                lbp(rgb2gray(extTrainDataSet(k1).data{k2
    }(26:75,26:75,:)),3,neighbors,mapping,'nh')];
        end
      end

      for k1 = 1:length(extEvalDataSet)
        for k2=1:max_eval
          tmp_eval_feat(max_eval*(k1-1) + k2,:) = [lbp(rgb2gray(extEvalDataSet(k1
    ).data{k2}(26:75,26:75,:)),1,neighbors,mapping,'nh'), ...
                                lbp(rgb2gray(extEvalDataSet(k1).data{k2
    }(26:75,26:75,:)),2,neighbors,mapping,'nh'), ...
                                lbp(rgb2gray(extEvalDataSet(k1).data{k2
    }(26:75,26:75,:)),3,neighbors,mapping,'nh')];
        end
      end

      train_features = [train_features, tmp_train_feat];
      eval_features = [eval_features, tmp_eval_feat];

    otherwise
      disp('Feature not recognized')
  end

end


if normalize
  [train_features_norm, m_t, std_t] = normalizeF(train_features);
  eval_features_norm = normalizeF(eval_features, m_t, std_t);
end

if save_data
  if exist('filename','var')
    if isempty(strfind(filename,'.mat'))
      filename = strcat(filename,'.mat');
    end
  else
```

```
    filename = 'features.mat';
  end

  if normalize
    save(filename,'selected_features','train_features','train_classes','
     eval_features','eval_classes','train_features_norm','eval_features_norm');
  else
    save(filename,'selected_features','train_features','train_classes','
     eval_features','eval_classes');
  end
end
```

Listing 2: extractFeatures.m

.3 Experiments

Listings

Il listato (o solo parti rilevanti di questo, se risulta particolarmente esteso) con l'autodocumentazione relativa.

Website Implementation

Manuale utente per l'utilizzo del sistema

Use case

Un esempio di impiego del sistema realizzato.

Datasheet

Eventuali Datasheet di riferimento.