



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

---

Escola Tècnica Superior d'Enginyeria  
de Telecomunicació de Barcelona

# MASTER DEGREE THESIS

## DEEP LEARNING NEURAL NETWORKS IN MALARIA DIAGNOSIS

---

Master's degree in Telecommunications  
Engineering (MET)

**Author:** Jaume Fernàndez García

**Advisors:** Elisa Sayrol Cloles, Marga Cabrera Bean

**Barcelona, June 2014**



UNIVERSITAT POLITÈCNICA  
DE CATALUNYA  
BARCELONATECH



## **Abstract**

Malaria is a serious disease mostly spread in tropical and subtropical areas that causes 438.000 deaths per year. Current malaria diagnosis relies primarily on microscopic examination of stained blood films. This method is time consuming and prone to human error, even in experienced hands. Thus, there is a need for the development of an automatic technique that is able to detect malaria in a sensitive and unsupervised manner.

Deep learning networks are a novel field that promises to have a key role in this automatic detection. In this thesis, we propose a system that collects much of the research conducted about this issue and that proposes new schemes to enhance the performance. In particular, a solution based on convolutional neural networks has shown a clear improvement of the results in the detection of malaria.

## **Acknowledgements**

Firstly, I would like to express my sincere gratitude to professors Marga Cabrera, Elisa Sayrol and Josep Vidal for giving me the opportunity to work on this thesis and for advising me during the process. I am also thankful to all the students, professors, doctors and many other people that in some way or another have helped in the development of this project.

Finally, I would like to extend a word of thanks to my friends and family that have helped and encouraged me during this master's degree.

Thank you all.

## **Table of contents**

Abstract.....	2
Acknowledgements .....	3
Table of contents.....	4
List of Figures .....	6
List of Tables .....	7
1. Introduction.....	8
1.1. Motivation.....	10
1.2. Objectives .....	10
1.3. Document structure .....	12
2. State of the art of the technology.....	13
2.1. Support Vector Machine (SVM).....	13
2.1.1. Support Vector Classifier.....	13
2.1.2. Support Vector Machine.....	16
2.2. Convolutional Neural Networks (CNN).....	17
2.2.1. CNN architecture and layers .....	18
2.2.2. Training of a CNN.....	25
3. Methodology and project development.....	29
3.1. Work environment.....	30
3.2. Image acquisition .....	31
3.3. Pre-processing techniques .....	31
3.3.1. Hue binarization.....	31
3.3.2. Cell segmentation.....	35
3.4. Feature extraction .....	41
3.5. Classification.....	44
3.5.1. Support Vector Machine (SVM) .....	45
3.5.2. Convolutional Neural Networks (CNN) .....	45
3.6. Other developed work.....	51
4. Results.....	53
4.1. Results with SVM classification .....	54
4.1.1. Hue binarization.....	54
4.1.2. Cell segmentation.....	55

4.1.3.	Combination of results in SVM classification .....	56
4.1.4.	Comparison of SVM classification methods .....	57
4.2.	Results with CNN classification .....	60
4.2.1.	Hue binarization.....	60
4.2.2.	Cell segmentation.....	62
4.3.	Final comparison of the results .....	63
5.	Budget .....	65
5.1.	Development costs .....	65
5.2.	Comparison with current diagnosis methods .....	67
6.	Conclusions and future development .....	69
6.1.	Objectives covered .....	69
6.2.	Conclusions .....	70
6.3.	Future development.....	71
	Bibliography.....	72

## List of Figures

### Chapter 2

<b>Figure 2.1:</b> Example of maximal margin hyperplane shift	14
<b>Figure 2.2:</b> Examples of soft margin classification	15
<b>Figure 2.3:</b> Non-linear class boundary	16
<b>Figure 2.4:</b> CNN architecture	19
<b>Figure 2.5:</b> Convolutional layer operation	20
<b>Figure 2.6:</b> Input and output of a convolutional layer	21
<b>Figure 2.7:</b> Filters of a trained AlexNet	22
<b>Figure 2.8:</b> Max pooling example	24
<b>Figure 2.9:</b> Dropout example	28

### Chapter 3

<b>Figure 3.1:</b> System block diagram	29
<b>Figure 3.2:</b> Malaria thin blood smear	30
<b>Figure 3.3:</b> Block diagram of hue binarization pre-processing	32
<b>Figure 3.4:</b> (a) Original image (b) Hue image (c) Binary image (d) Potential parasites	33
<b>Figure 3.5:</b> Blood smear with clumped cells	35
<b>Figure 3.6:</b> Block diagram of cell segmentation pre-processing	36
<b>Figure 3.7:</b> Original and binary image on cell segmentation	36
<b>Figure 3.8:</b> (a) Cells with hole (b) Broken cells (c) Clumped cells	37
<b>Figure 3.9:</b> Example of undesired hole filling	38
<b>Figure 3.10:</b> Example of an opening filter breaking cells	39
<b>Figure 3.11:</b> Example of opening filter breaking cells	39
<b>Figure 3.12:</b> Sub-images with different background	40
<b>Figure 3.13:</b> Sub-images after cell segmentation	41
<b>Figure 3.14:</b> Features selected for SVM training	42
<b>Figure 3.15:</b> CNN architecture of the system	46
<b>Figure 3.16:</b> Number of hue binarization sub-images in the CNN training	48
<b>Figure 3.17:</b> Number of cell segmentation sub-images in the CNN training	49
<b>Figure 3.18:</b> Evolution of the training and validation errors	50
<b>Figure 3.19:</b> Malaria thick blood smear	51

## List of Tables

### Chapter 2

<b>Table 2.1:</b> Summary of the main activation functions	23
--	----

### Chapter 3

<b>Table 3.1:</b> Most relevant features in SVM classification	43
--	----

<b>Table 3.2:</b> Summary of the layers on the CNN	47
--	----

### Chapter 4

<b>Table 4.1:</b> Summary of the results of SVM methods: Sub-images	58
---	----

<b>Table 4.2:</b> Summary of the results of SVM methods: Images	59
---	----

<b>Table 4.3:</b> Summary of the results of CNN methods	63
---	----

### Chapter 5

<b>Table 5.1:</b> Summary of the development costs of the project	66
---	----



## 1. Introduction

Malaria is a serious infectious disease that is widespread in tropical and subtropical regions around the equator, including much of Sub-Saharan Africa, Asia, and the Americas. According to the World Health Organization, there were 214 million cases of malaria globally in 2015 (uncertainty range 149–303 million) and 438 000 malaria deaths (range 236 000–635 000) [1]. Malaria is caused by parasites of the genus *Plasmodium* that are transmitted to people through the bites of infected female mosquitoes. Parasites are introduced via its saliva into the circulatory system, and ultimately to the liver where they mature and reproduce [5].

There are various techniques to diagnose malaria of which manual microscopy is considered to be the gold standard. Thus, malaria diagnosis relies primarily on microscopic examination of Giemsa-stained thick and thin blood films. The staining process slightly colorizes the red blood cells but highlights *Plasmodium* parasites, white blood cells, platelets and other artefacts. This method requires trained technicians to efficiently detect and classify the malaria parasite species. However, due to the number of steps required in manual assessment, this diagnostic method is time consuming (leading to late diagnosis) and prone to human error (leading to erroneous diagnosis), even in experienced hands [6].

Giemsa-stained thin blood smear are the most common type of blood films and in this project, we will mainly work with this kind of smears. Nevertheless, there are a large variety of stains and two different smear preparations. Thick blood smears are most useful for detecting the presence of parasites, because a larger sample of blood is examined. On the other hand, in thin smears, the drop of blood is spread across a large area of the slide and so do the red blood cells, enabling to see the parasites within them. This technique facilitates species differentiation and allows to discover what specie of *Plasmodium* is causing the infection.

In addition to this diversity of diagnostic tests, there are four distinct parasite species that can cause malaria on humans: *Plasmodium falciparum*, *P. vivax*, *P. ovale* and *P. malariae*. These species show different morphology when looked under the microscope and can be hard to differentiate even for a trained eye. Furthermore, parasites pass through different stages in their biological cycle where they present distinct characteristics

and shapes. An accurate recognition of the species and stages is crucial to diagnose and treat malaria.

Manual microscopy for diagnosing malaria is also very time consuming in many of its stages. Training a technician able to diagnose malaria requires a training of 4-5 weeks according to the World Health Organization [1]. Moreover, about 30 minutes are necessary for the preparation of a thin blood smear, including extension and staining. Then, searching and finding *Plasmodium* parasites can take from 2 to 5 minutes depending on the parasitaemia density. Furthermore, in remote areas, this training is even more difficult and technicians must be renewed periodically.

With this overview in mind, there is a need for the development of an automatic technique that is able to detect malaria in a robust, unsupervised and sensitive manner and that solves many of the drawbacks of current diagnostic methods. Some research has been conducted, especially in the area of image processing [4], [5], [6]. However, all the previously mentioned variety of scenarios makes it difficult to find a unique and standard solution that is able to deal with all the issues.

The focus of this study is to propose an automatic system based on deep learning solutions, in particular convolutional neural networks (CNN). Deep learning methods are novel techniques that have shown great results in image classification and other related fields. This project has also compiled some of the research done in image processing in order to propose our own method and compare the performances of the different techniques. The aim of the project is to check if the outstanding performance of the convolutional neural networks in other areas can be extended to the malaria detection problem.

During the development of this thesis, we have been in contact with the Tropical Medicine and International Health Unit at Drassanes, who have helped by giving us information and a dataset of images. In particular, a Biology student was developing her final degree thesis at this center which consisted on digitalizing malaria images from blood films. Those images are the ones we have used in this project and they correspond to thin blood smears.

## 1.1. Motivation

This project has been developed as a final thesis of the Master in Telecommunications (MET). The original idea of the project was proposed by a group of professors of the Signal Theory and Communications Department. Some proposals for the detection of malaria were already being developed, such as a method based on crowd-sourcing through the platform MalariaSpot. Moreover, the aim was to create an interdisciplinary working group that focused on neglected diseases. With this background, working on an automatic technique for the diagnosis of malaria makes perfect sense.

My particular interest in this project comes from my will to expand my knowledge on machine learning. In my studies, I had the opportunity to learn its main aspects but I wanted to be able to develop a system where all these concepts were applied. In addition to that, I am really interested in medicine and this project has given me the opportunity to work on both subjects.

## 1.2. Objectives

The main and primary objective of the project was to develop an automatic system capable to distinguish infected images from non-infected ones, by using a convolutional neural network. This statement raises several issues that should be faced before and during the progress of the thesis. First, we required information about malaria, especially regarding image diagnosis, types of species, etc. We also needed to expand our knowledge on deep learning techniques, which is a relatively new concept. But most importantly, we needed a database with a considerable amount of images that made it possible to train and test our automatic systems. Bearing in mind all of this, we defined the initial goals of the project which are listed below.

- Study the different imaging tests for diagnosing malaria and evaluate which of them is more suitable to accomplish the following objectives.
- Investigate what research approaches have been conducted on automatic techniques for the diagnosis of malaria. Propose a solution based on these conventional image processing techniques.
- Research on deep learning techniques and focus on the implementation of a CNN-based solution.

- Compare the results and performance of the used techniques.

The first objective has been accomplished with no difficulties, since much information is available on malaria. As we explained before, this project has been mostly developed with images corresponding to Giemsa-stained thin blood smears. These images are the most appropriate for malaria diagnosis and species differentiation.

We would like to mention that during the first steps of this thesis we didn't have access to the necessary amount of images in order to progress and test our algorithms. This is the hardest drawback we have faced and was partially solved with the acquisition of the previously mentioned database. With a total of 87 images, we have been able to train and test all the proposed techniques, as well as to extract useful results. However, more images are needed in order to check the accuracy of these results and to keep progressing in the project. In particular, CNNs operate with big amounts of images, such as thousands and tens of thousands.

Regarding the second objective, there are some articles about automatic techniques for the detection of malaria but none information about CNN on the disease. We have found CNN applied to other medical imaging problems, the closest ones applied to different cell detection problems [7]. All the methods found for malaria detection focus on image processing tools. Mostly segmentation techniques and conventional machine learning techniques such as support vector machines are applied [4], [5], [6]. In this project we have sought and compiled this information and evaluated which methods are more viable and suitable. Then, our own methods have been proposed, some of which share similarities with the previous ones.

The two last objectives have finally been achieved, even though using CNN has required time and dedication in order to understand the concepts and make the technique work. As we will mention in future sections, we have used a library with little documentation and it has been difficult to find solutions to the problems faced during the process. In the end, all the objectives initially set have been achieved to a significant extent.

Finally, we would like to make a consideration about the project. This thesis focuses on the diagnosis of the species *Plasmodium falciparum*, in particular on the trophozoite stage. This *Plasmodium* is the one causing most of the deaths in the world and besides,

we have access to more images of this class than any other. However, most of the work developed in this thesis can be extended to other species and stages of the parasite.

### **1.3. Document structure**

This document is composed of six chapters, the first of which is this introductory section. It also contains a table of contents, a list of figures, a list of tables and a bibliography as well.

The second chapter is entitled State of the art of the technology and aims to contextualize the technologies used in this project. We summarize information about image processing and machine learning techniques but also about deep learning and convolutional neural networks. Due to the novelty of the technologies, the explanations given in this section are important for the comprehension of the thesis.

The third chapter explains the methodology used during the project and the main part of its development. Firstly, we present the work environment and the architecture of the system. Then, each block that composes the system is further explained in its own subsection: image acquisition, pre-processing, feature extraction and classification. Finally, we explain other work lines that have been followed during the development of this thesis.

Chapter 4 contains the results that have been generated during the development of this diagnosis system. They are organized in categories and in each section, the process that has generated the results is explained. Additional comments and conclusions are also added to this chapter.

The fifth chapter is the budget and consists on an estimation of the costs of developing the project. We also include a comparison between our diagnosis system and the current techniques based on manual microscopy.

Finally, in chapter 6 we present the conclusions of the project, as well as we check the fulfilment of the initial goals. We also expose some possible improvements, extensions and future development that may be done.

## 2. State of the art of the technology

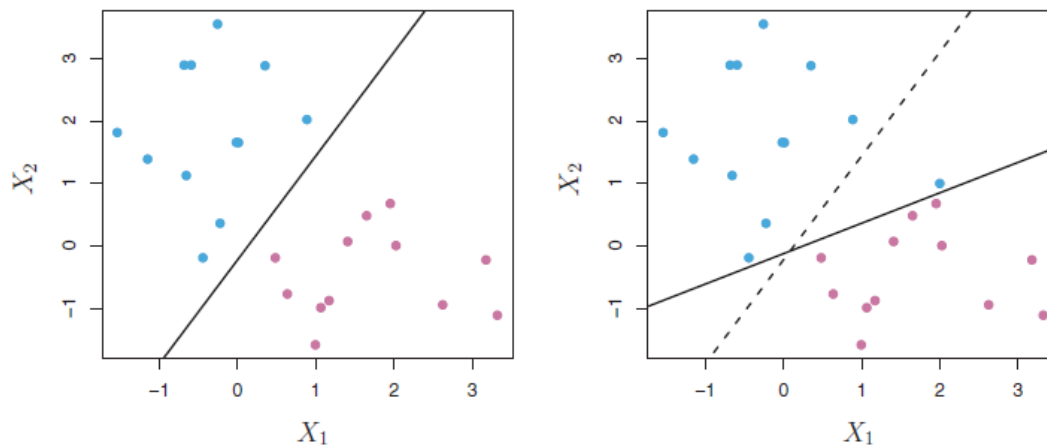
The purpose of this section is to present and explain the technologies used in the course of the project, as well as to highlight its main characteristics. The techniques explained below are related with machine learning, especially with classification. Firstly, we will talk about Support Vector Machine, a common but useful solution when doing classification with two classes. Afterwards, we will focus on convolutional neural networks, a novel technique that is just emerging and whose performance will be tested in this project. In this regard, deep learning techniques have shown outstanding results in other computer vision applications and represent a great opportunity to improve the performance of malaria diagnosis.

### 2.1. Support Vector Machine (SVM)

Support Vector Machines are supervised learning techniques that allow to perform classification, usually into two categories [8]. Given a set of training examples, each marked for belonging to one of two classes, an SVM training algorithm builds a model that assigns new examples into one category or the other. Data in a SVM model belongs to an n-dimensional space and for ease of discussion is usually represented in a 2D map. Firstly, we will introduce the Support Vector Classifier and then, we will extend its concept to the Support Vector Machine.

#### 2.1.1. Support Vector Classifier

When doing linear classification, the common choice would be to define a hyperplane and use it as a decision boundary. However, observations that belong to two classes are not necessarily separable by a hyperplane. In fact, even if a separating hyperplane does exist, there are instances in which a classifier based on a separating hyperplane might not be desirable. A classifier based on a separating hyperplane will necessarily perfectly classify all of the training observations; this can lead to sensitivity to individual observations [2]. An example of this case is shown in Figure 2.1. In the left figure, two classes of observations are shown in blue and in purple, along with the maximal margin hyperplane. In the right figure, an additional blue observation has been added, leading to a dramatic shift in the maximal margin hyperplane shown as a solid line. The dashed line indicates the hyperplane that was obtained in the absence of this additional point.



**Figure 2.1: Example of maximal margin hyperplane shift**

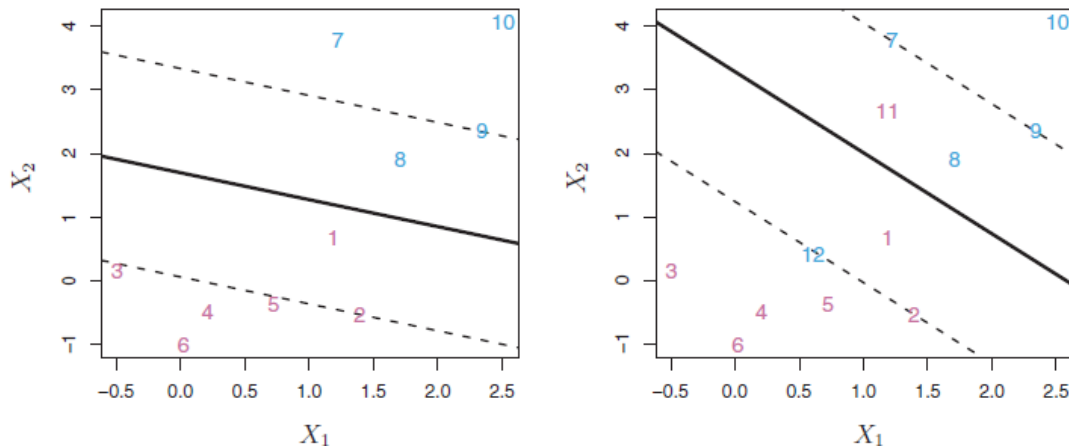
The resulting maximal margin hyperplane is not satisfactory as it has a tiny margin. This is problematic because the distance from an observation to the hyperplane can be seen as a measure of the confidence that the observation was correctly classified. Moreover, the fact that the maximal margin hyperplane is extremely sensitive to a change in a single observation suggests that it may have overfit the training data.

With these issues in mind, we might consider a classifier based on a hyperplane that does not perfectly separate the two classes. By doing this, we want to obtain greater robustness to individual observations and better classification of most of the training observations. In other words, it could be worthwhile to misclassify a few training samples in order to do a better job in classifying the remaining observations.

The support vector classifier, sometimes called a soft margin classifier, does exactly this. At first, it defines a ‘tube’ containing the hyperplane and two margins. The hyperplane defines the separation edge between the two classes. The region between the hyperplane and each margin allows for a certain amount observations. In other words, the margin is *soft* because it can be violated by some of the training observations. Rather than seeking the largest possible margin so that every observation is on the correct side of the hyperplane, we instead allow some observations to be on the incorrect side of the margin, or even the incorrect side of the hyperplane [2].

In Figure 2.2, we can observe two examples of support vector classification, where the hyperplane is plotted as a solid line and the margins are shown as dashed lines. In the left figure, all the observations are on the correct side of the hyperplane and most of them

are also on the correct side of the margin. Only observations 1 and 8 cross the margin but are still well classified. In the right figure, the same dataset is plotted but with the addition of samples 11 and 12 that are in the wrong side of the hyperplane.



**Figure 2.2: Examples of soft margin classification**

The process of finding this tube, with its hyperplane and margins, is the solution to a minimization problem that can be written using Lagrange multipliers in order to incorporate the desired constraints. In this optimization process, there are key parameters, such as the width of the tube and the budget. The budget is a measure of the amount that the margin can be violated by the observations. As the budget increases, we become more tolerant of violations to the margin, and so the margin will widen. Conversely, as it decreases, we become less tolerant of violations to the margin and so the margin narrows. In practice, this parameter is treated as a tuning parameter that is generally chosen via cross-validation [2].

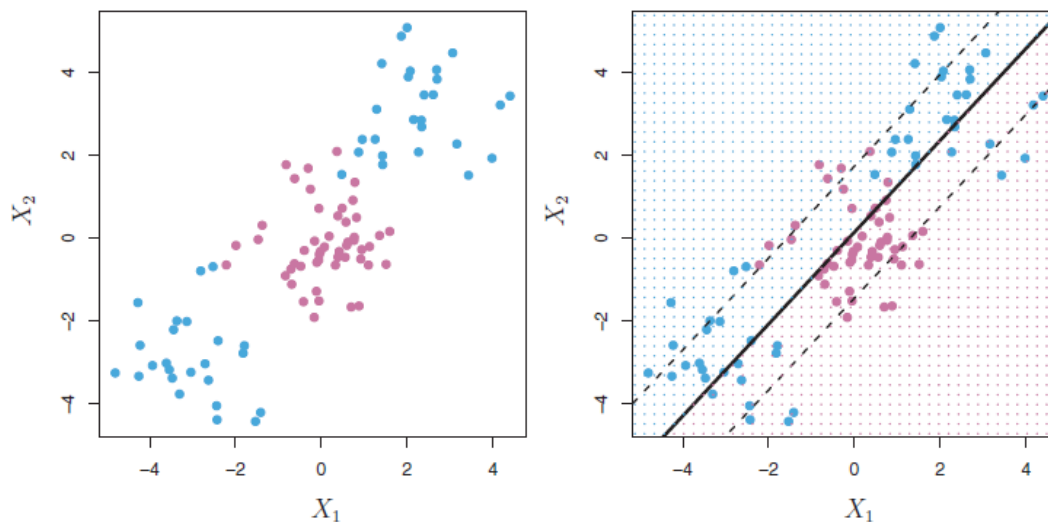
This optimization problem has a very interesting property: Only observations that either lie on the margin or that violate the margin will affect the hyperplane, and hence the classifier obtained. In other words, an observation that lies strictly on the correct side of the margin does not affect the support vector classifier. Changing the position of a correctly classified observation does not change the classifier at all, provided that its position remains on the correct side of the margin. Observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as support vectors and they do affect the support vector classifier [2].



The fact that the support vector classifier's decision rule is based only on a potentially small subset of the training observations (the support vectors) means that it is quite robust to the behaviour of observations that are far away from the hyperplane. This is a remarkable property that makes support vector classifier distinct from other classification methods and ideal for certain classification problems.

### 2.1.2. Support Vector Machine

The support vector classifier is a useful approach for two-class classification, if the boundary between the two categories is linear. However, in practice we often face non-linear class boundaries. For instance, consider the data in the left-hand panel of Figure 2.3. It is clear that a support vector classifier or any linear classifier will perform poorly here. Indeed, the support vector classifier shown in the right-hand panel of Figure 2.3 is useless [2].



**Figure 2.3: Non-linear class boundary**

We see that the performance of linear classification can suffer when there is a nonlinear relationship between the predictors and the outcome. In that case, we consider enlarging the feature space using functions of the predictors, in order to address this non-linearity. These goals are achieved by the support vector machine (SVM), which is an extension of the support vector classifier that results from enlarging the feature space in a specific way, using kernels.

We have not discussed in depth how the support vector classifier is computed because the details become somewhat technical. However, it turns out that the solution to the support vector classifier involves only the inner products of the observations. Then, suppose that every time an inner product appears in the representation or in the calculation of the solution for a support vector classifier, we replace it with a generalization of the inner product of the form  $K(x_i, x_i')$ , where  $K$  is some function that we will refer to as a kernel. This procedure allows to enlarge the feature space and is also known as the kernel trick [2], [3].

A kernel is a function that quantifies the similarity between two observations. The support vector classifier is linear in the features and thus, its kernel is known as linear kernel. But there are other viable options for the kernel function  $K$ . For instance, every instance of the inner product can be replaced by a polynomial of a certain degree. This is known as a polynomial kernel of degree  $d$ , and it essentially aims to fit a support vector classifier in a higher-dimensional space involving polynomials of degree  $d$ , rather than in the original feature space. Another popular choice is to use a Gaussian function to compute the distance between two observations. This is known as radial kernel, radial basis function kernel, RBF kernel or Gaussian kernel.

One advantage of using kernels is the reduction of computational operations, as it only requires to compute the values of  $K(x_i, x_i')$  for the different pairs of observations. This can be done without explicitly working in the enlarged feature space. This is important because in many applications of SVMs, the enlarged feature space is so large that computations are intractable. For some kernels, such as the radial kernel, the feature space is implicit and infinite-dimensional, so computations would be even impossible there. When using a RBF kernel, the width of the Gaussian must be chosen via cross-validation.

## **2.2. Convolutional Neural Networks (CNN)**

A convolutional neural network (CNN or ConvNet) is a type of feed-forward artificial neural network in which the connectivity patterns between its neurons is inspired by the organization of the human visual cortex [12]. CNNs consist of multiple layers of small neuron collections that process portions of the input image, called receptive fields. This functioning makes it ideal for operating with images because unlike, other machine

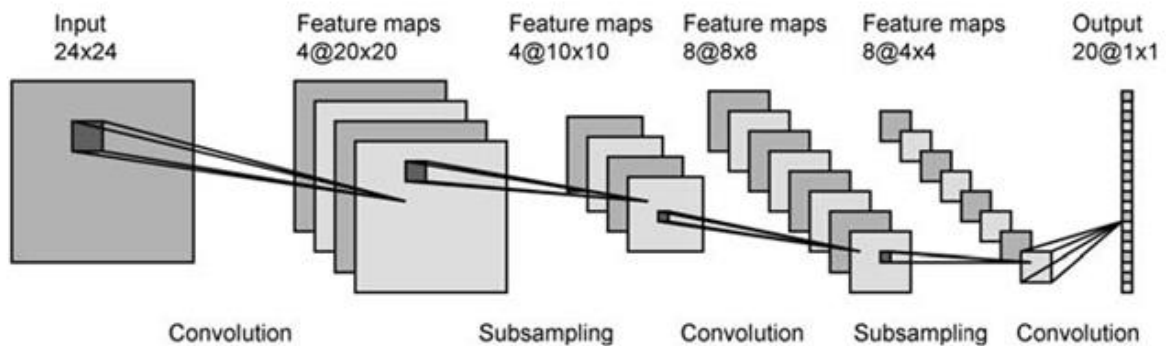
learning techniques, it works on pieces of the original image. The main applications of convolutional neural networks are related with image recognition such as image classification, object detection, facial recognition. However, CNNs are also used in other fields like video analysis, natural language processing and even in helping to understand biological processes.

CNN architectures are very powerful due to the big amount of parameters used in its convolutional layers. This fact can be a disadvantage though, because it can easily lead to overfitting. Another important benefit of CNNs is location invariance, i.e. when doing object detection or classification, no matter where the object appears in the image. Compositionality is one of the major advantages too. Each filter composes a local patch of lower-level features into higher-level representation. It makes intuitive sense that edges are being built from pixels, shapes from edges, and more complex objects from shapes [10]. All of these concepts will be further explained in the following subsections.

In CNNs there are two stages, as it happens in conventional neural networks. During the training phase, a CNN automatically learns the values of its filters based on the task that aims to perform. During the estimation or inference stage, the networks tries to perform the action required using the values learnt on the training, given a new input. An example of this may be classifying a new image to a class based on a CNN trained to distinguish several classes.

### **2.2.1. CNN architecture and layers**

A convolutional neural network is mainly composed by a set of consecutive convolutional layers. In a traditional feed-forward neural network we connect each input neuron to each output neuron in the next layer [10]. That is also called a fully connected layer, or affine layer and in CNN architectures, their use is reserved for the final layers. As its name states, convolutional layers are the core of CNNs and they operate by using convolutions over the input image to compute the output. The structure of a convolutional neural network is shown in Figure 2.4. It is important to notice that when representing a CNN the different convolutional layers or other layers are not plotted. Instead, the inputs and outputs of the stages are shown by clearly stating its dimensions. For instance, the figure below shows an input image of  $24 \times 24 \times 1$  pixels and an output after the first convolution of  $20 \times 20 \times 4$  pixels. Then, the size of the filters can be deduced from these numbers.



**Figure 2.4: CNN architecture**

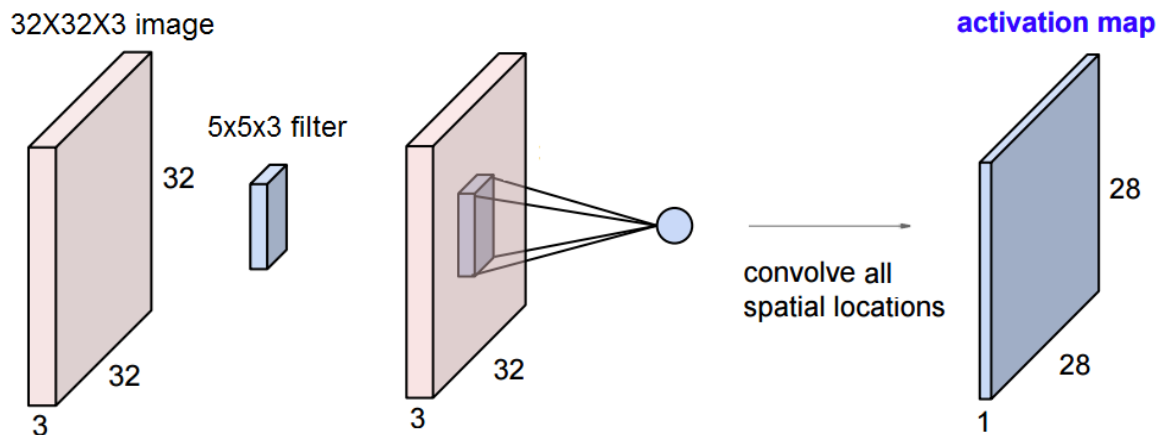
CNN architectures can change in size and complexity but the basic structure is typically the same. Several convolutional layers are concatenated and a non-linear activation function is applied to the results of each layer. A subsampling or pooling mechanism is done between each of these convolutional layers. Thus, it is very typical to see concatenated structures composed by three layers: convolutional, activation function, pooling. However, the activation function is usually not considered as a layer itself but as a part of the convolutional one. After this, some fully connected layers are often added to the end, just before the final layer which depends on the problem the CNN is trying to solve. This final layer can be a classifier with as many outputs as classes or a linear regression among others.

In the following sections, we will present the different existing layers, explain how they work and comment their purposes and advantages.

### **Convolutional layer**

The convolutional layer is the most important and the core block of a CNN. First of all, notice that in conventional neural networks or in fully connected layers, the inputs are depicted as a vertical line of neurons. Instead of doing this, in a CNN we structure the inputs as a square of neurons whose values correspond to the pixel intensities of the input image. In addition to the two dimensions (width and height), these squares or matrix also have a depth. When the input image is grey, the depth is equal to one, whereas an RGB image has a depth of three. However, all three dimensions vary in size as they advance through the network and the different layers are applied.

Convolutional layers are composed by several filters of the same width and height. The depth of these filters must be equal to the depth of the input volume. Meanwhile, the width and height of the filters are smaller so they can be shifted during the convolution. In Figure 2.5, we can observe the elements involved in a convolutional layer and notice the details explained above.

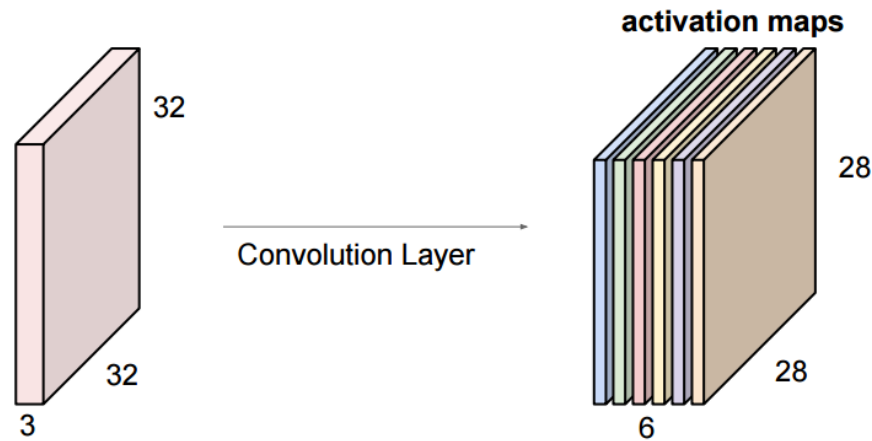


**Figure 2.5: Convolutional layer operation**

As it can be seen in the figure, when doing the computation each filter only is applied to a small number of pixels at each step. Thus, each neuron at the output is only connected to a small region of the input neurons, which is called the receptive field [11]. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input, and producing a 2-dimensional activation map of that filter. Each connection or neuron learns a weight and at each layer an overall bias is learnt as well. Notice that the output width and height are smaller than the input ones as a result of the convolution applied.

Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. In particular, there is one activation map for each filter and it is important to note that the output depth is determined by the number of filters of the layer. Alternatively, the depth of each filter is determined by the input depth. All these ideas are shown in Figure 2.6, where we can see the input and output of a convolutional layer. It can be seen that six activation maps have been produced, meaning that six filters were applied to the input. Regarding the depth, all these filters must have a depth of three in order to match the input dimensions. If another convolutional layer was applied to the

output, whatever number of filters could be applied but all of them should have a depth equal to six [9].

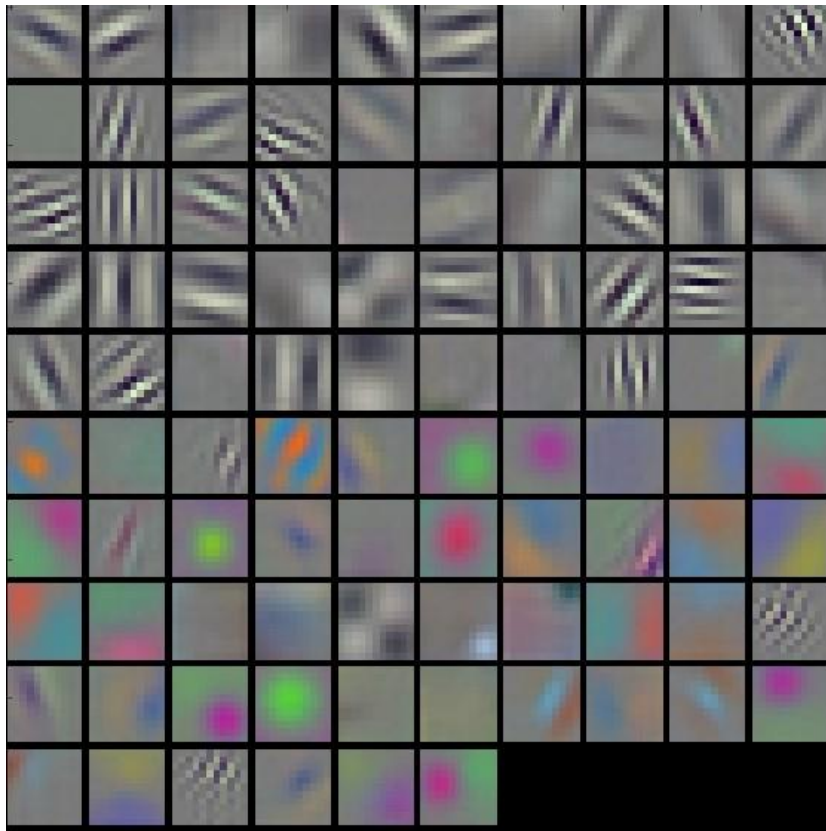


**Figure 2.6: Input and output of a convolutional layer**

Returning to the concept of local receptive fields, we should recall that every output neuron is connected to a small region of the input. When computing an activation map, connections are local in space (along width and height), but always extend along the entire depth of the input volume. Such an architecture ensures that the learnt filters produce the strongest response to a spatially local input pattern. As a result, the network learns filters that activate when they see some specific type of feature at some spatial position in the input.

Having in mind the structure of the activation maps, we can reinterpret the meaning of the output volume. For each value of the depth, we have a different activation map that is computed by a filter that aims to recognize specific shapes or patterns. Meanwhile, for a certain position of width and height, we have a number of values along the depth that correspond to the same input region. Then, each of these values corresponds to the application of different filters.

In Figure 2.7, we show the typical-looking filters on the first convolutional layer of a trained AlexNet, which is a well-known CNN. These plots indicate what shapes and colors is every filter targeting and trying to detect. For instance, there are filters acting on green blobs or vertical frequencies. In deeper layers, this interpretation is complicated because filters are less shaped and defined.



**Figure 2.7: Filters of a trained AlexNet**

Finally, it is worth mentioning two other hyperparameters that are relevant in convolutional layers, despite they are also present in other layers:

- **Stride:** This parameter specifies the number of rows or columns that the filter is shifted during the convolution. In the previous explanations we have assumed a stride of one but this parameter can take any value that makes the following expression result in an integer.

$$\frac{N + F}{stride} + 1$$

$N$  is the input size and  $F$  corresponds to the filter size. Thus, not all the values for the stride are possible given an input and a filter. If the previous condition is not fulfilled, when shifting the filter, at the last step, the filter edges do not match the input edges.

- **Zero-padding:** This mechanism consists on padding the input with zeros, in other words, adding zeros on the borders of the input volume. This allows to control the spatial size of the output volumes. In particular, sometimes it is desirable to

exactly preserve the spatial size of the input volume. This method is also useful in order to apply a certain value of stride. If the mentioned condition is not fulfilled for the current values, zero-padding may be a solution as it increases the value of N.

## Activation Functions

These functions are added at the output of each convolutional layer and also after fully connected layers. Their goal is to provide a non-linear activation of the outputs of a convolution and general non-linearity to the CNN. Therefore, all activation functions must be non-linear. They should also have some desirable properties such as being continuously differentiable in order to enable gradient-based optimization methods. Each of them shows better performance in certain cases and has their own benefits. For example, ReLU results in the neural network training several times faster, without making a significant difference to generalisation accuracy. Anyway, the design of activation functions and the selection of the most appropriate one in each case are complex issues that are out of the scope of this thesis. The most relevant activation functions are shown in Table 2.1 together with their equations and plots. The most used is the ReLU, although the sigmoid and the hyperbolic tangent are common too [9].

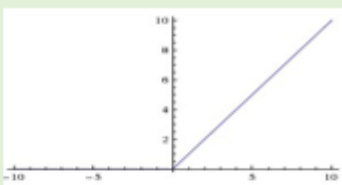
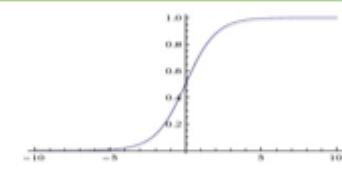
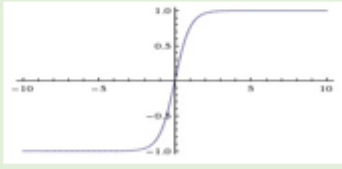
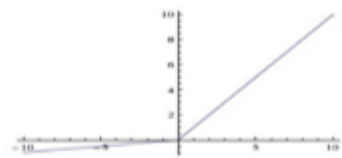
Name	Equation	Plot
<b>Rectified Linear Unit (ReLU)</b>	$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ $f(x) = \max(0, x)$	
<b>Sigmoid</b>	$f(x) = \frac{1}{1 + e^{-x}}$	
<b>Hyperbolic tangent (tanh)</b>	$f(x) = \tanh(x)$	
<b>Leaky ReLU</b>	$f(x) = \max(\alpha x, x)$	

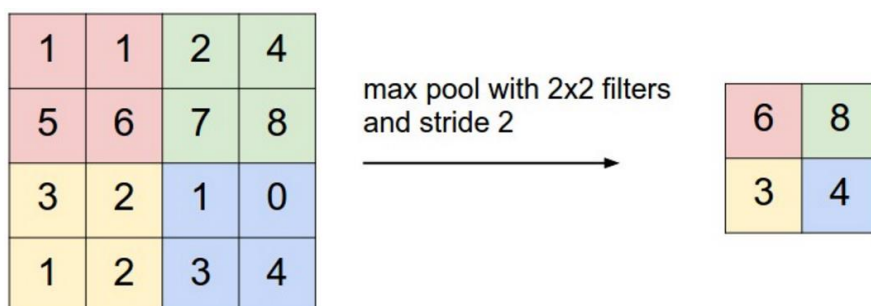
Table 2.1: Summary of the main activation functions



## Pooling layer

Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling among which max pooling is the most common. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The intuition is that once a feature has been found, its exact location isn't as important as its rough location relative to other features. The function of the pooling layer is to progressively reduce the spatial size of the representation by down-sampling the results at some layers. It also aims to reduce the amount of parameters and computation in the network, and hence to also control overfitting. It is common to periodically insert a pooling layer in-between successive convolutional layers in a CNN architecture.

The pooling layer operates independently on every depth slice of the input and resizes it spatially. The most common form is a pooling layer with filters of size  $2 \times 2$  applied with a stride of 2. This leads to a reduction of 75% of the inputs, as only one out of four elements is selected. It is important to note that the depth dimension remains unchanged because the pooling is applied over each activation map independently. Although max pooling is the most common and used, the pooling layers can also perform other functions, such as average pooling and even L2-norm pooling [9].



**Figure 2.8: Max pooling example**

In Figure 2.8, an example of max pooling is represented, showing the operation on a single activation map. A  $2 \times 2$  filter is being used which is convolved using a stride of 2. The different colours shown corresponds to each position of the filter in the input activation map. Then, the maximum value at each location is taken as the output.

## Fully connected layer

As said before, neurons in a fully connected layer are connected to all activations in the previous layer, as it happens in traditional neural networks. In CNN architectures, these layers are usually placed at the end of the network, after several convolutional and pooling layers. In particular, fully connected layers can be seen as a subgroup of convolutional layers where the filters have the exact same size as the input volume. Then, the filter can be interpreted as a set of coefficients that multiply all the input elements, followed by a bias offset. In this case, the output size is  $1 \times 1$  with a depth equal to the number of filters and hence, it is represented as a column.

### 2.2.2. Training of a CNN

The training of a convolutional neural network is for the most part very similar to the training of a conventional neural network. In this section we will explain its main aspects trying to focus on those parts which are specific for CNNs. Anyway, much research is still being conducted in order to improve aspects of the training such as regularization techniques and parameter updating, but also to find new methods than can increase the performance.

The process of training is composed by two stages: the forward and the backward pass. During the forward pass the network computes the outputs by applying convolutions, activation functions, pooling layers until the final result. Typically this stage works the same as the inference of a new image and the estimation of its result, although it can present some modifications. On the other hand, the backward pass is executed afterwards and aims to correct the values on all the layers by defining a loss function that compares the value obtained on the forward pass with the expected one.

This process with both stages is repeated over a number of epochs that is a parameter of the training. At each epoch, all the elements in the train dataset are used to update the weights of the CNN. In general terms, a higher number of epochs implies a deeper training and more accurate results. Moreover, there are many parameters that can be tuned in order to customize the training and adjust to the current problem and data. In the following subsections we will explain most of issues that have been commented before and present some important considerations to do a proper training of a CNN.

## Backpropagation and parameter updates

Backpropagation is an abbreviation for "backward propagation of errors" and it is the most common method of training neural networks used in conjunction with an optimization method such as gradient descent. As it names states, this method propagates the errors through the whole network to update the weights on the layers. The errors are generated by a loss function that compares the obtained result with the expected value. Then, the goal of backpropagation is to modify the weights on the layers by minimizing the loss function. Backpropagation requires a known, desired output for each input value in order to calculate the loss function. It is therefore usually considered to be a supervised learning method, although it is also used in some unsupervised networks.

Regarding the optimization method, we have commented that gradient descent is the usual choice. The method calculates the gradient of a loss function with respect to all the weights in the network. The gradient is fed to the optimization method which uses it to update the weights, in an attempt to minimize the loss function. The parameter updates are done by using the chain rule to iteratively compute the gradient for each layer. Thus, backpropagation requires that the activation functions used in the CNN are differentiable in order to calculate its gradient.

Gradient descent with backpropagation is not guaranteed to find the global minimum of the error function, but only a local minimum. Moreover, the convergence of the algorithm can be very slow depending on the surface of the gradient. Although these inconveniences may not be a problem when dealing with small networks and datasets, they can worsen the training as the dimensions grow. For this reason, other algorithms for parameter updates have been proposed. Momentum update allows to keep a velocity as the updates go on. Then, if the value of a gradient changes drastically, the effect on the direction of the updates will not be so sharp. Other related methods that try to further improve the convergence of backpropagation are Nesterov momentum, AdaGrad, RMSProp and Adam update [9].

Apart from the technique chosen to apply backpropagation, it is also important to select their parameters correctly. This is the case of the learning rate, which measures how much will the weights be updated in the gradient direction. A high learning rate allows to move quickly in the desired directions but causes jitter when near minimums. On the

other hand, low values can slow down the convergence and get stuck in local minimums. Thus, the ideal situation is to select a high learning rate for the first epochs and then decrease its value to make the convergence easier.

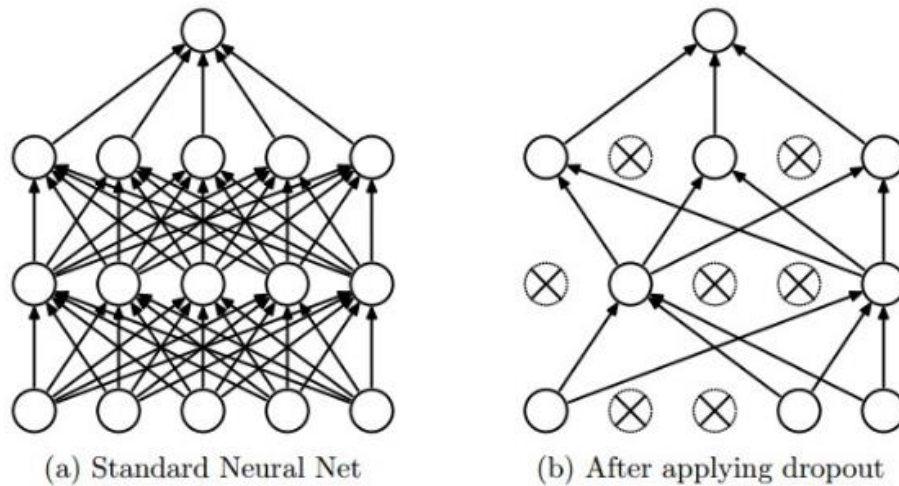
## Regularization

Overfitting is a common problem in the training of neural networks that appears when a model is excessively complex, such as having too many parameters relative to the number of observations. A model that has been overfit has poor predictive performance, as it overreacts at minor fluctuations in the training data. Increasing the amount of training data or reducing the size of the network are ways of reducing overfitting. However, large networks have the potential to be more powerful than small networks. Fortunately, there are other techniques which can reduce overfitting, even when we have a fixed network and fixed training data. These are known as regularization techniques [9].

The idea of many regularization techniques is to add an extra term to the loss function, a term called the regularization term. This method is called weight decay and its most common examples are L2 and L1 regularization [9], [12]. In L2 regularization, the added term is the sum of the squares of all the weights in the network. On the other hand, L1 regularization consists in adding the sum of the absolute values of the weights. The regularization term can be different depending on the technique but it is always multiplied by a coefficient  $\lambda$ .

Intuitively, the effect of regularization is to ensure that the network prefers to learn small weights. Large weights will only be allowed if they considerably improve the first term of the loss function. In other words, regularization can be viewed as a way of compromising between finding small weights and minimizing the original cost function. The relative importance of the two elements of this compromise depends on the value of  $\lambda$ .

Another technique to avoid overfitting is dropout that can be applied to complement the regularization techniques mentioned before. While training, dropout is implemented to keep some neurons with a certain probability and deactivate others. Only the reduced network is trained on the data in that stage. The removed nodes are then reinserted into the network with their original weights. During testing there is no dropout applied.



**Figure 2.9: Dropout example**

In Figure 2.9, a dropout example is represented for a traditional neural network with fully connected layers. It can be seen that about half of the neurons has been deactivated after applying dropout.

By avoiding to train all the nodes with all the available training data, dropout decreases overfitting in neural nets. The method also improves significantly the speed of the training. The technique seems to reduce the complex, tightly fitted interactions between nodes, leading them to learn more robust features which better generalize to new data.

### Batch normalization

During the training step, all the train data is not used simultaneously but divided into groups that are called batches or mini-batches. Then, each of these batches is used independently in the training. Setting a value to the batch size is an issue to consider before the training because the performance of the process can vary significantly.

Recent researches suggest that a change in the distribution of activations because of parameter updates slows the CNN learning [13]. This phenomenon is called “internal covariate shift” and can be alleviated by implementing batch normalization. During the training, each activation of the mini-batch is centred to zero-mean and unit variance. The mean and variance are measured over the whole mini-batch, independently for each activation. The application of these measures potentially allows for faster learning and higher overall accuracy.

### 3. Methodology and project development

In this chapter we will explain all the relevant methods and procedures that have been implemented during the project and that have led to the obtained results. Firstly, we will introduce the work environment as well as some software tools that have an important role in the development of this thesis. Then, the core of the project will be explained. In Figure 3.1, a block diagram of the developed system is shown with all the stages that will be explained in the following sub-sections.

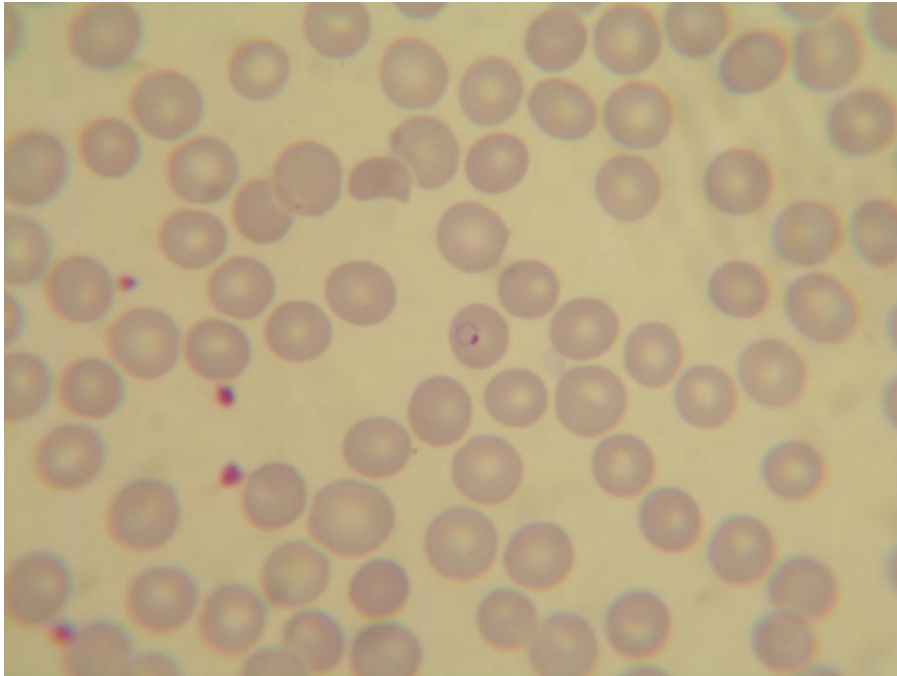


**Figure 3.1: System block diagram**

At first, we will briefly comment on the image acquisition stage. Afterwards, we will focus on the image pre-processing step and explain the two different pre-processing methods that we have designed. The two following blocks are feature extraction and classification, which are very connected. Regarding the classification block, we have also proposed two different techniques.

As a summary, there are two pre-processing and two classification techniques, which can be combined between them obtaining four possible system configurations. We will expose them independently, commenting its benefits and limitations. Some considerations on their performance will be done but the results will be further explained in chapter 4.

Finally, we would like to highlight that the methods developed, especially regarding pre-processing techniques, highly depend on the available images. These procedures try to detect malaria on thin blood smears with Giemsa stain because these images are the most common and have available a large number of examples. Moreover, the number of images on a dataset influence the performance of the classification techniques. All these limitations must be considered before and during the development of this project. In Figure 3.2, an example of the images which we have worked with is shown. In the middle part of the image, there is a red blood cell containing a parasite. We can also observe some platelets in left side.



**Figure 3.2: Malaria thin blood smear**

### **3.1. Work environment**

The project in its entirety has been developed in a laptop using Matlab R2014a. The major part of the algorithms relies on libraries and tools that are available in this version of Matlab. In this regard, there are many tools for image processing that have been useful when trying to improve the pre-processing techniques.

In relation to convolutional neural networks, we have used a toolbox called MatConvNet that allows to train your own CNNs, as well as using pre-trained models. Although some other options were available, we have chosen MatConvNet because it was compatible with Matlab. This library also gives the option of using a GPU for improving the speed when working with CNNs. Despite we encountered some velocity limitations, no GPU was needed nor used in the project because our network is relatively small. Moreover, the pre-processing techniques applied make the training easier.

Before starting to explain the details of the project development, we would like to make a consideration on the technologies used. CNN is a novel technique and, although there is much theoretical information, it can be difficult to find information about practical implementations. During the first steps of the project, it has been necessary to spend a considerable amount of time on learning to use CNNs and MatConvNet.

### **3.2. Image acquisition**

The first block of our system is responsible for the acquisition of the images. As this task is not developed by the author of this thesis, we will briefly comment it but without going into the details. As explained in the introduction section, the images used in this project have been provided by a Biology student developing her final thesis at the Drassanes Tropical Medicine and International Health Unit.

All the images correspond to thin blood smears. After the drop of blood is extended, it is stained in Giemsa and prepared for the visualization in the microscope. The extensions are then examined with a 100x magnification and oil immersion under an optic microscope. Finally, the images are captured with a camera that is attached to the microscope by an adaptor.

### **3.3. Pre-processing techniques**

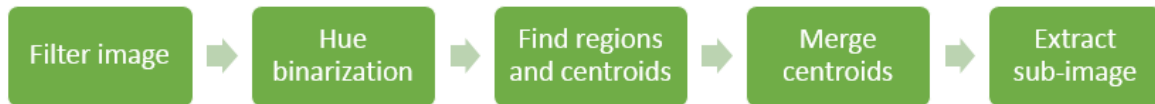
As it has been mentioned, two pre-processing methods have been developed in this thesis. The purpose of these techniques is to analyse the images and extract valuable information that helps to detect parasites. The pre-processing step aims to prepare and modify the images in order to make the job of the feature extraction and classification blocks easier, especially when a limited number of images is available. The first proposed technique tries to detect parasites by performing an image binarization based on the hue component. On the other hand, the second method aims to do cell segmentation to detect and separate the red blood cells. Afterwards, the cells are processed to see whether or not they contain parasites. In both cases, the output of the pre-processing technique is a set of sub-images that are susceptible to be or contain a parasite. The decision of whether or not the sub-image is classified as a parasite will be taken by the chosen classification method.

#### **3.3.1. Hue binarization**

In Giemsa stained images, parasites appear highlighted in purple colour. This difference in colour can be exploited when performing binarization in order to segment the parasites from the background. However, the contrast of the images is too low to obtain good results consistently if the binarization is applied to grey images. Binarization on red, green and blue components has also been tried with no success, but applying this technique to



the hue component of the HSV image has shown excellent results. Before entering into details, let us see a summary of the steps this pre-processing method takes and explain them in an orderly manner.

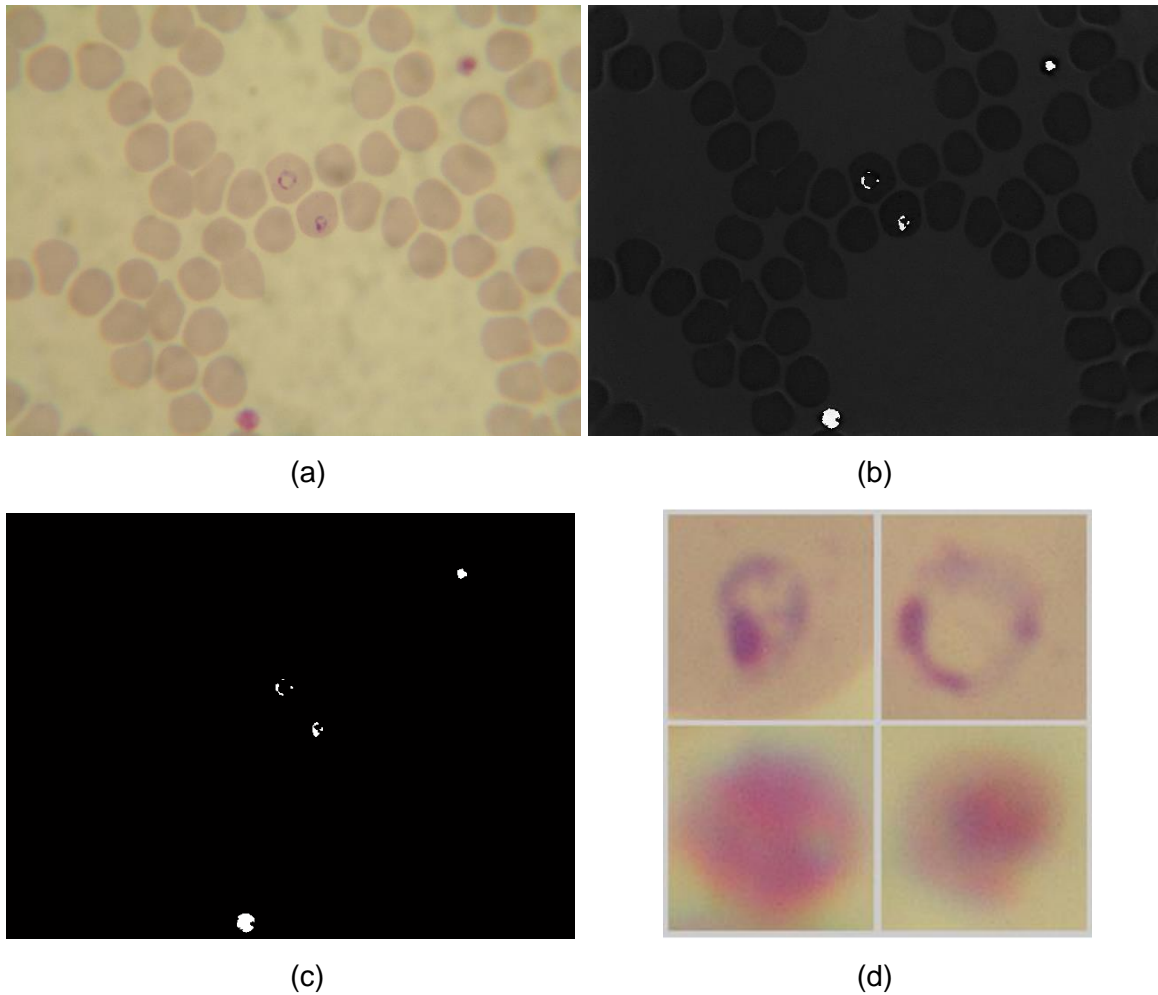


**Figure 3.3: Block diagram of hue binarization pre-processing**

Firstly, the original image is filtered and enhanced. A median filter is used for smoothing of colour image and a Laplacian filter is applied for edge sharpening [4]. The median filter is a non-linear digital filtering technique, used to remove noise from images. In median filtering, each pixel is replaced with the median of its neighbouring pixel values. In this case, the size of the median filter used is 3x3, so the image is not modified considerably. Laplacian filters take the second order derivative of the pixel. These filters highlight regions of rapid intensity change and are often used for edge detection. In our case, the result of the Laplacian filter is subtracted from the filtered image to achieve an improvement in sharpness.

After these enhancements, the image is converted to HSV format and the hue component is extracted. In Figure 3.4, examples of an original image (a) and its hue component (b) are shown. Note that the grey-scale image of the hue component is very shiny on those areas where there are parasites, platelets or white cells but very dark on those areas where none of them is present. This high contrast allows for a good binarization of the image that finally segments stained element from the foreground.

To perform this binarization into black and white images, a value for the threshold must be assigned. One common option in this cases is Otsu's threshold. This threshold is the optimal value that separates the pixels into two classes so that their combined spread (intra-class variance) is minimal. However, Otsu's threshold presented errors in a small number of cases that were finally solved by multiplying the threshold value by a coefficient around 0.95. Another viable option is to set a fixed value for the threshold that binarizes correctly all the images, although this method would be very experimental. Figure 3.4 (c) shows an example of a binary image obtained at the end of this process.



**Figure 3.4: (a) Original image (b) Hue image (c) Binary image (d) Potential parasites**

Once the binary image is obtained, we proceed to extract the sub-images of the candidates. We name candidates to the detected stained areas that may or may not be a parasite. First, we need to detect the bright areas resulting from the binarization and the position of their centroids. Apart from the already mentioned blood elements, there are artefacts in the original image that produce bright areas on the binary image. In general, this is not a problem as these artefacts will be treated as candidates and probably classified as non-parasites. However, in some cases, they generate a large amount of very small areas which have negative influence on the results. For this reason, only regions with an area bigger than 10 pixels are considered and smaller ones are discarded. This value is chosen considering the size of the parasites and the generated areas.

In addition to this, another problem that must be faced is the fact that stained objects usually create more than one area. In other words, a single parasite may create several

bright areas that if treated independently, would lead to the detection of several parasites when in fact, only one parasite is present. In order to solve this problem, areas that are close to others are merged and considered as one. In fact, not the areas but the centroids are merged. When the distance between two centroids is under a threshold, they are combined and create a new centroid with their average positioning. The value of this threshold is an important parameter to consider. A small value will produce no joins, while a high one will produce too many. After several trials, a value of 70 pixels is chosen, as it performs the best.

When this process is ended, we have a set of centroids that correspond to candidates to parasites. Finally, we take the surrounding pixels around the centroids to create a sub-image of the candidate. The size of the sub-images is 121x121 pixels. This size is decided taking into account two considerations. Firstly, when the candidate corresponds to a parasite, the entire area of the parasite must fit in the sub-image. Secondly, the size must not be too big, in order to avoid that other elements are included in the sub-image. In particular, we want to avoid that non-parasite elements are included in parasite sub-images and vice versa, because that would hinder the classification.

It is also important to consider those centroids near the edges of the image. When taking the surroundings of these centroids, we could be exceeding the image margins. The solution applied to this problem is to slightly move the centroid so the whole sub-image fits inside the original image. By doing this, we guarantee that the candidate is still in the sub-image and that all the pixels in the sub-image correspond to real pixels in the original image.

In Figure 3.4 (d), the final parasite candidates are shown. Although the binary image has a big amount of small areas, their centroids have been combined into four candidates that correspond to the stained regions in the original image. The two sub-images above contain parasites while the two below correspond to non-parasite elements.

These sub-images are the input elements to the feature extraction block. Before the training stage, the sub-images are labelled into parasite-class or non-parasite class. This procedure is easily done by checking if each sub-image is near a labelled parasite in the original image. During the test stage, sub-images are obviously not labelled because we do not know if they correspond to a parasite and that is what we try to discover.

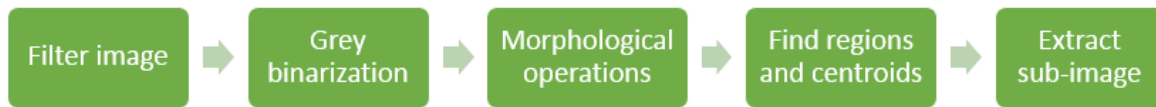
### 3.3.2. Cell segmentation

In thin blood smears, red blood cells are spread along the film and parasites can be seen within them. The goal of this pre-processing technique is to segment all the cells from the foreground and then process them individually. In an ideal situation, cells can be easily distinguished and separated from each other. However, in many cases, red blood cells appear superimposed and accumulated in certain regions. The Figure 3.2 at the beginning of this chapter shows an image where all elements are spread and easy to segment. On the other hand, in Figure 3.5, we can see an example of a smear where cells have been clumped together. As we will further explain, this is a problem that we must face and that makes the segmentation difficult.



**Figure 3.5: Blood smear with clumped cells**

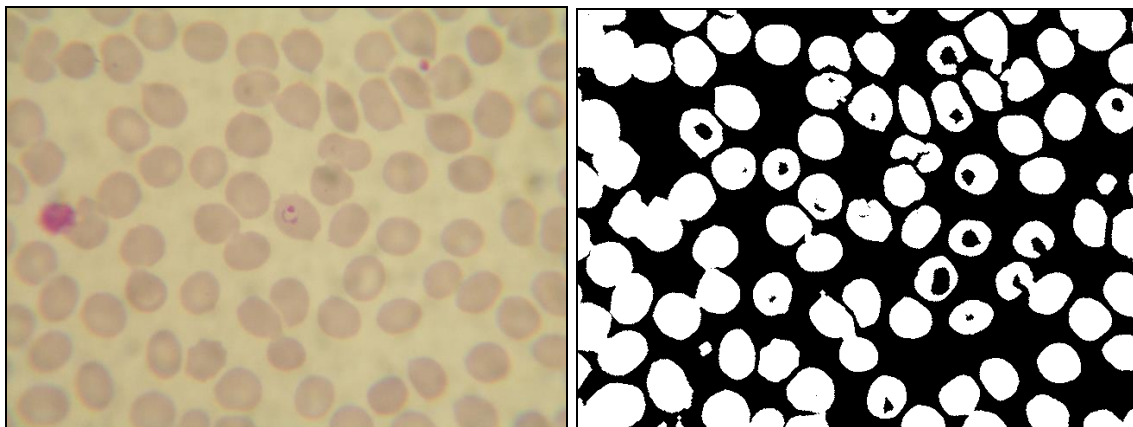
Most of the research conducted in cell segmentation for malaria diagnosis uses images with low cell density and low clumping [4]. Indeed, this pre-processing technique is very powerful with images where the cells are well spread. However, dealing with difficult images requires the implementation of new ideas and algorithms that can face all the different situations. In the following lines, we will explain which kind of problems we have encountered and the proposed solutions for each of them. But before, let us see a scheme of the steps this pre-processing method takes.



**Figure 3.6: Block diagram of cell segmentation pre-processing**

Firstly, as it was done in the hue binarization pre-processing, the original image is filtered and enhanced. A median filter is used for smoothening of colour image and a Laplacian filter is applied for edge sharpening.

After the quality of the image is improved, we perform a binarization of the grey-scale image. The aim of this process is not to segment the parasites but the cells, so we do not focus on the stain colour in this pre-processing technique. Binarization on other components has been tried but with worse results regarding the shapes of the cells obtained. The binarization has been performed using the Otsu's threshold. Afterwards, another 3x3 median filter is applied to the binary image in order to remove noisy small elements. In Figure 3.7, the original image and the binary image obtained after this process are shown.



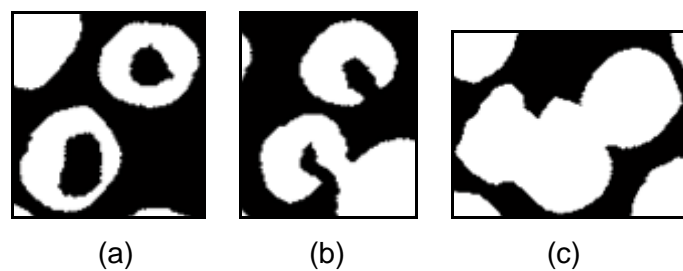
**Figure 3.7: Original and binary image on cell segmentation**

We can appreciate many independent areas in the binary image that correspond to cells in the original one. Ideally, these areas are elliptical, with no holes and not connected to other elements. However, it can also be seen that some areas present some complications that we enumerate and explain below:

- **Cells with holes:** When binarized, the central part of some cells appears in black, as if it was not part of the cell.

- **Broken cells:** Similarly to the holes, some bordering parts of the cell appear in black. In this case, the dark region is not contained inside the bright area.
- **Clumped cells:** When cells in the original image are superimposed, the binary image presents large white areas corresponding to several cells. Usually, only two or three cells are joined in the same area but there are cases of large masses that contain tens of cells.

Examples of these issues are shown in the pictures at Figure 3.8.



**Figure 3.8: (a) Cells with hole (b) Broken cells (c) Clumped cells**

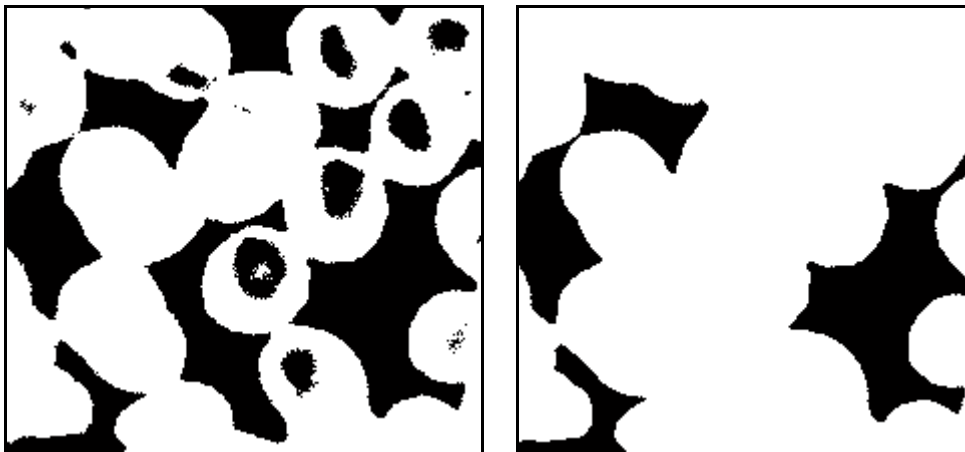
In order to solve all these problems, morphological operations are performed. To understand the proposed solutions, it is important to realize that the final goal is to select some areas that must fit in a sub-image size and that should correspond to cells. Keeping this in mind, let us comment the options we have tried to deal with the problems and the measures finally taken.

Filling the holes inside an area does not represent a great challenge because there are morphological functions that provide the option to do it. Then, those cells with a hole inside can be converted into the compact bright area that corresponds to the real cell. In the other hand, filling the spaces on broken cells is complicated. Notice that the detection of these open holes when they are not totally contained on white areas is really difficult. After trying several algorithms, we have decided not to modify this broken cells. As we will explain in a few lines, all areas are treated as potential cells and if they fulfil some conditions, they will be selected. Hough transform may be a solution to this problem in some cases as it can be applied to detect circular shapes.

Regarding the problem of clumped cells, different morphological operations have been tried to split the masses into independent cells. The final decision was to use an opening

filter with a diamond structuring element. This filter allows to split some cells that are in contact but its size must be checked in order to avoid deformations on the cell shapes.

Therefore, we will implement two measures at an initial stage: filling holes and applying an opening filter. However, the order of these two operations has a relevant impact on the results (morphological operations are non-linear). When the function that fills holes is executed, it does not distinguish between holes inside cells and holes created by the space between cells. Differentiating both cases is difficult and as a consequence, large masses can be created, worsening the problem of clumped cells. An example of this is shown in Figure 3.9, where a group of cells (left) is finally converted into a big white area (right) because of an undesired filling of intra-cellular space.

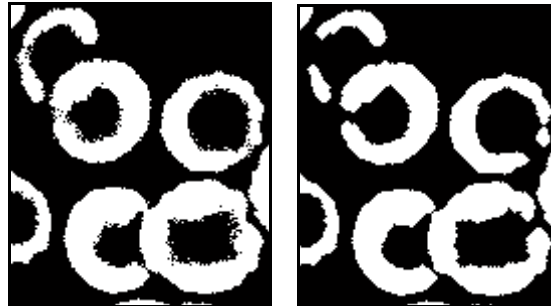


**Figure 3.9: Example of undesired hole filling**

On the other hand, when doing a morphological opening, some joins between areas are broken. This eases the cell segmentation but aggravates the broken cell problem, as new cells are broken. Hence, when doing the opening before, it is impossible to fill the holes on many of these new broken cells. This issue is showed in Figure 3.10. On the left, we see a group of cells with a hole inside and after the filtering, they appear opened on the right.

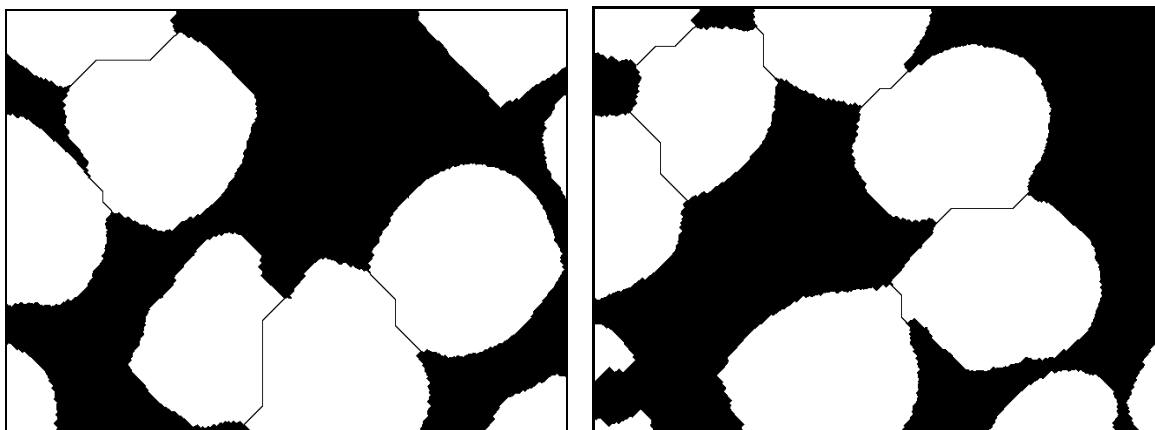
At the end, the decision taken is to apply a small opening filter that does not break too many joins and afterwards do the filling of holes. This choice is made trying to preserve as many cells as possible and considering that more techniques will be applied to solve the problem of clumped cells and big areas. In fact, a combination of distance transform and watershed segmentation is applied. Distance transform allows to create an image

where each pixel has a value that measures its closeness to a border. Then, by applying watershed segmentation, we can find local minimums that help to perform segmentation.



**Figure 3.10: Example of an opening filter breaking cells**

The raw watershed transform is known for its tendency to oversegment an image. As we do not want to deal with small areas, a common trick in watershed-based segmentation methods is to filter out tiny local minima and then modify the distance transform so that no minima occur at the filtered-out locations [14]. This is called minima imposition and allows for very good results in our case. Two different examples of this are shown in Figure 3.11. In both cases the watershed algorithm has split the regions into areas similar to the original cells.



**Figure 3.11: Example of opening filter breaking cells**

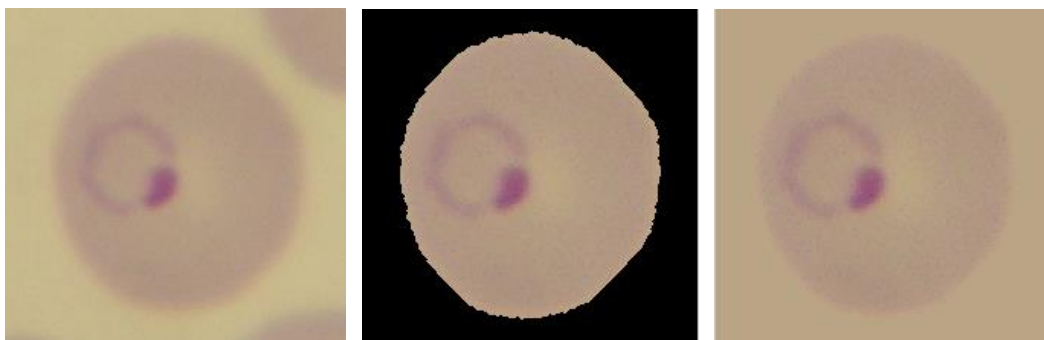
Once the segmentation is finished, we proceed to extract the segmented areas and their centroids. Because of the problems mentioned before, we can find small areas corresponding to pieces of cells or even platelets. On the other hand, big areas where segmentation has not been achieved may also be present. For this reason, the selected areas must be inside a range that guarantees that they can be a red blood cell. These



margins must not be very restrictive in order to allow that broken cells are considered as candidates. Considering the typical area of a cell, the final values taken for the margins are 20.000 and 50.000 pixels, so every area value between these values will be selected.

Finally, the last step consists on the sub-image extraction. We want to create a bigger image because we must capture the whole cell and not only the parasite. Thus, a 221x221 sub-image is generated for each candidate, but there are different ways to do it. Firstly, we can take the surrounding pixels of each centroid to create the sub-image. However, when doing cell segmentation, this does not make true sense because we only want the inside of the cells. In fact, when the 221x221 surrounding pixels are taken, other elements outside the cell are also included in the sub-image, such as platelets or white cells that harm the subsequent processing. After analysing the results, we can affirm that this option does not perform well enough.

The correct approach is to generate a 221x221 sub-image and copy only those pixels inside the detected region. Then, a decision must be taken for the values outside the cell area. The two proposed options are to set a black background or to assign the average value of the image. Both approaches are viable but an average background performs better in all the cases. In Figure 3.12, the different options just mentioned are shown. From left to right: no changes in background, black background and average background.



**Figure 3.12: Sub-images with different background**

In the left picture, we can appreciate that other portions of cells are also contained in the sub-image. This is not desirable because these foreign elements can disturb the classification step. Black background images do not provide good results because feature extraction methods focus on the high contrast between background and foreground and not on the cell details.

Finally, in Figure 3.13 we show some examples of sub-images that have been generated using this cell segmentation technique. The three left pictures are cells without parasites but notice that the third one contains part of a platelet. The red blood cell and the platelet are superimposed in the original image and the algorithm interprets the platelet as part of the cell. The right picture corresponds to an infected cell with a parasite inside.



**Figure 3.13: Sub-images after cell segmentation**

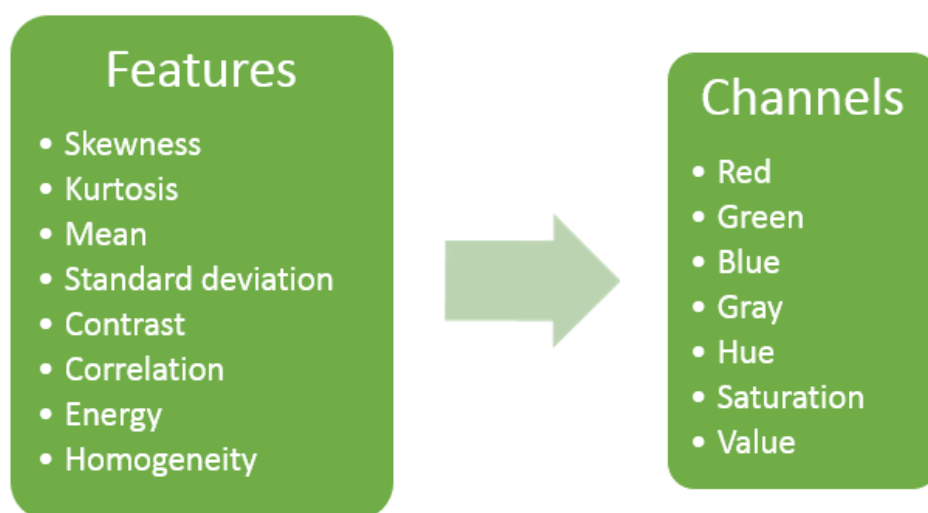
It is important to mention that this pre-processing technique has its limitations. We have just commented the case of the platelet attached to the cell but the most important drawback is the fact that, when the algorithm is unable to segment big areas of clumped cells, they are discarded. This causes that some parasites are not selected in the pre-processing and then, will not be processed. In fact, around 28% of the parasites are discarded by this pre-processing technique. As we will comment in the Future development chapter, new algorithms and solutions need to be considered in order to improve the results of the cell segmentation method. Despite this inconvenience, this technique shows outstanding results when sub-images are processed. The fact that cells are very uniform and similar between them helps the processing tools to learn which cells contain parasites.

### **3.4. Feature extraction**

After the pre-processing stage is finished, we proceed to apply feature extraction techniques to the sub-images just obtained. The goal of this phase is to extract useful data that feeds the classification block and allows for a correct classification. As we will see in the following sub-section, two different techniques have been proposed for the classification block. The first one is Support Vector Machines (SVM) that performs classification from an input set of features. Therefore, feature extraction is an indispensable step before using SVM.

On the other hand, convolutional neural networks are the other classification technique. The inputs to these networks are the sub-image generated in the pre-processing block and no previous feature extraction is required. In fact, during the training, the network learns to find some edges, shapes and textures by itself. This procedure is in some way similar to a feature extraction. Taking this into account, the feature extraction explained in the following section is only applied when the classification method is SVM.

It is clear that choosing an appropriate set of features is the most determining step for a posterior correct classification. Features which give predominant differences between normal and infected cells must be identified and used for training purposes. After consulting all the research regarding that issue [4], and considering our own proposals, we have chosen a set of features based on colour and statistical concepts. Geometrical features have been considered as well but they did not improve the performance in our case. For instance, selecting the area, perimeter and radius of the cells are reasonable options when working with cell segmentation images. In fact, infected cells tend to have larger area and we could exploit that fact as a feature. However, in practice, these parameters do not enhance but even worsen our classification results. This is probably due to the great variety of sub-images obtained after the pre-processing. Another reason for that is the number of problems that we commented in the pre-processing section, especially regarding cell segmentation. The area found on a sub-image could not match the real area of the cell because of cell clumping or bad segmentation.



**Figure 3.14: Features selected for SVM training**

The selected features are colour and statistical based, which means that they mainly work on histograms. We have finally selected 8 different features that are calculated over 7 different colour components, making a total of 56 different features. In Figure 3.14, we show a list of these features and a list of the colour components which they are applied to. All these parameters have their own optimized implementation in Matlab, which allows for a faster training and testing.

Ideally, these magnitudes should be very different when calculated over parasites and non-parasite sub-images. For instance, the interior of a cell containing a parasite is less homogeneous than an empty cell. However, this may be true for a certain colour component but false for another. Then, it is useful to know which features are the most important for the classification, or in other words, which features does the SVM really use when classifying. To achieve this, feature selection has been applied. The main goal of feature selection is to reduce the number of features by selecting the more relevant ones. It also helps in preventing overfitting when the number of features is too big compared to the number of observations. In our case, we do not want to eliminate features but to know their relevancy. There are different methods to implement this idea such as forward selection or backward elimination. We have executed a forward selection where we have finally obtained the 10 most relevant features. This operation has been done separately for both pre-processing techniques. In Table 3.1, we summarize the most important features for both pre-processing methods.

Rank	Hue Binarization	Cell segmentation
1	Skewness - Saturation	Skewness – Green
2	Skewness - Hue	Mean - Blue
3	Skewness - Blue	Kurtosis - Red
4	Skewness - Green	Skewness - Blue
5	Standard Deviation - Red	Skewness - Hue
6	Energy - Saturation	Homogeneity - Red
7	Homogeneity - Red	Homogeneity - Green
8	Energy - Red	Energy - Red
9	Mean - Saturation	Skewness - Saturation
10	Homogeneity - Grey	Contrast - Green

**Table 3.1: Most relevant features in SVM classification**

We can observe that some of the most relevant features for each pre-processing technique are common and some others differ. From the table results, we determine that skewness is a relevant parameter for many colour components. It can also be seen that features related to RGB components are more relevant than those applied on HSV components. However, these results are obtained from a relatively small dataset. With more images and several executions, we could determine more accurately the relevance of these features.

To end up with the feature analysis, let us just say that all features are normalized when training and testing. In other words, each sub-image feature is divided by the same feature calculated over the whole image. With this practise we are preventing the system from failing when working with images with slight changes in luminosity.

### **3.5. Classification**

Once the feature extraction is done, we proceed to perform classification. The aim of this block is to train an automatic system that is capable to classify the sub-images into parasites or non-parasites. To do that, two different techniques based on machine learning have been proposed. The first one is Support Vector Machines (SVM) that performs classification from an input set of features. On the other hand, convolutional neural networks are an example of deep learning techniques that aim to do classification by training a neural network. We are somehow trying to work on two different models, a conventional and a novel one, in order to check their performance and compare results.

In both cases, the classification technique (or the previous feature extraction block) can be fed with the input sub-images of both pre-processing methods. The only consideration to make is that in one case we are pre-processing parasites and in the other, we are pre-processing cells. When doing classification, this will not be a problem, but during the extraction of results we must bear this in mind. For example, a cell containing two parasites will be treated once if cell segmentation is applied. However, the sub-images of both parasites will be considered when doing hue binarization.

In all of these machine learning techniques, the available dataset must be divided into train and test. A validation set is often necessary too but it is usually extracted from the train dataset. The training set is used in the first step and allows the system to learn how

to classify properly. Afterwards, we use the test dataset to evaluate the trained system and compute the results.

### 3.5.1. Support Vector Machine (SVM)

As it has been commented in previous sections, Support Vector Machines perform classification from a set of features and not from the raw image. In particular, given the class and a list of features for each sub-image, the SVM can learn to separate the two categories. Therefore, in the training step, the SVM is given a matrix whose rows correspond to different observations (sub-images) and whose columns are the calculated features. In other words, each column contains the information of a particular characteristic. In addition, a column containing the classes of each observation is also entered. Then, when doing classification, the SVM is only fed with a row containing the features of the sub-image to classify.

Although the training of a support vector machine is simply achieved with these two inputs, there are many parameters that can be tuned and that allow this technique to adapt to each problem. In particular, the choice of the kernel function used to train the SVM is a critical issue that clearly influences the posterior classification. A linear kernel that defines a hyperplane in the feature space has been tried at first. However, enlarging the feature space with a RBF or a polynomial kernel has shown better results, as it increases the feature dimensions. More flexibility in the hyperplane and margins definition is achieved and thus, better classification results. In particular, the Gaussian kernel (RBF) is the one that performs the best for both types of pre-processed sub-images.

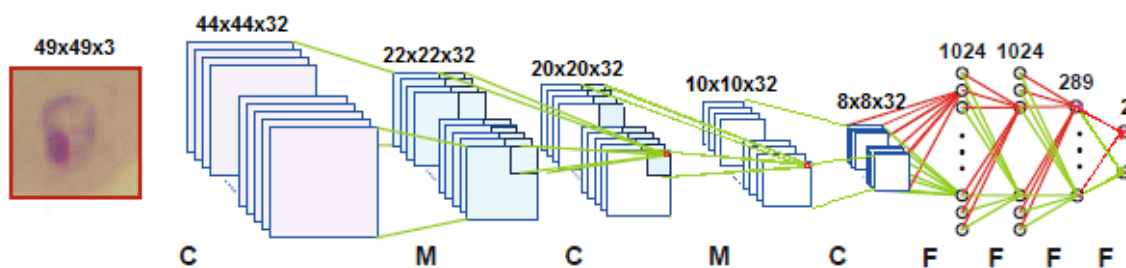
The implementation of a SVM is very simple because a previous task of feature extraction has been performed. When training the system and when classifying a new sub-image, the relevant and necessary information has been extracted in the previous block. Therefore, the task of the SVM is only to find a proper classification model with the features previously found.

### 3.5.2. Convolutional Neural Networks (CNN)

The alternative to the traditional SVM classification is the novel structure of convolutional neural networks. As we have explained before, the input to these networks is the raw sub-image generated in the pre-processing block and no previous feature extraction is

applied. In the training step, we must also know the class to which the image belongs. Then, new images can be classified with the trained CNN that we obtain. Before entering the details of the training, let us present the network used in this project to perform the classification.

In Figure 3.15, the main architecture of the CNN proposed is shown. Notice that the displayed sizes correspond to the inputs and outputs of the layers and the type of layer is stated with the initials C (convolutional), M (max pooling) and F (fully connected). In other words, the activation maps and their dimensions are represented in the picture and the layers are implicit between them. After the last fully connected layer, there is a softmax classifier that decides into which category is every sub-image assigned.



**Figure 3.15: CNN architecture of the system**

In order to decide the structure of the CNN, we have examined the research conducted that was related with malaria, cells or microscopy. This particular structure is an adaptation of a proposal [7], which has been modified in order to fit our data and to perform binary classification. However, other options have been tried during the development of this project. For instance, training a well-known CNN structure, such as LeNet, has shown poorer results.

Regarding the network structure, the first issue to comment is related with the input. The sub-images obtained in the pre-processing block are inputted directly without a previous feature extraction, as we have already mentioned. The same network is used for both hue binarization and cell segmentation pre-processing techniques. The only consideration is that the required input size of this CNN is 49x49, while the size of the obtained sub-images is 121x121 or 221x221, depending on the pre-processing method applied. This mismatch has been solved by simply resizing the sub-image before entering the network.

The architecture of the proposed CNN is quite common. At first, some convolutional layers and pooling layers are interleaved; in particular, three convolutional layers and two max pooling layers. Then, a group of four fully connected layers progressively diminish the size of the outputs that are finally classified by a softmax. The filter size is chosen as 6x6 for the first convolutional layer and 3x3 for the remaining two. The max pooling layer uses a window of size 2x2 with a stride of 2, which reduces the height and width of the activation maps by half. The activation function of all the convolutional and fully connected layers is a ReLU, except from the last fully connected layer, whose activation function is chosen to be a sigmoid. A summary of all the layers and its main characteristics is shown in Table 3.2.

Num.	Type of layer	Input size	Output size	Activation Function
1	Convolutional	49x49x3	44x44x32	ReLU
2	Max. Pooling	44x44x32	22x22x32	-
3	Convolutional	22x22x32	20x20x32	ReLU
4	Max. Pooling	20x20x32	10x10x32	-
5	Convolutional	10x10x32	8x8x32	ReLU
6	Fully connected	8x8x32	1024	ReLU
7	Fully connected	1024	1024	ReLU
8	Fully connected	1024	289	ReLU
9	Fully connected	289	2	Sigmoid

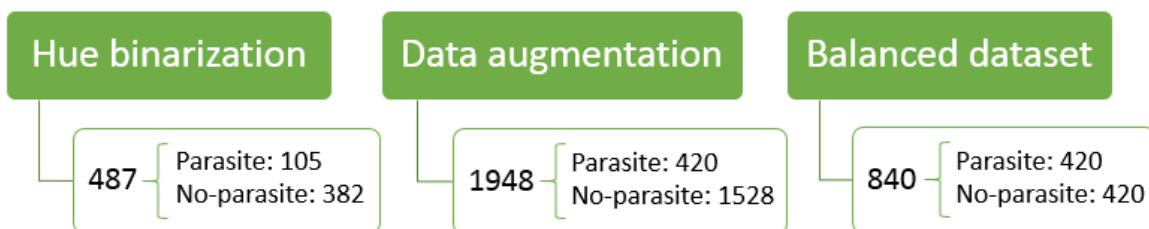
**Table 3.2: Summary of the layers on the CNN**

The first step is to perform a training of the convolutional neural network from the available sub-images. One particular characteristic of CNNs is that they often require a big amount of images in order to be trained correctly. Notice that these networks have a lot of parameters and the process of learning all their values is complicated and thus, they require a considerable number of images. As we have access to a limited amount of sub-images, we have performed data augmentation techniques in order to enlarge our dataset. In particular, we have applied three different modifications to each sub-image: a vertical flip, a horizontal flop and both of them simultaneously. In total, we have multiplied by four the total number of sub-images as three new images have been generated from each original one.



Another important aspect to consider is the fact that there are more sub-images corresponding to non-parasites. This situation is more severe in the cell segmentation pre-processing, as many non-infected cells are present. This imbalance can lead to incorrect trainings as the network learns based on the class probabilities and not on the image contents. Therefore, a measure that has been studied and considered is to balance the dataset used for the training and testing. This is simply done by selecting the same number of parasite and non-parasite sub-images. Finally, this measure has been adopted as it has been proved that it enhances the classification results. In the following lines, these issues will be commented individually for each pre-processing technique.

After the hue segmentation pre-processing is finished, a total of 487 sub-images are generated. From these, 105 belong to the parasite category and 382 to the non-parasite category. After the data augmentation is done, we obtain a total amount of 1948 sub-images with 420 parasites and 1528 non-parasites. Then, when we balance the database, we obtain 840 images that are equally distributed into the two classes. A summary of these figures is shown in Figure 3.16.

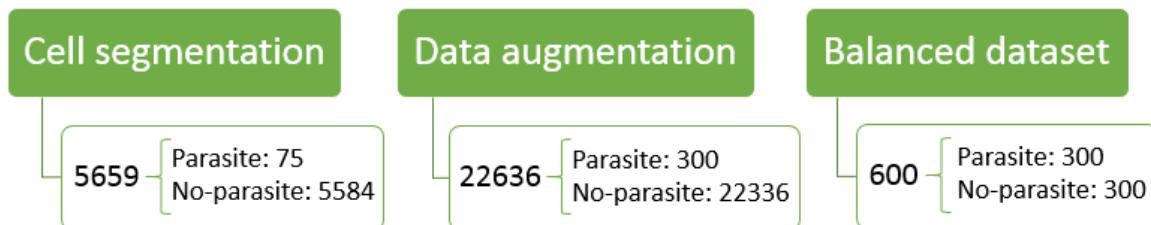


**Figure 3.16: Number of hue binarization sub-images in the CNN training**

As we have mentioned before, the final choice is to work with a balanced dataset that will allow for a more accurate training and better classification results. The 840 sub-images are divided with a proportion of 80% for train and 20% for test. This results in 672 images for the training step and 168 for the testing. In order to check the obtained results, we have done cross-validation by randomly assigning the sub-images to train or test according to the previous percentages. This process has been repeated 10 times, so 10 different divisions of the dataset have been tried.

In the case of the pre-processing by cell segmentation, a total of 5659 sub-images are generated, from which 75 correspond to parasites and 5584 to non-parasites. After the data augmentation process, we obtain a total amount of 22636 sub-images with 300

parasites and 22336 non-parasites. Then, when we balance the database, we obtain 600 images that are equally distributed into the two classes. These numbers are summarized in Figure 3.17.



**Figure 3.17: Number of cell segmentation sub-images in the CNN training**

In this case, we apply the same percentages for train and test, as well as the same cross-validation method. Then, we use 480 images for the training step and 120 images for the testing.

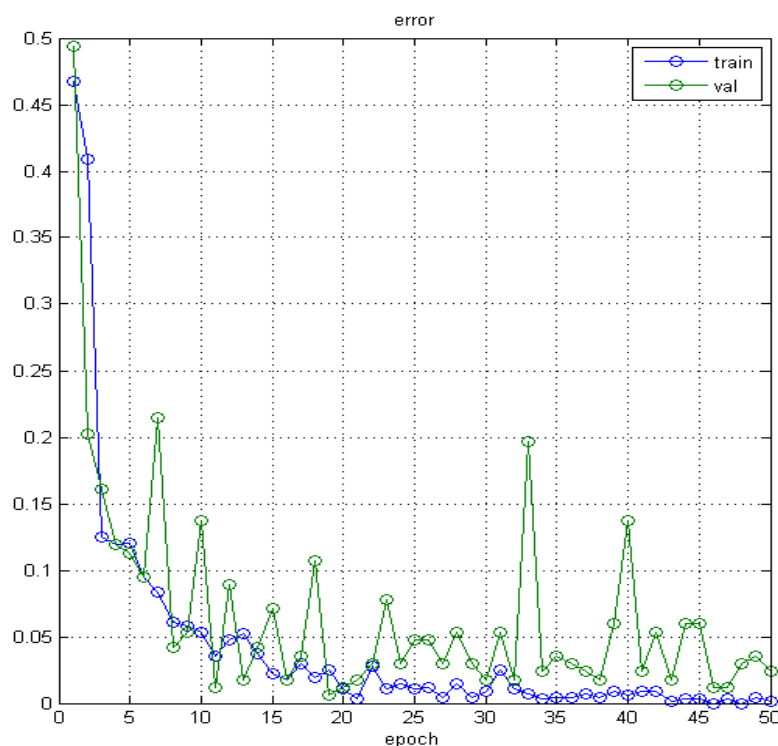
It can be seen that the usage of cell segmentation pre-processing generates more images than hue binarization technique. This happens because there are far more cells (infected or not) than stained areas that can be a candidate to parasite. However, it is important to notice that the amount of parasite elements is lower when using the pre-processing that segments cells. There are two reasons for this, the first of which is the fact that some cells may contain more than one parasite. The second reason is the already commented problems occurring on cell segmentation pre-processing that lead to the discard of many infected cells.

Let us now comment the details of the training, with the chosen options and the values for the relevant parameters. The input images are divided into batches of 20 and batch normalization is applied at the input and after each ReLU function. Remember that batch normalization aims to solve some training problems related with the distribution of the activations.

The optimization method used for backpropagation is the gradient descent algorithm with a learning rate of 0.001. Regularization is added to the training with a weight decay of 0.0005. Different values for these rates have been tried but these have proven to be the most appropriate ones. Another important issue that must be considered to do a correct training is the network initialization. All the weights must be set to a value different from

zero at the beginning of the training step. In this case, we assign a random value to all the weights, which is obtained from the size of the network and the coefficient 1/100.

The convolutional neural network is trained for 50 epochs. This means that the process of forward computation and backpropagation of the error is repeated 50 times with all the images of the training set. In Figure 3.18 an example of the training results is shown. We can observe the evolution of the training error and the validation error through the different epochs of the training.



**Figure 3.18: Evolution of the training and validation errors**

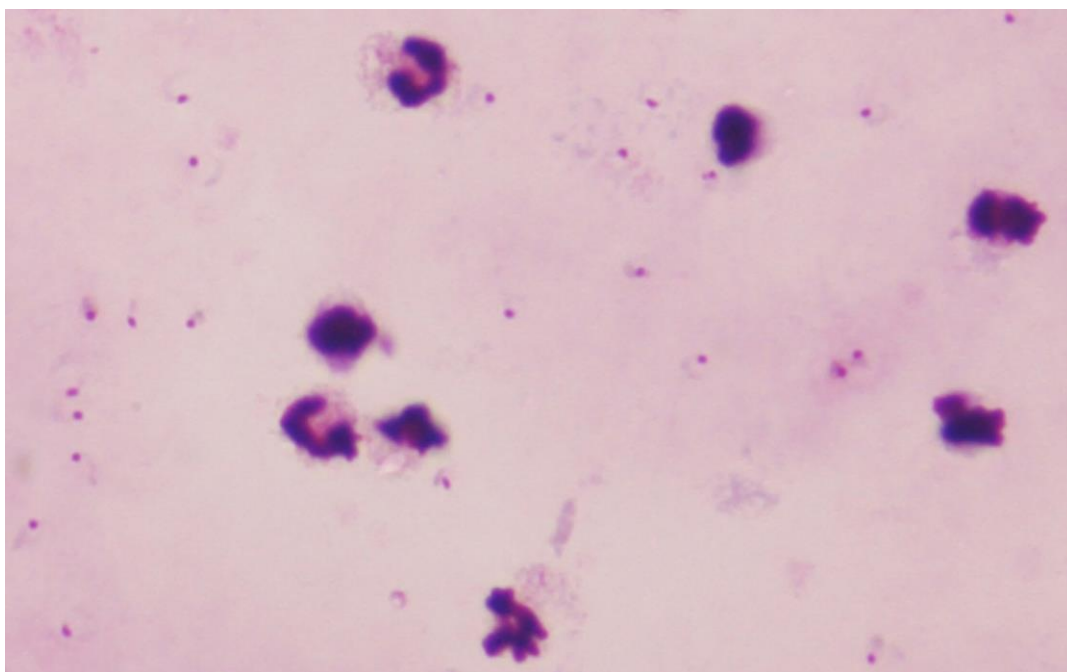
The training error is displayed in blue and the validation error is plotted in green. The first thing to note is that in general terms, the validation error diminishes along the training but it flickers locally. We have chosen a plot where the validation error flickers considerably in order to highlight this issue, but usually the convergence is more obvious. Each point corresponds to a particular training for the network and each training is not necessarily better than the previous one. Therefore, strategies must be defined to decide when to stop the training and which specific training is chosen for the CNN. This decision should be made considering that we want to minimize the validation error. Some possible

options are to take the global minimum or to look for a stable minimum that does not flicker for a number of epochs.

The whole training of the network along 50 epochs takes about 400 seconds in a laptop without GPU. The duration of this process would be longer if more images were added to the dataset. Anyway, the training must only be done once, so the fact that is time-consuming is not a problem. After the training is finished, the network is able to classify new images into the two categories. The time necessary for a new image classification is insignificant, so once trained, the system is very fast.

### 3.6. Other developed work

Before having access to the thin blood smear images, we worked on images corresponding to thick blood smears, like the one shown in Figure 3.19. At first, we only had access to a couple of these images so we started the project by trying to implement pre-processing and classification tools on them. In these images, a greater amount of blood is analysed and the red blood cells are not visible. Instead, parasites can be distinguished directly as they appear as little dots over a uniform background. In the figure shown, a considerable number of these parasites is present, as well as some white cells, which have also been stained but have a bigger area.



**Figure 3.19: Malaria thick blood smear**

The developed work regarding these images has a very similar structure as the system presented before. Firstly, the grey image is binarized with the Otsu's threshold and the bright areas are taken to generate sub-images. This is the same process done in the hue binarization pre-processing but it is applied to grey images. Similar pre-processing filters, such as median and Laplacian filters, are also added at the beginning.

After the sub-images are generated from the centroids and the areas are obtained, we proceed with the feature extraction. The selected features are the same ones that have been explained in the previous sections. Then, these features feed a support vector machine that is trained to perform binary classification.

As we only have two images of this type, we use one for the training and one for testing and thus, we cannot extract conclusive results. However, we observe that the pre-processing tools work correctly with these images because the sub-images are generated properly. Moreover, the feature extraction and the training of the SVM work fairly well considering the limitations of the dataset. Many sub-images are classified into their category and fewer are misclassified, generating some false positives and false negatives.

In conclusion, the aim of this part of the project is not to extract conclusive results but to show the possibility of working with this different type of malaria images. We believe that a bigger dataset will allow to extract useful results as it has been possible with the thin blood smear images.

## 4. Results

The goal of this chapter is to expose all the results obtained from the development of the project. Then, we aim to analyse the performance of all the proposed techniques in the diagnosis of malaria. We will give numbers and percentages of the results of our classification methods and we will also comment their benefits and drawbacks. Additional comments on the different blocks of the system will be made but always related with the results.

Firstly, we should recall that all the available images that we have correspond to positive samples. In other words, there is at least one infected cell with a parasite in each image. Therefore, the extraction of some results will be complicated as we do not have negative images. For this reason, the results commented in this chapter mainly focus on the sub-images obtained from the pre-processing techniques. The results extracted from the sub-image datasets will probably be more valuable and trustable. In any case, we will comment all the results and specify their origin.

The most important parameters that we will evaluate in this chapter are the sensitivity and specificity. We will calculate their values for each technique and give their confidence interval. In particular, we will always calculate the 95% confidence interval for these proportions (sensitivity or specificity). This interval contains the true proportion 95% of the times that the procedure for constructing the confidence interval is employed. The method used for the calculation is the Clopper-Pearson interval [15].

The information in this chapter is divided into two sections, depending on the classification method applied. Inside each of these chapters, there are more sub-sections depending on the type of sub-images used for the extraction of the results. Then, the goal is to compare the classification methods but also to investigate which pre-processing methods show better results.

Before entering the details of the performance results, let us make some comments about the speed of the used techniques. As we have commented, CNNs are very rapid in the classification but their training requires some time. The training of support vector machines is considerably faster and the classification is almost immediate. However, the training of a classification tool must be done only once, so a slower training is not a

problem. Regarding the pre-processing techniques, hue binarization is done very fast (about 1 second). On the other hand, applying cell segmentation on an image takes about 10 seconds. Then, when doing this technique, this amount of time will be spent in each classification. Although it is considerably slow compared to hue binarization, taking 10 seconds for a malaria diagnosis is very remarkable.

#### **4.1. Results with SVM classification**

This section summarizes all the results regarding support vector machine techniques. Firstly, we will comment these results when using each of the two pre-processing methods of our system. After this, a new option that combines both methods and improves the results will be proposed and explained. We will comment the results on each sub-section individually, and at the end of this section, Table 4.1 and Table 4.2 summarize all these results. A comparison of the techniques based on the obtained results is also done at the end.

In all the methods where the SVM is used as a classifier, the dataset of original images has been divided into 49 images for the training and 38 images for the testing. After the system is trained, the test images are inputted to the system, where they go through all the blocks in the system architecture. Finally, the sub-images generated from the original image are classified into parasite or non-parasite. It is important to mention that when working with SVM, the datasets are not balanced. The results do not substantially change when balancing the number of images in each category.

##### **4.1.1. Hue binarization**

In the following lines we will expose the results obtained when classifying the hue binarization sub-images with a SVM. To begin with, it is important to recall that this pre-processing technique focuses on detecting the presence of parasites. This means that we must evaluate the results considering the exact number of parasites in each image. This is relevant when cells contain more than one parasite inside themselves. In these cases, all the parasites must be classified, meaning that two or more sub-images must be considered.

After the training is finished, we evaluate the results of 38 images that correspond to 180 sub-images. From these candidates, 46 belong to the parasite category and 134 correspond to non-parasite. Then, let us comment firstly the results based on the sub-images, also called candidates. Among the 46 parasite sub-images, 28 are classified correctly when these methods are applied, leading to a sensitivity of 60.87% and a confidence interval of [45.37%, 74.91%]. On the other hand, 133 out of the 134 non-parasite sub-images are classified correctly. Thus, the specificity in this classification is 99.25% and the confidence interval is [95.91%, 99.98%]. Based on these results, we observe that only one false positive is generated, as only one non-parasite candidate is misclassified into the parasite category. This means that only a 0.75% of the non-parasite sub-images is badly classified and generates a false positive.

Regarding the original images, we can also obtain some partial results. From the 38 parasitized images that we initially have for testing, we detect a parasite in 27 of them, which means a hit rate of 71.05%. Additionally, in one case out of 38, a false positive is generated. Although this false positive does not harm the performance because all the images in our dataset are positive, we can consider that the false positives will occur in negative images in the same manner. Therefore, in a 2.63% of the original images, a false positive is produced.

#### **4.1.2. Cell segmentation**

In this sub-section, we will expose the results obtained when classifying the cell segmentation sub-images with a SVM. Unlike the hue binarization case, we do not focus on a direct detection of parasites, but on an analysis of the cells to see if they are infected. This means that those cells containing more than one parasite are only analysed once and thus, the number of positive sub-images is smaller.

In particular, the results are extracted from a total amount of 2497 sub-images generated from the same 38 original images. From these candidates, 44 belong to the parasite category and 2453 correspond to non-parasite. Notice that the total amount of sub-images is much higher in this case because there are significantly more cells than stained areas in each image. However, the number of positive sub-images is slightly lower as some of these cells contain more than one parasite.



In this case, 28 candidates are classified correctly among the 44 positive sub-images. This implies that the sensitivity with cell segmentation and SVM classification is 63.64% and its confidence interval is [47.77%, 77.59%]. On the other hand, 2450 out of the 2453 non-parasite sub-images are well classified. Thus, the obtained specificity with these methods is 99.88% with an interval of [99.64%, 99.97%]. A slight increase in the number of false positives is produced as we observe that three non-parasite candidates are misclassified into the parasite category. However, this means that only a 0.12% of the non-parasite sub-images is badly classified and generates a false positive.

Although the global results are the ones just shown, a deeper analysis is necessary to understand better the performance of this technique. As we have mentioned in previous chapters, the proposed cell segmentation method fails in some cases, causing the discard of cells. In particular, this pre-processing technique discards an infected cell in about 28% of the cases and is effective 72% of the time. With this in mind, we can deduce that the hit rate for the SVM classifier is about the 88.5%, but the global rate decreases to 63.63% due to the previous pre-processing efficiency of 72%.

As done in the previous case, we can also contextualize these results in the original images. From the 38 parasitized images that we initially have for testing, we detect a parasite in 28 of them, which means a hit rate of 73.68%. On the other hand, we obtain a false positive in 3 of these images, meaning that a 7.89% of the images generate a false positive.

#### **4.1.3. Combination of results in SVM classification**

After analysing the previous results, we observe that the sensitivity of both methods may not be high enough. Therefore, we propose a new method that joins the results of both pre-processing sub-images. The proposal consists on doing an addition of results similar to a logical function OR. If any of the two techniques classifies a sub-image as a parasite, we decide that the region correspond to a parasite. Then, we do not need that the results of both methods match, because they are added.

When classifying a sub-image, we know its exact coordinates on the original image. Thus, we can list the detected parasites and its positions in the image for the two methods explained before. The combination of results is done by merging those positions whose

distance is under a certain value. This means that when two detected parasites are close enough, they are considered as one. If both methods detect a parasite, the positions are merged and the parasite is detected. Moreover, if only one of the methods detects the parasite, we keep the detected position and consider that region as a parasite. The threshold that decides when to merge two parasite positions is an important parameter that has finally been set to 110 pixels. This value has been chosen taking into account the typical size of parasites and cells.

It is important to note that although the parasite detection is improved, the false positives are also added. If only one method generates a false positive, this detection is considered as correct, increasing the false positive rate. This method based on the addition of results can be applied because both methods have a low rate of false positives. If not, it would perform poorly and show a low specificity.

As a summary, this combination of results aims to sum the results of two techniques that do not excel. As close parasites are finally merged, the number of candidates is very similar to the case of cell segmentation. For a better understanding, we take the same total numbers as the cell segmentation case, consisting on 2497 sub-images from which 44 are parasites. With this new method we detect correctly 35 of them, leading to a sensitivity of 79.55% and a confidence interval of [64.70%, 90.20%]. The specificity is 99.84% and its interval is [99.58%, 99.96%], but these values do not provide relevant information in this case. It is more important to mention that the number of false positives generated has increased as a result of this combination. In fact, we obtain 4 false positives, which is the exact sum of the numbers obtained in the two combined methods. This implies a false positive rate of 0.16% in the sub-images.

Regarding the 38 original images, 33 of them are well classified and a hit rate of 86.84% is obtained. However, the percentage of images that generate a false positive increases to 10.53%.

#### **4.1.4. Comparison of SVM classification methods**

In this sub-section we will make some additional comments on the obtained results, focusing on the comparison between the methods proposed. In table 4.1, we summarize all the results related with the sub-images that have commented before. Regarding the

sensitivity, we can observe that cell segmentation sub-images generate slightly better results than hue binarization candidates. Obviously, the combination method improves considerably the sensitivity of the previous techniques, achieving a hit rate of 79.55%.

The specificity is also better with cell segmentation sub-images if analysing the relative frequencies (percentages). However, the absolute value of false positives is higher for segmented cells. This occurs because the number of non-parasite candidates in this case is drastically higher, as there are many non-infected cells. Then, although the probability of misclassifying a negative candidate is lower, the big amount of sub-images of this type leads to a higher generation of false positives.

When using the combination method, the false positives of both methods are aggregated, worsening this problem. Thus, the specificity is lower than the cell segmentation case. However, the specificity of the combination method is higher than the hue binarization one, because there are more negative candidates, like in the cell segmentation case.

Results	Hue binarization	Cell segmentation	Combination
Total num. of candidates	180	2497	2497
Num. of parasite candidates	46	44	44
Num. of parasite candidates detected	28	28	35
Percentage of parasite candidates detected (sensitivity)	60.87%	63.64%	79.55%
Num. of no-parasite candidates	134	2453	2453
Num. of no-parasite candidates well classified	133	2450	2449
Percentage of no-parasites well classified (specificity)	99.25%	99.88%	99.84%
Num. of no-parasite candidates wrongly classified (false positives)	1	3	4
Percentage of no-parasite candidates wrongly classified (false positives)	0.75%	0.12%	0.16%

**Table 4.1: Summary of the results of SVM methods: Sub-images**

Let us now analyze the global results regarding the original images, which are specified in Table 4.2. These results are directly related with the ones explained before. In the first row, we show the number of images that were classified as parasite from the total 38. We can observe that a better sensitivity is obtained with cell segmentation pre-processing rather than hue binarization. The best hit rate is achieved by the combination of both methods though. However, we can see that the hue binarization pre-processing presents the lowest generation of false positives, which would finally traduce in a better specificity.

As a reminder, our dataset only contains positive images. We are calculating the percentage of images that generate a false positive but these misclassifications do not harm the performance because false positives cannot harm a positive image. However, we can assume that this percentage is the same for positive and negative images. Then, the obtained rates of false positive generation are a measure of the specificity of the system.

In fact, when a false positive occurs on a positive image, it improves the classification. In some cases, the parasite may not be detected but the false positive helps to classify the image as positive. For all these reasons, the results extracted from the global images have a relative importance. Negative images are needed to extract complete results and give certainty to the obtained ones.

Results	Hue binarization	Cell segmentation	Combination
Num. of images classified as parasite	27	28	33
Percentage of images classified as parasite (sensitivity)	71.05%	73.68%	86.84%
Percentage of images with false positive	2.63%	7.89%	10.53%

**Table 4.2: Summary of the results of SVM methods: Images**

Once we have shown all the results we can choose the best method from the SVM classification group. Considering the sensitivity, the combination method is the best choice as it improves considerably the hit rate. However, this is achieved at expense of a lower specificity and a higher generation of false positives. As we will see, CNNs perform better, so we do not need to make a final choice yet.

## 4.2. Results with CNN classification

In this section, we summarize all the results extracted from the application of convolutional neural networks. Firstly, we will analyse the results when feeding the CNN with each of the two pre-processing methods of our system. In each sub-section, we will comment the results individually. At the end of the chapter we will compare all the methods proposed and explain the final choice with its benefits. A summary of the results is shown in Table 4.3.

As it has been explained in chapter 3, it has been necessary to work with a database of sub-images independently from the original images. Then, the major and most useful part of the results is extracted from this set of sub-images. In order to improve the training and testing results, we have implemented data augmentation techniques and balanced the databases. A set of these sub-images is used to train the CNN and another set is used to do the test and perform classification. The main aspects of these methods will be specified in each sub-section.

Anyway, some tests with original images have been done in order to extract results. In this case, the images must go through all the blocks in the system, from the pre-processing to the classification. If we had a higher number of images, we could obtain more consistent results and in particular, do a better evaluation of the results obtained from the original images.

### 4.2.1. Hue binarization

When we train our network with hue binarization sub-images, we work with a balanced dataset of 840 candidates, meaning that there are 420 sub-images of each category. As they are divided with a proportion of 80% for train and 20% for test, this results in 672 candidates for the training step and 168 for the testing. Therefore, we test our trained network with 84 positive sub-images and 84 negative sub-images.

During the training of the CNN, the validation error decreases in general terms but flickers in every epoch. Then, we must decide which epoch and which trained network we finally choose. A possible choice is to select the final epoch, in our case the 50<sup>th</sup>, and use the last training for the network. Another viable option is to select the network with the lowest

validation error from all the epochs. With this choice, the CNN usually shows zero validation error.

As we do not have a big amount of sub-images, we have repeated the process of training, selecting the trained network and evaluating the results 10 times. In each iteration, the division into train and test is randomly modified but always keeping the same proportions. With this procedure, we aim to obtain more consistent results and to avoid outstanding or terrible performances due to a particular dataset division.

When we select the trained network from the last epoch, the results are the following. From the 84 parasite candidates, only 2.2 on average are misclassified, leading to a sensitivity of the 97.38% with a confidence interval of [91.30%, 99.63%]. Regarding the negative sub-images, an average of 82.8 candidates is well classified and thus, the resulting specificity is 98.57% with an interval of [93.15%, 99.94%]. This means that on average 1.2 negative sub-images generate a false positive among the total 84 negative candidates. The obtained rate of false positives is 1.43%.

On the other hand, if the epoch with lowest validation error is chosen, the trained network generates the following results. The sensitivity increases to 99.76%, as an average of 83.8 positive images is classified correctly from the total 84. Its confidence interval in this case is [95.23%, 100%]. The specificity grows to 99.88% because only 0.1 negative sub-images are badly classified and the obtained confidence interval is [95.47%, 100%]. This implies that a false positive is generated only in 0.12% of the candidates. We can observe that the performance clearly improves when choosing the best training for the network.

When trying the detection on original images, the results match the numbers just shown. Depending on the training, one or at most two images may not be classified as parasite. However, we require more images to perform an independent training, validation and test that could show the exact results on original images. As expected, false positives are generated with a low rate.

#### 4.2.2. Cell segmentation

In the case of pre-processing with the cell segmentation technique, we have many more images after the data augmentation. In particular, the dataset is composed by 22636 sub-images with 300 parasites and 22336 non-parasites. However, we work with a balanced database and our final dataset is composed by 600 candidates with 300 sub-images from each category. As done in the previous case, sub-images are divided with a proportion of 80% for train and 20% for test, resulting in 480 candidates for the training step and 120 for the testing. Therefore, we test our trained network with 60 positive sub-images and 60 negative sub-images.

When we feed the CNN with these sub-images, the results are outstanding with regard to the sensitivity. The positive candidates are always classified correctly and thus, the sensitivity is always 100% with a confidence interval of [94.04%, 100%]. We have validated this results by changing the sub-images that belong to train and test and the 60 positive sub-images are always well classified.

On the other hand, some negative sub-images are misclassified and generate false positives. In order to obtain more accurate results, we have tested all the negative sub-images in the non-balanced dataset. Remember that only 300 negative candidates from the total 22336 have been used until now in these method. This evaluation shows that 0.22% of the negative sub-images generate a false positive. Therefore, the specificity in this case is 99.78% and its confidence interval is [99.71%, 99.89%].

As it happens in all the classification methods that use cell segmentation sub-images, the results are worse if analyzed from a global point of view. As it has been explained before, this pre-processing technique discards some cells and is effective only in 72% of the infected cells. In this case, this 72% is the final sensitivity because the hit rate of the classification is 100%.

If these methods are tried in original images, we observe a behavior that matches the explanations just given. In about 75% of the images, the correct parasites are not found because of pre-processing issues. Moreover, the generation of false positives is quite high because there are many non-infected cells. In particular, about 10% of the original images present a misclassification that leads to false positive detection.

A summary of the results explained in this sub-section and the previous one are shown in the Table 4.3. Notice that these results are the ones extracted from the sub-images. In cell segmentation sub-images, we do not distinguish two different trainings because the results do not substantially change between these epochs.

Results	Hue binarization (last epoch)	Hue binarization (best training)	Cell segmentation
Sensitivity	97.38%	99.76%	100%
Specificity	98.57%	99.88%	99.78%
False positive rate	1.43%	0.12%	0.22%

**Table 4.3: Summary of the results of CNN methods**

### 4.3. Final comparison of the results

As a summary, we have proposed several methods that combine pre-processing techniques with classification methods. Three different methods rely on a SVM classifier while two of them are based in a CNN. We have commented its benefits and limitations in the previous sections and the final choice and the reasons that motivate it are exposed now.

Firstly, we observe that convolutional neural networks show a great performance in the classification of our sub-images. Checking if an improvement in classification was possible with CNNs was the main goal of this project and it has been proven with the obtained results. The sensitivity and specificity with our CNN are much better than the ones obtained using a SVM. In this section, we do not further comment the SVM results as they have already been exposed and they are not as good as the CNN results.

Once that the better performance of CNN is demonstrated, we must choose which pre-processing sub-images allow to obtain the best results. If we only focus on the classification results, cell segmentation seems to be a promising option because of its perfect sensitivity. However, the already exposed problems related with cell segmentation pre-processing make that the results are not so brilliant. Firstly, the actual sensitivity considering the discard of cells is about 72%. Secondly, the false positive rate is not so



low if we consider that a lot of non-infected images are generated by this pre-processing method.

With this in mind, the final choice is to use the hue binarization pre-processed images to feed our CNN. If we select the best training epoch, the sensitivity in the sub-images is 99.76% and the specificity 99.88%. With the current images, these are the best results we are able to achieve and they are very satisfactory. More trustable results can be obtained if the number of available images increases.

## 5. Budget

The aim of this chapter is to analyse the costs of the project, especially focusing on its development. At first, we will expose all the expenses that must be considered in order to develop the project. Then, we will compare the costs of the current methods for malaria diagnosis with the costs of diagnosing malaria with our system. Many of the figures in this chapter are estimated and hence, the main goal of this budget is to do an analysis of the costs and the financial viability of the project.

### 5.1. Development costs

In this sub-section, we discuss the costs of developing this project of malaria diagnosis. The expenses and figures given correspond to the costs of developing this project on a real work environment. In Table 5.1 this information is summarized and in the following lines, we explain the different costs that are listed.

The first and main cost to analyse is the salary of the employees. We have considered that the development of the system requires an engineer that is fully dedicated to this task. A project coordinator that advises and supervises the progress of the project is also necessary. Based on the amount of hours actually spent in the project development, we have decided that the developer needs to work 576 hours and the coordinator should only dedicate 108 hours. We have considered that they spend 32 and 6 weekly hours respectively during 18 weeks. The salary received by the coordinator is higher as it demands more responsibility.

We also must take into account the expenses related with hardware and software. We suppose that a laptop is rented during 4 months at the price shown in Table 5.1. A GPU and a server are also necessary in the development of the project for the training and testing of images. Finally, the Matlab license is one of the most relevant costs to consider.

Additionally to the development of the system, which is the core of the product, we must also elaborate some other related tools whose cost must be included. A Graphical User Interface and a user guide are needed in order to facilitate the usage of our system. Moreover, we must consider the cost of preparing a training course for the diagnosis technicians.

<b>I. Personnel</b>		<b>Total</b>
Project Developer	576 hrs x 15€/hr	8.640€
Project Coordinator	108 hrs x 20€/hr	2.160€
	<b>Subtotal</b>	<b>10.800€</b>
<b>II. Other Direct Costs</b>		
<b>Hardware and Software</b>		
Laptop	4 months x 80€/month	320€
GPU		200€
Server		400€
Matlab license		2.000€
	<b>Subtotal</b>	<b>2,920€</b>
<b>Associated Tools</b>		
GUI development		600€
User guide development		300€
Training course preparation		400€
	<b>Subtotal</b>	<b>1.300€</b>
<b>III. Indirect costs</b>		
Office rent	450€/month x 4 months	1.800€
Light, internet and telephone	120€/month x 4 months	480€
	<b>Subtotal</b>	<b>2.280€</b>
	<b>TOTAL</b>	<b>17.300€</b>

**Table 5.1: Summary of the development costs of the project**

Finally, we compute the indirect costs related to the renting of an office, as well as the light, telephone and internet expenses. These costs are calculated considering the duration of the project development which is 4 months. With all these expenses in mind, the final estimated cost of developing the project is 17.300€.

It is important to note that only development costs are considered in this section. If the system is finally distributed, some additional costs should be regarded. For example,

there are expenses associated with conducting the training course. Furthermore, there are also costs regarding the maintenance service, such as a hotline service.

## **5.2. Comparison with current diagnosis methods**

The purpose of this section is to evaluate the costs of malaria detection with our system in comparison with the current diagnosis methods. Then, we must analyse both techniques, focusing on its costs and the time spent on the medical tests. In this comparison, we focus on the diagnosis of malaria in medical centers of our country. Some details of this analysis are different if we compare our system with the diagnosis in underdeveloped regions.

Current methods for the diagnosis of malaria rely on manual microscopy, i.e. after the smear is prepared, a doctor analyses under the microscope and checks the presence of parasites. The first thing to note is that the preparation of the blood film is exactly the same in both cases. The methods differ once the smear is allocated under the microscope and is ready to be observed.

The existing methods require that a member of the medical staff spends 2-5 minutes viewing the sample under the microscope. Depending on the parasitaemia density, the duration of this analysis can vary. On the other hand, our system is very fast as it only requires to take some pictures of the smear. The processing of the obtained images and the further detection of malaria are almost immediate. Therefore, the main advantage of our system is that it is very time-efficient. Moreover, our system can be used by a technician, who may not have superior medical knowledge and whose salary may be lower. The two elements just commented entail a reduction of the salary costs.

However, our system requires an initial investment that must be considered. Firstly, the digitalization of the images requires some support. For example, the images used in this project are obtained with a camera attached to the microscope with an adaptor. Another option is to take the pictures with the mobile but an adaptor is also required then. Anyway, a moderate expense has to be considered in this regard. Furthermore, the acquisition of the software and the conduction of a training course are also costs to take into account.

As a summary, our system allows to reduce the costs on malaria diagnosis in the long-term due to a reduction on the time needed for the parasite detection. However, it requires an initial investment to acquire the software, the necessary knowledge and some additional components. In underdeveloped countries, the daily number of malaria tests that must be diagnosed is higher. Therefore, this reduction in the time spent for the diagnosis is crucial in such environments.

## **6. Conclusions and future development**

Until now we have only presented the specific conclusions related with the results. In this chapter, we will expose the general conclusions of the project. Firstly, we will evaluate the fulfilment of the project objectives and afterwards, we will present the conclusions. Finally, we will explain the next steps that should be taken and possible extensions to further develop the project.

### **6.1. Objectives covered**

During the execution of such a long project, there are inconveniences and difficulties that force to reconsider the initial goals and their scope. However, in our particular case, these objectives have not been greatly modified. In the following lines, we comment their main aspects and their level of compliance.

#### **1. Study the different imaging tests for diagnosing malaria and evaluate which of them is more suitable to accomplish the following objectives.**

The first objective does not present many difficulties by itself, as there is much information available about malaria diagnosis. The problem was to have access to a appropriate collection of images that we could exploit in our processes. In this regard, the assistance provided by the colleagues in the Tropical Medicine and International Health Unit at Drassanes was key. Then, the decision was to work with Giemsa-stained blood thin smear images.

#### **2. Investigate what research approaches have been conducted on automatic techniques for the diagnosis of malaria. Propose a solution based on these conventional image processing techniques.**

After analysing several articles and publications, we have concluded that most of the research conducted on malaria diagnosis focuses on image processing. Moreover, the machine learning technique used for the classification is usually SVM. We have compiled much of this research and proposed our own solution based on support vector machines. In fact, we have developed two solutions corresponding to the two different pre-processing techniques.

### **3. Research on deep learning techniques and focus on the implementation of a CNN-based solution.**

We have found a considerable amount of theoretical information about these networks from several sources. However, as it is a novel technique, it is hard to find practical information and examples that help in the development of a CNN model. We have found articles that applied these networks on cell detection problems. Finally, a system based on CNN has been proposed and analysed with several configurations, input images, etc.

### **4. Compare the results and performance of the used techniques.**

Once the two techniques have been correctly developed, we have compared the results and demonstrated that convolutional neural networks are able to improve the performance of malaria diagnosis.

## **6.2. Conclusions**

After this thesis is finished, we can draw conclusions about the work developed and analyse its main aspects:

1. Malaria is a severe disease causing many deaths per year and its diagnosis needs to be improved. Current methods are time consuming and prone to errors but automatic techniques are able to solve many of these inconveniences.
2. The blood smear images can vary in type and stain. Moreover, there are four species of parasite that can affect the humans and they can appear in several biological stages. All this diversity implies that the developing of an automatic technique is not simple and straightforward.
3. Convolutional neural networks offer the possibility to improve considerably the results of many classification problems and in particular, the detection of *Plasmodium* parasites. Although SVM is a useful tool, CNNs have shown an outstanding performance.
4. From the two pre-processing techniques proposed in this thesis, hue binarization allows to obtain better results. However, the classification results with cell segmentation are also great and its main drawback is the ineffectiveness it shows to select all the cells.

5. The automatic techniques proposed in this thesis are a major opportunity to improve the diagnosis of malaria in the current health system, because they lead to a reduction of the time spent in the analysis of smears. Moreover, they are even more relevant in underdeveloped regions, where the amount of malaria cases is higher.

### **6.3. Future development**

As it has been commented throughout this document, this thesis aims to be the first approximation for the improvement of malaria diagnosis, but much work still remains to be done. In this sub-section, we will enumerate some of the next steps and extensions of the project.

1. The cell segmentation pre-processing promises to be a useful technique. We have encountered problems that lead to discard of cells and parasites, but if they were solved, the performance of this tool could improve. There are some image processing algorithms that may help in this task. For example, Hough transform can be applied to detect circular or elliptical shapes such as the cell contours.
2. A possible improvement is to establish more complex criterions than a binary classification. For instance, we could evaluate infected areas and compute the percentage of parasitized areas to evaluate the parasitaemia density. Another option is to give a score for the classification of parasites, so that we have a measure of the certainty of the detection.
3. All the work developed must be extended to other stages of the *P. falciparum* and other *Plasmodium* species. In addition, there are still many diseases that require for an automatic diagnosis technique, such as Chagas disease, Leishmaniasis and tuberculosis. Many of the tools exposed in this thesis can be exploited to improve the detection of these diseases.
4. In this vein, the techniques of this project should be adapted to deal with the different types of images and stains that are present. In order to do this, it is essential to have a big amount of images of every type.
5. In a final stage, an analysis on the system deployment should be made. An option is to load the software into a mobile that takes the photo of the smear. It is also viable that the images are sent to a computer that executes the system.



## **Bibliography**

- [1] WHO. World Health Organization. [<http://www.who.int/topics/malaria/en/>]
- [2] G. James, D. Witten, T. Hastie, R. Tibshirani. “*An Introduction to Statistical Learning: with applications in R*”, Springer, 2015.
- [3] F. van der Heijden, R.P.W. Duin, D. de Ridder, D.M.J. Tax. “*Classification, Parameter Estimation and State Estimation*”, Wiley, September 2004.
- [4] S. S. Savkare, S. P. Narote, “*Automatic Detection of Malaria Parasites for Estimating Parasitemia*”, International Journal of Computer Science and Security (IJCSS), Volume (5) : Issue (3) : 2011, pp: 310-315.
- [5] Pallavi T. Suradkar, “*Detection of Malarial Parasite in Blood Using Image Processing*”, International Journal of Engineering and Innovative Technology (IJEIT) Volume 2, Issue 10, April 2013, pp:124-126.
- [6] Yashasvi Purwar, Sirish Shah, Gwen Clarke, Areej Almugairi and Atis Auehlenbachs, “*Automated and unsupervised detection of malarial parasites in microscopic images*”, Malaria Journal, BioMed Central, Vol 10, Dec 13, 2011.
- [7] Yuanpu Xie, Fuyong Xing, Xiangfei Kong, Hai Su, Lin Yang, “*Beyond Classification: Structured Regression For Robust Cell Detection Using Convolutional Neural Network*”, 18th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), Munich, Germany, 8 pages, 2015.
- [8] Support Vector Machine. [[https://en.wikipedia.org/wiki/Support\\_vector\\_machine](https://en.wikipedia.org/wiki/Support_vector_machine)]
- [9] CS231n: Convolutional Neural Networks for Visual Recognition. Stanford University (2016). [<http://cs231n.stanford.edu/syllabus.html>]
- [10] CNN. [<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>]
- [11] CNN. [<http://neuralnetworksanddeeplearning.com/chap6>]
- [12] CNN. [[https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](https://en.wikipedia.org/wiki/Convolutional_neural_network)]
- [13] S. Ioffe, C. Szegedy, “*Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariance Shift*”. [<https://arxiv.org/pdf/1502.03167.pdf>]
- [14] Watershed transform. [<http://blogs.mathworks.com/steve/2013/11/19/watershed-transform-question-from-tech-support/>]
- [15] Binomial proportion confidence interval: Clopper-Pearson interval. [[https://en.wikipedia.org/wiki/Binomial\\_proportion\\_confidence\\_interval](https://en.wikipedia.org/wiki/Binomial_proportion_confidence_interval)]