Deep Learning for Identifying Malaria Parasites in Images

Carlos Sánchez Sánchez

Master of Science

Artificial Intelligence

School of Informatics

University of Edinburgh

2015

Abstract

The purpos of this project is to build a detection system for malaria parasites in images. This would avoid some of the current obstacles of malaria diagnosis such as the lack of expert technicians in low-resource areas. In contrast to most of the previous work we use thick blood smears instead of thin smears. These are the preferred ones for diagnosis, although they make the task harder.

We use Convolutional Neural Networks, a particular type of deep neural networks, to classify individual patches as containing parasites or not. Then, we use computer vision techniques such as non-maxima suppression to integrate the results and locate each parasite. The results are compared to those obtained with Extremely Randomized Trees and hand-designed features as used in previous work on this dataset (Quinn et al, 2014 [1]).

Our results are notably better than the models proposed previously. For a recall of 90% we have obtained a gain in the precision from 37% to 78%, which is enough to already improve the work conditions of laboratory technicians working on malaria diagnosis. Furthermore, the system can be easily extended to the detection of other parasites or blood components without much effort, so several tests may be run at the same time on the blood smears.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Chris Williams, for his guidance throughout all this project. I would also like to thank Pol Moreno for attending our meetings and providing useful comments and suggestions.

I would like to thank John Quinn from Makerere University, who originally proposed this project, for his cooperation and constructive comments.

I am grateful to my parents, Ángel and Manoli, and my sister, Cristina, for their support and for making it possible for me to come to The University of Edinburgh. Finally, thank you to my girlfriend, Lucía, as this MSc thesis would not have been possible without her continued encouragement.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Carlos Sánchez Sánchez*)

Table of Contents

Chapter 1

Introduction

Malaria is still a serious health problem in many areas of the world and its diagnosis is a key aspect to combat the disease. Microscopic analysis of blood samples is the preferred method. An expert must examine several blood smears looking for parasites to declare a patient infected or not, and the lack of such experts has been identified as one of the main obstacles in the diagnosis [2].

Computer vision has been proposed as a way to make this process easier and enhance the results of the diagnosis. If a computer is capable of locating the parasites in images the workload of the experts would be drastically reduced. In this project we propose a system to identify malaria parasites in images using state-of-the-art techniques. The system is intended to be used in low-resource areas, as it is there where it would be more useful. An example of how it might be used if it is not possible to enterly remove the experts is as a decision-support system. The computer vision program would finds candidates to be a parasite, so the technician only accepts them or not instead of looking for parasites in the entire image.

One particularity of this project is the type of blood smears used. While most of the previous work is focused on thin blood smears [3, 4, 5] we will use the dataset collected by *Quinn et al.* [1], which is composed by thick smears. These are the indicated smears for parasite detection, but they make the parasites harder to identify. The images have been taken under field conditions in health centres of Uganda and, as a result, many of them may not be in ideal conditions for the diagnosis of the disease. However, they reflect the samples used in this kind of installation, a requirement considering the purpose of the project.

All the methods used so far are ”shallow” methods. The authors design a set of features to build an appropriate representation of the image to feed a classifier. As a result, the performance of the detection system will depend on how good the representation is. The authors of the previous work on this dataset designed a set of shape features (elongation, perimter and others). Then, those features where used in a Extremely Randomized Trees (ERT) classifier. Contrary, in this project we use deep learning techniques. This means that a useful representation is not designed but learned by the model.

We use Convolutional Neural Networks (CNN), a particular case of deep neural networks which has been proved to be useful for object recognition [6, 7], being now one of the preferred machine learning methods for image related tasks. These are neural networks with a particular structure that makes them especially suitable for the spatial structure of images.

Similar to *Quinn et al.* [1], we will classify small patches of the image as containing a parasite or not. However, we will add a second stage where the results of classifying all the patches of an image are integrated to get the location of each parasite. The detection process will be built upon the non-maxima suppression technique of *Felzenszwalb et al.* [8].

Thus, as described above, the goals of this project are to build a full system to detect malaria parasites in images and compare the performance of CNNs to the results of ERT and hand-designed features. At the same time, we will analyse the effect of different aspects of the model in the results. Apart from the parameters of CNNs and the non-maxima suppression algorithm, we will analyse in depth what is the best way to extract patches from images and how to assign a score to each detection.

The results achieved by our system are notably better than previous work on this dataset. Furthermore, as we use thick smears, it would be more suitable for malaria diagnosis than other work focused on thin smears. The results of Convolutional Neural Networks yield a considerable improvement even when our detection stage is applied to the ERT model. Considering this, we will propose how the system could already be used to efficiently improve the work of laboratory technicians.

1.1 Dissertation outline

The outline of the dissertation is presented as follows.

Chapter 2 provides the needed background to use computer vision techniques in malaria detection. We introduce the relevant aspects of microscopic analysis, review previous work in malaria detection, and describe the state-of-the-art techniques of machine learning and computer vision that we use throughout the project.

In Chapter 3 we cover the first stage of the proposed system, which consists in the classification of individual patches. We first analyse different ways to extract the patches from the original images. Then we describe the training procedure and the details of the CNN models that we have considered. Finally, we include an evaluation section that describes the results obtained and compares the deep learning techniques to other models.

Similarly, Chapter 4 details the second stage of the system, which is the detection of parasites. In the first part we describe the methods we use. The second part is dedicated to the evaluation of the results of the full system.

Finally, in Chapter 5 we summarize the outcomes of the project, point out to some aspects that might be included in future work and discuss the immediate applications of the system.

Chapter 2

Background

In section 2.1 we review some of the basic aspects of malaria diagnosis and how computer vision has been applied to the identification of malaria parasites in blood samples. We present the data we have been working on and we focus on previous work with the same images.

Then, the rest of the chapter describes the particular techniques we will use to detect malaria parasites and the aspects that make them useful for this project. In section 2.2 we introduce deep learning and Convolutional Neural Networks, which we will use to classify image patches as containing parasites or not. Finally, in section 2.3 we describe the methods used to integrate the results of the classification of many patches to find the position of each parasite, how to use those results to improve the model and how we evaluate the system.

2.1 Malaria Diagnosis

Malaria disease is caused by *Plasmodium* parasites, which infect human red blood cells. According to the latest World Malaria Report around 198 million cases occurred during 2013 leading to 584,000 deaths [9].

Although treatments exist and the estimated number of cases has decreased by 30% since 2000 [9], the lack of appropriate diagnosis methods is still one of the main drawbacks in some developing countries such as Uganda [10].

The most effective method to diagnose malaria consists in examining a blood sample

to find malaria parasites using a microscope. Although the most common alternative are Rapid Diagnosis Tests (RTDs), they are not always effective and cannot compete with microscopic diagnosis [2].

2.1.1 Microscopic analysis

The microscopic analysis consists in looking for parasites in blood samples. There are four species of the *Plasmodium* parasite: *P. falciparum, P. vivax, P. ovale* and *P. malariae*. Furthermore, each parasite can appear in three different stages: trophozoite, schizont and gametocyte. As a result, the parasites may present different shapes. It is notable that *P. falciparum* and *P. vivax* account for 95% of the cases. Moreover, for *P. falciparum* only trophocites can be found in most cases. Sketches of the parasites are presented in figure 2.1. The most characteristic shape is composed by one or two bright red areas (chromatin) surrounded by a pale blue area (cytoplasm).

To examine the blood and to look for parasites a set of blood films need to be prepared. If the process is not carried out correctly the quality of the samples will be reduced, which commonly happens in health centres with limited resources. In these cases identifying the parasites will be harder. Apart from the natural components of the blood (white and red cells and platelets) other elements appear on the slides including bacteria, spores, fungi or dust. They can look like malaria parasites and cause confusion.

For the diagnosis, two different kinds of films are used. The most common type to look for malaria parasites is the thick film. It consists of many layers of cells where the red blood cells are destroyed and a large amount of blood can be quickly analysed. In thin films, however, only one layer of cells is present. The density of parasites is much lower, so it is not commonly used to detect parasites. It is used for other tasks such as to confirm the species of the parasite as the parasites there are much easier to identify [11].

Another important step in the preparation process is the dyeing of the sample. It makes it easier to recognize the parasites. Although Giemsa is the most common stain, in Uganda and some other countries, Fields is preferred as it is faster and cheaper [1]. Fields stain is the one used for the images of this project, which complicates the recognition task. Some examples of parasites can be seen in figure 2.2.

Figure 2.1: **a)**: sketches of trophozoite *P. falciparum* (thin film). **b)**: sketches of trophozoite *P. falciparum* (thick film). **c)**: sketches of parasites of other species or in a different stage (thick film) . From [11]

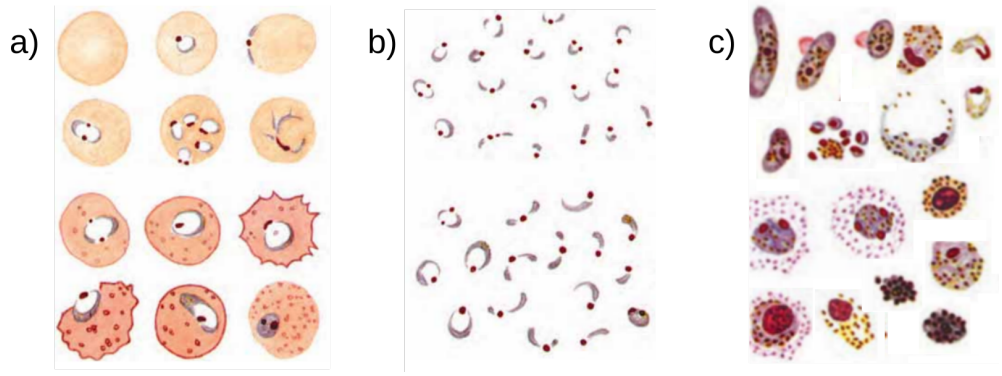
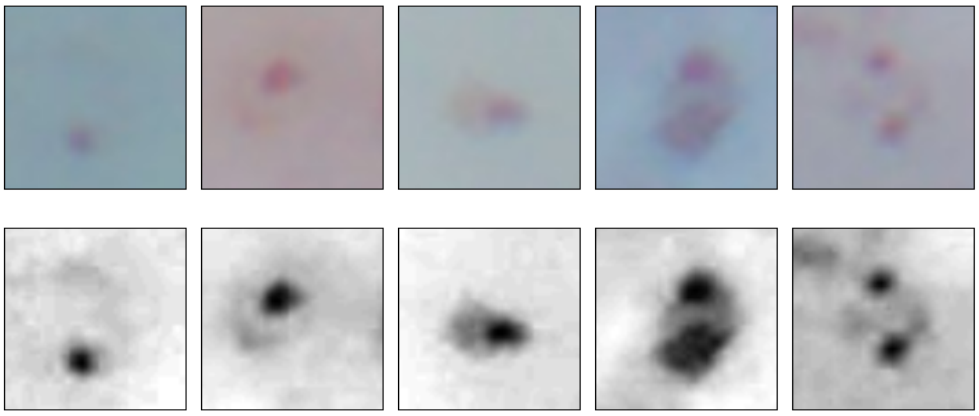


Figure 2.2: **Top:** images of parasites in the thin blood films used in this project stained with Fields A, B. **Bottom:** Same images in greyscale, each image has been scaled so that the brightest colour of the image is white and the darkest is black.



Once the samples are ready, a trained technician has to examine them looking for parasites. It is recommended that at least 100 fields (i.e. the area visible under the microscope) are examined before the sample is considered negative [11]. This involves a workload that makes it impossible to examine the samples appropriately for health centres with low resources.

Technicians with 5 weeks formal training to detect malaria parasites are expected to achieve an accuracy of 90% when classifying slides as containing parasites or not, with a set where approximately 40% of the slides are negative. Apart from identifying if there are malaria parasites present, it is usually necessary to quantify the number of parasites, which helps to know the severity of the infection and how the patient

responds to the treatment [11]. This is done by counting the number of parasites per white blood cell.

2.1.2 Computer vision

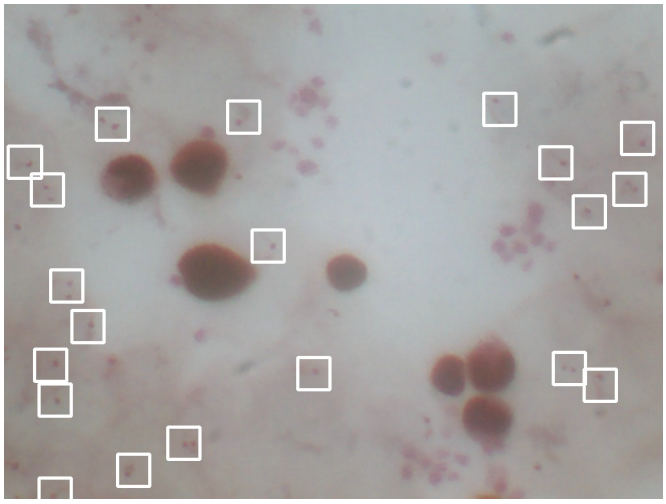
The lack of trained staff and the inability to cope with the workload to examine malaria slides are listed as two of the main limitations of malaria microscopy [2]. In particular, in the Kabarole District of Uganda, in only 17% of health facilities where laboratory services where available there were qualified laboratory technicians present [10]. The use of automatic computer vision diagnosis is an inexpensive solution. However, a computer program to recognize parasites has to deal with the differences in position, shape and colour of parasites while avoiding false positives caused by objects that look like parasites.

Even when thin blood smears are discouraged to diagnose malaria, most of the previous computer vision work has focused on them. In thin films the aim is to find infected red blood cells, which is considerably easier than detecting parasites in thick films. The common procedure consists of the segmentation of red blood cells to further classify them as infected or not. Several methods have been reviewed in [4].

Different features (color, shape, size, texture) are extracted from the red blood cells. Then, a classification algorithm is used to discriminate between infected and non-infected cells. Using k-nearest neighbors the authors of [3, 5] achieve a recall of 72.4% with a precision of 85.8%. This means that they find 72.4% of the total number of parasites, but 14.2% of detections are false positives. For another model using neural networks as the classifier a recall of 85.13% and a precision of 80.84% is reported [12]. They also address the problem of species differentiation.

This project is based on one of the very few studies with thick blood smears and computer vision [1], and is detailed in the following section. Another attempt to use thick blood samples was done in [13]. They used a mechanical system that allowed a computer to examine many fields of a slide to search for parasites. Their model extracted candidates according to the image histogram. Then the candidates were classified as parasites if their size matched the cromatin of a parasite. They do not report the performance of the parasite detection, but how well a slide is classified as positive (containing parasites) or negative after having examined many images. The accuracy for positive

Figure 2.3: A sample image of the dataset used in [1] and in this project. The white squares identify the parasites.



and negative samples is 95% and 68.5% respectively: the recall is high, but having many false positives.

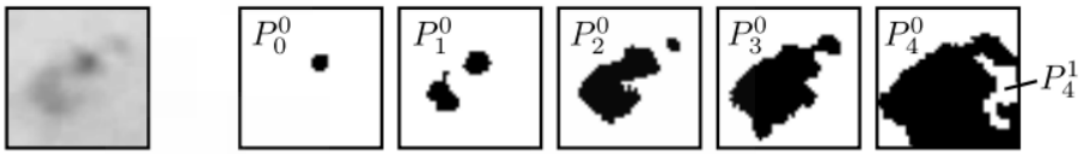
2.1.3 Previous work with thick blood samples

In [1], *Quinn et al.* propose a method to automatically diagnose malaria parasites in images. Their project is intended to work in under-resourced health centres with images taken from a phone camera. The images are taken under non- ideal field conditions and include thick films with Fields A stain. Both characteristics (the conditions and the stain) increase the complexity of the problem, but match more accurately the real situation of those places where a system like this one would be useful.

The dataset is composed by 2,703 1,024×768 images of 133 individuals. A group of annotators recorded the coordinates of 50,255 parasites. No information about the species or the stage of the parasites is available. An example image can be seen in figure 2.3.

This model classifies 50×50 patches as containing a parasite or not. It is composed by two stages: feature extraction and classification. First, the patches are thresholded at different levels (figure 2.4). For each one of the patches a set of shape features is extracted such as elongation, jaggedness and perimeter. The features are then used to train Extremely Randomized Trees, which are a particular case of a random forest. Many decision trees are computed, and for each of them only a random subset of

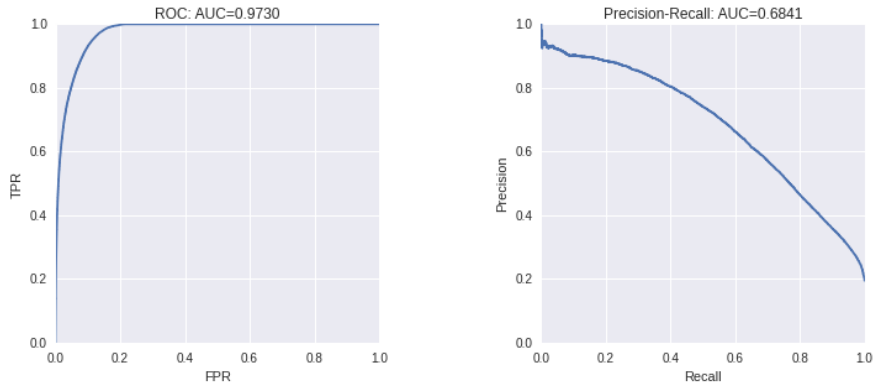
Figure 2.4: In the previous work the images are thresholded at different levels to extract shape features. From [1].



features is used with a random threshold. The trees with the best performance are retained while the rest are discarded.

This method is evaluated on how well the patches are classified, obtaining an area under the Precision-Recall curve of 0.69 (figure 2.5). To have a precision of 90% they recall only 20% of the infected patches. The authors claim that the number of fields that need examining is still lower than in previous studies with thin films. They do not attempt to locate the position of the parasites nor to count how many of them there are on each image. To do this task, it would be necessary to merge and discriminate between close positive patches.

Figure 2.5: Results for the classification of patches in [1]. The results have been replicated for a new split of the dataset that will be used through this project and the same results are obtained. **Left:** Receiver operating characteristic curve. **Right:** Precision-recall curve. AUC indicates the area under the curve.



2.2 Deep Learning for Image Recognition

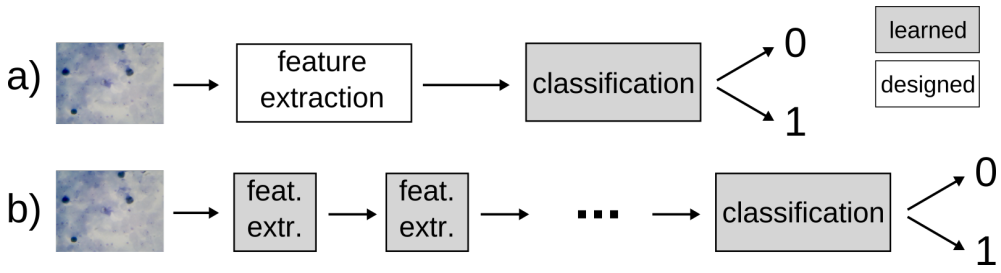
One of the drawbacks of the previous methods is the use of hand designed features. Many of them are computed and the best ones are picked with some feature selection

method. If that could be done automatically, the system would be easily extended to other parasites and would not rely on how good the features we have created are.

Deep learning techniques allow us not only to solve a typical machine learning problem (classification in this case) but also to create an appropriate representation of the input data, as illustrated in figure 2.6. Deep learning includes neural networks with multiple hidden layers where each one provides a new representation of the data. In this case we use Convolutional Neural Networks (CNNs). In this section we review the most important aspects of CNNs as well as their training procedures.

The success of such models for image recognition have been proven several times. Some of the most remarkable results have been achieved on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [14], where they have been used to recognize up to 1,000 different object classes. The model proposed by *Krizhevsky et al.* [6] on the ILSVRC has been frequently taken as a reference. It included 5 convolutional layers followed by 3 fully connected layers, although deeper models have also been used [7].

Figure 2.6: In shallow methods only the classification is learnt (**a**). In deep methods both the feature representation and the classification are learnt (**b**).

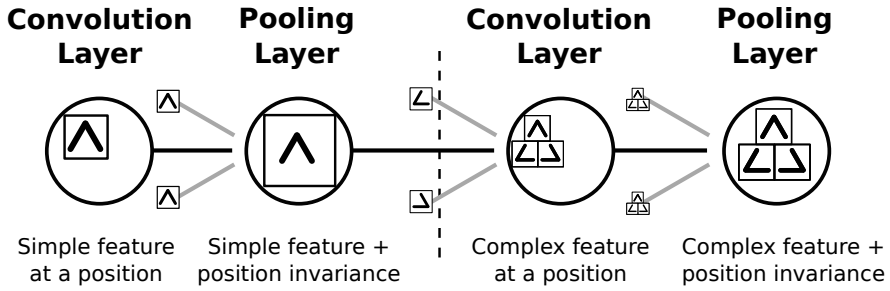


2.2.1 Convolutional neural networks

The origin of Convolutional Neural Networks (CNNs) is the Fukushima Neocognitron [15]. This model is inspired by the visual system of mammals and is characterized by having two different types of units. The first ones are activated (after training) when a certain pattern is present on a certain location. The second type is activated by the same pattern but in any of the possible places of a broader area (Figure 2.7).

This model was then expanded by *LeCun et al.* [16] to do handwriting recognition and included all the basic aspects of CNNs:

Figure 2.7: Each unit of a convolutional layer learns more complex features than the previous one by combining previous features. Each pooling layer achieves position invariance from inputs of the same feature at different but closeby locations.



- **Local receptive fields**: in fully connected networks each unit has connections from every point the previous layer. However, in CNNs each unit only covers a subset; in images they correspond to an area of the image.
- **Shared weights**: Not all the units will have different weights. The units of a layer are split into sets. Each one of the sets will share the same weights while each unit will be applied to a different area of the previous layer.
- **Sub-sampling**: The output of the same unit at close positions is summarized so that the exact location of the feature is not relevant, achieving position invariance.

Taking this into account, the number of parameters is much lower compared to fully connected layers, due to sparse connectivity (local receptive fields) and weight sharing. Both factors reduce the complexity of the model and allows us to train the network with much less data.

Furthermore, CNNs have some benefits that make them particularly good for images. They take advantage of the spatial structure of the image as a result of the local receptive fields. We can expect adjacent pixels to be correlated and thus learning local features makes sense, so we are introducing prior knowledge of the structure of the input data: this cannot be done with fully connected networks. Besides that, CNNs achieve invariance to non-relevant changes on the input image as a result of sub-sampling. As a consequence, small changes on the position of the features do not affect the result.

In convolutional networks we find four different components that can be seen as different layers: convolution, non-linearity, normalization and sub-sampling. A set of stages composed by those four layers are usually followed by fully connected layers that act as a typical classification model. These components are described below.

Convolution

The shared weights and local receptive fields can be implemented as a convolution of the output of the previous layer and a filter. The filter is a matrix with the weights that are learnt. The result of the convolution corresponds to a matrix resulting of the inner product between the filter and a sub-matrix of the input data centered at each point . A bias term is usually present and is added to the result of the convolution.

Therefore, for the filter f of the layer l, we have a set of weights W_{lf} of size $k \times k$ and a bias term b_{lf}. For a two-dimensional input matrix I we have a two-dimensional output O_{lf} where:

$$O_{lf,ij} = b_{lf} + \sum_{m=-\lfloor \frac{k}{2} \rfloor}^{\lfloor \frac{k-1}{2} \rfloor} \sum_{n=-\lfloor \frac{k}{2} \rfloor}^{\lfloor \frac{k-1}{2} \rfloor} W_{lf,mn} I_{i+m,j+n}$$

Sometimes, instead of applying the filter to each unit of the previous layer it is done with a stride [6]. Apart from this value, it is necessary to set the number of filters and the size (k, which has to be even). Using several convolutional layers leads to hierarchical features (the features of one layer are composed by features of the previous one). Previous work shows that using at least two stages generally improves performance of the network [17].

Non-linearity

After each convolution there is a non-linearity that makes the model capable of solving a classification problem without a linear solution. The selection of the non-linearity has been reported as the decision with the highest impact on the performance[17].

Although $tanh()$ has been frequently used, Rectified Linear Units (ReLu) have been preferred for large datasets as them are easier to compute and this allows the network to train faster [6]. The ReLu is computed as $f(x) = \max(0,x)$.

Sub-sampling

Initially, the sub-sampling method consisted in averaging the outputs of the activities around a certain location. However, it was shown that taking the maximum value

(max-pooling) over those activities leads to better results for different datasets [18].

It is necessary to determine the width of the pooling kernel (i.e. how many units are summarized). A large pooling width would mean higher invariance but we could be losing useful information. It is also necessary to establish the overlapping between pooling units: this can improve the performance but, at the same time, increments the number of units making it harder to train the model [18].

The pooling layers can follow all or some of the convolutional layers. For instance, in the model of *Krizhevsky et al.* there are 5 convolutional layers and only 3 pooling layers [6].

Normalization

It is common to have some normalization layers [6, 17]. In this case we use Local Response Normalization as in [6]. It creates competition between the activities of the filters at the same position during learning and forces the filters to learn to respond to different features. It is computed after some of the ReLu non-linearities. If a_i and b_i are the input and the output of the ReLu:

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

The parameters k, n, α and β should by found by validation [6]. In particular, n indicates the number of filters that are involved in the normalization. They are ordered at random at the beginning and the n adjacent filters are used in each unit.

Fully connected layers

After all the convolutional and pooling layers, several fully connected layers can be used as a traditional classifier. The connections are computed as the inner product between all the outputs of the previous layer and a set of weights, followed by the addition of a bias term. After all the fully connected layers except the last one there is a non-linearity in the same way as with convolutional layers.

2.2.2 Training deep neural networks

All the trainable parameters of the model are in the convolutional and fully connected layers. To find the best values the most common method is to use gradient descent as the optimization technique and backpropagation to find the gradients of the weights. We also consider data augmentation as a means of having more data to avoid overfitting.

Backpropagation and gradient descent

The objective function to optimize is the multinomial logistic loss of the softmax of the final layer's outputs [6], which corresponds to maximizing the log-likelihood of the data. In this case, the softmax function replaces the non-linearity of the last fully connected layer, which will have as many units as existing classes. For the z_k outputs of the last layer, the softmax function for z_i is defined as:

$$\sigma_S(z_i) = \frac{e^{z_i}}{\sum_k e^{z_k}}$$

The values $\sigma_S(z_i)$ correspond to the probabilities of belonging to a certain class i. The selected class will be $y = argmax_i(\sigma_S(z_i))$. Except when we do hard negative mining (see Section 4.2.4), all the models are for binary classification. In this cases, when only two classes are present, the softmax with two outputs is equivalent to having just one output value $\sigma(z) = 1/(1+\exp(-z))$.

Then, the log-likelihood $\mathcal{L}$ for a set of parameters θ and the dataset of values X_i and labels y_i is:

$$\mathcal{L} = \sum_i \log(P(y = y_i|X_i, \theta)) = \sum_i \log \sigma_S(z_{(y_i)})$$

Finally, the loss $\ell = -\mathcal{L}$ can be used in stochastic gradient, where each parameter is updated following the rule:

$$w' \leftarrow w - \eta \frac{\partial \ell}{\partial w}$$

Momentum and weight decay can also be included in the update rule [6].

Data augmentation

When using larger networks more data is required and with data augmentation we can create new images. In this case we consider rotating and mirroring the images, although other methods exist such as elastic and colour transformations[6, 19]. For each image we will take the original image and other three resulting from 90 rotations. For each one of them we also take the horizontal mirrored image. It is not necessary to mirror them vertically as that would produce repeated images.

2.3 Object Detection

The previous work of Quinn et al. [1] used a sliding window approach to find the parasites. A patch of 50×50 pixels is taken every 40 pixels, although the authors do not attempt to find the location of the parasites. This would be necessary if we want to count the parasites in an image, and requires to merging close detections if they correspond to the same parasite. Here we describe a simple method to achieve this, and the procedures to evaluate the results.

2.3.1 Non-maxima suppression

Non-maxima suppression is an algorithm to merge a set of detections that correspond to the same object. In the greedy method of *Felzenszwalb et al.* [8] each detection is associated with a score (i.e. probability) and iteratively the top one X is picked. Then, all the detections Y having an overlap with X above a threshold will be merged and removed from the original set.

A similar version of non-maxima suppression is used by *Ciresan et al.* [20]. They compute the probability of being the desired object for each pixel of the input image. This allows them to create a probability map which is used to find the location of the objects (i.e. the probability peaks) similarly to the previous method.

2.3.2 Hard negative mining

Most of the negative patches extracted from a blood smear do not contain any objects that look like parasites as they only cover background areas. Moreover, the number of negative samples is much higher (above 95%) than the number of positive patches. Hard negative mining is a way to find patches that represent the challenging cases that the model will have to face. First, a model is trained with random patches and then it is trained again using the patches wrongly classified with a high probability.

The process can be repeated iteratively and run until convergence [8] or just once [20]. In the last case the probability of picking one of the hard-negative mined examples to feed a CNN was proportional to the probability assigned by the previous model. We will use a simpler version of hard negative mining where we train the model with all the negative patches with a probability above a threshold and a few samples taken at random from the full dataset. We will as well compare if assigning a new label to the difficult negative cases improves the performance.

2.3.3 Evaluation of object detection

To evaluate the performance of the system we will use the same procedures as in the PASCAL VOC Challenge [21]. The Average Precision (AP) is used to summarize the performance and corresponds to the area under the precision-recall curve of the detections with an associated score value. The precision of the results is defined as the ratio between the true positives and the total number of detections. The recall is the ratio between the true positives and the ground truth number of objects.

A detection is considered correct when the overlap between the parasite and the detected bounding box is above a fixed threshold. The overlap ratio between X and Y is defined as the area of the intersection divided by the area of the union of the two bounding boxes: $\frac{Area(X \cup Y)}{Area(X \cap Y)}$. The threshold for the PASCAL VOC Challenge was 0.5. If several detections correspond to the same object all of them but one are considered false positives.

Furthermore, we will use bootstrapping to find out if the difference between models is significant [21]. To do this, a new dataset is created by taking n samples from the original dataset with replacement, where n is the size of such dataset. Then, the difference between the APs of two models is computed. These two steps are repeated

many times. It is assumed that the two methods are equivalent with a confidence of $(1-\alpha)$ if zero is contained in the interval between the percentiles α and $(1-\alpha)$.

Chapter 3

Parasite Classification

In this chapter we detail how to classify individual patches as having parasites or not. Then we evaluate the results and make comparisons to previous work. There are two reasons to individually evaluate the classification stage. The first one is that previous work was focused on classifying individual patches and this allows us to compare our results to theirs. The second one is that we use the results of the evaluation to identify the best CNN models and we will use those models in the detection stage.

In Section 3.1 we analyse different alternatives to extract the patches and describe the details of the datasets we use. In Section 3.2 we focus on the training details. Then, in Section 3.3 we discuss the CNNs used and analyse the details of each component of the model. We include as well some non-convolutional neural networks for comparison purposes. In Section 3.4 we evaluate the final results, motivating the need of specific detection techniques to find the location of parasites (which will be covered in Chapter 4). Finally, implementation details of the languages and third party libraries used are provided in Section 3.5.

3.1 Dataset

For this project we use the dataset created by *Quinn et al* [1]. It is composed by 2,703 images with a resolution of 1,024$\times$768. We use grayscale images to train the models as color is not expected to be very informative with the particular stain (Fields A and B) used on the blood smears. Originally, 80% of the images were used to train the models and 20% as a test set. In this case we modify the split to create a validation set

that we will use to tune the hyperparameters of the models. Hence, we split the set of images as 60/20/20 (train/validation/test). As it has not been possible to keep the same splits we have repeated the experiments of *Quinn et al.* using the new sets. We use the same the number and depth of the trees as in the code shared on GitHub by the authors (see section 3.4) to get comparable results. The annotation of the parasites includes a bounding box around each parasite. Most of the bounding boxes have a size of 50×50 pixels but there are some exceptions, the majority of them corresponding to parasites which bounding box is cut by an edge of the image.

In the following subsections we explain the relevant aspects of the dataset. First, we analyse how to extract patches from images. Then, we introduce a new test set that reflects better the performance of the system. Finally, we explain how we use data augmentation and how modifying the percentage of positive patches can be used to improve the results.

3.1.1 Patch Selection

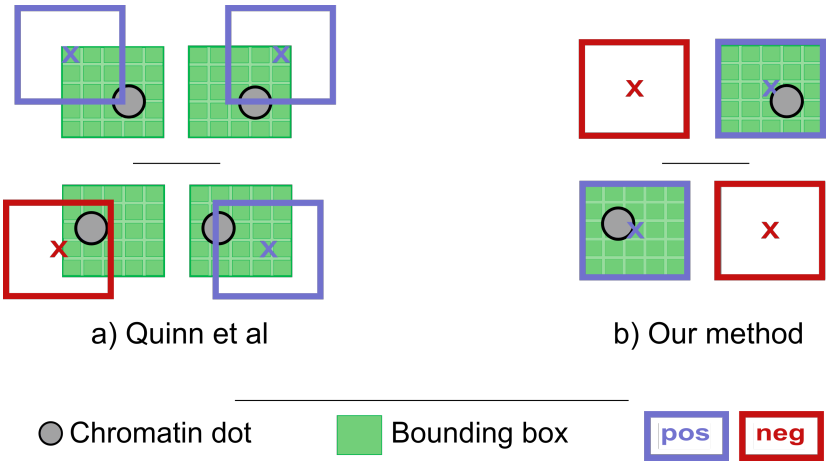
Originally, a sliding window approach was used to generate positive and negative patches. The size of the patches was 50×50. For each image, patches are taken every 40 pixels both in the horizontal and the vertical axes. The label associated with each patch is 1 if the center of the bounding box of a parasite is contained in the patch or 0 if not.

The chromatin is the most characteristic part of the parasite and its size is approximately 14 pixels. The chromatin may be excluded in some of the patches as it is not centered on the bounding box that identifies the parasites. Therefore, we consider alternatives for taking the parasites both for training and evaluating the results.

Our proposed method consists in taking one positive patch centered on each parasite. The negative patches are generated using a sliding window as before, but excluding any patch that overlaps with the bounding box of a parasite. This way we avoid patches with different labels that cover similar areas of a parasite. In Figure 3.1 we illustrate this problem, the blue and red squares show positive and negative patches that cover similar areas of the parasites.

The new way of extracting patches improves the performance of both Convolutional Neural Networks and the previously used Extremely Randomized Trees. The differ-

Figure 3.1: **a)** Method used by *Quinn et al* [1] to select the positive and negative patches. The positive patches are taken every 40 pixels, they are positive if they contain the centre of a parasite's bounding box. **b)** Our alternative, where the positive patches are centered at the parasite and the negative patches do not overlap with the parasite bounding boxes.



ences are evaluated in depth in Section 3.4.1. Similarly, we consider the new patches to be a fairer measure to evaluate the performance of the model, and thus the validation and test sets will be composed of these patches as well. However, all the positive patches are centered at parasites while the negative patches are not centered at other objects. Given that the positions of parasites is unknown for new images, in the next section we introduce a new test set that tries to solve this problem.

3.1.2 Test sets

We will evaluate the model on two different sets that differ on the negative samples. We use both of them as test sets for the classification stage, but they are created from the images of the the validation set, as the results will be used to select the best architectures that will be considered for the detection stage. The first set (*full* test set) will contain all the patches created as described in the previous section. The second one (*difficult* test set) includes all the positive samples, which account for approximately 60% of the new set, but the negative samples are created using a technique similar to hard negative mining.

The negative patches of the *difficult* set correspond to the detections with high probability that do not correspond to a parasite. They are obtained with the full system as

described in Chapter 4. The classifier is a CNN which architecture[1] does not match any of the ones evaluated on this dissertation. Convolutional Neural Networks might score worse results on the *difficult* set that other models if all CNN tend to make similar mistakes. However, we have observed that models with the best results on this dataset perform better on the overall detection process.

3.1.3 Positive/negative ratio and data augmentation

The dataset created is highly unbalanced, approximately with only 4.5% of the patches being positive. We evaluate how using a rebalanced dataset affects the results obtained. The new training set contains all the positive samples and negative patches are taken at random so they account for 66% of the data.

Additionally, we use data augmentation to create new samples. The shape features extracted by *Quinn et al.* are fully invariant to rotations and mirroring; CNNs, however, are not. For this reason we create new samples with all four 90° rotations and horizontal mirroring. As a result, we consider for evaluation four datasets:

- **Original**: 619,276 negative patches and 28,579 positive patches.
- **Original and augmented**: 2,477,104 negative patches and 114,316 positive patches.
- **Rebalanced**: 85737 negative patches and 28,579 positive patches.
- **Rebalanced and augmented**: 342,948 negative patches and 114,316 positive patches.

3.2 Training Details

The networks are trained with Stochastic Gradient Descent as described in section 2.2.2. Each batch contains 256 samples.

We include momentum and weight decay as was done in the model of the 2012 ImageNet [6] . The value of momentum is 0.9 (i.e. 0.9 of the value of the previous update

[1]The architecture of the CNN is not relevant and was not carefully optimized, but we leave it here for reference. It was composed by the following layers. 1st convolutional layer with 24 kernels of size 4 and stride 1. 1st max-pooling layer with kernel size 4 and stride 2. 2nd convolutional layer with 32 kernels of size 5 and stride 2. 2nd pooling layer of size 5 and stride 2. The convolutional layers are followed by a ReLu non-linearity and a LRN layer. Only one fully connected layer is present.

is added on each step) and the weights decay by 0.0005 of its value weighted by the learning rate on each step. Thus, the formula of the update v_i for the weight w_i is:

$$v_{i+1} := 0.9v_i - 0.0005w_i\eta - \eta\langle\frac{\partial\ell}{\partial w_i}\rangle_{D_i},$$

$$w_{i+1} = w_i + v_{i+1},$$

where η is the learning rate and $\langle\frac{\partial\ell}{\partial w_i}\rangle_{D_i}$ is the average of the gradients over a batch D_i. The learning rate is initially set to 0.01 and is reduced so that at step i we have $\eta = 0.01/(1+0.001\cdot i)^{0.75}$. All the models are trained for 500,000 iterations, which has been enough for all of them to converge.

The weights are initialized with random values from a zero-mean Gaussian with standard deviation of 0.01 (same values as [6]). The bias values are initialized as 0.

The mean of all patches is computed and subtracted from every image in both training and testing stages.

3.3 Models

This section covers the different versions of Convolutional Neural Networks we have used and other models that we include for comparison purposes. We describe models with 1, 2 or 3 convolutional layers in Sections 3.2.1/3, a CNN with hand-designed filters in 3.2.4 and fully connected networks in section 3.2.5.

3.3.1 One convolutional layer

The simplest network we have used contains only one convolutional layer. We will evaluate networks with different parameters to know how they affect the results. The network will consist of one convolutional layer followed by a ReLu non-linearity, a max-pooling layer and a Local Response Normalization Layer. The details of each layer are as detailed in Chapter 2. The local response normalization (LRN) layer after every convolutional layer has the same parameters as in the model of *Krizhevsky et al.* [6] (although there it only follows some of the layers). They are $k = 2$, $n = 5$, $\alpha = 10^{-4}$

and $\beta = 0.75$. The convolutional layers will be followed by one or two fully connected layers.

The parameters that affect the results the most are the width and the stride of the convolution kernel and thus we will compare different networks to evaluate them, the rest of parameters are as follows. Each convolutional layer will have 4 filters, as augmenting the number of filters (to 8) does not improve the performance of the networ (i.e. the area under the precision recall curve is not significant). The kernels of the pooling layer will have a size of 5 with an stride of 3 pixels, as incrementing it to a size of 7 (with an stride of 4) or decreasing it to 3 (with an stride of 2) has little or no effect on the results.

For the convolutional kernels we will compare networks with kernels of 7, 14 and 21 pixels and strides of 1, 2 and 4 pixels. The networks are given names that represent these values. For instance, a network with kernels of 14 pixels, stride of 2 and 1 fully connected layer will be: (14, 2)1C-2FC. Following this, we indicate if it has been trained with the augmented original train set (O) or the augmented and rebalanced set (R).

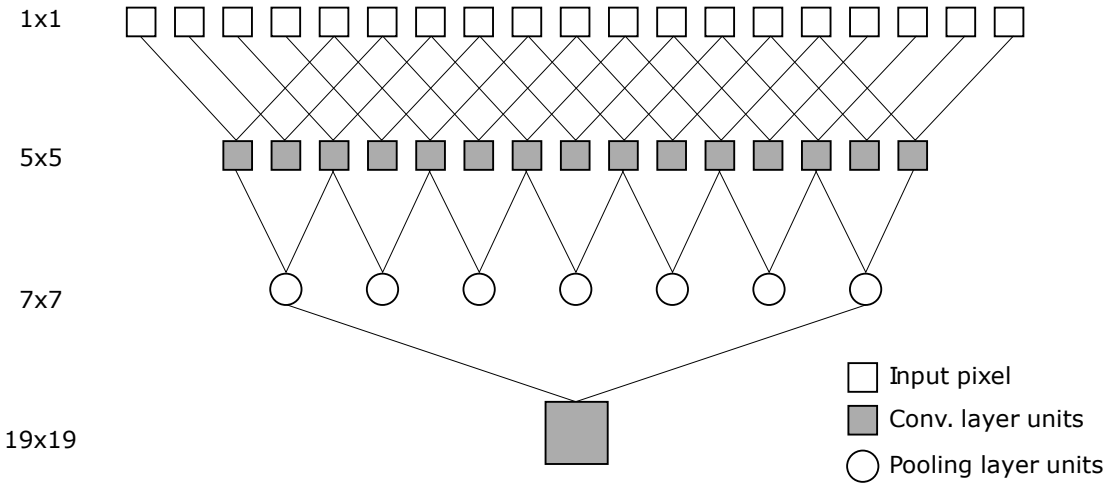
3.3.2 Two convolutional layers

After experimenting with the parameters of a deeper network, we find an structure with two convolutional layers that outperforms the 1 convolutional layer network. The first layer has 12, 5×5 filters applied to every pixel of the input images. The pooling layer has kernels of size 3 and they are applied with an stride of 2 pixels. The second layer has 24 7×7 filters with a stride of 1. The corresponding pooling layer has kernels of size 5 and they are applied every 3 pixels. .

As a result, each unit of the second layer has inputs from an area of $19\times$ 19 pixels (see Figure 3.2), which is similar to the area covered by the first layers of the best models with one convolutional layer (14-21 pixels).

In this case we will also compare models both with one and two fully connected layers. These networks will be named in a similar way as before (e.g. 2C-2FC (R)).

Figure 3.2: Units up to the second convolutional layer (top to bottom). Numbers on the left are the dimensions of the area of the input image covered by each unit.



3.3.3 Three convolutional layers

Finally, we have trained a much larger network with three convolutional layers. The first one with 16 5×5 filters, the second one with 32 5×5 filters and the third one with 64 3×3 filters. Only the two first layers are followed by pooling layers, both of them with kernels of size 3×3 and a stride of 2.

The convolutional layers are follwed by two fully connected layers. The hidden layer contains 512 units. The name of this network will be 3C-2FC.

3.3.4 Hand-designed convolutional network

To check if the network is really learning useful filters we train a network with filters shaped as blob detectors. This network has a convolutional layer with 4 24x24 kernels applied to each pixel of the input image. After that, there is one LRN layer, one pooling layer (with a width of 3 and a stride of 2) and 1 fully connected layer.

Each filter is initially set to a Difference of Gaussian filter, then the model is finetuned for 150,000 iterations so that the weights of the fully-connected layer connections are adjusted. To keep the original filters we reduce the learning step by a ratio of 0.1. The initial weights correspond to the difference of two Gaussian functions centered at the middle of the kernel. The positive Gaussians have standard deviations of 1, 3, 5, 10

and the negatives 2, 6, 10, 20. The result of the difference is negated and scaled so that they detect dark blobs and the minimum value of the filter is 1.

3.3.5 Fully connected networks

Finally, to check if it is necessary to use a CNN instead of a traditional fully-connected neural network we train neural networks with one and two hidden layers. The first one will have 36 hidden units and the second one 512 in each of the hidden layers. In both cases they have ReLu non-linearities as in the case of CNNs.

3.4 Evaluation

In this section we evaluate the decisions made in this chapter and compare the results obtained with the different models and previous work. In Section 3.4.1 we analyse the decisions related to the dataset: how the patches are selected and how rebalancing and augmenting the dataset affect the results. Then, in Section 3.4.2 we evaluate the components of the CNNs and discuss what useful information the networks may be learning. We compare the results to non-convolutional models in Section 3.4.3 and to the previous work in 3.4.4. Finally, we include a brief conclusion in Section 3.4.5.

We evaluate the performance in the two test sets introduced in Section 3.1.2: the *full* test set and the set the *difficult* set. The main measure will be the area under the precision recall curve (AUC) as used in the previous work [1]. We use bootstrapping [21] with 1,000 iterations and a confidence of 95% to check if the difference between results is significant.

Additionally, we compute the F1 score, which corresponds to the maximum harmonic mean between the precision and recall values that the model achieves:

$$\frac{2 \cdot \text{ precision } \cdot \text{ recall}}{precision + recall}$$

The F1 score is included as it is easier to understand, it shows the best result we can get with a balance between the precision and recall values.

Table 3.1: Area under the precision-recall curve for the ERT and CNN models. For each model it shows the value corresponding to random/centered patches for both training and testing. All the results are significantly different.

		Train			Train			
		Random	**Centered**		**Random**	**Centered**		
Test	**Random**	0.652	0.663		0.552	0.350	**Random**	Test
	Centered	0.824	0.848		0.796	0.980	**Centered**	
		Extremely Randomized Trees			Convolutional Neural Network			

3.4.1 Selection of the dataset

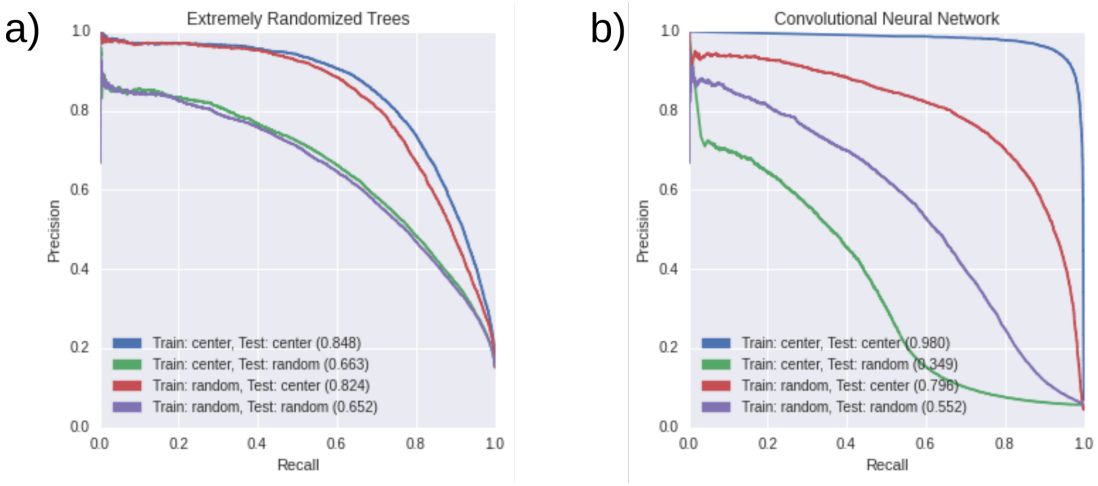
The first thing we evaluate is how to extract the patches and if we should balance and augment the dataset. As it is not feasible to train every network with all the datasets we use a simple CNN for this purposes. It correspond to the CNN with one convolutional (4 filters of size 14 and stride 2 and max pooling of size 5 and stride 3) followed by one one fully connected layer. In the case of how to extract the patches we include the Extremely Randomized Trees (ERT) as well.

3.4.1.1 Evaluation of patch selection

The results of the CNN and ERT models with both ways of taking patches is summarized in Table 3.1 and in Figure 3.3. For both models we see that all the results are higher when evaluated on the set with patches centered at parasites, which makes sense because this task is easier. In the case of the ERT model, the performance is better when evaluated on the original dataset even if it has been trained with centered patches. This shows that, for this model, training the model with the new patches do not harm the performance in any case. Although the difference is small, using bootstrapping as described shows that the difference is small but significant.

For the CNN model the performance on the new patches is much higher. Now we have an AUC of 0.980 (compared to 0.848 of the ERT) if the model is trained with centered patches and 0.796 if it is trained with randomly taken patches. It is interesting to note that if trained with centered patches and tested with random patches the performance is lower than in any other case. It is also lower than the ERT when trained and tested

Figure 3.3: **a)** Precision-recall curve for the ERT model when the training patches are taken as random or centered, and it is evaluated with patches taken as random or centered. **b)** The same for a CNN with one convolutional layer.



in centered patches.

The results above make sense if both the CNN is modeling the data better than the ERT and if taking the positive patches with a sliding window produces imprecise labels. Figures 3.4 and 3.5 support this idea. They correspond to the results of the CNN trained with centered patches and tested on random patches. Figure 3.4 shows the probabilities assigned to positive and negative patches. Most mistakes are due to positive patches classified as negative. Two samples of these patches are shown in the figure. These mistakes correspond to patches labeled as positive that only contain a minimal part of a parasite.

It also happens frequently that there is an incorrect detection next to a correct one as seen in Figure 3.5. Both detections partially cover the parasite, but one of them has a positive label and the other not. It could be argued if both of them should be positive or not. This also shows that a way of merging close detections is necessary, which will be addressed in chapter 4.

Given these results, we accept the initial hypothesis that the centered patches better represent the difference between parasites and negative cases of parasites. Because of this, we will use the new way to generate patches to train the models. Similarly, we consider them to be a fairer measure to evaluate the performance of the model, and thus the validation and test sets will be composed of these patches.

Figure 3.4: Histogram of the probabilities assigned to positive (green) and negative (blue) patches. The patches of the test set are taken at random and the model (CNN) has been trained with centered patches. Most of the false negatives correspond to patches that contain a minimal part of a parasite as those shown on the image (the green box is the patch and the gray box the true parasite).

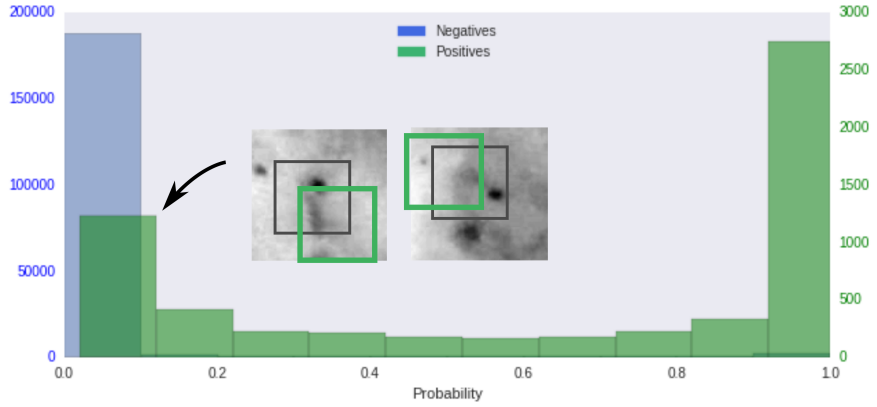
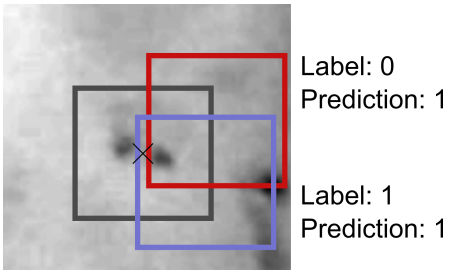


Figure 3.5: Two patches (red, blue) that cover different areas of a parasite. Both are classified as positive but only one the blue is labeled as positive.



3.4.1.2 Evaluation of data augmentation and dataset rebalancing

When we test the performance on the full test set we find minor differences between the four datasets (original/rebalanced, non-augmented/augmented). The area under the curve for the four of them only varies between 0.9769 and 0.9798. Because of this, we only show their results on the *difficult* set. The results for the same CNN as in the previous section are included in Table 3.2.

Using a balanced dataset reduces the performance of the model from 0.952 to 0.936. Nevertheless, if the dataset is balanced and then augmented with rotations and mirroring, the performance is improved to 0.958. These differences are significant when tested with bootstrapping with a confidence of 95%. Augmentation of the unbalanced dataset, however, does not change the results significantly. Although the area under

Table 3.2: Area under the precision-recall curve for a CNN trained with datasets with differing proportions of positive patches, and including or not augmentation. * indicates that two results are not significantly different using bootstrapping with a confidence of 95%.

	Original	**Rebalanced**
Non-augmented	0.952*	0.936
Augmented	0.948*	**0.958**

the precision-recall curve of the unbalanced set is lower than the one of the augmented balanced set the difference is small, so it is worth it to look closer at the results of both cases.

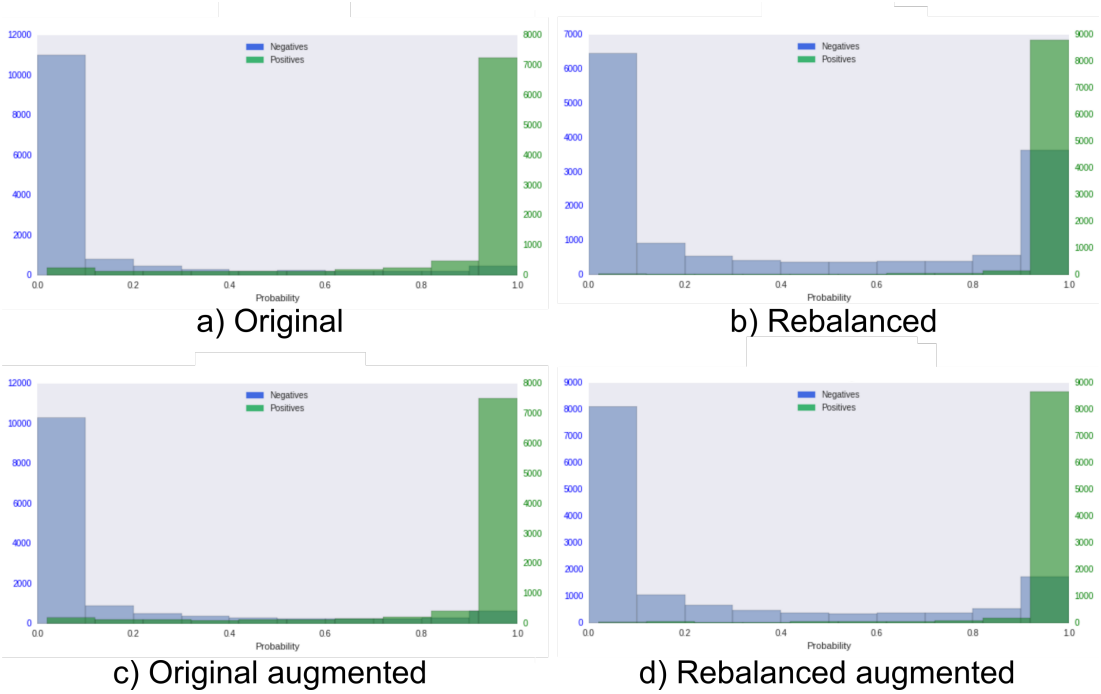
In Figure 3.6 we show the histograms of the probabilities assigned to positive and negative patches. Looking at the top row, it is clear that using the dataset containing all the negative patches instead of the balanced one reduces the number of negative patches classified as positive (the blue bar on the high probabilities). Nonetheless, this means that some of the positive patches are assigned low probabilities causing false positives.

When any of the datasets are augmented (bottom row) the number of false positives is reduced further, in this case without any negative effect. For the unbalanced set, both histograms look very similar (left column), which supports the statement made before of them being non significantly different.

Similar results appear when we use larger networks. The network trained with the unbalanced dataset usually has more false negatives while the one trained with the balanced dataset has more false positives. This means that the network that sees a larger proportion of parasites tends to assign this class to the patches more that the other one.

The augmented version of both datasets will be considered in further experiments as they produce similar scores but their effects are different.

Figure 3.6: Histograms of the probabilities assigned by a CNN trained with the balanced/unbalanced and original/augmented datasets. Green represents positive patches and blue represents negative patches.



3.4.2 Evaluation of CNN models

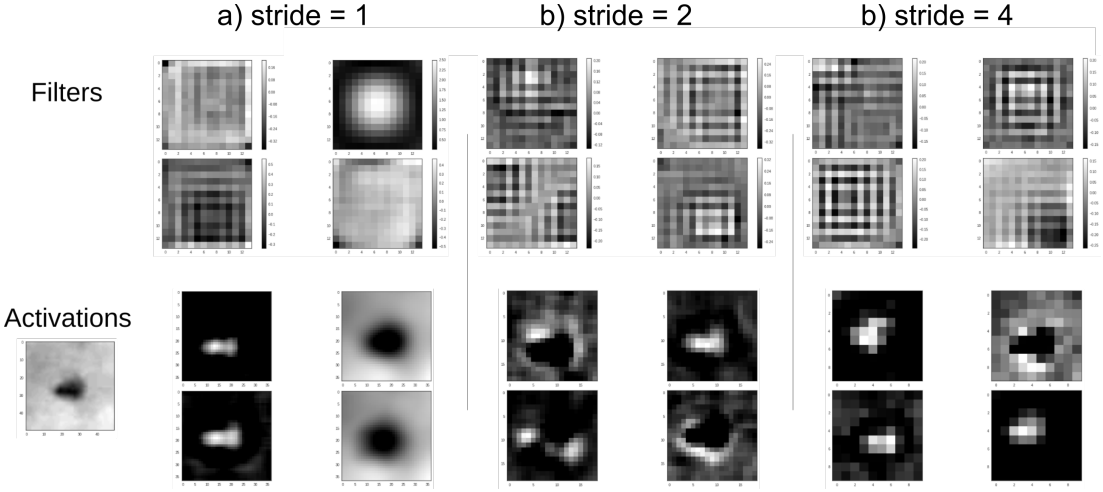
In this section we analyse several aspects of the CNNs described on Section 3.3. First, we evaluate the effect of the stride and size of the kernels in CNNs with one convolutional layer. Then, we study how adding more fully-connected layers improves the results. Finally, we describe the effect of removing the Local Response Normalization layers.

3.4.2.1 Stride in convolutional layers

For the CNNs with one convolutional layer (as an example we use one with 4 14×14 filters) applying the filters to every two pixels instead of to all of them is important. Applying the kernels to each pixel results in filters shaped as simple blob detectors (see a) in Figure3.7) and the performance is much lower.

However, if we apply them to every two pixels it results in the characteristic shape shown in Figure 3.7 b), which looks like blobs with stripes or a plaid pattern. The

Figure 3.7: Filters and activations for three CNN models with different strides for the application of the convolutional layer filters. The activation images correspond to the positive patch displayed on the bottom left.



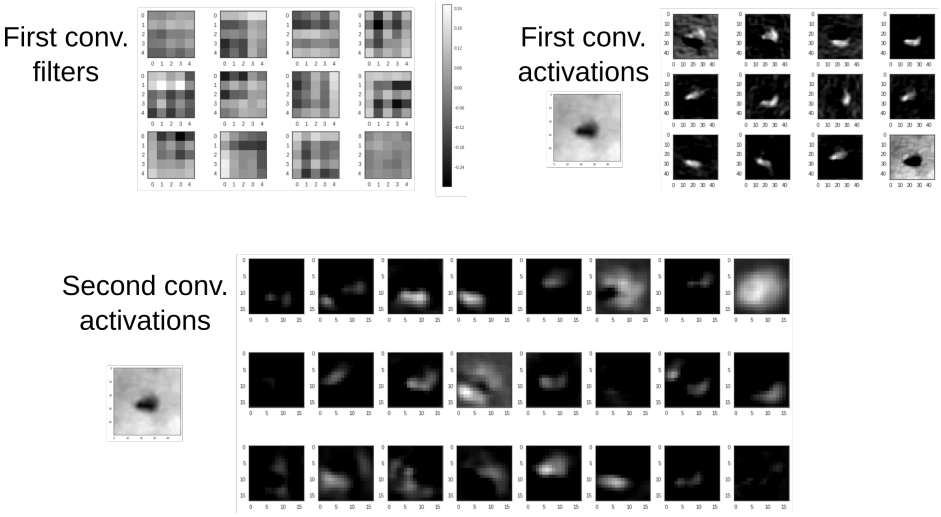
activations apparently correspond to the blob made by the chromatin dot and the surrounding area, which could be a way to find the cytoplasm of the parasite.

Firt we hypothesized that such patterns may the result of the competition caused by the LRN layer. However, removing this layer still produces the plaid pattern. Increasing the stride to 4 pixels leads to similar filters without a relevant difference in performance (see c) in the same figure as before).

Evaluating them on the *full* test set, the area under the precision-recall curve with a stride of 2 is 0.9761 compared to 0.8046 with a stride of 1. The results for strides of 1 and 2 are displayed with the other relevant models with 1 convolutional layer in Table 3.3.

For a CNN with two convolutional layers, the learned convolutional filters and their activations are displayed in figure 3.8. In this case an stride of 1 or 2 has little or no effect. Although it is not clear what the network is learning, the first layer filters look like edge detectors while having the characteristic striped pattern. The activations of these filters are different representations of the parasite, but it is not easy to see clearly how they are useful for the classification. The same happens with the activations of the second layer. It is not possible to display the learned filters of this layer as with the previous layer as they are three dimensional. The visualization could be extended with the method proposed by *Zeiler et al.* [22] to better understand what patterns produce the maximum activations.

Figure 3.8: Filters and activations for the first convolutional layer. Activations for the second convolutional layer. The filters of the second layer are three dimensional and thus are harder to visualize and interpret.

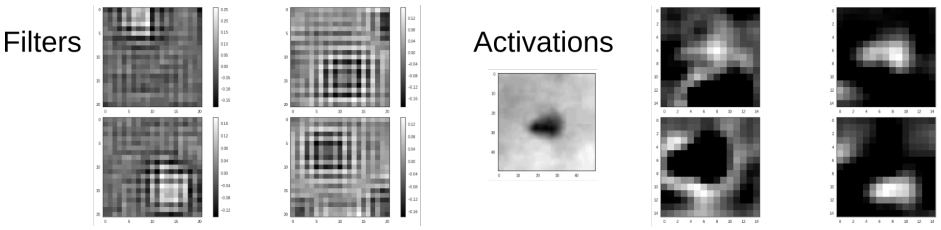


3.4.2.2 Size of convolutional kernels

Initially we picked the size of 14x14 pixels because it is approximately the size of the chromatin dots and we expect filters to learn to detect them. Reducing the size considerably harms the performance, however, we notice that incrementing it to a value of 21x21 pixels makes the results better in some cases (see Figure 3.4).

Nevertheless, the best results are still obtained with filters of size 14×14. The most interesting fact is that the big filters still contain shapes similar to blob detectors around 14 pixels, and we still get the stripped pattern as we display in 3.9. In particular, two of the filters apparently extract light and dark blobs while the other two look like the result of a translation of these, probably caused by the LRN layer, which forces them to learn different things.

Figure 3.9: Filters and activations of the model with larger filters (21x21). The activations correspond to the positive patch displayed on the image.



3.4.2.3 Depth of the network

We compare the networks with different number of convolutional and fully connected layers.

Adding another fully connected layer (with 36 hidden units) to the networks with one convolutional layer can improve the results. We include all the relevant networks in Table 3.3. On the *full* test set the difference is not significant, however in the *difficult* it increases from 0.9568 (best result with 1 fully connected layer, trained on the rebalanced training set) to 0.9860 (in this case the best result is obtained with the original set).

The best network with 1 convolutional layer is (14,2)1C-2FC (O), which we will name just 1C-2FC (O). This achieves the best results in both datasets, although it the case of the full test set there are others which results are not significantly different.

In Table 3.4 we show the results for CNNs with two convolutional layers. The results are better for these networks that for the models with just one convolutional layer. It is notable that the model with 1 fully connected layer works better if it is trained with the full dataset. However, it is not the case for the model with two fully connected layers, which works better if trained with the balanced dataset. This may mean that having a balanced dataset is more important the more complex is the model, which will be confirmed for models with three convolutional layers.

We have also included the performance on the train set as this networks are more complex. It is slightly better than the performance on the test sets, but the difference is small, which means that the network are not suffering of overfitting.

The AUCs for the best models are 0.9859 (*difficult* set, by 2C-2FC (R)) and 0.9894 (*full* set, by 2C-1FC (O)). We will consider both networks in the following sections and in the detection stage.

Finally, also in Table 3.4, we include the results for the network with 3 convolutional layers. The AUC for this model on the training set is 0.9980, higher that any previous model. This is expected, as the model is more complex and can model the input data better. If it is tested on the full test set the AUC is 0.9776 with an F1 score of 94.13%, which is worse than some of the previous models. However, on the difficult test set it is higher than any other with an AUC of 0.9925 and an F1 score of 97.26%.

Table 3.3: Results of the CNN models with 1 convolutional layer for different strides, kernels sizes, number of fully connected layers and training dataset (R)ebalanced or O)riginal). The area under the PR curve (AUC) and the the F1 scores are shown for both the *full* and the *difficult* test set. *#params* indicates the number of trainable parameters of the model. In each column, if two values have the same superscript that means that they are not significantly different using bootstrapping with a confidence of 95 %.

	AUC (full)	F1 (full)	AUC (diff. set)	F1 (diff. set)	#params
(14,1)1C-1FC (R)	0.8046	82.88	0.8708	84.03	1,365
(14,2)1C-1FC (R)	**0.9782**[1]	**94.01**	0.9568	89.60	933
(14,2)1C-1FC (O)	0.9769	94.07	0.9485	87.93	933
(14,2)1C-2FC (R)	0.9689[3]	92.14	0.9789[1]	93.86	6,045
(14,2)1C-2FC (O)	**0.9793**[1]	**93.78**	**0.9860**	**95.35**	6,045
(21,2)1C-1FC (R)	0.9737	93.13	0.9744[2]	93.10	1,869
(21,2)1C-1FC (O)	0.9646	91.95	0.9748[2]	93.21	1,869
(21,2)1C-2FC (R)	0.9712[2]	92.57	0.9782[1]	94.44	5,441
(21,2)1C-2FC (O)	0.9699[2,3]	92.13	0.9697	92.61	5,441

It is notable that in this case, if the network is trained with the full unbalanced dataset, the performance drops to an AUC of 0.7416. This matches the previous results that show that for complex models it is necessary a balanced dataset: the networks 1C-2FC and 2C-1FC (three layers with trainable parameters) get the best results with the original dataset while the networks 2C-2FC and 3C-2FC (four and five layers with trainable parameters) provide the best results when trained with the rebalanced dataset.

We conclude that deeper and more complex networks can achieve better results, but we will consider smaller networks as well as they are much faster to compute and their performance is close to the best networks.

3.4.2.4 Importance of LRN layer

To find if the Local Response Normalization (LRN) layer is really necessary we have trained the model with two convolutional layers and two fully connected layers as described before, but without the LRN layers.

The performance of such model is not as good as before (the area under the curve is

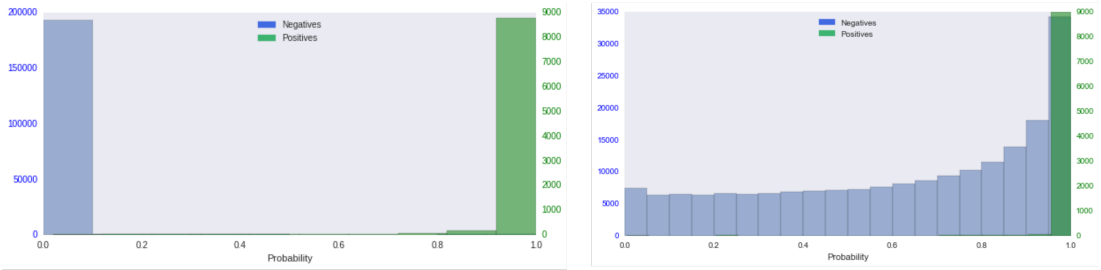
Table 3.4: The results of the relevant CNN models with two and three convolutional layers. We show the results for networks with 1 and 2 fully connected layers, both trained with the original (O) and rebalanced (R) datasets. The area under the precision-recall curve (AUC) and the the F1 scores (the harmonic mean at the operating point where it is maximum) are shown for both the full and the *difficult* test set. *#params* indicates the number of trainable parameters of the model. We show the AUC for the training set as well. In each column, if two values have the same superscript that means that they are not significantly different using bootstrapping with a confidence of 95 %.

	AUC (train)	**AUC (full)**	**F1 (full)**	**AUC (diff)**	**F1 (diff)**	**#params**
2C-1FC (R)	0.9750	0.9718	93.16	0.9816^{1}	94.46	15,049
2C-1FC (O)	0.9826	0.9823	94.50	**0.9894**	**96.35**	15,049
2C-2FC (R)	**0.9870**	**0.9859**	**95.30**	0.9836	95.74	33,713
2C-2FC (O)	0.9717	0.9766	94.00	0.9799^{1}	96.06	33,713
3C-2FC (R)	**0.9981**	**0.9776**	**94.13**	**0.9920**	**97.20**	1,638,401
3C-2FC (O)	0.9890	0.7416	70.68	0.9004	83.12	1,638,401

0.920 for the full dataset, compared to 0.9718 in Table 3.4) although it is still relatively high. However, if we look at the histogram of probabilities of the positive and negative patches (Figure 3.10), we can see how the probability of the negative patches is spread from 0 to 1. The reason why the AUC is not lower is that most of the positive patches have a probability higher than 0.9999 (87%) and only 0.15% of the negative patches are above that value. So, in this case we have a much higher threshold to separate the positive and negative patches instead of a probability value close to 50%.

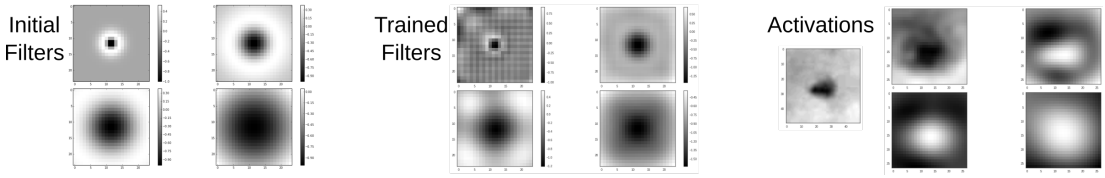
These results show the importance of the local response normalization layer, as the performance of the model is harmed and the boundary is not 0.5 as expected. However, if only one of the two LRN layers is removed, we do not see this results. We can conclude that at least one normalization layer is necessary but they do not necessary need to follow each convolutional layer, as it was done in the model of *Krizhevsky et al.* [6].

Figure 3.10: Histogram of the assigned probabilities to positive and negative patches for a CNN with 2 convolutional layers with and without LRN layers.



a) with LRN b) without LRN

Figure 3.11: Filters and actvations of the model initialized with a Difference of Gaussian blob detector.



3.4.3 Comparison with non-convolutional and hand designed networks

For the CNN with hand-designed filters, the resulting filters and activations after the model is fine tuned are displayed in Figure 3.11. We can see how they are acting as blob detectors.

The resulting performance is lower than for the any model with learned filters with an area under the precision-recall curve of 0.8198 on the *full* test set. The equivalent CNN with only one layer of filters and one fully connected layer achieved an AUC of 0.9782.

More carefully designed filters could provide better results, however, an advantage of deep learning methods is to substitute the hand-designed feature extractors. In any case, with this we show that the network is learning something more complex than a simple blob detector.

With fully connected networks we get better results, but they are still worse than CNN networks. For a model with one hidden layer with 32 units, the area under the precision recall curve is 0.938 when tested on the *full* test set.

For the model with two hidden layers with 512 parameters each one, the performance raises to 0.953, but it has not been possible to get higher performance with this architecture.

They are worse than the simplest CNN with 1 convolutional layer and 1 fully connected layer (i.e. without hidden fully connected layers) but have many more trainable parameters. The model with two layers has 80,062 parameters and the model with three layers have 1,543,681 parameters. The simple CNN model had only 933 parameters.

3.4.4 Comparison with previous work

In this section we compare the best models with the previous work on the same dataset by *Quinn et al.* [1] with hand-designed features and ERT. We cover the methods with 2 and 3 convolutional layers, as they have provided the best results. We also include methods with 1 convolutional layer as they are much simpler and have the advantage of being faster to compute. For comparison purposes we present as well the the results of the non-convolutional network.

The results for the full test set are included in Table 3.5. Clearly, CNNs outperform non-convolutional networks and the Extremely Randomized Trees of *Quinn et al.* [1]. CNNs with two layers provide the best results. They correspond to the best networks trained with the balanced and unbalanced datasets.

The best network (two convolutional layers and trained with the balanced dataset) allows as to recall 95.29% of the parasites with a precision of 95.31% at the point that maximizes the F1 score. This is a significant improvement compared to the ERT model, which achieves a recall of 75.27% with a precision of 79.74%. The network with three convolutional layers, which is the one with the best performance on the other dataset, is capable of recalling 96.12% of the parasites with a precision of 92.31%, which is not far from the previous CNN.

In Table 3.6 we include the evaluation on the *difficult* set. Again, we see that CNNs provide better results than any of the other models. In this case, the ERT model is closer to the neural networks, being equivalent to one the simplest CNNs. However, as this dataset is created using a CNN (even if that network is not any of the ones we compare here), if all the networks tend to make similar mistakes, this dataset could be harder for CNNs than for the ERT model.

Table 3.5: Results for the full test set. It shows the area under the precision recall curve (AUC) and the F1 score associated to it. The median AUC for bootstrapping with 1.000 replications and its associated ranking is included. If two models share the same rank they are not significantly different with a confidence of 95%. We include the best models with 1,2 and 3 convolutional layers, the fully-connected network with 3 layers (3FC) and the ERT model.

	AUC	F1	AUC (med.)	Rank
2C-2FC (R)	**0.9859**	**95.30**	**0.9859**	**1**
2C-1FC (O)	0.9823	94.50	0.9823	2
1C-2FC (O)	0.9823	94.49	0.9793	3-4
3C-2FC (R)	0.9776	94.13	0.9775	3-4
3FC (R)	0.9535	90.88	0.9534	5
ERT	0.8479	77.44	0.8482	6

Table 3.6: Results on the *difficult* test set. It shows the area under the precision recall curve (AUC) and the F1 score associated to it. The median AUC for bootstrapping with 1.000 replications and its associated ranking is included. If two models share the same rank they are not significantly different with a confidence of 95%. We include the best models with 1,2 and 3 convolutional layers, the fully-connected network with 3 layers (3FC) and the ERT model.

	AUC	F1	AUC (med.)	Rank
3C-2FC (R)	**0.9920**	**97.20**	**0.9921**	**1**
2C-1FC (O)	0.9894	96.35	0.9894	2
1C-2FC (O)	**0.9860**	95.35	0.9815	3
2C-2FC (R)	0.9836	95.75	0.9836	4
ERT	0.9589	89.18	0.9589	5
3FC (R)	0.9417	88.41	0.9416	6

As before, deeper networks work better than networks with one convolutional layer. In particular, the network with three convolutional layers is the one with the best performance. Furthermore, all the CNNs are better than the fully connected neural network.

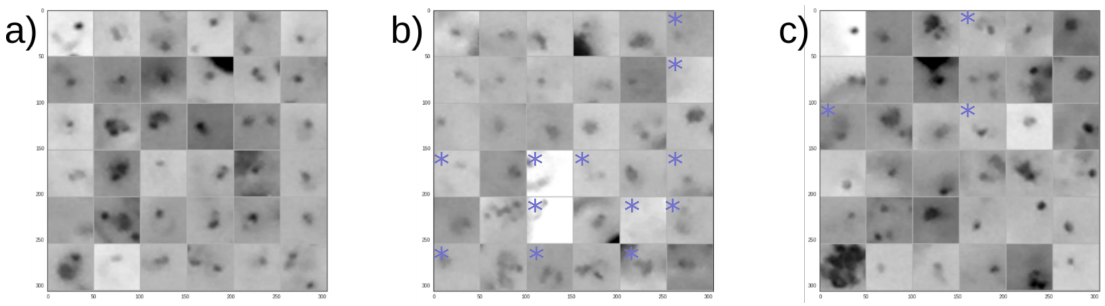
In this case, the CNN with two layers that achieved the best performance on the *full* dataset has a recall of 94% with a precision of 97%. The ERT model has a recall of 91% with a precision of 86%. The CNN with three layers recalls 98% of the parasites with a precision of 97%.

3.4.5 Conclusions of the evaluation

In Figure 3.12 we show some of the patches correctly classified by the CNN with two convolutional layers. The patches that are wrongly classified as parasites look like real parasites. Furthermore, all the missed parasites are difficult to distinguish for various reasons, mainly because the contrast of the patch is very low or because they are partly covered by other structures. Some of this patches have been identified as labelling mistakes (marked with a blue star), which may be a limiting factor for our models.

Although these results are useful to evaluate the networks they do not reflect the real performance of the network if we want to use it to diagnose malaria. In both datasets the positive patches are centered at parasites and those positions are unknown for new blood samples. In the case of the *difficult dataset* there are also patches centered at other objects that are similar to parasites. However, this dataset contains only the objects that were found by one particular model and those may not be representative enough.

Figure 3.12: **a)** parasites correctly classified by a CNN with 2 convolutional layers. **b)** parasites that are missed. **c)** patches wrongly classified as parasites. The blue star indicates parasites identified as mistakenly labeled



Because of this, it will be necessary to run an sliding window and to merge close detections that correspond to the same parasite. This problem is addressed in the next chapter. As various CNNs with different structures have asimilar performance on this task, we will consider all of them for the detection stage. We will optimize the detection for each model and compare them again.

3.5 Implementation Details

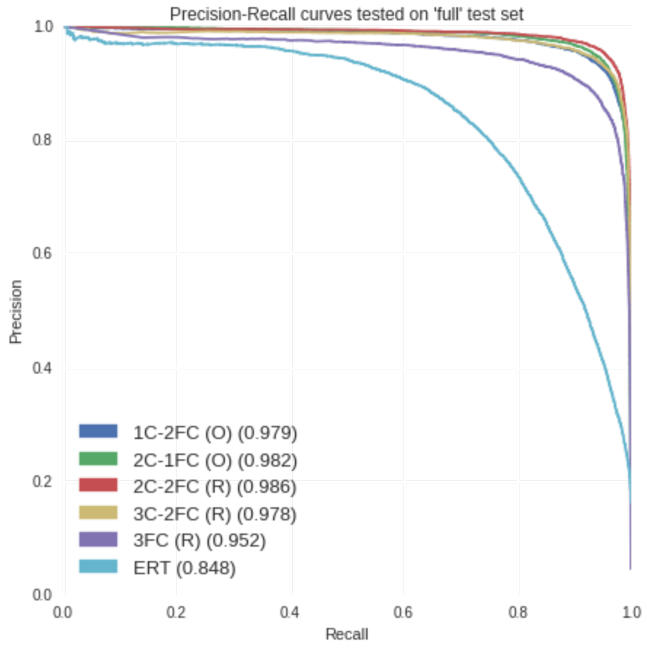
The deep learning framework Caffe [2] has been used to implement all the deep neural networks. Although similar third party libraries exist, it has been chosen because it makes easy to implement all the neural network layers needed for the project. Furthermore, it allows the models to be trained on GPUs, which makes the training process much faster.

A set of Jupyter Notebooks [3] using Python [4] have been created to build the datasets, evaluate and compare the models, visualize the results and automatize other tasks.

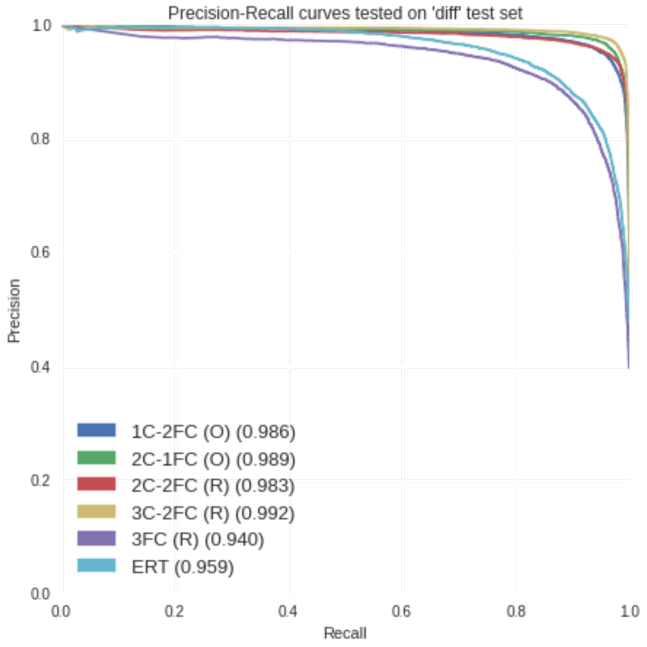
We have used the implementation of *Quinn et al.* model [1] available on GitHub [5].

[2] http://caffe.berkeleyvision.org
[3] https://jupyter.org
[4] https://www.python.org
[5] https://github.com/jqug/mobile-microscopy

Figure 3.13: Precision-recall curves of the models included in tables 3.5 and 3.6.



a) 'full' test set



b) 'difficult' test set

Chapter 4

Parasite Detection

In this chapter we use the classifiers built in Chapter 3 to find the position of each parasite in the images. We consider several convolutional neural networks as well as the other classifiers that we have presented, and we optimize the detection stage for each one of them. CNNs followed by a detection stage achieve better results than the previous ERT model and the fully connected networks.

In Section 4.1 we describe the methods we have used. They include the probability maps created by classifying the patches at each point of an image and how the results can be merged using a non-maxima suppression method. Additionally, we describe some attempts to improve the performance of the model by using the results of the detection process using hard negative mining.

The evaluation of the methods and the results that support our decisions are presented in section 4.2. In the same section we analyse how good the performance of the system is in different images and we compare the results of the all the models.

4.1 Methods

The main part of the detection stage consists of a non-maxima suppression method as described in section 2.3.1, based on the work of *Felzenszwalb et al.* [8]. However, we first create a probability map that makes it possible to use some pre-processing techniques before running the algorithm (explained in sections 4.1.1 and 4.1.2). Section 4.2.3 describes how we have tried to improve the performance of the model by doing

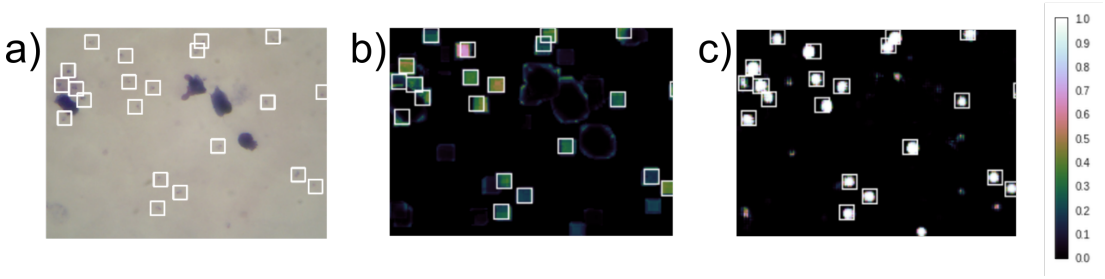
hard negative mining.

4.1.1 Probability maps

The probability maps show how probable is for a point of an image to be a parasite. We expect this probability to be higher the closer it is to the center of a parasite.

To create the probability maps we use a sliding window and classify a patch every 5 pixels. The map can be displayed as an image (in Figure 4.1 for a CNN with two layers and an ERT). The maps of the two models are different. The high probability areas of the CNN are much closer to 1 and the shapes of the high probability areas are rounded and irregular. For the ERT model, however, they are squared, which may be explained because they are invariant to translations.

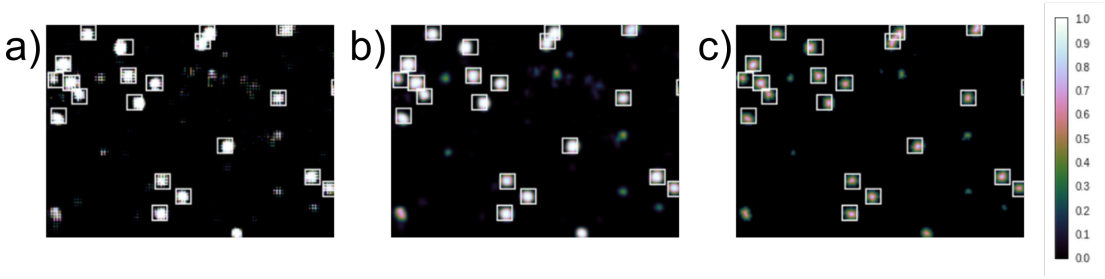
Figure 4.1: **a)** A sample image. **b)** Probability map of the ERT. **c)** Probability map of a CNN with 2 convolutional layers.



Having the full probability map instead of just the position of the patches where we detect a parasite (those above a certain threshold) allows us to preprocess the map before the non-maxima suppression algorithm. We convolve the map with a Gaussian kernel with standard deviation $\sigma = 1$ pixel, which smooths the changes on the map as it can be seen in Figure 4.2. A similar approach can be found in the work by *Ciresan et al.* [20]. This improves the results of non-maxima suppression depending on the particular parameters of the algorithm (see the evaluation section).

It is possible to use alternative methods to process the probability maps although we have not found them useful. In particular, we have run the non-maxima suppression using a Euclidean distance transform where each point of the map will be transformed into its distance to the background. The background includes all the pixels with values below a threshold. This filter is usually used before the Watershed algorithm for object segmentation [23]. Similarly, here we are segmentating the probability map. The result

Figure 4.2: **a)** Raw probability map for a CNN. **b)** Probability map after the Gaussian filter. **c)** Probability map after a Euclidean distance transform.



of this filter is that we lose the information about the probability, which may be more informative than the distance to the background (i.e. the size of the blobs), while the Gaussian filter preserves it.

4.1.2 Non-maxima suppression

The non-maxima suppression algorithm (as described in section 2.3.1) involves two different parameters: the threshold that indicates what patches are accepted as positive and the overlap to merge the detections. We also consider several ways to compute the score of a detection.

Originally, the score of a detection was the probability of the maximum value, the other detections are removed and not considered. We compare that to other three options. The first one is the average over a region of 40×40 centered at the maximum. The other two options only consider the values above a threshold (the probabilities of the *initial detections*) and are computed as the average value of the detections and the sum of the detections. All three alternatives have the advantage of considering not only the maximum value but also the surrounding area. Furthermore, the mean over a region and the sum of the *initial detections* take into account the size of the high-probability region.

To set the two parameters we use a grid search. For each classifier and combination of the other parameters (the filter and the way to compute the score) we execute a first search with values spaced by 0.2 (i.e. 0.1, 0.3, 0.5, 0.7 and 0.9). Then, we run the same search by values spaced by 0.05 (e.g. if the best value for a parameter was 0.3 we test the values: 0.20,0.25,0.3,0.35,0.40).

Figure 4.3: Images of an area with two parasites (white sqaures) and detections after having applied different overlap thresholds (purple rectangles). **a)** The overlap threshold is too small (0.1) and the points corresponding to both parasites are suppressed by the first one **b)** The threshold is correct (0.3) and both parasites are detected. **c)** The threshold is too large (0.4) and many duplicated detections appear.

For the overlap threshold, if the value is too large, then we could be suppressing points that belong to a neighbouring parasite. However, if it is too small, we will not suppress some patches that might be detected as another parasite leading to a wrong detection. Figure 4.3 shows three images to illustrate the above situations.

If we apply the Gaussian filter and take the maximum value it is not necessary to set a minimum value to accept detections. We can just consider all the points with a probability greater than 0. Nonetheless, with the current greedy implementation it will take longer if the value is very small, so we use a value of 0.05 as smaller values do not present a relevant improvement. For other cases this value must be tuned as well. For instance, if we take the sum of the positive patches, which is the best method without the Gaussian filter, it works better if we exclude points with low probability.

4.1.3 Hard negative mining

Hard negative mining (as described in section 2.3.2) allows us to find the difficult negative patches and feed the network with them. Using the results of the detection stage of the CNN with 2 convolutional layers we create a new training set composed by all the positive patches (25% of the images), negative patches taken at random (50% of the images) and patches taken at the center of the objects wrongly identified as parasites with a probability score about 0.9. As there were vert few wrong patches than positive ones we repeat each patch of them 8 times to account for 25% of the new dataset.

To label the new dataset we propose two alternatives. One consists in labeling the non-parasite objects with the same label as the negative patches. The other in labeling them with a third label, which is considered the same as the negative in test time.

4.2 Evaluation

In this section we evaluate the results of the full detection pipeline. We consider several of the models described in the classification chapter: the CNN with the best results for 1, 2 and 3 convolutional layers, the fully-connected network and the ERT model from *Quinn et al.* [1]. For each one of them we optimize the parameters of the detection stage so we can find the best overall detection system.

The main measure we use is the average precision (AP), which corresponds to the area under the precision-recall curve in the context of object detection [21]. Thus, the optimization consists on maximizing the AP. A detection is considered a true positive if it overlaps with the bounding box of a parasite above a threshold. All the detections are assigned a bounding box of 50×50 to compute the overlap, as this is the size of the patches. If two detections are assigned to the same parasite, they count as one true and one false positive.

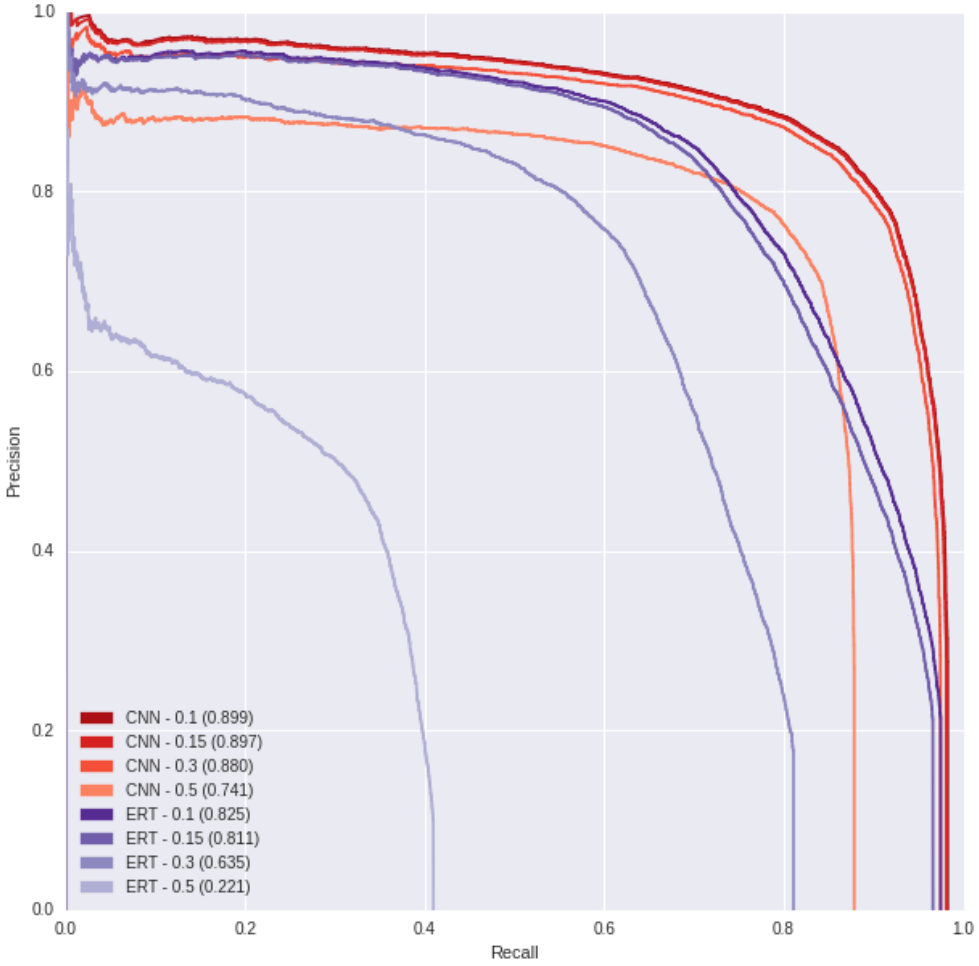
We set the minimum overlap of a true positive as 0.15 as defined in Section 2.3.3 (it is the intersection divied by the union of the two bounding boxes). Both the detections and the true parasites have a size of 50×50 (except for some parasites that are cut off by an edge of the image). This means that, approximately, the detection has to cover at least 25% of the bounding box of the parasite ($25^2/(2\cdot50^2-25^2)=0.15$ and a square of 25×25 is the 25% bounding box).

This threshold is similar, although not identical, to the one used by *Quinn et al.* to assign a positive label. In that case the patches are positive if the center of a parasite is inside them, so the overlap is at least 0.15, although not all patches with an overlap of 0.15 are accepted. As the labels are not intended to be precise and we are not interested in finding the exact location, we consider this an adequate value.

Incrementing the minimum overlap to 0.3 (i.e. the detection covers approximately 50% of the parasite of the bounding box) has little effect on the performance of CNN as shown in Figure 4.4. An overlap of 0.5 (i.e. covering approximately a 67% of the parasite) reduces the AP from 0.89 to 0.74. For the ERT, however, the AP is reduced drastically when the overlap threshold is incremented, from 0.811 to 0.221 when the threshold is incremented from 0.15 to 0.5. This suggests that the CNN model does not only detect more parasites but also finds more precise locations.

In sections 4.2.1 to 4.2.3 we evaluate how the AP is affected by the pre-processing

Figure 4.4: Precision-recall curves of the CNN and the ERT for different overlap thresholds to accept detections. The legend shows the method (CNN or ERT), overlap thresholds and AP (in brackets).



of the probability map and the method to compute the score, the threshold to accept detections and the overlapping threshold to merge detections. As the probability maps for all the CNNs are similar, we use the CNN with three layers to illustrate these sections. Section 4.2.4 describes the results obtained with hard negative mining.

Finally, in section 4.2.5 we evaluate the results of the model, including an analysis of the results per image instead of looking only at the AP of the parasites detected. In this section we study the behaviour of the system depending on the characteristics of the image and how well we are able to count the parasites.

4.2.1 Preprocessing of probability maps and score

Table 4.1 shows the best results for each way to compute the score both with and without the Gaussian filter. If it has not been applied the summation of the probability values is the best score: an AP of 0.883 compared to 0.860 of taking the maximum. However, with the Gaussian filter, taking the maximum is the best method (0.900).

If the Gaussian filter is not applied to the probability map the best results are provided by the sum of the probability of the merged pixels. Nevertheless, after applying the Gaussian filter, taking the maximum value outperforms the other options.

This confirms that having information of both the maximum peak and the surrounding area is a better score. Furthermore, the Gaussian filter summarizes that information better than the sum or the mean of the values over a certain threshold.

Table 4.1: Best AP values after obtained for each way of computing the score of a detection both for the raw probability map and after applying a Gaussian filter.

	No filter	**Gaussian**
Max	0.860	**0.900**
Mean	0.858	0.886
Sum	**0.883**	0.884
Area	0.746	0.744

It has not been possible to get a good performance by computing the score as the mean of the area surrounding the center, but tuning the size of the area and weighting each value by its distance to the maximum should report similar results.

4.2.2 Threshold to accept detections

For the best choice of the previous section (Gaussian filter and the maximum as the score) having a lower threshold always improves the results. This means that taking the values of all the patches improves the performance. However, that requires more computations as it is necessary to compute the overlap between every two points of the probability map. Although more efficient implementations would solve this, we select a minimum value of 0.05 as the difference of the performance is not relevant. In Table

Table 4.2: AP for different values of the threshold to accept detections. We show the results for both the maximum score after a Gaussian filter and for a sum score without any filter.

Threshold	**Max. (Gauss)**	**Threshold**	**Sum.**
0.9	0.836	**0.99**	0.844
0.5	0.890	**0.95**	0.877
0.25	0.898	**0.9**	**0.882**
0.1	0.900	**0.5**	0.880
0.05	**0.901**	**0.1**	0.874

4.2 we show the AP for different overlap thresholds; the overlap threshold to merge has been fixed to a value of 0.3.

For other choices a lower threshold does not mean better results. For instance, if we do not apply the Gaussian filter and we use the sum score, the best value is not the lowest one but an intermediate one (0.9) as displayed in Table 4.2. This may be explained because the areas with low probability are caused by noise and do not really correspond to patches containing a relevant portion of the parasite.

4.2.3 Overlap to merge

Using the Gaussian filter and the maximum score, the best value for the overlap threshold to merge is 0.3 (which holds for all the neural networks). This overlap is not the same value as the overlap to accept a detection described in section 2.3.3 and used here for evaluation; it is the percentage of a the bounding box of a parasite that is covered by a detection. Given that the bounding boxes of all detections have the same size both measures are equivalent (the value of 0.3 would be 0.176 using the same equivalence as in the introduction of Section 4.2). The AP for different thresholds is displayed in Table 4.3.

Table 4.3: AP value for different overlap thresholds to merge detections.

Threshold	**Max. (Gauss)**
0.1	0.881
0.25	0.899
0.3	**0.901**
0.35	0.899
0.40	0.894

Table 4.4: Results when hard negative mining is used both with two and three classes. The values are the AUC on the classification train set and the AP on the detection validation set.

	Original	**HNM (2 classes)**	**HNM (3 classes)**
Classification (train set)	0.987	1.000	0.999
Detection (validation set)	0.896	0.849	0.895

4.2.4 Hard negative mining

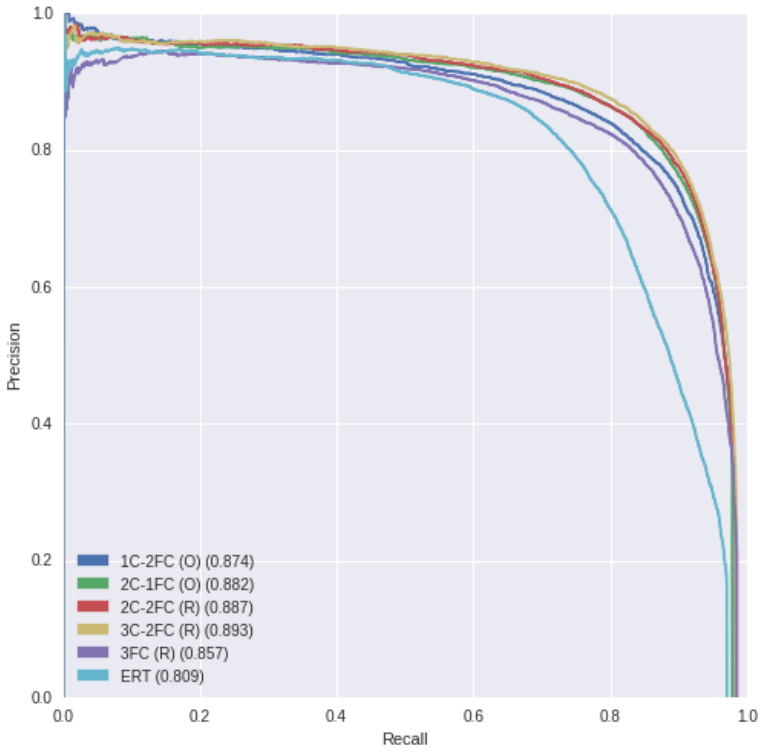
We evaluate the results when hard negative mining (HNM) is used by applying to the previous wrong detections the negative label and when a new label is used as described in section 3.1.3.

In both cases, HNM improves the performance on the classification training set (even when it is trained only with a subset of such set) to a value close to 1. However, in the test set and after optimizing the non-maxima suppression method HNM reduces the performance (with two classes) or has no effect (with three classes). The resulting values are displayed on Table 4.4.

4.2.5 Classification models

As it has been previously said, we optimize the detection stage for several classifiers. We present the results for both the validation set (used to tune the parameters) and

Figure 4.5: Precision-recall curves for the classifiers considered (on the test set).



the test set. For all the methods, the values are slightly higher on the validation set, although the difference small (Table 4.5).

All three CNN with multiple convolutional layers that we consider achieve better performance than the other models. They correspond to a network with two convolutional layers and one fully connected layers trained with the *rebalanced* dataset (2C-2FC (R)), the same network with two fully connected layers and trained with the original dataset (2C-2FC (O)) and the CNN with three convolutional layers (3C-2FC (R)).

The results of these three networks are very close. We use bootstrapping with 500 replications and sort the results by the median value of all the executions. All the differences are significant in this case. The one with the highest median AP is the network with three convolutional and two fully connected layers (0.8943 on the validation set).

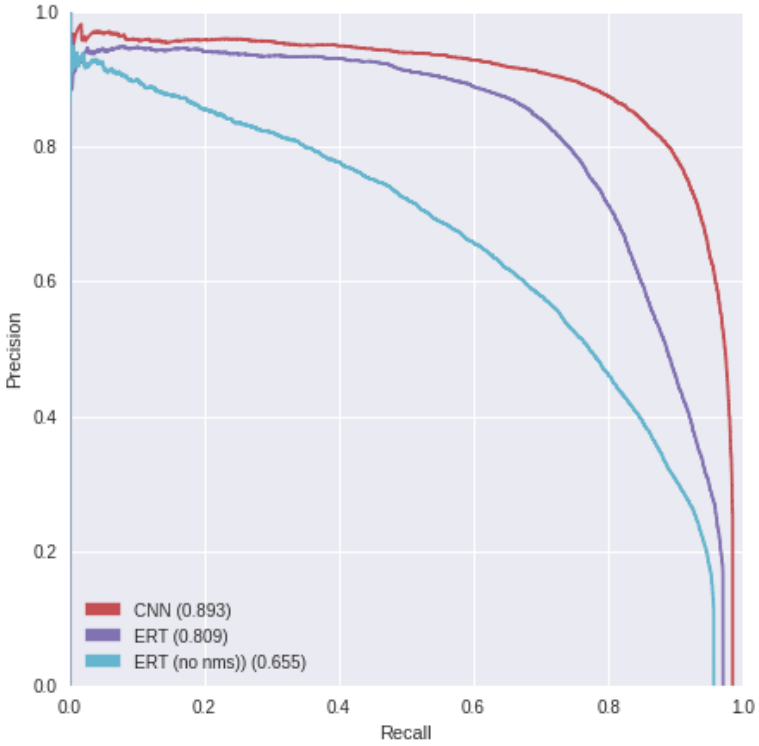
The CNN with one convolutional layer (with a median AP of 0.8768) is still better than the fully connected network (NN-R) and significantly better than the ERT model (0.8107). The precision-recall curves of all the models are displayed in Figure 4.5.

Table 4.5: Average precision of several models on both the validation and test set. They are ranked by its median value after bootstrapping, all the results are significantly different.

	Validation set		
	AP	**Median (boots.)**	**Rank**
3C-2FC (R)	**0.9006**	**0.9004**	**1**
2C-2FC (R)	0.8965	0.8964	2
2C-1FC (O)	0.8942	0.8939	3
1C-2FC (O)	0.8884	0.8884	4
3FC (R)	0.8714	0.8711	5
ERT	0.8105	0.8108	6

	Test set		
	AP	**Median (boots.)**	**Rank**
3C-2FC (R)	**0.8927**	**0.8943**	1
2C-2FC (R)	0.8867	0.8883	2
2C-1FC (O)	0.8817	0.8835	3
1C-2FC (O)	0.8745	0.8768	4
3FC (R)	0.8557	0.8593	5
ERT	0.8092	0.8106	6

Figure 4.6: Precision-recall curves of the best CNN, the ERT model with non-maxima suppression and the original ERT model.

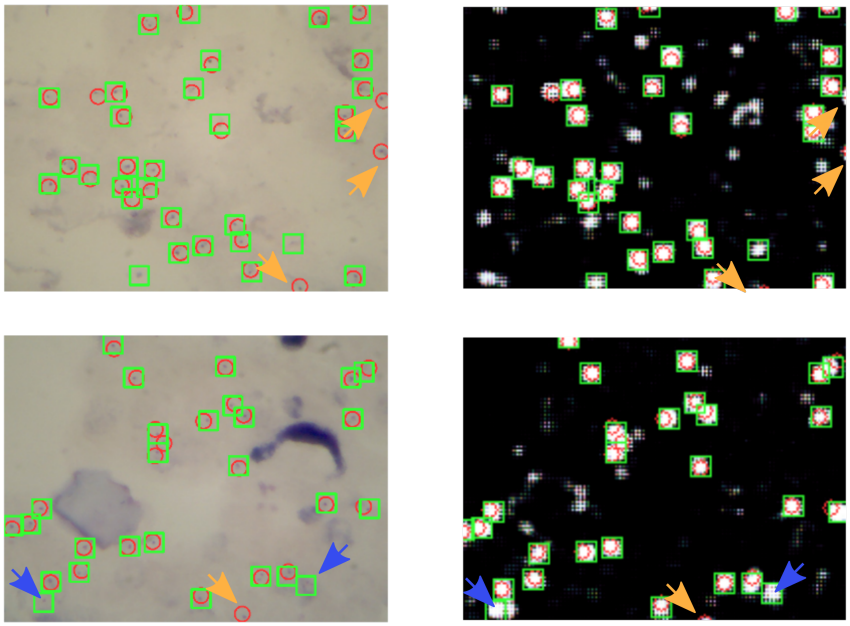


4.2.6 Evaluation of the best model

In this section we evaluate the best parasite detector (the CNN 3C-2FC (R) followed by non-maxima suppression) in depth. As stated, this model achieves an average precision of 0.8956 on the validation set. The F1 score on this dataset is of 0.8541 with a threshold of 0.9264 to consider a detection as a valid parasite. This corresponds to a precision/recall of 82%/88%. The same procedure with the ERT model leads to a precision/recall of 80%/73%. In the same way, if we want to recall 90% of the parasites identified by the experts we have a precision of 78% with the CNN and 46% with the ERT.

For most images the results are similar to those displayed in Figure 4.7, where the green rectangles represents the detections and the red circles the true parasites. Some detections are clearly wrong, see those marked with a blue arrow. Many of the missed parasites, however, are due to their position close to an edge, those identified with a orange arrow. If we ignore parasites in a margin of 10 pixels on the edges of the image the AP increases from 0.8927 to 0.9026.

Figure 4.7: Samples of the detections in typical images. Red circles represent the true parasites and green rectangles the detections of the best CNN model with non-maxima suppression. Blue arrows indicate false positives and orange arrows indicate false negatives caused by the position of the parasite too close to an edge.

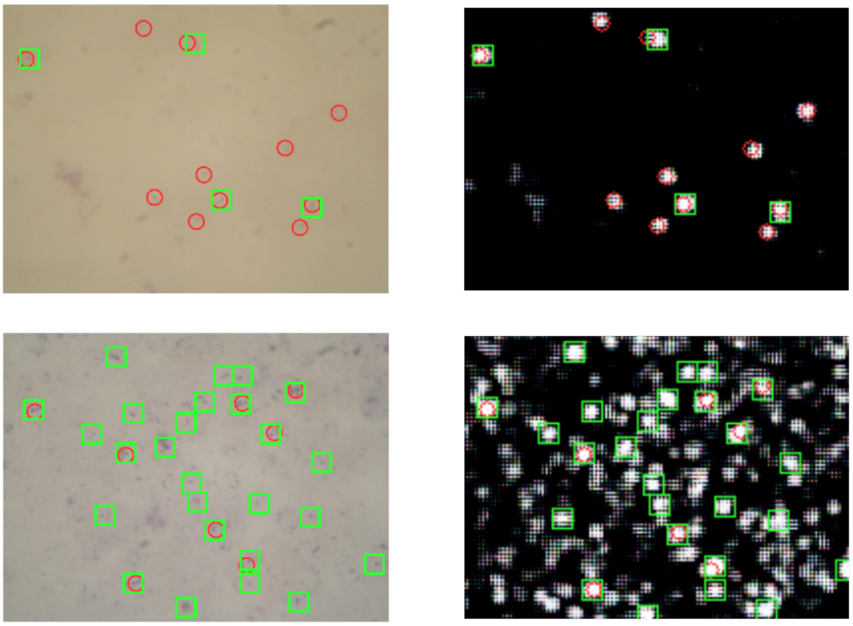


To find the worst cases we look for the images with the lowest recall and precision. We show one example of each case in Figure 4.8. Most of the images with low recall (as the one on the top of the figure) have low contrast and it is difficult distinguish the parasites. The parasites still correspond to areas with high probability, but they have been suppressed to avoid a high number of false positives on other images. The images with low precision (i.e. many false positives) are like the one on the bottom of the figure. These have many objects that look similar to parasites but are not labeled as so. A lower threshold would increase the recall on the top figure but decrease the precision on the bottom one.

In the rest of this section we analyse this results in terms of different charasteristics of the images. If we can detect what are the images with low recall/precision we could automatically adjust the threshold for each one of them and significantly improve the performance.

To have a better insight of the images where the model is not working properly we analyse the results in terms of the number of parasites of each image. Figure 4.9 shows the number of parasites detected compared to the true number of parasites. The Pear-

Figure 4.8: Image with the lowest recall (top row) and the lowest precision (bottom row). Red circles represent the true parasites and green rectangles the detections of the best CNN model with non-maxima suppression.



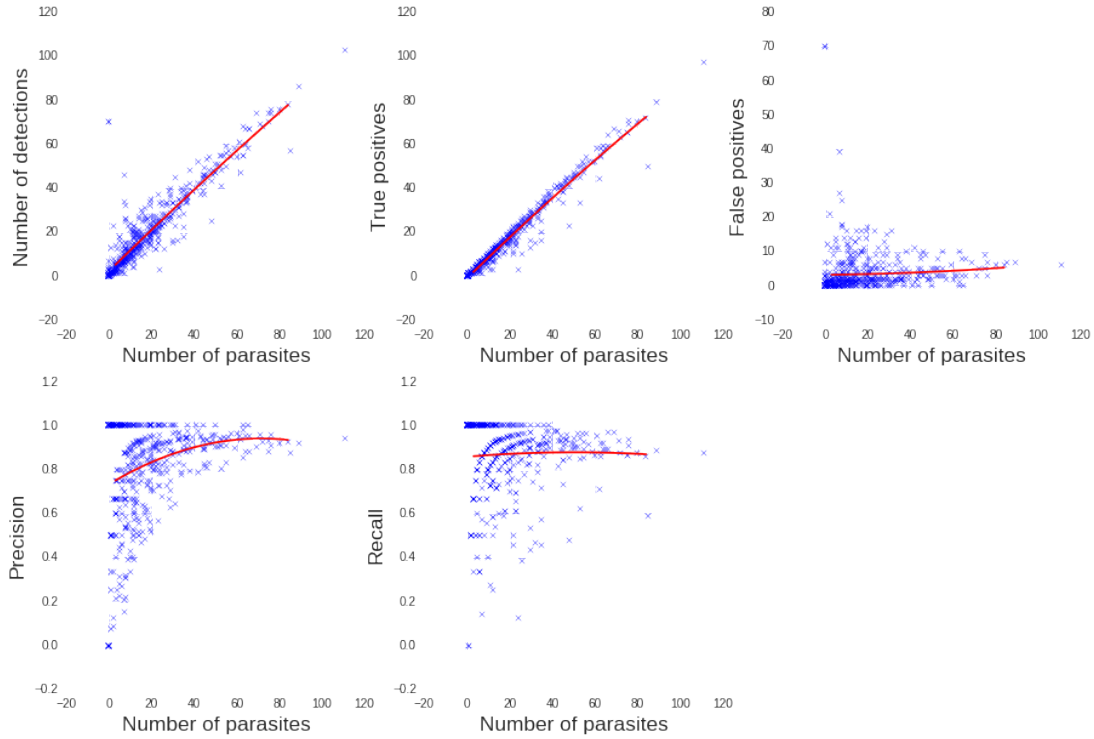
son product-moment correlation coefficient (PCC) (the covariance of the two variables divided by the product of variances) between them is 0.93, which means that there is an strong linear relation.

The same figure includes the number of true and false positives, the precision and the recall plotted against the number of parasites. The number of true positives is very close to the number of parasites for all values. The median number of wrong detections is similar across all number of parasites, which explains why the precision is lower for images with a low number of parasites. For example, five wrong detections in an image with only one parasite leads to a precision of 16% but if there are 10 parasites the precision is 62%. All the values have been obtained by accepting detections with a probability above the threshold established by the F1 measure computed on the validation set.

The mean absolute error (MAE) between the real and the detected number of parasites is 3.39. However, if we optimize the threshold for this measure we obtain a value of 3.24 and it could be even lower if we optimize the full system, and not only this threshold, using this measure. This could be useful if we are interested in predicting the number of parasites.

]

Figure 4.9: Different measures in terms of the number of parasites. On the top row: number of detections, true positives and false positives. On the bottom row: precision and recall. The red lines correspond to fitting a second order polynomial to the mean for each value of the number of parasites by minimizing the squared error.



There are only 52 images without parasites out of 475 images on the test set. Using the same F1 measure we compute the number of them that are correctly classified as infected or not. We get a precision of 97% and a recall of 99%. 2 out of 52 of the non-infected are classified as having parasites, while 17 out of 489 of the infected images are missed. As with the number of parasites, optimizing the full system for this goal will likely improve the results.

Finally, we show the same variables as in Figure 4.9 but for the entropy of the histogram of the grayscale image (Figure 4.10) instead of the number of parasites. We expect to find differences on the performance in terms of the complexity of the image. The entropy has been more informative than the contrast (computed by subtracting the minimum value from the maximum value). It is computed as $-\sum_i p_i \log(p_i)$ where p_i is the percentage of pixels of the image that have the value p_i. Images with a high entropy have a more parasites, as each parasite contributes to the complexity of an image: a plain background would have the minimum entropy and adding a parasite will

increase it.

Images with a high entropy have a lower precision. Thus, the threshold could be tuned depending on the entropy of the image. However, this relation is weak with a PCC of 0.21 between the entropy and the number of false positives and we have not been successful on this.

In Figure 4.11 we display the images with the highest and lowest entropy. On the top row we see two with high entropy, one of them corresponds to a broken smear and the other one to an image with a blueish tone. The images on the bottom have low entropy and have very few or none objects. This technique may be useful to find unusual images that may not be valid for the diagnosis of malaria.

Figure 4.10: Different measures in terms of the entropy of an image. On the top row: number of detections, true positives and false positives. On the bottom row: precision and recall. The red lines correspond to fitting a second order polynomial to the mean for each value of the entropy (grouped in bins) by minimizing the squared error.

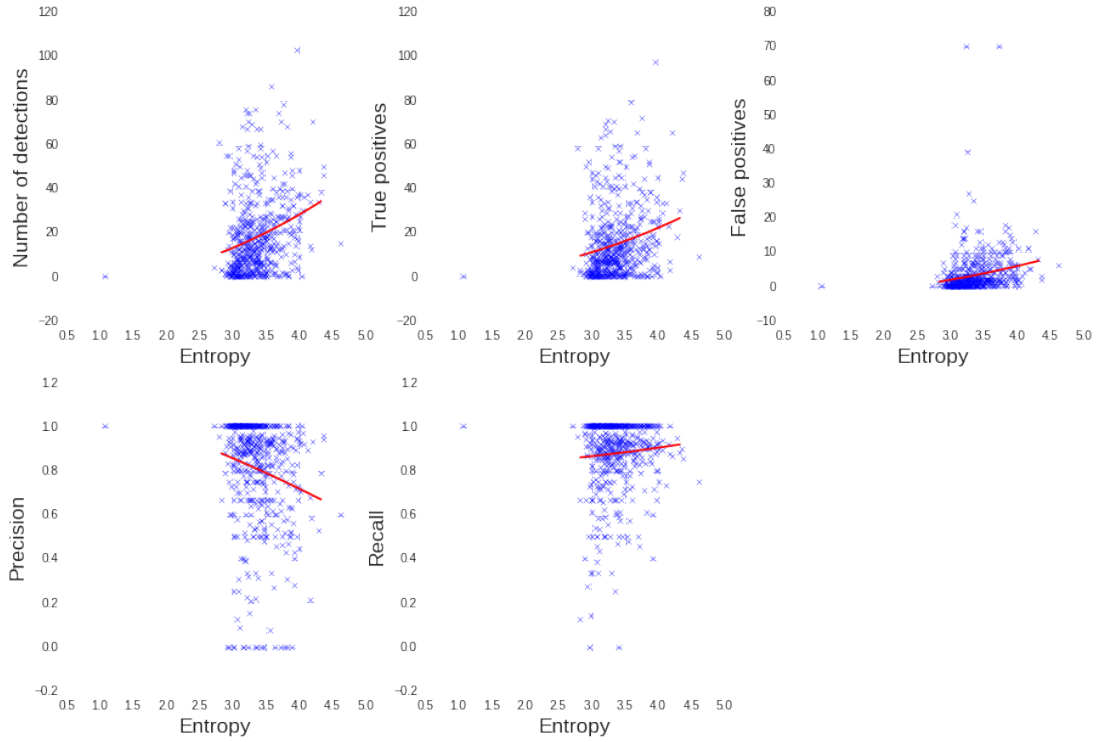
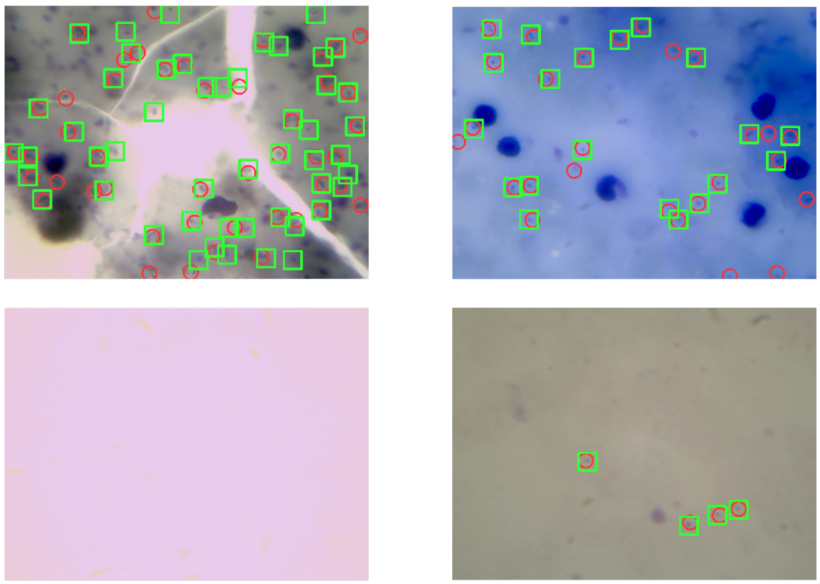


Figure 4.11: Images with the highest entropy (top row) and the lowest (bottom row).



Chapter 5

Conclusions and Future Work

During the previous chapters we have proposed a system to detect malaria parasites based on state-of-the-art methods in the field of image recognition. Considering the importance of malaria diagnosis and the lack of resources and trained laboratory technicians in developping countries, computer vision techniques can be of much help for health workers.

Our system was divided in two stages: classification and detection. The classification stage consists in predicting whether an small portion of an image contains a parasite or no. For this we have used deep learning methods, in particular Convolutional Neural Networks). Using CNNs has lead to considerable improvements both compared to the model previously used on this dataset (Extremely Randomized Trees) and to non convolutional networks.

In the detection stage we extract the position of each parasite from a probability map created by running the classifier with a sliding window through the original image. This method, based on non-maxima suppression, allows us to merge the results of close detections and discriminate between parasites. We have selected several networks and then we optimized the parameters of the detection stage for each one of them to select the best architecture.

The best classifier has been a CNN with 3 convolutional layers and 2 fully connected layers. In order to find this model we have analysed different aspects of the network such as the width of the kernels, the stride used to apply them or the number of convolutional layers. We have paid special attention to the way of extracting the patches from images to train the network. In the same way, we have analysed the effects of the

parameters of non-maxima supression, and how the probability map can be processed to maximize the performance. Additionally, we have proposed and compared several ways to assign an score of confidence to a detection.

One of the particularities of this project is the use of images taken in real field conditions, which hinders the detection task. We are able to detect parasites with an average precision (AP) of 0.89 and a precision/recall of 82% / 88%. Using this system for diagnosis purposes requires a high recall, and to find 90% of the parasites that the experts have found we get a precision of 78%, a significant improvement compared to the previous work [1], which achieved a precision of 37%.

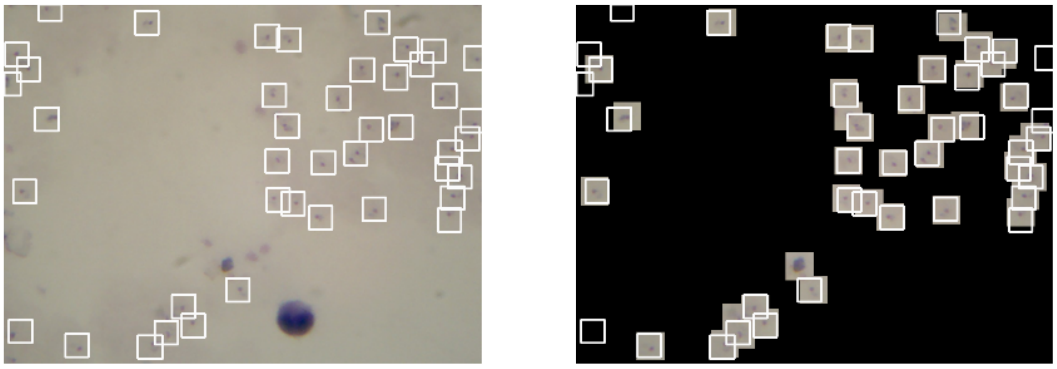
To consider the system for automated diagnosis more data would be necessary and it should be evaluated on the full data of every patient. Nonetheless, it can be already useful to improve the work of the laboratory technicians. Looking at blood samples for a long time is a fatiguing task and the technicians are required to look at the data of each patient for at least ten minutes. Using an operating point with high recall would allow us to display only the portions of the image where we find candidates to be a parasite. This would drastically reduce the time they need to look at each sample and would make the task easier.

In Figure 5.1 we show a sample where only the areas corresponding to parasites are visible. In this image, those areas correspond only to 17.7% of the total image. The authors of the dataset we use have already been working on low cost adapters that allow a mobile phone to be attached to a microscope [1]. This way, the identification of parasite candidates could be done in real time.

One advantage of deep learning methods is that we do not need to hand craft an appropiate representation of the image; they learn it as well. Adapting the ERT model to a new kind of parasite or to detect other objects in blood samples might require designing a new set of features. However, with CNNs, if our architecture is learning useful features, we can expect that it will work as well in other cases. A system based on the work presented here could be useful for the diagnosis of various diseases and more data is already being collected with this goal.

Examining the results of the network, as we have shown, has revealed some mistakes on the labels of images. The results of the system might be used to revise the dataset. The experts could check the objects labeled as parasites that have a low probability in our system and those detections that do not correspond to labeled parasites. Further-

Figure 5.1: The left image shows a sample blood smear. On the right, only the areas where the system has detected candidates to be parasite are displayed, which account for 17.7% of the image. The white squares indicate the location of the parasites.



more, as the parasites appear in different stages and can belong to different species, having a more detailed annotation could lead to an improvement of the results. However, determining the stage and species is not always easy in thick films and the task could involve too much work.

Looking at Figure 5.1 it is notable that the missed parasites are close to the edges of the image. This happens as it is not possible to extract enough patches from those areas and the probability maps do not include them. Some alternatives can be found on the literature, in the work of *Ciresan et al.* for mitosis detection in breast histology images, they mirror the images on the edges to solve this problem [20]. This one or another alternative is likely to improve the performance.

Both our system and the previous work on this dataset use grayscale images. Color is not as informative with the stain used on the images as with other stains [1], but using color images might improve the results.

Even when the above suggestions might improve its performance, the system is already providing results good enough to be useful in malaria diagnosis as we have described. Computer vision systems such as this one are a inexpensive and fast way to enhance the diagnosis of various diseases and can considerably reduce the workload of experts. This is particularly important in the case of malaria infections, given the global impact of the disease and the lack of resources in the places where it is endemic.

Bibliography

[1] J. A. Quinn, A. Andama, I. Munabi, and F. N. Kiwanuka, "Automated blood smear analysis for mobile malaria diagnosis," in *Mobile Point-of-Care Monitors and Diagnostic Device Design*, vol. 114, pp. 21–32, CRC Press, 2014.

[2] J. Hutchinson and World Health Organization, *Malaria microscopy quality assurance manual*. World Health Organization, 1st ed., 2009.

[3] F. B. Tek, A. G. Dempster, and I. Kale, "Malaria parasite detection in peripheral blood images," *BMC Bioinformatics*, vol. 13, pp. 347–356, 2006.

[4] F. B. Tek, A. G. Dempster, and I. Kale, "Computer vision for microscopy diagnosis of malaria," *Malaria Journal*, vol. 8, p. 153, 2009.

[5] F. B. Tek, A. G. Dempster, and İ. Kale, "Parasite detection and identification for automated thin blood film malaria diagnosis," *Computer vision and image understanding*, vol. 114, no. 1, pp. 21–32, 2010.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems 25*, pp. 1097–1105, 2012.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[8] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[9] W. H. Organization, *World Malaria Report 2014*. 2014.

[10] M. Tumwebaze, "Evaluation of the capacity to appropriately diagnose and treat malaria at rural health centers in kabarole district, western uganda," *Health Policy and Development*, vol. 9, pp. 46–51, 2011.

[11] World Health Organization, *Basic malaria microscopy*. World Health Organization, 2nd ed., 2010.

[12] N. Ross, C. Pritchard, D. Rubin, and A. Dus, "Automated image processing method for the diagnosis and classification of malaria on thin blood smears," *Med Biol Eng Comput*, vol. 44, no. 5, pp. 427,436, 2006.

[13] S. Kaewkamnerd, C. Uthaipibull, A. Intarapanich, M. Pannarut, S. Chaotheing, and S. Tongsima, "An automatic device for detection and classification of malaria parasite species in thick blood film," *BMC Bioinformatics*, vol. 13, no. suppl. 17, p. 18, 2012.

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," 2014.

[15] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[16] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[17] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best architecture for object recognition?," *Computer Vision and Pattern Recognition, 2009 IEEE 12th International Conference*, pp. 2146–2153, 2009.

[18] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," *Proceedings of the 20th International Conference on Artificial Neural Networks: Part III*, pp. 92–101, 2010.

[19] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, 2003.

[20] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Mitosis de-

tection in breast cancer histology images with deep neural networks," in *MICCAI*, vol. 2, pp. 411–418, 2013.

[21] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

[22] M. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), vol. 8689 of *Lecture Notes in Computer Science*, pp. 818–833, Springer International Publishing, 2014.

[23] S. Beucher and F. Meyer, "The morphological approach to segmentation: the watershed transformation," *Optical Engineering*, vol. 34, pp. 433–481, 1992.