Detecting Intestinal Parasites in Images

Efstathios Lempesis

Master of Science

School of Informatics

University of Edinburgh

2016

Abstract

In this project we aim to build a detection system for intestinal parasites in microscopy images. The output of this system could overcome some of the current difficulties of intestinal parasite diagnosis such as the lack of laboratory experts and resources in the developing countries. In contrast to previous work on this task which only classifies the microscopy images, we also detect the parasites in the images. This of course makes our task harder.

We use various techniques in order to tackle several problems that we encounter at this project. The difficulties of this projects are the high background noise, the high dimensionality of the feature vectors and the small number of positive training instances. We apply widely used classification algorithms, such as logistic regression and Support Vector Machines, to classify individual patches as containing parasites or not. Then, we use computer vision techniques such as non-maxima suppression to merge the detections and locate each parasite.

For a recall of 90% we have gotten precision 66% for the parasite detection task, which is good enough to make the working conditions of experts working on intestinal parasites diagnosis better. Moreover, our model can be easily extended to the detection of other parasites, so various experiments may be run using our parasite detection system.

Acknowledgements

Foremost, I would like to express my deepest sense of gratitude to my supervisor, Chris Williams, for his continuous support throughout all this project. I would also like to thank Lukasz Romaszko for providing useful and very valuable comments and suggestions.

I would like to thank John Quinn from Makerere University, who created and provided me with the dataset.

Finally, I am grateful to my family for their support in all my pursuits and for making it possible for me to come to The University of Edinburgh. Thank you.

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.

(*Efstathios Lempesis*)

Table of Contents

Chapter 1

Introduction

Usually, we don't realize how common is to contract parasites. Consumption of contaminated food, water, fruits and vegetables and undercooked meat can be the reason that someone can contract parasites. If someone is infected by a parasite, it's very likely that he will pass it along. For example, parasites eggs can easily be transmitted onto anything we touch.

There are two groups of parasites, the first is the intestinal and urogenital and the second is the blood and tissue parasites (Rai et al., 1996). Jacobsen et al. (2007) investigated the existence of intestinal parasites in Quichua children in the highlands of rural Ecuador and in total 203 stool specimens were examined from children aged 12-60 months and found that 85,7% of them had at least one parasite. The World Health Organization[1] approximates the number of children that suffer from these parasites to more than 880 million. So, diagnosis is a essential aspect to cope with the diseases and microscopy analysis is a preferred method. However, the lack of experts who can examine stool samples to identify if someone is infected, and the decreasing importance of most intestinal parasites at the fecal examinations due to the high background noise, which results in an insufficiency of training material, are the biggest obstacles in the diagnosis in the developing countries.

Because it is not always possible an intestinal disorder to be diagnosed symptomatically, the parasitic infection can only be identified by laboratory investigation. For this task an adequately expert staff, who are able to examine fecal specimens, is needed, but this manual inspection might well lead to erroneous detections when dealing with large amount of samples. Rema and Madhu (2013) indicate that the presence of fecal impurities, errors in analysis procedures and the level of expertise of the human an-

[1] http://www.who.int/intestinal_worms/more/en/

alyzer can also lead to inconsistent results. When the trained staff examine the fecal samples for parasites, they usually search for eggs of helminths and cysts of protozoa. If the specimens are positive, they examine the sizes, shapes, and numbers to identify the parasites. The most obvious problem is that it is very time consuming for any lab staff to manually detect parasites. For this reason, parasite detection could take advantage of the automatic image processing techniques, but fecal impurities in microscopy images make this task a challenging one.

Computer vision could be used to locate the parasites in the microscopy images so that the experts will have less work to do. This project has as a goal to propose a system to detect intestinal parasites in images using state-of-the-art techniques, which are expected to provide high degree of reliability and objectivity, and be an improvement on the simple detection methods. Moreover, our intention is that we will positively contribute to the diagnosis, building a model that automatically detect parasites in intestinal microscopy images which have high background noise.

1.1 Dissertation outline

The outline of the dissertation is presented as follows.

Chapter 2 demonstrates the background needed for the completion of this dissertation. We introduce the relevant aspects of intestinal parasites analysis and the three parasites that we need to detect, review the previous work on the intestinal parasites diagnosis, and describe the methods and techniques that we use throughout this project.

Chapter 3 introduces the dataset on which we build our models. We presents how many images we have to train the models and how many parasites these images contain. We analyze how we split the dataset in training and test sets and how we select the training patches.

Chapter 4 covers the parasite detection task. We first describe the models and the training procedure we use. Then, we evaluate the models and the impact of the various methods that we apply on these models. All these methods have as a goal to improve the overall performance. Finally, we evaluate the performance of our final model.

In chapter 5 we cover the Hookworm parasite detection task. About the 80% of the parasites in the dataset are Hookworm parasites. Similarly, we evaluate the models and the methods we apply on them.

Finally, in chapter 6 we summarize the outcomes of this project and discuss the applications of the system and future work.

Chapter 2

Background

Section 2.1 presents the two main fecal examination methods and the two types of intestinal parasites. We introduce the three parasites that we need to detect in this project and we emphasize the importance of the microscopy image recognition in the parasite diagnosis. In section 2.2 we present the previous works with fecal samples carried out by researchers. We focus on the differences between these works and the task of our project, and the difficulties that we have to cope with in comparison with the previous works that were free of these difficulties.

Then, the rest of the chapter describes the particular methods and techniques we use to detect intestinal parasites and the reasons that make them useful for our project. Finally, in section 2.8 we describe how we evaluate the model.

2.1 Intestinal parasites diagnosis

Fecal examination is the most common diagnostic method. There are two types of stool examination. First, the macroscopy stool examination where experts search for blood and mucus, and second the microscopy stool examination that relies on the detection of trophozoites, cysts, eggs and larvae. The traditional microscopy method is the most widely used.

There are two classes for the intestinal parasites (Avci and Varol, 2009). These are Protozoa and Helminthes. Protozoa are single celled. The Protozoa are separated to many categories[1] such as Entamoeba, Giardia, Trichomonas, Cryptosporidium, Isospora, Pneumocystis and Balantidium. The Helminthes are separated to many

[1] http://www.southampton.ac.uk/ ceb/Diagnosis/Vol1.htm

categories[2] such as Fasciola, Schistosoma, Ascaris, Hookworm, Trichuris, Taenia and Enterobius. The normal stage for examination is the egg stage (our case).

Taenia, Hookworms and Hymenolepsis nana are among the most common Helminthes. More than half a billion people suffer from the Hookworm parasite globally (Fenwick, 2012) mainly from the developing countries because of poor sanitation. Dirt contaminated by feces is the main reason that humans can contract Hookworm parasites. The main symptom of the Hookworm parasite is the anemia. Taeniasis is an intestinal tapeworm infection caused by eating contaminated beef or pork. The symptoms of Taenia are pain, unexplained weight loss, blockage of the intestine and digestive problems. Hymenolepis nana is the smallest of all tapeworms found in humans and is most common in temperate zones infecting humans, especially children. Hymenolepiasis is infestation by Hymenolepis nana and the symptoms sometimes include vomiting, weakness, diarrhea, insomnia, muscle spasms e.t.c. (Haque, 2007).

The microscopy image recognition of parasite eggs is one of the most important ways of diagnosis for humans infected by parasites. In recent years, because of the lack of experts and experience in manual identification parasite researchers and computer researchers have been trying to apply automatic recognition of parasite eggs and real-time diagnosis in microscopy images to minimize the inaccurate identification rate. Moreover, the use of automatic computer vision diagnosis is an inexpensive solution and this is very important for the developing countries.

2.2 Previous work with fecal samples

Many researchers have focused on how to classify specific kinds of eggs. Most of them are concentrated on the automatic segmentation of parasite eggs for the extraction of characteristics and the classification of parasite eggs in later stages. The majority of the research papers implement the automatic segmentation by applying a thresholding method to the parasite egg objects, but also there are papers where humans take part in the segmentation of parasite eggs so that parasite eggs can accurately be extracted from images.

Yang et al. (2001) started research into the recognition of common helminth eggs in fecal samples, but the dataset, they used, was free of high background noise. Four morphometric features were selected on three morphological characteristics. These morphological characteristics represented shape, shell smoothness, and size, and two dif-

[2] http://www.southampton.ac.uk/ ceb/Diagnosis/Vol1.htm

ferent artificial neural networks were used for classification. They had 87 microscopy patches containing one out of seven common human helminth eggs and 100 patches with various artifacts. They obtained 88.2% success classification rate.

Sengur and Turkoglu (2004) have used pattern recognition techniques to recognise eight different types of parasite eggs. A feature extraction method based on invariant moments and a neural network classifier was used obtaining more than 90% correct recognition rates.

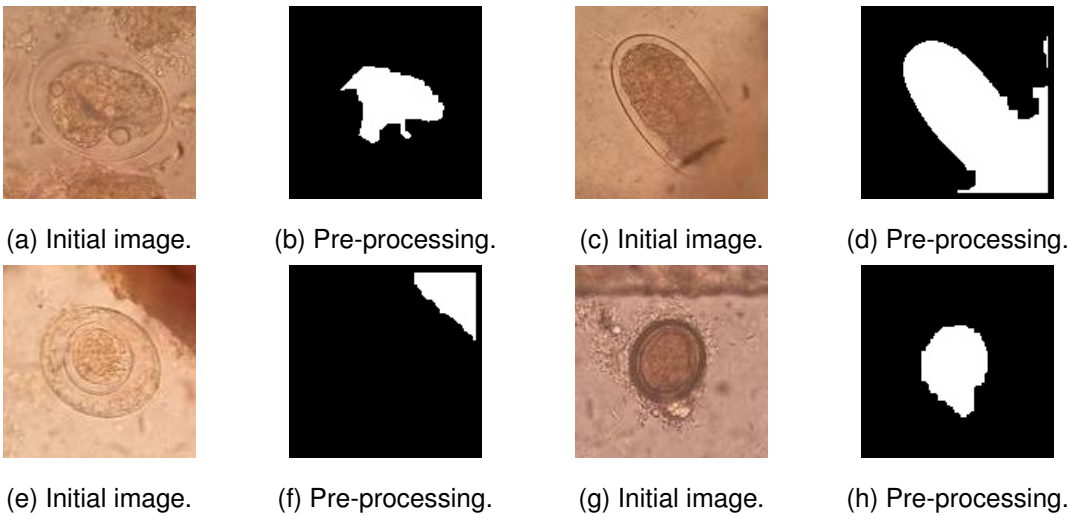
Dogantekin et al. (2008) proposed a fuzzy inference method based on invariant moments for recognition of parasite eggs in microscopy images. They classified parasite eggs of sixteen different types. The overall success rate was almost 95%, tested on small images having a parasite in their center.

Avci and Varol (2009) proposed an diagnosis system for classifying parasite eggs based on a multi-class SVM and invariant moments. This method consists of a pre-processing stage, a feature extraction stage, a classification stage, and a testing stage. Finally, the authors obtained a 97.70% overall success rate, using the same dataset with Dogantekin et al. (2008).

We tried to apply the techniques from paper (Avci and Varol, 2009) on our dataset. Figure 2.1 illustrates the results of our attempt. We can see from this figure that this pre-processing stage has bad results on our dataset's patches because we have high background noise in comparison with the dataset of (Avci and Varol, 2009). So, the image segmentation and thresholding computer vision techniques are not suitable for our training patches, since we cannot extract representative shapes for our parasite due to the high background noise. For this reason, in this project we don't use these types of pre-processing techniques.

Ghazali et al. (2013) proposed a human fecal parasite detection technique called Filtration and Steady Determinations Thresholds System (F-SDTS). They segmented and classified two different types of parasite eggs, with one hundred images for each parasite, from their microscopy images based on the F-SDTS classifier. The recognition method included three stages (image pre-processin, feature extraction and classification). However, the algorithm presented in this paper is limited to these two types of parasites.

Zhang et al. (2014) indicate that the parasite egg segmentation method used in Avci and Varol (2009) was based on too simple threshold operations. Moreover, Zhang et al. (2014) indicate that egg images from clinical testing samples are often obtained from fecal extracts (our case); these samples have the same features regarding the surround-



(a) Initial image. (b) Pre-processing. (c) Initial image. (d) Pre-processing.

(e) Initial image. (f) Pre-processing. (g) Initial image. (h) Pre-processing.

Figure 2.1: Effect of Dogantekin et al. (2008) pre-processing on the our dataset's training patches.

ing and internal contents. Also, due to the heavy background noise if we use threshold segmentation, it is usually hard to precisely recognize egg boundaries from fecal extract images. Zhang et al. (2014) apply a series of methods so that they can identify the boundary regions of the parasite eggs. The biggest challenge for the automatic segmentation is the background noise and our dataset does have high background noise. All these papers say that the most important thing to identify parasite eggs automatically is the automatic segmentation of parasite eggs in images for the extraction of the eggs' features. Castañón et al. (2007) isolate the parasite manually by using an image processing program (Gimp or Adobe Photoshop), and in Rema and Madhu (2013) the images are manually segmented by an expert in parasitology. These segmentations are used as the ground truth to estimate the accuracy of the proposed method. The accuracy is calculated by comparing the automatically segmented results with the manually-segmented results.

However, in the majority of these works the datasets only consist of images that in their center have a parasite and low background noise and are free of fecal impurities. Also, in our task we want to detect the parasites, so we have to slide a window through the whole image to detect the patches that contain parasites. But our dataset is highly imbalanced, approximately only 10% of the images contain parasites and moving the sliding window on the images we have to cope with the background patches and build a model that can distinguish between the patches that do contain a parasite and the patches that do not contain.

2.3 Object detection

The goal of object detection is to localize objects of known classes in images. In our case, we want to detect Hookworm, Taenia and Hymenolepsis Nana parasites in the microscopy images. Amit and Felzenszwalb (2014) note that we can regard each detection as the location of the object or the bounding box which contains the object. In this project our goal is to locate the parasites in the images using bounding boxes to declare their location. Previous work on classifying specific intestinal parasite eggs does not use the sliding window technique and the researchers do not attempt to find the location of the parasites. To locate and count the parasites that an image contains, we apply the non-maxima suppression (NMS) algorithm to merge the detections that correspond to the same parasite.

2.3.1 Non-maxima suppression

NMS (Felzenszwalb et al., 2010) is an algorithm that merges a set of detections that are assigned to the same object. NMS involves one parameter, the overlap threshold that determines if two overlapping windows will be reduced to one. This method is very useful and necessary because the detection algorithms make several detections near the real location of an object. When we use the sliding window technique, usually we get multiple windows with high scores close to the correct location of objects because of the generalization of object detectors. Using the NMS algorithm we can manage to keep only one bounding box per group of window detections. The most common approach is greedy NMS which chooses the best scoring window and then the bounding boxes that are too close to the selected window are suppressed.

We use the NMS implementation of Malisiewicz et al. (2011). This implementation is a modification of the implementation of Felzenszwalb et al. (2010), but it is $100\times$ faster by eliminating an inner loop. They replaced this inner for-loop with vectorized code. However, the code doesn't return the bounding box with the higher predicted probability. Malisiewicz et al. (2011) compute the area of the bounding boxes and sort the bounding boxes by the bottom-right y-coordinate of the bounding box. We modified the code and we use the fast non-maxima suppression but returning the bounding box with the highest predicted probability.

2.4 Image descriptors

We need to describe the properties of the objects in a unique way, if we want to successfully detect or classify objects in different classes. The descriptors can achieve this goal, if they have certain properties. First of all, if objects belong to the same class, they must have similar descriptors. Then, the detections should be invariant to rotation, illumination, scale and shift. They should describe special properties such as the shape, the color, the texture and should be compact containing information that makes the object unique.

2.4.1 Histogram of Oriented Gradients descriptor

Dalal and Triggs (2005) introduced the Histogram of Oriented Gradients (HOG) descriptor which is robust to changes in illumination and local changes in geometry. They proposed a human detection algorithm using HOG descriptor and showed that the HOG descriptor significantly out-performed existing feature sets for human detection even though partial occlusions occurred. The HOG feature is calculated by taking orientation histograms of edge intensity in a local region and is a dense descriptor computed on overlapping blocks along a grid of cells. The basic intuition behind the HOG descriptor is that the distribution of intensity gradients can be used to describe objects and shapes in images. These cells can be rectangular leading to R-HOG descriptor, or circular leading to C-HOG descriptor. The HOG descriptor is effective and widely used for classification tasks. The main disadvantage is that to get the HOG descriptor is computationally inefficient, and we apply dimensionality reduction to ease this computational load. Figure 2.2 illustrates the procedure of how HOG descriptor is computed. Figure 2.3 illustrates the HOG descriptor of some training patches and the HOG descriptor creates the features of the parasite and background patches respectively.

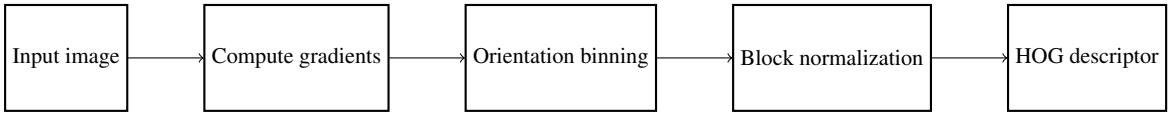


Figure 2.2: Creation of Histogram of Oriented Gradients descriptor.

(a) Hookworm.

(b) Hookworm HOG descriptor.

(c) Taenia.

(d) Taenia HOG descriptor.

(e) Hymenolepsis Nana.

(f) Hymenolepsis Nana HOG descriptor.

(g) Background.

(h) Background HOG descriptor.

Figure 2.3: Training patches and their HOG descriptor.

2.5 Dimensionality reduction

The goal of dimensionality reduction is to project the data on a lower dimensional space while retaining the important properties of the data for the task under consideration. As we know data of high dimensions can lead to computationally intensive and slow procedures. One well-known problem is the Curse of Dimensionality where if the feature space is extremely large, it may result in lower performance compared to that of a lower dimensional feature space. The term Curse of Dimensionality was introduced by Richard E. Bellman (Bellman, 1957) who considered problems caused by additional dimensions to Euclidean space. One widely used solution is the dimensionality reduction which can provide us with computational benefits obtained by processing lower dimensional data. But, it can also offer a secondary benefit that some noise and redundancies in the features are eliminated after reducing the feature space. As a result, the

data is projected on a lower dimensional space and is represented in a more compact and effective way, which can improve classifier performance. Moreover, since we have few training instances which are high-dimensional, it would be beneficial to reduce the dimensions.

2.5.1 Principal Component Analysis

Principal component analysis (PCA) transforms the data dimensions by retaining the first few principal components, which correspond to most of the variation of the initial data. In other words, PCA compresses the data. The linear projection created by PCA is a lower dimensional subspace of the initial high dimensional data maximizing the retained variance and minimizing the least square reconstruction error. In figure 2.4 we can see a toy example of how PCA compute the principal components of a dataset, the 1st principal component is the direction of the greatest variability and the 2nd direction is perpendicular to the 1st one. PCA can also recognize patterns in data and highlight their similarities and differences.

PCA has the following steps:

1. Before PCA we have correlated high dimensional data
2. Center the points (subtract mean from each attribute)
3. Compute points' covariance matrix C
4. Compute eigenvalues and eigenvectors of C
5. Pick the eigenvectors that account for the highest eigenvalues
6. Project points to those eigenvectors
7. After PCA we have uncorrelated low dimensional data

PCA is widely used in computer vision and one very famous application the eigenfaces (Turk and Pentland, 1991), where the eigenfaces are the principal components - eigenvectors of the covariance matrix of the set of face images.

Similar to Turk and Pentland (1991), we can plot the principal components of the parasites patches and background patches and we can also plot the principal components of all the dataset. Figure 2.5 shows these principal components. The purpose of these plots is to help us to understand the relevant parasite and background information. Since in section 3.1.2 we rotate our parasites 90°, 180° and 270° clockwise, we

can notice that the principal components of the parasite patches have a ball shape, but the background patches don't have any specific shape.

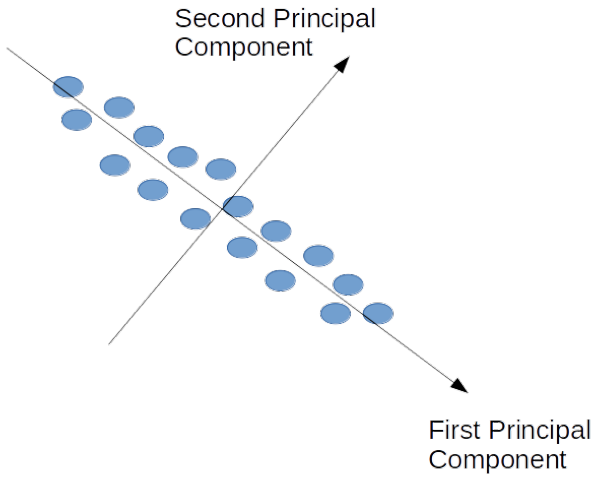


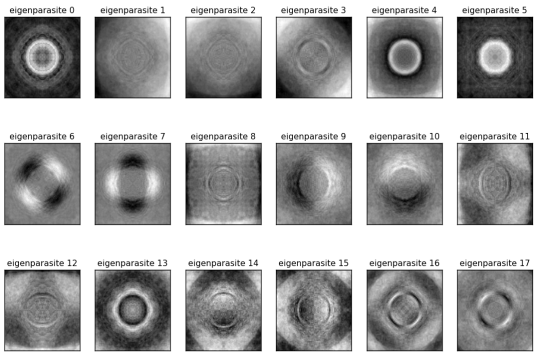
Figure 2.4: PCA on a toy example in two dimensions.

2.6 Classification

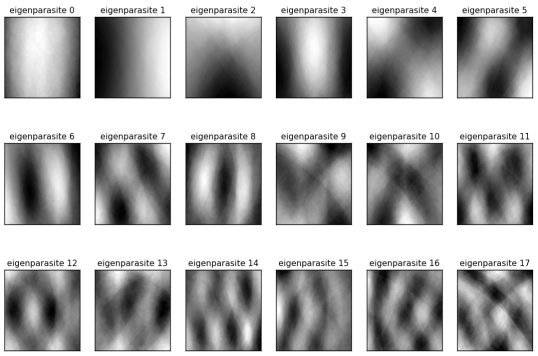
In machine learning, classification is the task of given a set of classes to predict which class a new instance belongs to. Training is conducted on a set of training data containing instances whose class is known. Classification is regarded as an instance of supervised learning. Clustering is the corresponding unsupervised algorithm which groups data points in classes depending on some measure of similarity or distance. A classifier, which usually refers to a mathematical operation that assigns input data instances to a class, is the algorithm that implements the classification. Figure 2.6 shows the procedure of how we can train a machine learning algorithm and then use it to classify new test instances.

2.6.1 Logistic regression

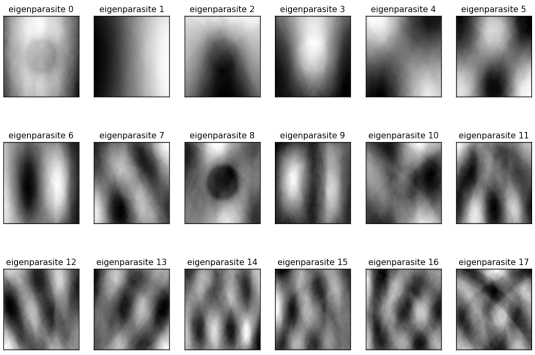
Logistic regression is a discriminative model. It models the probability $p(y|\mathbf{x})$, where $\mathbf{x}$ is the feature vector of a data point and y is the class label, directly without modelling the probability $p(\mathbf{x})$. Moreover, it is a linear model.



(a) Parasite eigenvectors.



(b) Background eigenvectors.



(c) Dataset's eigenvectors.

Figure 2.5: Eigenvectors.

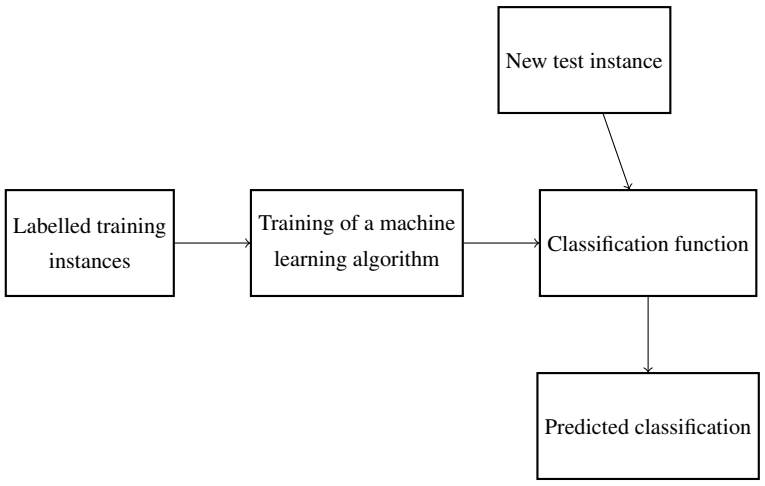


Figure 2.6: Training of a machine learning algorithm and classification.

Let consider a two-class linear classifier $y \in \{0,1\}$. We learn a function $f(\mathbf{w},\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$, $\mathbf{w}$ are the parameters that we learn from the training data. We can classify a new instance to belong to the class $y = 1$ if $f(\mathbf{w},\mathbf{x}) > 0$. So, our decision boundary

is the line $\{\mathbf{x}|\mathbf{w}^T\mathbf{x}+w_0=0\}$. In more than two dimensions, the boundary will be a hyperplane. The position of the hyperplane is shifted by the bias parameter w_0 and the angle of the hyperplane is affected by the direction of the vector $\mathbf{w}$. The hyperplane is perpendicular to $\mathbf{w}$. The magnitude of the vector $\mathbf{w}$ affects the certainty of the classification probabilities. The coefficients of the logistic regression algorithm are learned using maximum-likelihood estimation. The maximum-likelihood for logistic regression compares the probabilities predicted by the model to those in the data and tries to find values for the coefficients that minimize the error. The likelihood for logistic regression has a unique optimum, in other words it is convex.

We need a function that returns probabilities. Logistic regression is named after the logistic function. The logistic function $f(\mathbf{w},\mathbf{x}) = \sigma(\mathbf{w},\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$ provides this (Figure 2.7). As $\mathbf{w}^T\mathbf{x}$ goes from $-\infty$ to $+\infty$, $\sigma(\mathbf{w}^T\mathbf{x})$ goes from 0 to 1. Using the logistic trick, we can classify a new instance using the probabilities $P(y=1|\mathbf{x}) = f(\mathbf{w},\mathbf{x})$ and $P(y=0|\mathbf{x}) = 1-f(\mathbf{w},\mathbf{x})$. As a result, we can transform any regression model to a classification model.

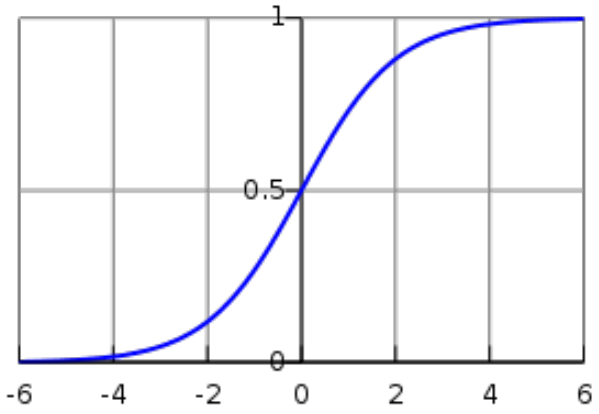


Figure 2.7: Logistic function.

2.6.2 Support Vector Machines

Support Vector Machines (SVMs) (Cortes and Vapnik, 1995) are one of the most effective algorithms and are applied to a wide variety of tasks. SVMs are a linear classifier, like logistic regression, but using the "kernel trick" SVMs can be a non-linear classifier. The two key ideas of the SVMs are:

1. Maximum margin classification

2. The “kernel trick”

Let consider the training instances $(\mathbf{x}_i, y_i), i = 1,...,n, y_i \in \{-1,+1\}$ and the hyperplane $\mathbf{w}^T\mathbf{x} + w_0 = 0$. The distance between the decision boundary and the closest training point is called margin. If the data is linearly separable, we can have a pair $(\mathbf{w}, w_0)$, using the classification rule $f(\mathbf{w},\mathbf{x}) = sgn(\mathbf{w}^T\mathbf{x} + w_0)$, such that:

$$\mathbf{w}^T\mathbf{x} + w_0 \geq +1 \qquad \text{for} \quad y_i = +1, \tag{2.1}$$

$$\mathbf{w}^T\mathbf{x} + w_0 \leq -1 \qquad \text{for} \quad y_i = -1 \tag{2.2}$$

If we combine the equations (2.1) and (2.2), we can have:

$$y_i(\mathbf{w}^T\mathbf{x} + w_0) \geq 1 \qquad \text{for} \quad i = 1,...,n \tag{2.3}$$

To maximize the distance between the decision boundary and the closest training point, we have to solve the optimization problem:

$$\max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|} \quad \text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x} + w_0) \geq 1 \quad \text{for all} \quad i \tag{2.4}$$

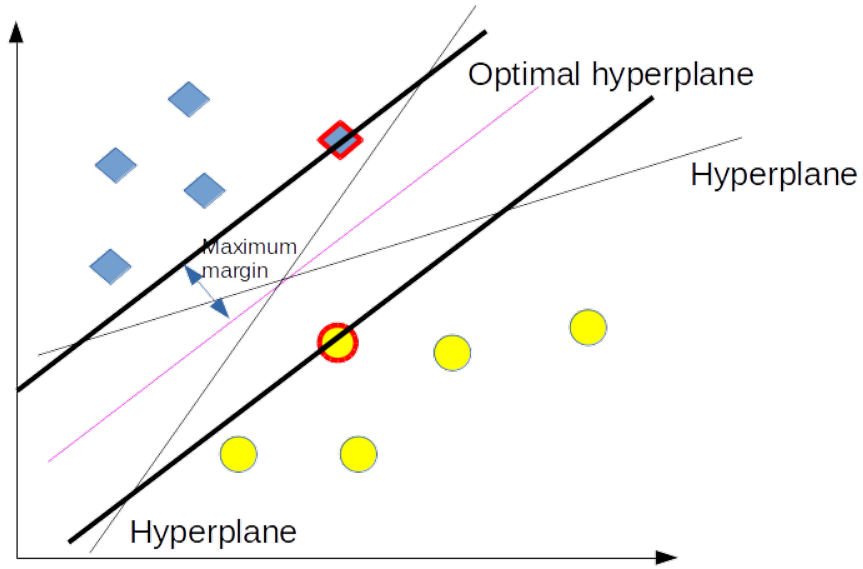


Figure 2.8: Maximum margin.

Also, maximizing $\frac{1}{\|\mathbf{w}\|}$ is the same as minimizing $\|\mathbf{w}\|^2$.

The solution is:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \tag{2.5}$$

The coefficient α_i's are non-negative only for the data points that lie on the optimal hyperplanes and these data points are called support vectors. As a result, we have a sparse solution. In figure 2.8, we can see the potential hyperplanes, the optimal hyperplane, the support vectors and the maximum margin. SVMs has a convex optimization problem and the solution is unique. To make predictions for a new instance, we use the function:

$$f(\mathbf{w},\mathbf{x}) = sgn(\mathbf{w}^T\mathbf{x} + w_0) = sgn\left(\sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + w_0\right) \tag{2.6}$$

A basis expansion can be used to make SVMs nonlinear. In an SVM, the basis expansion is easily implemented using the "kernel trick". The linear SVM depends only on $\mathbf{x}_i^T\mathbf{x}$. If we map the input to some other space, the algorithm will only depend on $\Phi^T(\mathbf{x}_i)\Phi(\mathbf{x})$ and we can use a kernel function such that $k(\mathbf{x}_i\mathbf{x}) = \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x})$. The goal is to transform a problem that it is not linearly separable in the initial space $\mathbf{x}$ to a linearly separable projecting the data to a higher-dimensional feature space. So, the equation (2.6) is modified to:

$$f(\mathbf{w},\mathbf{x}) = sgn\left(\sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + w_0\right) = sgn\left(\sum_i \alpha_i y_i k(\mathbf{x}_i^T \mathbf{x}) + w_0\right) \tag{2.7}$$

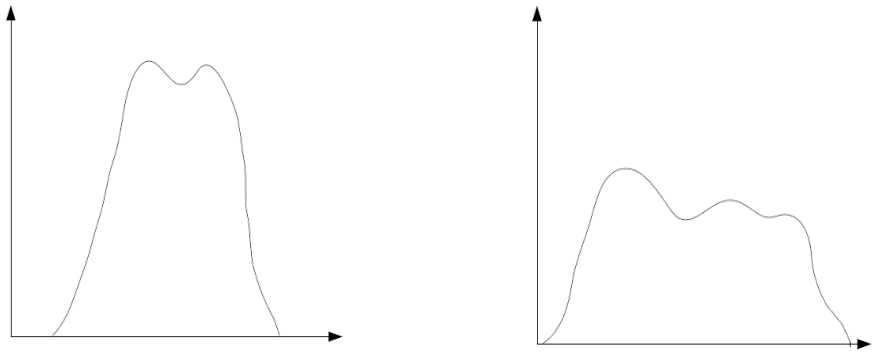
SVMs have been applied to a wide variety of problems. An example is the text classification, where Joachims (1998) has applied SVMs to classify news stories to categories.

2.7 Image pre-processing

Before the analysis, for example feature extraction and interest points, it is very common to pre-process the images. If we cleverly pre-process the images, we can get better local and global feature descriptions. We can think of the image pre-processing as feature normalization which is a widely used as a pre-processing step for feature extraction. Common image pre-processing methods are filtering and noise removal, thresholding, edge enhancement and illumination corrections.

2.7.1 Contrast Limited Adaptive Histogram Equalization

Adaptive Histogram Equalization (Pizer et al., 1987) (AHE) is a computer image processing method which enhances contrast in images. The difference between the AHE and ordinary histogram equalization is that AHE computes a separated histogram for a distinct section of the image. Histogram equalization doesn't work well when the histograms cover all the images and especially when the images have large intensity variation. So, usually the pixel transformation is based on the histogram of a square surrounding the pixel and for each image we can have several squares. In figure 2.9 we can see the effect on the pixels' value distribution when we apply histogram equalization. The areas of an image which have low contrast can get higher contrast.



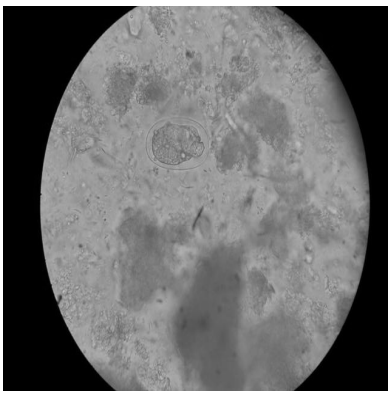
(a) Before equalization. (b) After equalization.

Figure 2.9: Effect of histogram equalization.

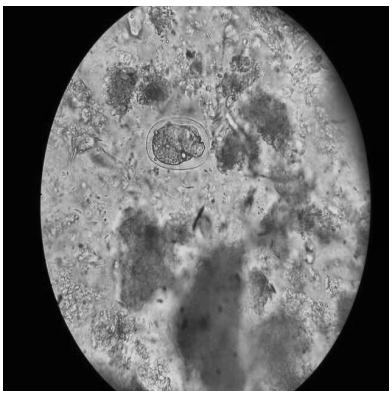
However, AHE has a drawback that amplifies noise if noise exists in a region of an image. Contrast Limited Adaptive Histogram Equalization (CLAHE) (Zuiderveld, 1994) prevents this by limiting this amplification. It clips the histogram bins that are above a specified contrast limit and these pixels are uniformly distributed to other bins. In figure 2.10 we can see an example of applying CLAHE on a image of our dataset.

2.7.1.1 Entropy of an image

We could compute the entropy of an intestinal microscopy image by computing at each pixel position (i,j) of the image the entropy of the pixels within a 2-D region centered at pixel (i,j). We convert this 2-D region to 1-D array and then we compute the entropy of the numbers in this array. The entropy is computed on grey-scale images. In the examples of figure 2.11 we have calculated and plotted the entropy of two example images that contain parasites. First, we convert the images to grey-scale and then we



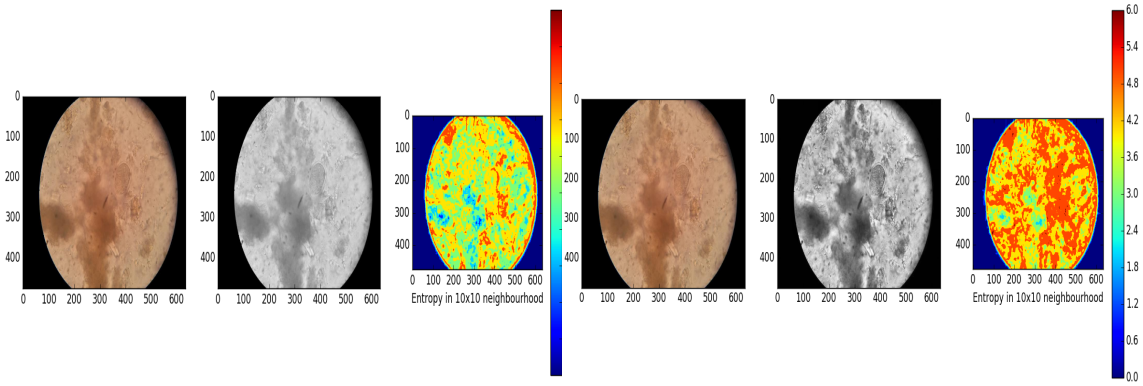
(a) Grayscale image.



(b) Grayscale image with CLAHE pre-processing.

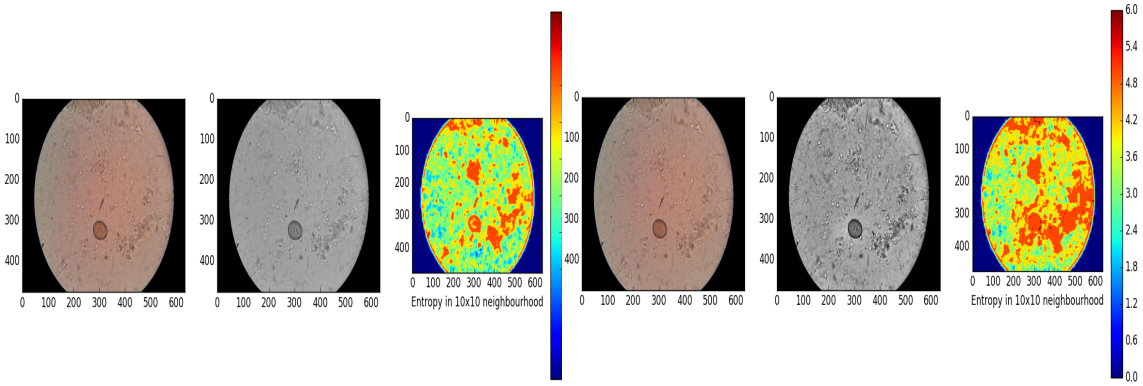
Figure 2.10: Effect of CLAHE pre-processing.

compute the entropy. The first column contains the example image and the second column contains the grey-scale version of the corresponding image. We have set the region size to be (10,10).



(a) Entropy of an image without CLAHE.

(b) Entropy of an image with CLAHE.



(c) Entropy of an image without CLAHE.

(d) Entropy of an image with CLAHE.

Figure 2.11: Effect of CLAHE pre-processing on the entropy of image.

We can see that homogeneous regions have low entropy. CLAHE image pre-processing has increased the entropy of the images and we will see in chapters 4 and 5 that CLAHE improves the overall performance of our models. This overall performance improvement can be a proof that CLAHE image pre-processing can be a beneficial pre-process step.

2.8 Evaluation of object detection

When we want to indicate how good is a trained model, we have to evaluate this model. Evaluation of a model's performance is one of the most important phases in a machine learning procedure. In the following subsections we present the evaluation measures and methods that we use in this project.

2.8.1 Main measures

The main measure to evaluate the performance is the area under the precision - recall curve (AUC), known as Average Precision (AP) and especially the precision at 90% recall which we name it in our experiments as PR@0.90R. AP is used to summarize the performance. Precision is the fraction of retrieved instances that are positive ($precision = \frac{\#correct\ boxes}{\#ground\ truth\ boxes}$), while recall is the fraction of positive instances that are retrieved ($recall = \frac{\#correct\ boxes}{\#all\ predicted\ boxes}$). Typically, precision and recall are inversely related and a balance between these two needs to be achieved by a system. Davis and Goadrich (2006) explain why we should prefer precision-recall AUC compared to receiver-operating-characteristic AUC for highly imbalanced datasets. The intuition is that since we are not really interested in how the model performs on the background class (which dominates our dataset). But, we want to make sure that every parasite detection is correct (precision), and that we get as many parasite detections as possible (recall). So, we should choose precision-recall AUC.

We use the intersection-over-union criterion (Everingham et al., 2010) to assess performance. A detection is correct if the intersection of the bounding boxes, divided by their union, is larger than 50%. The intersection-over-union is defined as $a_0 = \frac{B_p \cap B_{gt}}{B_p \cup B_{gt}}$ where B_p is the predicted bounding box and B_{gt} is the ground truth bounding box.

2.8.2 Stratified cross-validation

Cross-validation (CV) separates the dataset into n sub-samples (folds). Then, CV iteratively leaves one sub-sample out for the test set and trains on the rest. Finally, we should average the evaluation measures across all folds to estimate the performance of the algorithm on the task. The advantage of CV is that it efficiently uses the available data for testing, especially when data is limited as in our case.

Usually, when we randomly select training and validation sets, we want to make it sure that class proportions are retained in each set. Since we have a highly skewed dataset if we randomly split the dataset to training and validation sets, it is very likely that we will make the sets more imbalanced. Stratified cross-validation is able to retain the class proportions. First, it separates data points by class and then it randomly select data points from each class proportionally. Consequently, we don't make the problem of imbalanced classes more intense which is our goal when we create the stratified folds.

In this project when we demonstrate the results in tables, we compute the AUC averaging all the results from the five stratified cross-validation folds. We take all the detection probabilities of the test patches from the five folds and we give these probabilities to the function that computes the AUC of the precision - recall curve. That means we take into consideration the detection probabilities from all the patches from all the five folds. The AUC of CV is one result coming from all the detection probabilities of the five folds.

Chapter 3

Dataset

This chapter introduces the dataset we used in this project. In section 3.1 we present the number images and parasites of this dataset and we explain how we split it in training and test sets. Then, we explain how we selected the patches, the positive/negative ration and the data augmentation.

3.1 Details

For this project we use the dataset created by the Makerere Automated Lab Diagnostics Database. The images have been captured at the Mulago National Referral Hospital of Uganda. Each image can contain more than one parasite of a specific type and can also contain parasites of different types. Thus, in this project we have to detect any possible parasite and the class that these parasites belong to. For each image in our training set we know whether it contains any parasite or not, and if so, we know how many of them it contains and the type of these parasites. Moreover, if the images contain parasites the training set provides us with the information of the bounding boxes. So, for each parasite in an image we know its exact position since the bounding box provides us with the x_{max}, x_{min}, y_{max} and y_{min}.

Our dataset consists of 1,217 images. The images have two different resolutions. We have 1,067 images of resolution $4,128 \times 2,322$ and 150 images of resolution $3,264 \times 2,448$. 147 images contain parasites and the total number of bounding boxes is 162. That means we have images that have more than one parasite. Out of the 162 parasites we have 129 Hookworm, 25 Taenia and 8 Hymenolepsis Nana. That means we have three different kinds of parasites to detect. The annotation of the parasites includes a bounding box around each parasite. Table 3.1 summarizes all this informa-

tion.

Images with parasites	Bounding boxes	Hookworm	Taenia	Hymenolepsis Nana
147	162	129	25	8

Table 3.1: Number of images with parasites, bounding boxes and parasites.

70% of the images that contain parasites are used to train the models and 30% as a test set. Then, for each set we randomly take images that do not contain any parasite. The number of these negative images is two times[1] the number of the positive images. To tune the hyper-parameters of the models, we use stratified five-fold cross-validation on the training set. All the images are re-sized to have 476 pixels height so that we can have 128×128 pixels patches that can contain the bounding boxes of all the parasites. Figure 3.1 shows some examples of the resizing we have implemented. We computed the mean, max, min and the standard deviation value of the $(x_{max} - x_{min})$ and $(y_{max} - y_{min})$ of the bounding boxes. The results are shown in table 3.2.

	Height	Width
Mean	88	89
Max	123	128
Min	52	52
Standard devation	19	19

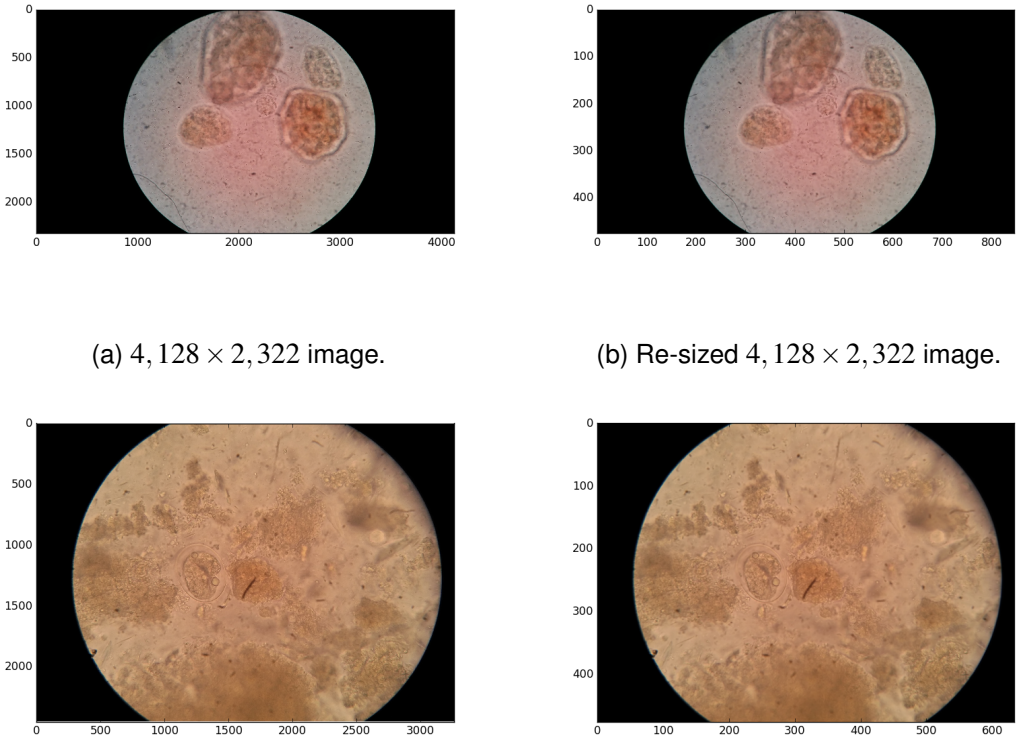
Table 3.2: Statistics on the bounding boxes size in pixels.

The number of Hookworm, Taenia and Hymenolepsis Nana parasites in the training and test sets is given in table 3.3.

Dataset	Images with parasites	Images without parasites	Hookworm	Taenia	Hymenolepsis Nana
Training	103	206	89	18	5
Test	44	88	40	7	3

Table 3.3: Number of images and parasites per dataset.

[1] We selected to use negative images two times the positive ones for computational complexity reasons. However, it would be more reasonable to also split the images that do not contain any parasite, 70% in the training set and 30% in the test set.



(a) $4,128 \times 2,322$ image.

(b) Re-sized $4,128 \times 2,322$ image.

(c) $3,264 \times 2,448$ image.

(d) Re-sized $3,264 \times 2,448$ image.

Figure 3.1: Initial images and re-sized images.

We should mention that there is not any image that simultaneously belongs to more than one set.

To cope with the problem of evaluating our models on a small validation set, we will apply stratified five-fold cross-validation on the training set to tune our parameters. We decided to use stratified cross-validation because for some parasites we have very few examples. For example, in our training set we only have five Hymenolepsis Nana parasites and there is the possibility that if we randomly create the folds to apply cross-validation, we may end up not having Hymenolepsis Nana parasites in the training or the validation set. In the stratified K-fold cross-validation the folds are made by preserving the percentage of samples for each class. Table 3.4 presents how the stratified five folds are created.

Table 3.5 presents the number of Hookworm, Taenia and Hymenolepsis Nana per cross-validation fold that training and validation set contain.

	Fold 1		Fold 2		Fold 3		Fold 4		Fold 5	
Set	Positive	Negative	Positive	Negative	Positive	Negative	Positive	Negative	Positive	Negative
Training	82	164	82	165	82	165	83	165	83	165
Validation	21	42	21	41	21	41	20	41	20	41

Table 3.4: Number of images that contain parasites or not per fold.

Table 3.5: Parasites per cross-validation fold.

(a) Fold 1

Set	Hookworm	Taenia	H. Nana
Training	72	14	4
Validation	17	4	1

(b) Fold 2

Set	Hookworm	Taenia	H. Nana
Training	71	14	4
Validation	18	4	1

(c) Fold 3

Set	Hookworm	Taenia	H. Nana
Training	71	14	4
Validation	18	4	1

(d) Fold 4

Set	Hookworm	Taenia	H. Nana
Training	70	15	4
Validation	19	3	1

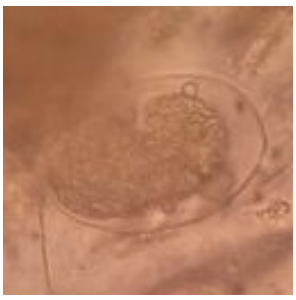
(e) Fold 5

Set	Hookworm	Taenia	H. Nana
Training	72	15	4
Validation	17	3	1

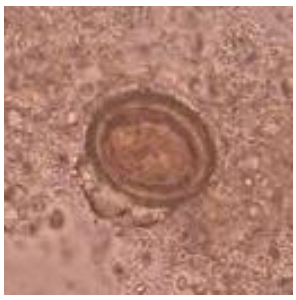
3.1.1 Patch selection

The size of the patches is chosen as 128×128 pixels. For each test image, patches are taken every 25 pixels both in the horizontal and vertical directions. To generate the training patches we use the approach from Sanchez (2015). Each positive patch is created centered on each parasite. The negatives patches are created using the sliding window technique as before, but rejecting 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. Figure 3.2 presents some examples.

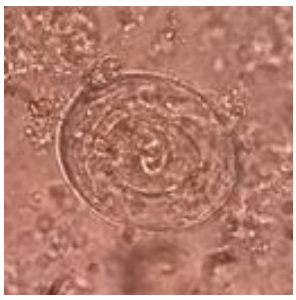
In these examples we can also see that we can have background noise in the parasite training patches. In the Hookworm patch background noise covers the Hookworm parasite. There is background noise around the Taenia parasite and we can see how difficult is to recognise the Hymenolepsis Nana parasite. This noise makes our parasite detection task difficult, since we have to extract features that are representative for the parasites but background noise might well decrease the quality of the feature extraction.



(a) Hookworm parasite.



(b) Taenia parasite.



(c) Hymenolepsis Nana parasite.

Figure 3.2: Centered positive patches.

3.1.2 Positive/Negative ratio and data augmentation

To create the training sets for the cross-validation we augment the positive patches with mirroring and 90°, 180° and 270° clockwise rotation. As a result, for every fold in the stratified cross-validation we have augmented training patches. For every fold, we use all the positive patches and we randomly select negative patches which are four times as many as the positive patches. Table 3.6 summarizes all this information.

Table 3.6: Training patches.

(a) Fold 1.

Set	Background	Hookworm	Taenia	H. Nana
Training	2880	576	112	32

(b) Fold 2.

Set	Background	Hookworm	Taenia	H. Nana
Training	2848	568	112	32

(c) Fold 3.

Set	Background	Hookworm	Taenia	H. Nana
Training	2848	568	112	32

(d) Fold 4.

Set	Background	Hookworm	Taenia	H. Nana
Training	2848	560	120	32

(e) Fold 5.

Set	Background	Hookworm	Taenia	H. Nana
Training	2912	576	120	32

After having tuned the parameters and selected the methods which improve the performance on the folds' validation sets, we train our best model using all the training set. We apply the same positive/negative ratio and data augmentation on the whole training set as we did for the cross-validation folds. Table 3.7 shows the number of training patches.

From the table we can see that we exploit all the training instances when we train our best model.

Set	Background	Hookworm	Taenia	H. Nana
Training	3584	712	144	40

Table 3.7: Training patches.

Chapter 4

Parasite Detection

In this chapter we detail how to detect parasites in the intestinal microscopy images. Section 4.1 introduces the model with which we made the experiments and how we name the models. In section 4.2 we introduce the Negative Enrichment technique that we use so that we are able to use informative background patches for the negative class. In section 4.3 we have separated the training procedure into two stages. In the first stage we have the Baseline models for which we have randomly selected background patches and in the second stage we have the models which have been trained using the Negative Enrichment technique. For both the steps we evaluate our models. In section 4.4 we comment on the effect of the Negative Enrichment technique on the overall performance. Finally, implementations details of the programming languages and third party libraries used are provided in section 4.5.

4.1 Models

This section covers the models we have used and what parameter values we have selected to train them. We describe the logistic regression and SVMs models. Figure 4.1 demonstrates an example of a detection model. In the feature extraction stage, the window of 128x128 resolution is moved over an image, and for each location a feature descriptor of 15876 dimensions is extracted.

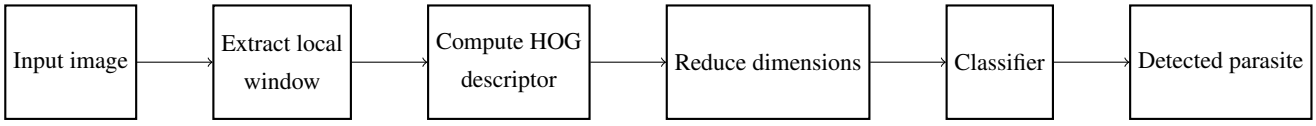


Figure 4.1: Sliding window detection process.

4.1.1 Logistic regression

To train the logistic regression models, we have used L2 regularization, one-vs-rest (OvR)[1] scheme and the liblinear[2] solver.

We name the logistic regression models as (LR). If the logistic regression model uses dimensionally reduced dataset, we name the model (LR-PCA:X) where X is the number of the principal components we have kept when we reduced the dimensions.

Because of the highly imbalanced dataset we have also made experiments using weights associated with the classes. We ran experiments using the "balanced" parameter, which automatically adjusts the weights inversely proportional to class frequencies. We name the logistic regression model that uses the "balanced" parameter as (LR-B). If the logistic regression model used both PCA and the "balanced" parameter, we name the model as (LR-PCA:X-B).

Moreover, if the model also uses CLAHE pre-processing, we name it (LR-C), (LR-B-C) and (LR-PCA:X-B-C) respectively.

The logistic regression models use the inverse regularisation parameter C. $\lambda = 1/C$ is the regularisation value and smaller values of C specify stronger regularization. To use regularisation during the training procedure we have trained models like (LR-IR:X) or (LR-PCA:X-IR:Y-C) depending on the combination of the techniques that have been used.

4.1.2 Support Vector Machines

To train the SVMs models, we have used the one-vs-rest (OvR)[3] scheme where in our case a binary problem is fitted for the parasite class and linear kernel. Kernels of higher order gave very poor performance.

We name the SVMs models as (SVM). If the SVMs model uses dimensionally reduced dataset, we name the model (SVM-PCA:X) where X is the number of the principal components we have kept when we reduced the dimensions.

We name the SVMs model that uses the "balanced" parameter as (SVM-B). If the SVMs model used both PCA and the "balanced" parameter, we name the model as (SVM-PCA:X-B).

[1] In our case `http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html` fits a binary problem for the parasite class

[2] `https://www.csie.ntu.edu.tw/~cjlin/liblinear/`

[3] In our case `http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html` fits a binary problem for the parasite class

Moreover, if the model also uses CLAHE pre-processing, we name it (SVM-C), (SVM-B-C) and (SVM-PCA:X-B-C) respectively.

The SVMs models that use the inverse regularisation parameter C. $\lambda = 1/C$ is the regularisation value and smaller values of C specify stronger regularization. To use regularisation during the training procedure we have trained models like (SVM-IR:X) or (SVM-PCA:X-IR:Y-C) depending on the combination of the techniques that have been used.

4.2 Negative Enrichment

We will notice in section 4.3.1.3 that if we randomly select the background patches, we get quite different performance between the trials. We will see in more details about problem in the next section. To cope with this problem, for each fold we randomly selected the background patches, we trained our model (our Baseline) and then we applied a technique that we named Negative Enrichment. We classified all the background training patches and we kept the incorrectly classified patches and the correctly classified with probability lower than 90%. If these patches were less than four times the positive ones, we randomly selected different background patches so that we could have four times the positive ones in total.

4.3 Evaluation

In this section we evaluate the methods we decided to use and we compare the results given by the various models we trained. In section 4.3.1 we evaluate our models for which we randomly selected the background patches and we analyze the results taken from these models. Then, in section 4.3.2 we evaluate our models which have been trained using the background patches that have been selected by the negative enrichment technique. In section 4.3.3 we evaluate our best model on the test set.

4.3.1 Baseline

We call this stage of the training procedure Baseline. In this stage we cannot know which background patches are the best to be used for training. We have created five stratified folds. For each stratified fold we randomly selected the background patches and it is quite likely that between these folds the patches in the negatives class can be

very different. We test the performance only for a logistic regression model because the dimensional space of the HOG descriptor is very large to train an SVM model. In section 4.3.1.5 where we apply PCA we also train SVM models.

4.3.1.1 Threshold to accept detections

In this subsection we tested the performance of the threshold to accept detections. When our model returns a detection, this detection should overlap with the bounding box of a parasite so that the detection can be consider correct. For NMS we use 0.4 overlap to merge. Table 4.1 demonstrates the results.

	CV	
Model	AUC	PR@0.9R
LR-THR:0.1	0.770	0.577
LR-THR:0.3	0.770	0.577
LR-THR:0.5	**0.770**	**0.577**
LR-THR:0.7	0.770	0.577
LR-THR:0.9	0.729	0.352

Table 4.1: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

From the table 4.1 we can see that we got the same results for the threshold 0.1, 0.3, 0.5 and 0.7. These thresholds gave the best results. However, we selected the 0.5. For all the next experiments in this project we use the threshold 0.5 to accept detections.

4.3.1.2 Overlap to merge detections

In this subsection we tested the performance of the overlap to merge detections. When a lot of detections correspond to the same parasite, these detections should be merged so that we can only have one detection for every individual parasite. Table 4.2 shows the results.

From the table 4.2 we can see that we got the best results for the overlap 0.4. For all the next experiments in this project we use the overlap 0.4 to merge the detections. So, for now our model is LR with performance 0.770 AUC for the precision recall curve and precision 0.577 at recall 0.90.

	CV	
Model	AUC	PR@0.9R
LR-OVR:0.2	0,745	0.437
LR-OVR:0.3	0,756	0.495
LR-OVR:0.4	**0.770**	**0.577**
LR-OVR:0.5	0,770	0.575

Table 4.2: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

4.3.1.3 Effect of the randomly selected background patches

Through our experiments we observed that if we keep the positives training instances steady and every time we train with different randomly selected background patches, we get quite different performance. Table 4.3 illustrates this effect for 5 trials with different randomly selected background patches in every trial. We ran the experiment for the fold 1 and fold 4 of the stratified cross-validation folds training a LR model for each trial.

	Trial 1		Trial 2		Trial 3		Trial 4		Trial 5	
Model	AUC	PR@0.9R	AUC	PR@0.9R	AUC	PR@0.9R	AUC	PR@0.9R	AUC	PR@0.9R
Fold 1	0.616	0.513	0.851	0.416	0.841	0.556	0.714	0.444	0.792	0.435
Fold 4	0.768	0.404	0.805	0.368	0.753	0.328	0.683	0.504	0.673	0.447

Table 4.3: AUC corresponds to the area under the curve of the precision - recall curve.

From the above results we observe that we get quite different performance between the five trials for both fold 1 and fold 4. So, we cannot have reliable results if we randomly select background patches. We try to cope with this problem in section 4.3.2 where we apply the Negative Enrichment technique.

4.3.1.4 CLAHE

We tested if an image pre-processing method can enhance the performance of a logistic regression model. In this section we evaluate the effect of applying CLAHE on the images. Table 4.4 shows the results of this experiment.

From table 4.4 we can see that CLAHE image pre-processing improved the over-

	CV	
Model	AUC	PR@0.9R
LR	0.770	0.577
LR-C	0.808	0.678

Table 4.4: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

all performance. Given these results, we accept the initial intuition that image pre-processing can improve the performance.

4.3.1.5 PCA

We saw that the main disadvantage of HOG descriptor is that it returns large feature vector and as a results to get the HOG descriptor is computationally inefficient. To ease this work load, we applied PCA dimensionality reduction. In this section we evaluate this dimensionality reduction on the HOG descriptors. We selected to keep 1200 principal components for all the five folds because this number of components is close to retain 95% of the data variation. We also test the performance on smaller number of principal components and more precisely for 50 principal components. Table 4.5 presents the performance on the stratified cross-validation.

	CV	
Model	AUC	PR@0.9R
LR	0.770	0.577
LR-PCA:1200	0.764	0.587
LR-PCA:50	0.628	0.301
SVM-PCA:1200	0.552	0.421
SVM-PCA:50	0.549	0.423

Table 4.5: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

Table 4.5 shows that using the 1200 first principal components that correspond to approximately the 95% of the data variation we got about the same performance with the model using all the HOG descriptor. However, if we reduce the dimensions of the dataset to 50 dimensions we get worse performance. That means reducing the

dimension to 50 we lost a lot of information about the dataset and for the baseline we cannot get a good model on these 50 dimensions.

4.3.1.6 Weight balancing

As we have seen our dataset is highly imbalanced. For this reason we made experiments where we used the "balanced" mode which automatically adjusts the weights inversely proportional to class frequencies in the dataset. With this option we hope that we can get better models. In table 4.6 we can see the performance of the model trained using the "balanced" mode.

	CV	
Model	AUC	PR@0.9R
LR	0.770	0.577
LR-B	0.445	0.337

Table 4.6: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

From the results we can see that this option gave worse performance as far as the baseline is concerned. However, we will see in section 4.3.2.3 that weight balancing along with Negative Enrichment can improve performance.

4.3.1.7 Regularisation

Regularisation is a technique to cope with the over-fitting problem in statistical models, by controlling the model complexity. We used the $l2$ regularisation on the weights of the model. It penalizes the model for higher weights. It seems necessary to use regularisation since we have few training examples and large feature vector. In these situations the large features vector can have good performance on training sets but not on test sets. We use inverse of regularization strength C (regularisation $\lambda = 1/C$), smaller values specify stronger regularization. The results are included in table 4.7.

From the results we observe that regularisation does not improve the performance as far as the Baseline is concerned.

	CV	
Model	AUC	PR@0.9R
LR	0.770	0.577
LR-IR:0.5	0.757	0.459
LR-IR:0.1	0.706	0.146

Table 4.7: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

4.3.1.8 Combinations

In this section we evaluate models that have been trained using combinations of the methods introduced in the Chapter 2. The goal is that if we combine the advantages of these methods, we can get better models. Table 4.8 presents the performance of these models.

	CV	
Model	AUC	PR@0.9R
LR-PCA:1200-C	**0.831**	**0.673**
LR-PCA:1200-C-B	0.377	0.325
LR-PCA:1200-IR:0.5-C	0,830	0.652
SVM-PCA:1200-C	0.515	0.395

Table 4.8: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

Table 4.8 indicates that using CLAHE image pre-processing models with PCA dimensionality reduction can give better performance than the simple logistic regression models. We saw in section 4.3.1.5 that PCA didn't improve the overall performance, but now using CLAHE it did. That means CLAHE can help the PCA technique to find better features and keeping the first 1200 principal components we have removed unnecessary information.

4.3.1.9 Evaluation of the best model

Since we tuned our models using the validation sets of the stratified folds, we evaluated the final performance of our model on the test set. Our best Baseline model was the LR-PCA:1200-C validated on the CV folds. We used 1200 principal components for

all the CV folds because these components correspond approximately to 95% variation of every fold. However, using all the training set the principal components that exactly correspond to 95% variation are 1473. So, we trained the model LR-PCA:1473-C on all the training set. We used all the parasite patches as the positive instances and we randomly selected background patches as the negative instances. Then, we applied PCA and we kept the 1473 first principal components. We applied CLAHE image pre-processing as well. The performance of the best model on the test set is included in table 4.9.

	Test	
Model	AUC	PR@0.9R
LR-PCA:1473-C	0.708	0.577

Table 4.9: Performance on the test set. AUC corresponds to the area under the curve of the precision - recall curve.

Figure 4.2 demonstrates the precision-recall curve for the CV and test set. As it was expected the area under the curve and the precision at 0.90 recall are bigger for the CV than for the test set, because we have tuned the parameters on the CV set.

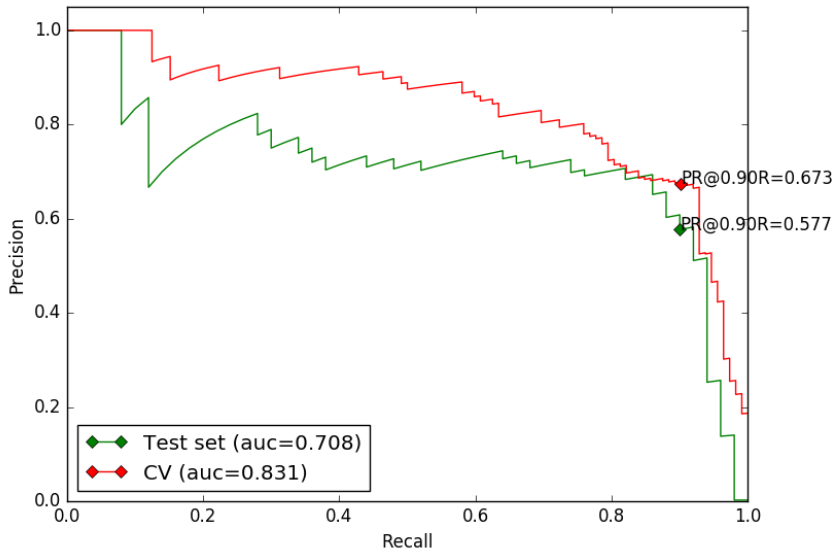


Figure 4.2: Performance comparison of the best model on CV and test set.

4.3.1.10 Conclusion of the evaluation

In this section we tested the performance of the models trained on randomly selected background patches. We found what were the best thresholds to accept and merge detections. We observed that if we train our model on different randomly selected background patches we get quite different performances. We saw that CLAHE image pre-processing improves both the AUC of the precision-recall curve and the precision at 0.90 recall. Moreover, if we apply PCA to reduce the dimensional space of our task, we noticed that if we keep the 1200 first principal components that correspond to approximately the 90% variance of our CV folds, we get about the same performance without dimensionality reduction. But, if we reduce the dimension keeping the 50 first principal components, we get worse performance. That's means we lost information reducing the dimensions to 50. Finally, weight balancing and regularisation can't give better results as far as the Baseline is concerned. The best Baseline had a combination of CLAHE image pre-processing and PCA using 1473 principal components. This indicates that although PCA didn't improve the performance, the application of CLAHE image pre-processing had an positive impact on the HOG feature creation which made PCA produce better dimensionality reduced data.

4.3.2 Negative Enrichment

In this stage of the training procedure we have classified all the background training patches and we have kept the incorrectly classified and the correctly classified with probability lower than 90%. If these patches are less than four times the positive ones, we randomly select different background patches so that we can have four times the positive ones in total. We did this procedure for every CV fold. The goal of the Negative Enrichment is that we can have background patches that are more informative, because we saw in section 4.3.1.3 that for different randomly selected background patches we got quite different values of performance. For Baseline we used the model LR-PCA:1200-C on the CV folds as we saw that it gave the best performance.

Some examples of the background patches that were selected through the Negative Enrichment step are included in figure 4.3. In this figure we can observe that these patches can easily be considered as parasites if a model hasn't been trained using these patches as negative training instances. As a result using Negative Enrichment we trained our model with these noisy patches and some other randomly selected patches as negative instances.

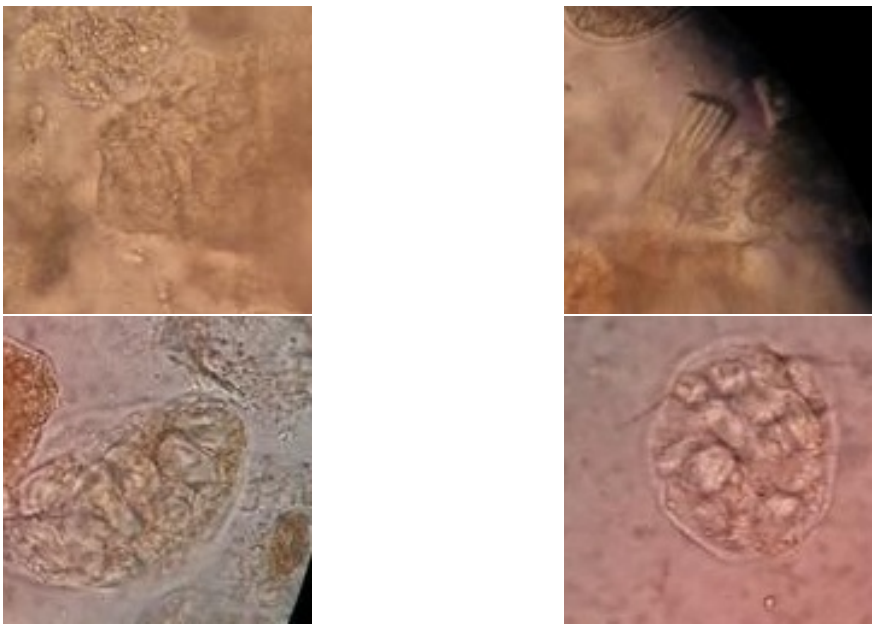


Figure 4.3: Background patches selected by Negative Enrichment.

For the same reason as for the Baseline we only train a logistic regression on the full HOG descriptor. The performance of the logistic regression model after applying Negative Enrichment is included in Table 4.10.

	CV	
Model	AUC	PR@0.9R
LR	0.862	0.398

Table 4.10: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

Comparing this performance with the best Baseline model we can see that now we have better precision-recall AUC but worse precision at 0.90 recall. However, this is just a simple logistic regression model, we will see later in this section that using the methods we described in chapter 2 we got better improvement than the best Baseline model.

4.3.2.1 CLAHE

Since we noticed in section 4.3.1.4 that CLAHE can provide us with enhanced performance, we also applied CLAHE image pre-processing after the Negative Enrichment step. Table 4.11 demonstrates the results.

	CV	
Model	AUC	PR@0.9R
LR	0.862	0.398
LR-C	0.885	0.532

Table 4.11: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

The results from table 4.11 match the previous results that showed CLAHE pre-processing can improve the performance of our models. That means image pre-processing can be beneficial to our task.

4.3.2.2 PCA

We also applied PCA after the Negative Enrichment step to reduce the dimensions. The results are included in Table 4.12.

	CV	
Model	AUC	PR@0.9R
LR	0.862	0.398
LR-PCA:1200	0.887	0.558
SVM-PCA:1200	0.855	0.385
LR-PCA:50	0.881	0.605
SVM-PCA:50	0.879	0.660

Table 4.12: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

Contrary to the Baseline we notice that in the case of the Negative Enrichment the dimensionality reduction can improve the performance of our model without applying CLAHE. This observation indicates that Negative Enrichment creates a set of background patches that can provide PCA with better features. More specifically, we got better performance for the 50 dimensions than the 1200 dimensions which contradicts with the Baseline where for 50 dimension we got much worse results.

4.3.2.3 Weight balancing

We also experimented the use of the "balanced" mode after applying Negative Enrichment. In table 4.13 we can see the performance of the model trained using the "balanced" mode.

	CV	
Model	AUC	PR@0.9R
LR	0.862	0.398
LR-B	0.872	0.612

Table 4.13: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

Contrary to the Baseline we notice that in the case of the Negative Enrichment the "balanced" mode can improve the performance of our model. In section 4.3.1 we saw that "balanced" mode gave bad results. This observation can be a proof that Negative Enrichment helped us to have informative background patches to train our model.

4.3.2.4 Regularisation

Similarly to Baseline we ran experiments using regularisation after applying Negative Enrichment. Again, we use inverse of regularization strength C (regularisation $\lambda = 1/C$) and smaller values specify stronger regularization. Table 4.14 illustrates the results.

	CV	
Model	AUC	PR@0.9R
LR	0.862	0.398
LR-IR:0,5	0.861	0.400
LR-IR:0,1	0.817	0.398

Table 4.14: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

From table 4.14 we observe that if set the inverse of regularisation to 0.5 we get approximately the same performance with the logistic regression model without regularisation. However we will see in section 4.3.2.5 that if we combine regularisation

with CLAHE and weight balancing we can improve the performance. If we use more regularisation specifically 0.1 we get worse results.

4.3.2.5 Combinations

In this section we present the results of the models built using combinations of the above methods after applying the Negative Enrichment. The results are included in table 4.15.

	CV	
Model	AUC	PR@0.9R
LR-PCA:1200-C	0.906	0.669
LR-PCA:1200-C-B	**0.884**	**0.849**
SVM-PCA:1200-C	0.857	0.564
SVM-PCA:1200-C-B	0.851	0.765
LR-PCA:1200-IR:0.5-C-B	0.888	0.842
LR-PCA:50-C	0.886	0.580
SVM-PCA:50-C	0.893	0.808
SVM-PCA:50-IR:0.5-C	0.884	0.664
LR-PCA:50-C-B	0.827	0.737
SVM-PCA:50-C-B	0.835	0.697

Table 4.15: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

From table 4.15 we notice that the model LR-PCA:1200-C-B gives the best overall performance on the validation sets. That means after having applied Negative Enrichment the combination of dimensionality reduction, weight balancing and CLAHE image pre-processing can enhance performance. We can also see that there are models trained on 50 dimensions like the LR-PCA:50-C and SVM-PCA:50-C that have higher AUC than the model LR-PCA:1200-C-B, but the model LR-PCA:1200-C-B has higher precision at 0.90 recall. One of our goal is to have high precision at 0.90 recall, that's why we selected the model LR-PCA:1200-C-B. Another observation is that the models which have been trained with the "balanced" mode have higher precision at 0.90 recall than the models that have been trained without the "balanced" mode. So, the weight balancing helps our models to have higher precision at 0.90 recall but slightly decreases the AUC.

4.3.2.6 Evaluation of the best model

Since we tuned our models using the validation sets of the stratified folds, we evaluated the final performance of our model on the test set. We used 1200 principal components for all the CV folds because these components correspond approximately to 95% variation of every fold. However, using all the training set the principal components that exactly correspond to 95% variation are 1413. So, we trained the model LR-PCA:1413-C-B on all the training set. We used all the parasite patches as the positive instances and we used the Negative Enrichment method in the same way as we did for the stratified cross-validation folds. We classified all the background training patches and we kept the incorrectly classified and the correctly classified with probability lower than 90%. If these patches were less than four times the positive ones, we randomly selected different background patches so that we could have four times the positive ones in total. Then, we applied PCA and we kept the 1413 first principal components. We applied CLAHE image pre-processing and weight balancing as well.

The value of the logistic regression coefficients represent the importance of some terms in the output predictions. We noticed in the section 2.5.1 how the principal components of the parasite patches have a round shape since we have rotated our parasites 90°, 180° and 270° clockwise and the principal components of the background patches don't have any specific shape. Before we apply PCA we can plot the logistic regression coefficients. Figure 4.4 illustrates the coefficients of the logistic regression model, plotted using the HOG descriptor function, which has been trained on our parasite and Negative Enrichment background patches. The most significant coefficients form a round shape, indicating the edge orientations that should be present in parasite patches, and this observation comes to agreement with the observation in the section 2.5.1.

Figure 4.5 demonstrates the mean grayscale training image and the average gradient training image of our dataset. We took the gradient images of our training dataset using the laplacian operator. From this figure we can see that the places on the image, which the logistic regression coefficients of figure 4.4 highlights, match with the round shapes that we can see in the figures 4.5.(a) and 4.5.(b).

The performance of the best model LR-PCA:1413-C-B on the test set is included in table 4.16.

Figure 4.6 demonstrates the precision-recall curve for the CV and test set. As it was expected the area under the curve and the precision at 0.90 recall are bigger for

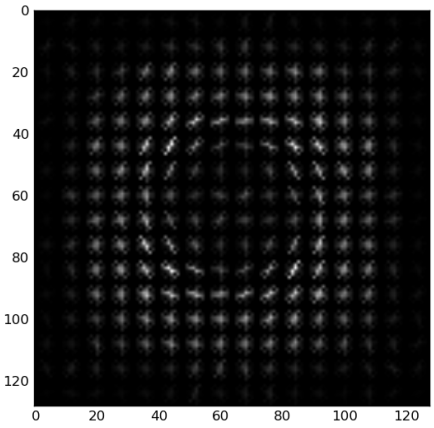
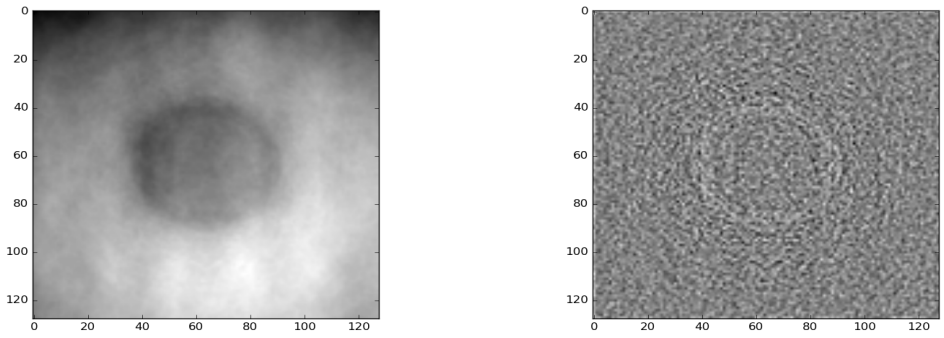


Figure 4.4: Logistic regression coefficients.



(a) Mean grayscale training image.

(b) Average gradient training image.

Figure 4.5: Mean grayscale and average gradient training images.

	Test	
Model	AUC	PR@0,9R
LR-PCA:1413-C-B	0.818	0.662

Table 4.16: Performance on the test set. AUC corresponds to the area under the curve of the precision - recall curve.

the CV set than for the test set. But, compared to the Baseline the curve of the test set is more close to the curve of the CV set. This result is good because it indicates that the model after the Negative Enrichment generalises better than the Baseline model.

From figure 4.6 we can observe that when recall is close to one, precision is zero. That means there are some false negative parasite patches that have very low prediction probability. We checked what test parasite patches were classified as false negatives

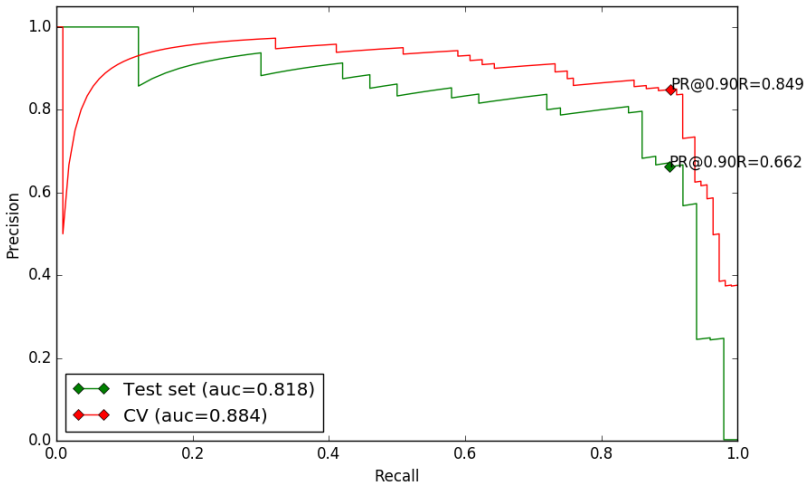


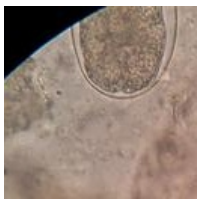
Figure 4.6: Performance comparison of the best model on CV and test set.

and had extremely low prediction probability. Out of the 50 parasites of the test LR-PCA:1413-C-B correctly classified 42 and 8 were false negative. Out of the 8 false negative parasite patches 5 were Hookworm parasites and 3 were Taenia parasites. Figure 4.7 illustrates the false negative patches with the prediction probabilities.

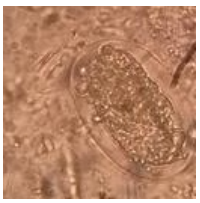
From figure 4.7 we can see that the parasite with the lowest probability 0.012 (a) is not even a whole parasite. It's a part of a Hookworm parasite and is located at the corner of the microscopy image. Our model cannot predict it as a parasite. This parasite patch is the reason that the precision is zero when recall is close to one. The other parasites with low prediction probabilities are the (b) and (c). Parasite patch (b) looks similar to background and the Hookworm parasite is not clear. Parasite (c) is located at the corner of the microscopy image and maybe this is the reason it has low probability prediction. When we extract the HOG descriptor this patch has different HOG features than the other parasites because of the corner of the microscopy image. Taenia parasites patches (d), (e) and (f) have low background noise but our model couldn't predict them as parasites.

Moreover, LR-PCA:1413-C-B incorrectly classified as parasites ten background patches. Figure 4.8 demonstrates the four background patches with the highest prediction probabilities.

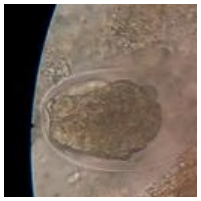
From figure 4.8 we can see that the patches with the highest prediction probabilities are (b) and (c). Patch (b) has 0.987 and patch (c) has 0.947 prediction probabilities. That means our model was incorrectly certain that these two patches contained para-



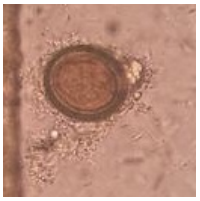
(a) Prediction probability 0.012.



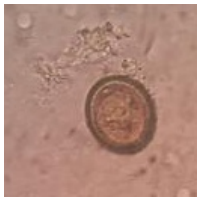
(b) Prediction probability 0.193.



(c) Prediction probability 0.189.

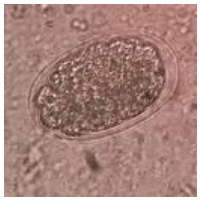


(d) Prediction probability 0.400.

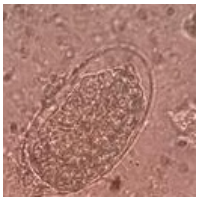


(e) Prediction probability 0.350.

(f) Prediction probability 0.401.

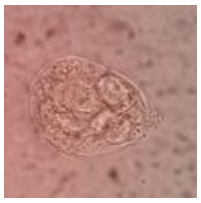


(g) Prediction probability 0.489.



(h) Prediction probability 0.429.

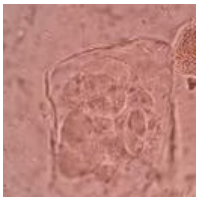
Figure 4.7: False negative patches.



(a) Prediction probability 0.828.

(b) Prediction probability 0.987.

(c) Prediction probability 0.947.



(d) Prediction probability 0.884.

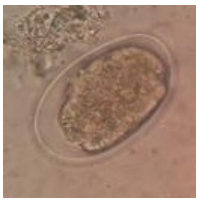
Figure 4.8: False positive patches.

sites. These two patches have background noise which has round shape. Also, patch (a) has background noise that looks like a parasite with round shape. We saw in section 2.5.1 where we applied PCA and we plotted the principal components of our dataset that the principal components of the parasites have round shape. Moreover, in section 2.6.1 where we plotted the logistic regression coefficients they formed a circle. It seems that since these patches have background noise which has round shape, our model predicts these patches as parasites. Patch (d) contains background noise which could also be regarded as a parasite.

We can also plot the test patches with the highest and lowest prediction probabilities. Figure 4.9 presents these patches. All of the patches with the highest prediction probabilities contain Hookworm parasites. This is reasonable because 80% of the parasites are Hookworm and most of the features that our model has taken as positive features come from the Hookworm parasites. Moreover, we can observe that these Hookworm parasites are not surrounded by background noise and this helps our model to give to these test patches very high probabilities. This is an indication that this system could work even better if we didn't have so much background noise. All of the patches with the lowest probabilities contain the corner of the microscopy image and only noise in the part of the microscopy images. All these observations are a proof that our model works as we expected when we started this project.

(a) Prediction probability 0.998.

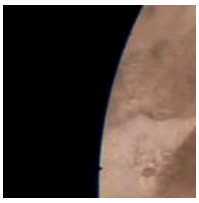
(b) Prediction probability 0.996.



(c) Prediction probability 0.997.

(d) Prediction probability 9.718×10^{-7}.

(e) Prediction probability 1.735×10^{-6}.



(f) Prediction probability 1.183×10^{-6}.

Figure 4.9: Test patches with the highest and lowest prediction probabilities.

Figure 4.10 shows the histogram of the prediction probabilities of the test patches. The histogram illustrates how the probabilities of both the parasite and background

test patches are distributed between zero and one. We can observe that almost 100% of the background patches are given probability between zero and 0.1 by our model LR-PCA:1413-C-B. The false positives are so few compared to the total number of background patches that we cannot even see the percentage of these false positive patches on the plot. Moreover, more than 40% of the parasite patches are given probability between 0.9 and one. The parasites patches that are given prediction probabilities between zero and 0.5 are the false negative patches that we saw in figure 4.7.

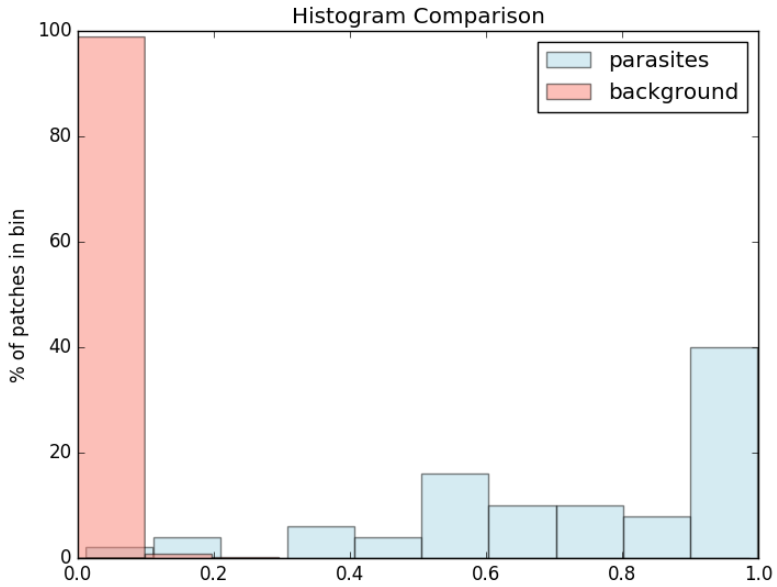


Figure 4.10: Histogram of both parasite and background patches.

Another way to test the performance of our model is to check what are the most similar training patches when we give test patches to our model. First, we extract the HOG descriptor of the test patch and we project this descriptor to the 1413 dimensions created by the PCA. Then, we find the training patches that have features that are similar to the test patch's features. Figure 4.11 illustrates in the first column the test patches and from the second to sixth column the corresponding most similar training patches. We randomly selected the test patches. As similarity measure we use the cosine similarity. We selected the cosine similarity because it returns an accurate measure of similarity regardless of the features' magnitude. From figure 4.11 we can observe that the most similar training patches match with the test patches. In the first row we have a Hookworm test patch and the most similar training patches are Hookworm patches except one which is background noise but this noise can return HOG

features that are similar with the HOG features of a Hookworm patch. In the second row we have a Taenia test patch and the most similar training patches consist of three Taenia, one Hookworm and one Hymenolepsis Nana patches. In the third row we have a Hymenolepsis Nana test patch and the most similar training patches consist of four Hookworm and one Hymenolepsis Nana patches. It's reasonable to have similar training patches that contain different parasites than the test patch because these parasite eggs have approximately the same shape and most of the parasites in our dataset are Hookworm parasites and this affects the result of the PCA dimensionality reduction. Another observation is that most of the parasites in the training patches have the approximately same orientation with their corresponding parasites in the test patch. This observation is a consequence of the HOG descriptor. A HOG descriptor is not a rotation invariant representation. In the last row we have a background test patch and all the most similar training patches are background patches.

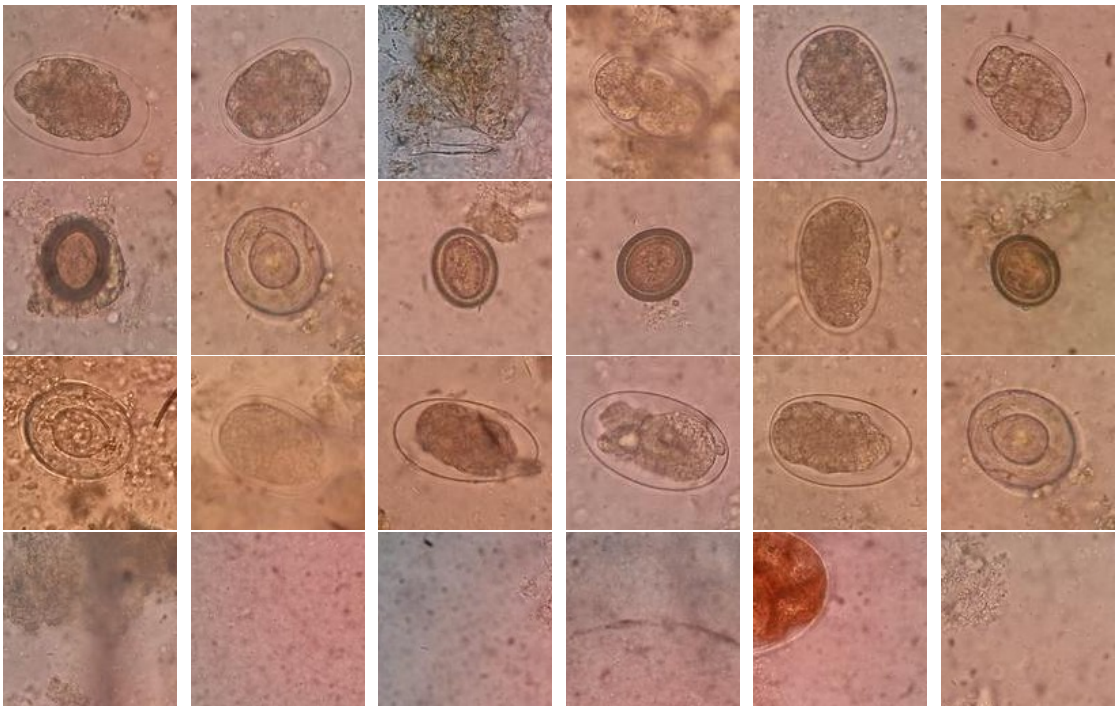


Figure 4.11: Test patches and the five most similar training patches based on HOG descriptors.

4.3.2.7 Conclusion of the evaluation

In this section we tested the performance of the models trained on background patches selected by the Negative Enrichment technique and some other randomly selected.

Similarly to the Baseline, we saw that CLAHE image pre-processing improves both the AUC of the precision-recall curve and the precision at 0.90 recall. Moreover, if we apply PCA to reduce the dimensional space of our task, we noticed that we get better overall performance for both the 1200 and 50 dimensions. The most interesting observation is that we get better performance for the 50 dimension than for the 1200 dimensions in comparison with the Baseline where reducing to 50 dimensions we got much worse results. The best model is LR-PCA:1413-C-B which has a combination of dimensionality reduction, CLAHE image pre-processing and weight balancing.

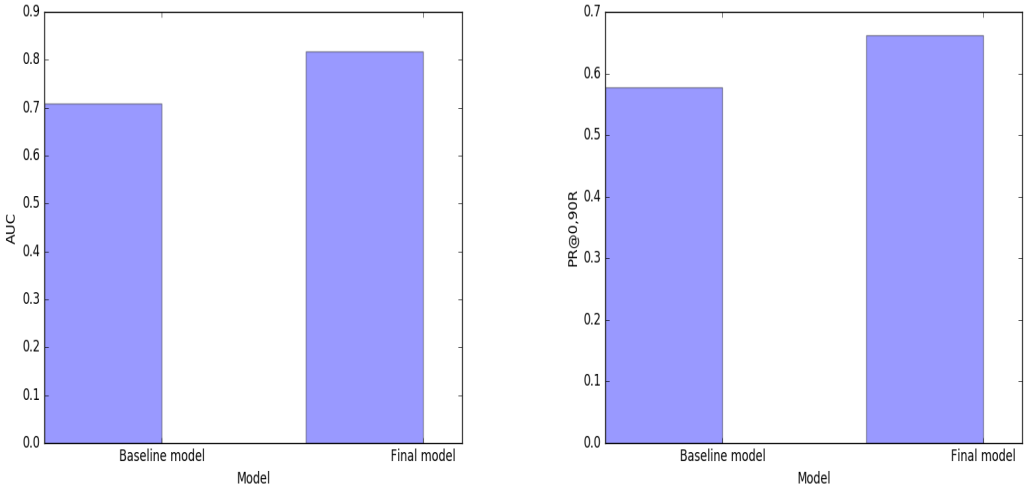
4.4 Effect of Negative Enrichment

Baseline model LR-PCA:1200-C had AUC of the precision-recall curve for the CV set 0.831 and LR-PCA:1473-C had 0.708 for the test set. LR-PCA:1200-C-B after the Negative Enrichment had AUC of the precision-recall curve for the CV set 0.884 and LR-PCA:1413-C-B 0.818 for the test set. We can see that after Negative Enrichment the difference between the AUC of the CV set and test set is smaller than the Baseline. That means Negative Enrichment helped us to get a model that generalise better. This is the goal of building a model after all, a model that can generalise.

An other improvement of the Negative Enrichment is that we got a model with higher precision at 0.90 recall. The Baseline gave 0.577 precision at 0.90 recall and after the Negative Enrichment LR-PCA:1413-C-B gave 0.662. That means when we have gotten 90% of the parasites of the test set, the model after the Negative Enrichment gives higher prediction probabilities for the parasite patches and lower prediction probabilities for background patches. In figure 4.12 we can see the overall performance improvement that Negative Enrichment gave to us.

Figure 4.13 illustrates the performance enhancement of the Negative Enrichment technique on the test set.

We applied PCA:50-C on the full training set and we plotted our data. van der Maaten and Hinton (2008) have created the technique t-distributed Stochastic Neighbor Embedding (t-SNE), which is a tool to visualize high-dimensional data. The authors have proven that t-SNE produces notably better visualizations than other existing methods. Intuitively, t-SNE converts a high-dimensional dataset to joint probabilities (a matrix of pair-wise similarities). The objective of t-SNE is the minimization of the Kullback-Leibler divergence between the joint probabilities of the high-dimensional data and the low-dimensional embedding. van der Maaten and Hinton (2008) indicate



(a) AUC of precision-recall curve.

(b) Precision at 0.90 recall.

Figure 4.12: Effect of Negative Enrichment.

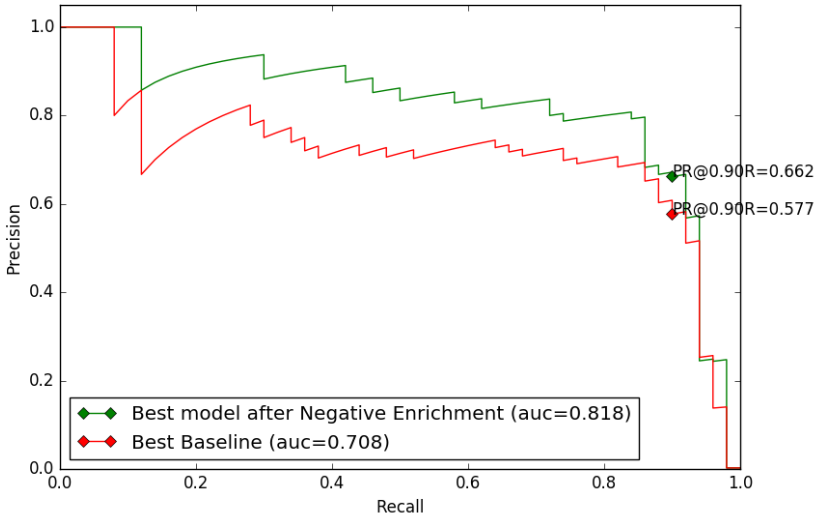
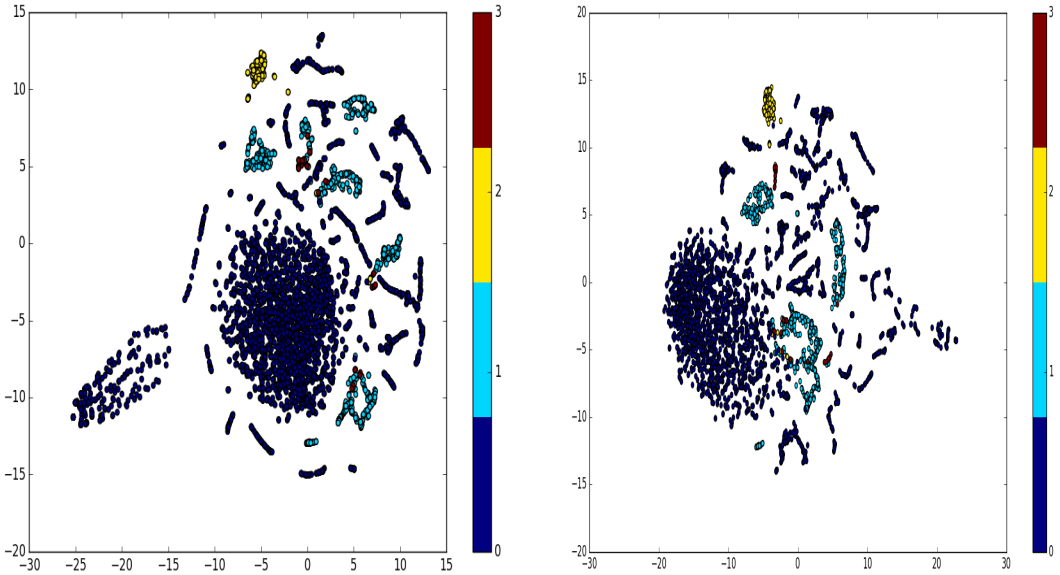


Figure 4.13: Performance comparison of Baseline model and best model on test set.

that t-SNE can reveal global structure such as clusters. But, t-SNE has a non-convex cost function. That means we can get different results for different initializations. To cope with non-convexity in figure 4.14 we have set the *random_state* to zero when we applied t-SNE for both the Baseline PCA:50-C dataset and the dataset PCA:50-C after the Negative Enrichment. Different initializations might result in different local minima of the cost function, so we kept the seed control steady for both t-SNE runs.

Moreover, it was necessary to use the dimensionality reduced data PCA:50-C to speed up the computation of pairwise distances between samples, otherwise the computations were very slow. Figure 4.14 illustrates our training samples in two dimensions after applying t-SNE. Labels 0, 1, 2 and 3 correspond to background, Hookworm parasite, Taenia parasite and Hymenolepsis Nana parasite patches respectively.



(a) Using PCA:50-C of the Baseline.

(b) Using PCA:50-C after applying Negative Enrichment.

Figure 4.14: Data visualization using t-SNE.

In subfigure 4.14.b we can see that our training samples are better clustered compared to the subfigure 4.14.a. Background patches form a big cluster instead of two big clusters, Hookworm parasite patches form 3 clusters instead of 6, Taenia parasite patches still form 1 cluster and Hymenolepsis nana parasite patches are merged with Hookworm parasite patches. This improved clustering is another indication that Negative Enrichment can be beneficial to building a model.

4.5 Implementation details

We used Python[4] as the programming language to complete this project. We mainly used the scikit-learn[5] library, which offers various and efficient tools for machine learning. We used scikit-learn to train our models (e.g. SVM, LR), apply PCA, create the stratified five-fold cross validation and visualize the data (e.g t-SNE). We used the OpenCV[6] library to apply CLAHE image pre-processing and the scikit-image[7] library to extract the HOG descriptors.

[4] https://www.python.org/
[5] http://scikit-learn.org/stable/
[6] http://opencv.org/
[7] http://scikit-image.org/

Chapter 5

Hookworm Detection

In this chapter we detail how to detect a Hookworm parasite in intestinal microscopy images. About the 80% of the parasites in our dataset are Hookworm parasites. So, it is interesting to experiment if we can train a model which regards the Hookworm parasites as the positive class and the Taenia and Hymenolepsis Nana parasites and the background as the negative class. The intuition of this experiment is that there are very few Taenia and Hymenolepsis Nana parasites so that a model can learn features from them. Similarly to 4.3, in section 5.1 we have separated the models into two stages. In the first stage we have the Baseline models for which we have randomly selected background patches and in the second stage we have the models which have been trained using the Negative Enrichment technique. For both the steps we evaluate our models on the validation set of the stratified folds. In section 5.2 we investigate the effect of Negative Enrichment on the Hookworm detection task.

5.1 Evaluation

In this section we evaluate the methods we decided to use and we compare the results given by the various models we trained but for the Hookworm parasite detection task instead of any kind of parasite. In section 5.1.1 we evaluate our models for which we randomly selected the background patches and we analyze the results taken from these models. Then, in section 5.1.2 we evaluate our models which have been trained using the background patches that have been selected by the negative enrichment technique. In section 5.1.3 we evaluate our best model on the test set. We used the same models as in in Chapter 4.

5.1.1 Baseline

For Hookworm parasite detection again we had the Baseline where for each stratified fold we randomly selected the background patches. So, it was quite likely that each model on this cross-validation procedure was trained on quite different patches that represent the background class. Table 5.1 shows the performance on the stratified folds.

	CV	
Model	AUC	PR@0.9R
LR	0.770	0.354

Table 5.1: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

Compared to the model in section 4.3.1.2 for the Hookworm detection task we got worse performance since in section 4.3.1.2 we got 0.577 precision at 0.90 recall.

5.1.1.1 CLAHE

As we saw in the section 4.3.1.4 that CLAHE image pre-processing can enhance the models' performance, in this task where we want to detect Hookworm parasites, we also applied CLAHE pre-processing. The results are included in table 5.2.

	CV	
Model	AUC	PR@0.9R
LR	0.770	0.354
LR-C	0.819	0.491

Table 5.2: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

As we can see the model with CLAHE image pre-processing gave better results than the model without CLAHE pre-processing. This enhancement agrees with the improvement we got on the performance for the parasite prediction task in section 4.3.1.4.

5.1.1.2 PCA

Similarly, we decreased the dimensions using the PCA technique since the HOG descriptor returns vectors of high dimension. Table 5.3 demonstrates the results of the Hookworm parasite detection for the case of the reduced dimensions.

	CV	
Model	AUC	PR@0.9R
LR	0.770	0.354
LR-PCA:1200	0.762	0.356
SVM-PCA:1200	0.549	0.338
LR-PCA:50	0.526	0.118
SVM-PCA:50	0.531	0.198

Table 5.3: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

Using the model with the 1200 first principal components we got about the same performance with the model using all the HOG descriptor. However, if we reduce the dimensions of the dataset to 50 dimensions we get worse performance. Matching the results in chapter 4 reducing the dimensions to 50 we lost a lot of information about the dataset.

5.1.1.3 Weight balancing

The problem of the imbalanced dataset is bigger for the case of the Hookworm parasite detection task since the patches of the Taenia and Hymenolepsis Nana parasites are considered negative for this task. As a result, the ratio of the positive samples is much smaller than the negative ones compared to the ratio for the parasite detection task. We experimented with the “balanced” mode. The results are included in table 5.4.

The results from table 5.4 match with the results from the section 4.3.1.6 where the weight balancing for the Baseline gave bad performance. However, similarly with the chapter 4 we will see in section 5.1.2.3 that weight balancing combined with Negative Enrichment can improve performance.

	CV	
Model	AUC	PR@0.9R
LR	0.770	0.354
LR-B	0.448	0.309

Table 5.4: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

5.1.1.4 Regularisation

For the same reasons with the parasite detection task we also used regularisation for the Hookworm detection test. We used inverse of regularization strength C (regularisation $\lambda = 1/C$), smaller values specify stronger regularization. Table 5.5 demonstrates the results.

	CV	
Model	AUC	PR@0.9R
LR	0.770	0.354
LR-IR:0,5	0.739	0.279
LR-IR:0,1	0.620	0.075

Table 5.5: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

From the results we observe that regularisation does not improve the performance as far as the Baseline is concerned.

5.1.1.5 Combinations

In this section we present the results of the models built using combinations of the above methods after applying the Negative Enrichment. The results are included in table 5.6.

Table 5.6 indicates that using CLAHE image pre-processing models with PCA dimensionality reduction can give better performance than the simple logistic regression models. We saw in section 5.1.1.2 that PCA didn't improve the overall performance, but now using CLAHE it did. That means CLAHE can help the PCA technique to find better features and keeping the first 1200 principal components we have removed

	CV	
Model	AUC	PR@0.9R
LR-PCA:1200-C	**0.804**	**0.476**
LR-PCA:1200-C-B	0.390	0.279
LR-PCA:1200-IR:0.5-C	0.798	0.419
SVM-PCA:1200-C	0.616	0.413

Table 5.6: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

unnecessary information. The model LR-PCA:1200-C matches with the Baseline of the parasite detection task.

5.1.1.6 Evaluation of the best model

Since we tuned our models using the validation sets of the stratified folds, we evaluated the final performance of our model on the test set. Also for the Hookworm detection task our best Baseline model was the LR-PCA:1200-C validated on the CV folds. We used 1200 principal components for all the CV folds because these components correspond approximately to 95% variation of every fold. To train our final model on all the training set, we have to randomly select the background patches. Since the selection is random we used the training patches that we used to build our model in section 4.3.1.9, but now we trained our model LR-PCA:1473-C regarding the Hookworm parasite patches as positive training instances and the Taenia parasite, Hymenolepsis Nana and background patches as negative training instances. The performance of the best model on the test set is included in table 5.7.

	Test	
Model	AUC	PR@0.9R
LR-PCA:1473-C	0.780	0.156

Table 5.7: Performance on the test set. AUC corresponds to the area under the curve of the precision - recall curve.

Figure 5.1 demonstrates the precision-recall curve for the CV and test set. As it was expected the area under the curve and the precision at 0.90 recall are bigger for the CV than for the test set.

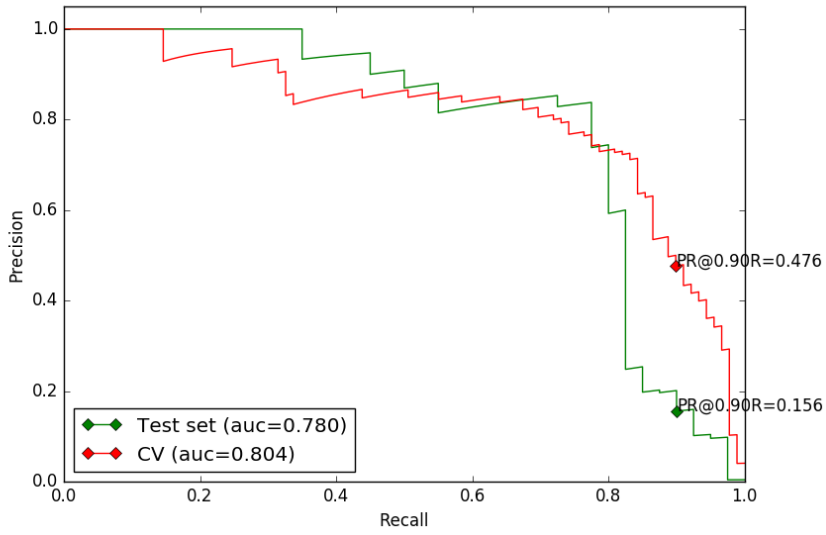


Figure 5.1: Performance comparison of the best model on CV and test set.

5.1.1.7 Conclusion of the evaluation

In this section we tested the performance of the models trained on randomly selected background patches. We made the same observations that we did in section 4.3.1. CLAHE image pre-processing improves both the AUC of the precision-recall curve and the precision at 0.90 recall. Moreover, if we apply PCA to reduce the dimensional space of our task, we noticed that if we keep the 1200 first principal components that correspond to approximately the 90% variance of our CV folds, we get about the same performance without dimensionality reduction. But, if we reduce the dimension keeping the 50 first principal components, we get worse performance. That's means we lost information reducing the dimensions to 50. Finally, weight balancing and regularisation can't give better results as far as the Baseline is concerned.

5.1.2 Negative Enrichment

An important conclusion from chapter 4 was that Negative Enrichment can improve performance. The motivation was that we wanted to cope with the problem of the randomly selected background patches. We applied Negative Enrichment for the Hookworm parasite detection as well for all the CV folds.

Some examples of the background patches that were selected through the Negative Enrichment step are included in figure 5.2. In this figure we can observe that

these patches can easily be regarded as parasites if a model hasn't been trained using these patches as negative training instances. As a result using Negative Enrichment we trained our model with these noisy patches, the Taenia and Hymenolepsis Nana parasites and some other randomly selected backround patches as negative instances.

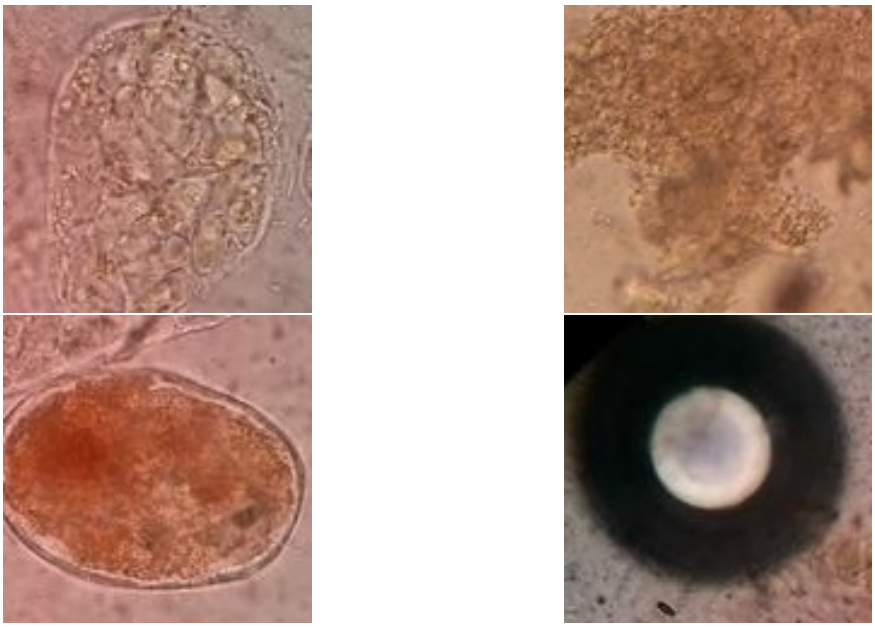


Figure 5.2: Background patches selected by Negative Enrichment.

The performance of the logistic regression model after applying Negative Enrichment is included in Table 5.8.

	CV	
Model	AUC	PR@0.9R
LR	0.807	0.462

Table 5.8: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

5.1.2.1 CLAHE

Since CLAHE provided us with performance enhancement for the parasite detection task after the Negative Enhancement step, we also applied CLAHE for the Hookworm parasite detection task after the Negative Enhancement step. Table 5.9 demonstrated the results.

	CV	
Model	AUC	PR@0.9R
LR	0.807	0.462
LR-C	0.838	0.516

Table 5.9: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

From table 5.9 we observe that CLAHE can also increase the overall performance for the Hookworm detection task after applying Negative Enrichment.

5.1.2.2 PCA

We saw in chapter 4 that PCA improved the overall detection performance after the Negative Enhancement step for the parasite detection task. So, we applied PCA after the Negative Enrichment step for the Hookworm detection. The results are included in table 5.10.

	CV	
Model	AUC	PR@0.9R
LR	0.807	0.462
LR-PCA:1200	0.822	0.488
SVM-PCA:1200	0.782	0.264
LR-PCA:50	0.772	0.340
SVM-PCA:50	0.764	0.369

Table 5.10: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

From table 5.10 we notice that, in the case of the Hookworm parasite detection, dimensionality reduction can improve the performance of our model after we have applied Negative Enrichment. This performance improvement matches with the enhancement we have in section 4.3.2.2 for the case of the parasite detection task.

5.1.2.3 Weight balancing

Since we have more imbalanced dataset for the case of the Hookworm parasite detection task, we also experimented the “balanced” mode after the Negative Enrichment

step. Table 5.11 illustrates the results.

	CV	
Model	AUC	PR@0.9R
LR	0.807	0.462
LR-B	0.851	0.667

Table 5.11: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

Matching the results of section 4.3.2.3 weight balancing can improve the performance of our model after applying Negative Enrichment. Compared to the Baseline we notice that in the case of the Negative Enrichment the "balanced" mode can improve the performance of our model.

5.1.2.4 Regularisation

Similarly to Baseline we ran experiments using regularisation after applying Negative Enrichment. Again, we use inverse of regularization strength C (regularisation $\lambda = 1/C$), smaller values specify stronger regularization. The results are included in Table 5.12.

	CV	
Model	AUC	PR@0.9R
LR	0.807	0.462
LR-IR:0.5	0.800	0.406
LR-IR:0.1	0,.744	0.408

Table 5.12: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

From table 5.12 we observe that if set the inverse of regularisation to 0.5 we get approximately the same performance with the logistic regression model without regularisation. However we will see in section 5.1.2.5 that if we combine regularisation with CLAHE and weight balancing we can improve the performance. If we use more regularisation specifically 0.1 we get worse results.

5.1.2.5 Combinations

In this section we present the result of model built using the combination of the methods after applying the Negative Enrichment. The results are included in table 5.13.

	CV	
Model	AUC	PR@0.9R
LR-PCA:1200-C	0.770	0.577
LR-PCA:1200-C-B	**0.843**	**0.721**
SVM-PCA:1200-C	0.778	0.420
SVM-PCA:1200-C-B	0.761	0.560
LR-PCA:1200-IR:0,5-C-B	0.837	0.714
LR-PCA:50-C	0.812	0.503
LR-PCA:50-C-B	0.812	0.615
SVM-PCA:50-C	0.818	0.460
SVM-PCA:50-C-B	0.684	0.540
SVM-PCA:50-IR:0,5-C	0.817	0.468

Table 5.13: Average performance on the cross-validation folds. AUC corresponds to the area under the curve of the precision - recall curve.

From table 5.13 we notice that the model LR-PCA:1200-C-B gives the best overall performance on the validation sets. That means after having applied Negative Enrichment the combination of dimensionality reduction, weight balancing and CLAHE image pre-processing can enhance performance. This model matches with the model we got in section 4.3.2.5 for the parasite detection task.

5.1.2.6 Evaluation of the best model

Since we tuned our models using the validation sets of the stratified folds, we evaluated the final performance of our model on the test set. We used 1200 principal components for all the CV folds because these components correspond approximately to 95% variation of every fold. However, using all the training set the principal components that exactly correspond to 95% variation are 1421. So, we trained the model LR-PCA:1421-C-B on all the training set. We used the Hookworm parasite patches as the positive instances and we used the Negative Enrichment method in the same way as we did for the stratified cross-validation folds. We classified all the background

training patches and we kept the incorrectly classified and the correctly classified with probability lower than 90%. If these patches were less than four times the positive ones, we randomly selected different background patches so that we could have four times the positive ones in total. The Tenia and Hymenolepsis Nana parasites also were regarded as negative instances. Then, we applied PCA and we kept the 1421 first principal components. We applied CLAHE image pre-processing and weight balancing as well. The performance of the best model on the test set is included in table 5.14.

	Test	
Model	AUC	PR@0.9R
LR-PCA:1421-C-B	0.778	0.218

Table 5.14: Performance on the test set. AUC corresponds to the area under the curve of the precision - recall curve.

Figure 5.3 demonstrates the precision-recall curve for the CV and test set. As expected the area under the curve and the precision at 0.90 recall are bigger for the CV set than for the test set. But, compared to the Baseline the curve of the test set is more close to the curve of the CV set. This result is good because it indicates that the model after the Negative Enrichment generalise better than the Baseline model.

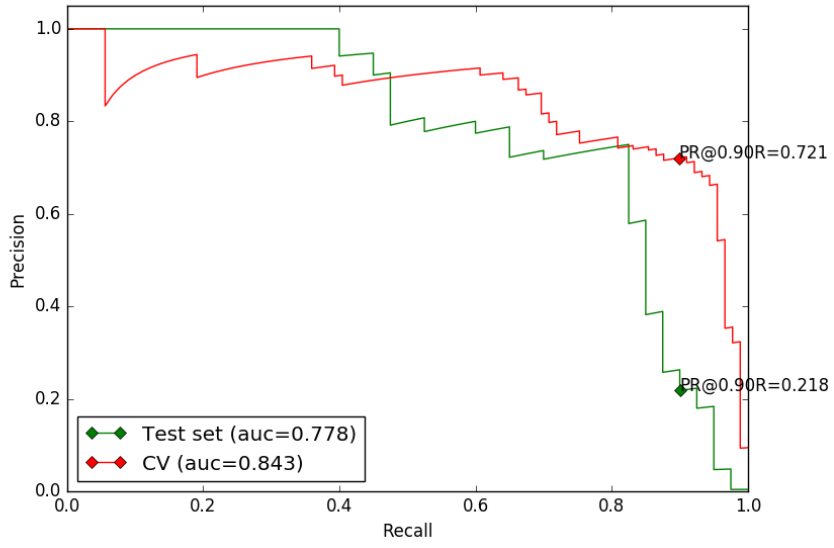
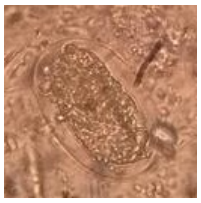


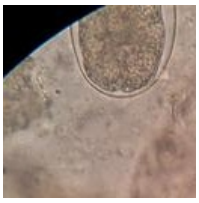
Figure 5.3: Performance comparison of the best model on CV and test set.

From figure 5.3 we can observe that when recall is close to one, precision is zero.

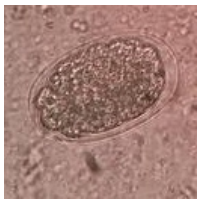
That means there are some false negative parasite patches that have very low prediction probability. We checked what test parasite patches classified as false negatives and had extremely low prediction probability. Out of the 40 Hookworm parasites of the test LR-PCA:1421-C-B correctly classified 33 and 7 were false negative. Figure 5.4 illustrates the false negative patches with the prediction probabilities.



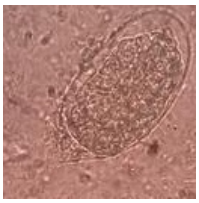
(a) Prediction probability 0.222.



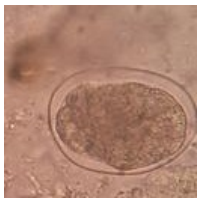
(b) Prediction probability 0.030.



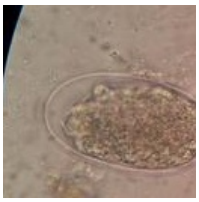
(c) Prediction probability 0.256.



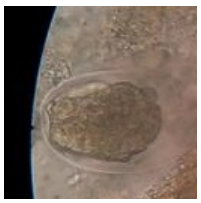
(d) Prediction probability 0.197.



(e) Prediction probability 0.408.



(f) Prediction probability 0.331.



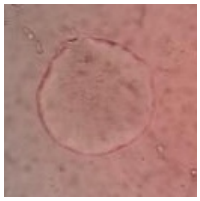
(g) Prediction probability 0.101.

Figure 5.4: False negative patches.

From figure 5.4 we can see that the Hookworm parasite with the lowest probability 0.030 (b) is not even a whole Hookworm. We saw this patch in section 4.3.2.6 where our model predicted a probability 0.012 for this patch. It's a part of a Hookworm parasite and is located at the corner of the microscopy image. Our model cannot predict it as a Hookworm parasite. This parasite patch is the reason that the precision is zero when recall is close to one. We also saw the Hookworm parasites (a), (c), (d) and (g) in section 4.3.2.6. It seems that these patches have features that are similar to the

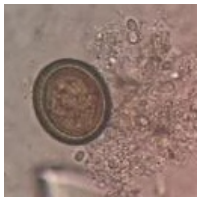
background patches' features.

Moreover, LR-PCA:1421-C-B incorrectly classified as Hookworm parasites thirteen Taenia, Hymenolepsis Nana and background patches. Figure 5.5 demonstrates the six incorrectly classified patches as Hookworm parasites with the highest prediction probabilities.

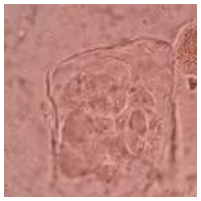


(a) Prediction probability 0.831.

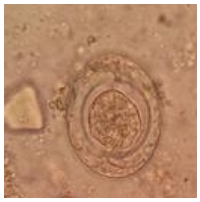
(b) Prediction probability 0.856.



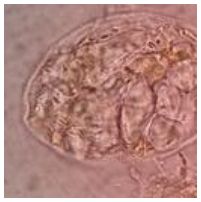
(c) Prediction probability 0.782.



(d) Prediction probability 0.790.



(e) Prediction probability 0.741.



(f) Prediction probability 0.891.

Figure 5.5: False positive patches.

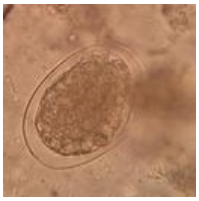
From figure 5.5 we also saw the patches (a), (b) and (d) in section 4.3.2.6. The patches (a) and (b) that have noise with round shape and the patch (d) that has noise that can be regarded as a parasite. Patch (f) has 0.891 prediction probability and it does have noise that looks like a parasite and it can have same features as the Hookworm parasite patches. Patch (c) contains a Taenia parasite and patch (e) contains a Hymenolepsis Nana parasite. It could be reasonable if someone expected that the model would return Taenia and Hymenolepsis Nana parasite patches as false positives, since Taenia and Hymenolepsis Nana parasites are Helminth eggs such as Hookworm parasites.

Like in section 4.3.2.6 for the parasite detection task we can also plot the test patches with the highest and lowest prediction probabilities for the Hookworm detection task. Figure 5.6 presents these patches. Compared to the parasite detection task we can observe that these Hookworm patches have lower detection probabilities

than the patches than in section 4.3.2.6. Probably, the reason is that the Taenia and Hymenolepsis Nana parasites that can have similar HOG descriptors now in this task are considered negative instances. This observation matches with the results in figure 5.8 where the distribution of the positive patches of the parasite detection task has more mass in the probability region [0.5-1.0] than the distribution of the Hookworm detection task. All of the patches with the lowest probabilities contain the corner of the microscopy image and only noise in the part of the microscopy images. This observation agrees with the results in the figure 4.9 but now the patches have higher detection probabilities than the patches in figure 4.9. We will see in figure 5.7 that the negative patches take higher detection probabilities in the Hookworm detection task than in the parasite detection task.

(a) Prediction probability 0.969.

(b) Prediction probability 0.986.



(c) Prediction probability 0.878.

(d) Prediction probability 1.608×10^{-5}.

(e) Prediction probability 3.527×10^{-5}.

(f) Prediction probability 2.120×10^{-5}.

Figure 5.6: Test patches with the highest and lowest prediction probabilities.

Figure 5.7 shows the histogram of the prediction probabilities of the test patches. Histogram illustrates how the probabilities of both the Hookworm parasites and background (included Taenia and Hymenolepsis nana) test patches are distributed between zero and one. We can observe that almost 100% of the background patches are given probability between zero and 0.2 by our model LR-PCA:1421-C-B. The false positives are so few compared to the total number of background patches that we cannot even see the percentage of these false positive patches on the plot. Moreover, the majority of the Hookworm parasite patches are distributed between the probability 0.5 and 1.0. The parasites patches that are given prediction probabilities between zero and 0.5 are the false negative patches that we saw in figure 5.4.

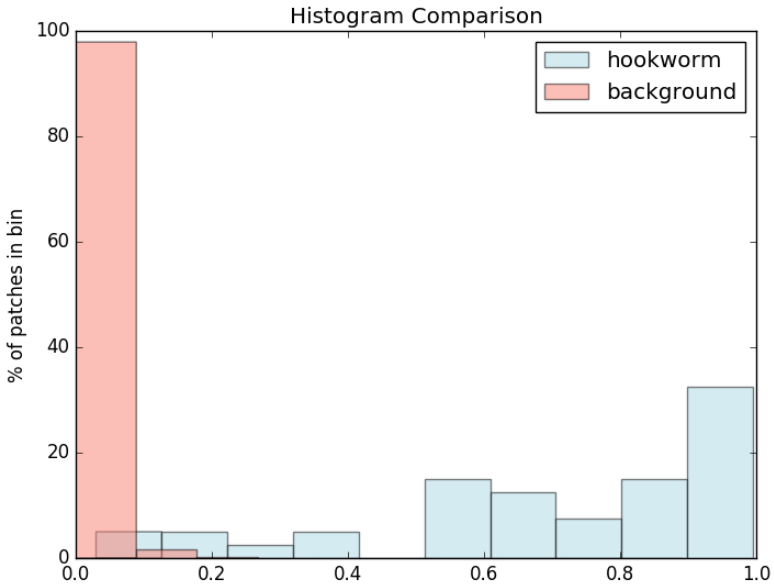


Figure 5.7: Histogram of both Hookworm and background (included Taenia and Hymenolepsis Nana) patches.

If we compare the figures 4.10 and 5.7 we can see that in figure 4.10 the distribution of the positive patches has more mass between the probabilities 0.5 and 1.0 compared to the distribution of the positives patches in figure 5.7. Moreover, in figure 4.10 the portion of the negative patches that are included in the probability region [0.0, 0.1] is bigger than the portion of the negative patches in figure 5.7. These observations indicate that the parasite detection model provide us with a better performance than the Hookworm parasite detection model.

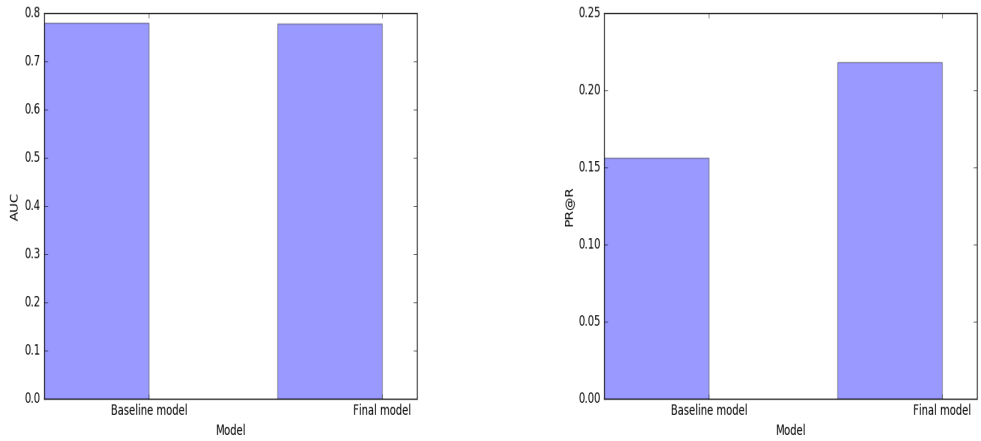
5.1.2.7 Conclusion of the evaluation

In this section we tested the performance of the models trained on background patches selected by the Negative Enrichment technique and some other randomly selected. Similarly to the Baseline, we saw that CLAHE image pre-processing improves both the AUC of the precision-recall curve and the precision at 0.90 recall. Moreover, if we apply PCA to reduce the dimensional space of our task, we noticed that we get better overall performance for both the 1200 and 50 dimensions. The most interesting observation is that we get better performance for the 50 dimension than for the 1200 dimensions in comparison with the Baseline where reducing to 50 dimension we got much worse results.

5.2 Effect of Negative Enrichment

Baseline model LR-PCA:1200-C had AUC of the precision-recall curve for the CV set 0.804 and LR-PCA:1473-C had 0.780 for the test set. LR-PCA:1200-C-B after the Negative Enrichment had AUC of the precision-recall curve for the CV set 0.843 and LR-PCA:1421-C-B 0.778 for the test set. In the case of the Hookworm detection task we didn't get better AUC after the Negative Enrichment step. This is an indication that the Hookworm detection task is a more difficult problem compared to the parasite detection task.

The improvement of the Negative Enrichment is that we got a model with higher precision at 0.90 recall. The Baseline gave 0.156 precision at 0.90 recall and after the Negative Enrichment LR-PCA:1421-C-B gave 0.218. That means when we have gotten 90% of the parasites of the test set, the model after the Negative Enrichment gives higher prediction probabilities for the parasite patches and lower prediction probabilities for background patches. In figure 5.8 we can see the overall performance improvement that Negative Enrichment gave to us.



(a) AUC of precision-recall curve.

(b) Precision at 0.90 recall.

Figure 5.8: Effect of Negative Enrichment.

Figure 5.9 illustrates the performance enhancement of the Negative Enrichment technique on the test set. The AUC of the precision - recall curve are approximately the same and Negative Enrichment didn't enhance the performance in contrast to the parasite detection task. However, similar to the parasite detection task Negative Enrichment also increased the precision at the 0.90 recall for the Hookworm detection task.

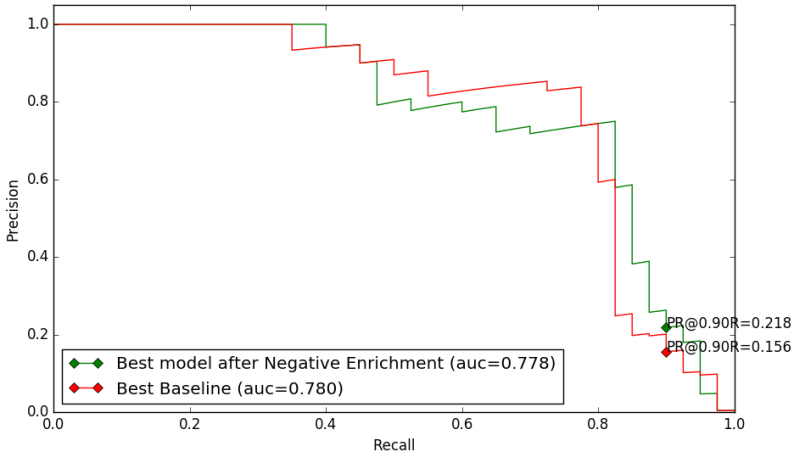


Figure 5.9: Performance comparison of Baseline model and best model on test set.

Chapter 6

Conclusions and future work

In this project we have proposed a system to detect intestinal parasites or Hookworm parasites based on image pre-processing, feature descriptor, dimensionality reduction and weight balancing techniques. Based on the importance of the intestinal parasite diagnosis and the lack of resources and laboratory technicians in low-resource areas, this project can contribute to the research on the automatic parasite detection in microscopy images. In contrast to previous work on automatic intestinal parasite classification we need to detect the intestinal parasites in a highly imbalanced dataset. Figure 6.1 illustrates two examples of our parasite detection system's output that we built in chapter 4. These examples are two test images, the first image contains a Hymenolepsis Nana parasite and the second image contains two Hookworm parasites. We can see that our model has highlighted the detected parasites with a bounding box and has returned a detection probability for the detected parasites. The main difficulty of this project, in contrast to the majority of previous work, is the high background noise, which decreases the quality of the feature extraction step and as a consequence the detection performance as well. In figure 6.1 we can see this high background noise in the intestinal microscopy images, but despite this high background noise our system is capable of detecting the parasites.

Our system, for both the parasite detection and Hookworm detection tasks, was divided in four stages: image pre-processing, feature extraction, dimensionality reduction and training. Through our experiments we observed that CLAHE image preprocessing provided us with performance enhancement. For this reason we applied CLAHE pre-procesing to build our system. In contrast with the deep learning methods we had to extract the appropriate representation of the training patches and we extracted these representation using the HOG descriptor. Due to the the high dimen-

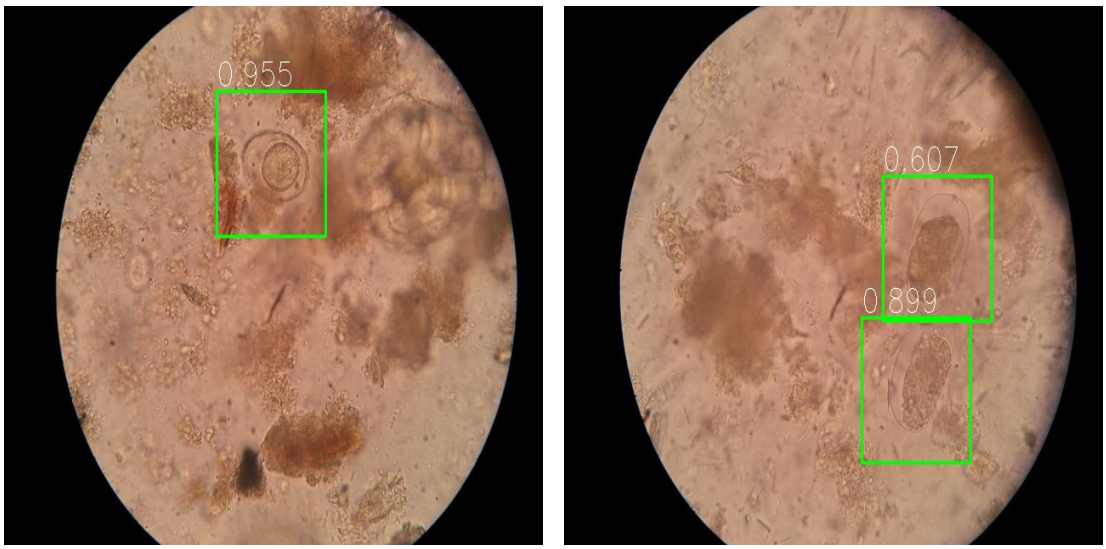


Figure 6.1: Parasite detection.

sional feature vectors of the HOG descriptor, in our case the HOG feature vector was of dimensionality 15876, we applied PCA dimensionality reduction keeping the principal components that correspond to the 95% variation of our training data. Finally, we trained our system using weight balancing since we had highly imabalanced dataset and weight balancing improved the performance.

These four stages where applied two times. The first time we built a model (Baseline) using randomly selected background patches for negative instance, but we saw that randomly selected background patches can create models with quite different AUC and PR@R performance. To cope with this problem we used the Negative Enrichment technique. Applying this technique we used the Baseline to predict detection probabilities for all the background patches. The incorrectly classified and the correctly classified but with a probability lower than 90% were used in the training negatives instances. This technique gave an overall better performance than the Baseline.

To obtain the best possible system we ran a lot of experiments to tune the parameters and carefully selected what methods we should apply on the system. We are able to detect parasites with 0.818 average precision and 0.662 precision at 0.90 recall. An ideal parasite diagnosis system requires a high recall and to find 90% we get a precision of 66%. This precision is quite good if we take into consideration the small amount of training data that we had available to train our system and the high background noise that we had to cope with when we were sliding the window to detect the parasites on the microscopy images. This system can already be useful for the work on the parasite diagnosis, but more data would be necessary for the training and test process for

reliable performance.

As future work it would be very promising to apply Convolutional Neural Networks (CNNs) on this project's task. CNNs are applied on various problems and it has been showed that CNNs outperformed a lot of other methods, especially on computer vision tasks. Of course, to apply CNNs on this project's task a much bigger training dataset will be needed. The small amount of training parasite patches was the main reason that we didn't try experiments using CNNs.

Even when CNNs, combined with a much bigger training dataset, might well improve the overall performance, our system gives results good enough to be useful in intestinal parasites diagnosis. Another important aspect is that our system is not only restricted to intestinal parasites. It can easily be applied on any other parasite detection, since the feature extraction stage is not specifically restricted to the intestinal parasites that we had to detect. A computer vision system such as this one can beneficially contribute to parasite diagnosis, especially in the developing countries where inexpensive diagnosis systems are needed.

Bibliography

Amit, Y. and Felzenszwalb, P. F. (2014). Object detection. In *Computer Vision, A Reference Guide*, pages 537–542.

Avci, D. and Varol, A. (2009). An expert diagnosis system for classification of human parasite eggs based on multi-class SVM. *Expert Systems with Applications*, 36(1):43 – 48.

Bellman, R. E. (1957). *Dynamic programming*. Princeton University Press.

Castañón, C. A. B., Fraga, J. S., Fernandez, S., Gruber, A., and da F. Costa, L. (2007). Biological Shape Characterization for Automatic Image Recognition and Diagnosis of Protozoan Parasites of the Genus Eimeria. *Pattern Recognition*, 40(7):1899–1910.

Cortes, C. and Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, 20(3):273–297.

Dalal, N. and Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893, Washington, DC, USA. IEEE Computer Society.

Davis, J. and Goadrich, M. (2006). The relationship between Precision-Recall and ROC curves. In *ICML 2006*, pages 233–240.

Dogantekin, E., Yilmaz, M., Dogantekin, A., Avci, E., and Sengur, A. (2008). A robust technique based on invariant moments for recognition of human parasite eggs in microscopic images. *Expert Systems with Applications*, 35(3):728–738.

Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The PASCAL Visual Object Classes VOC Challenge. *International Journal of Computer Vision*, 88(2):303–338.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D. A., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645.

Fenwick, A. (2012). The global burden of neglected tropical diseases. *Public health*, 126(3):233–236.

Ghazali, K. H., Alsameraai, R. S. H., and Mohamed, Z. (2013). Automated system for diagnosis intestinal parasites by computerized image analysis. *Modern Applied Science*, 7(5):98–114.

Haque, R. (2007). Human intestinal parasites. *Journal of Health, Population and Nutrition*, 25(4):387 – 391.

Jacobsen, K. H., Ribeiro, P. S., Quist, B. K., and Rydbeck, B. V. (2007). Prevalence of Intestinal Parasites in Young Quichua Children in the Highlands of Rural Ecuador. *Journal of Health*, 25(4):399 – 405.

Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142, London, UK, UK. Springer-Verlag.

Malisiewicz, T., Gupta, A., and Efros, A. A. (2011). Ensemble of exemplar-svms for object detection and beyond. In *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pages 89–96.

Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., Romeny, B. T. H., and Zimmerman, J. B. (1987). Adaptive Histogram Equalization and Its Variations. *Comput. Vision Graph. Image Process.*, 39(3):355–368.

Rai, S. K., Uga, S., Kataoka, N., and Matsumura, T. (1996). *Atlas of Medical Parasitology*. Kobe U. School of Medicine, Kyokuseisya Co.,Ltd. 1-2-7 Ninomiya-cho,Chuo-ku.

Rema, M. and Madhu, S. N. (2013). Segmentation of human intestinal parasites from microscopy images using localized mean-separation based active contour model. *Biomedical Engineering Letters*, 3(3):179–189.

Sanchez, C. S. (2015). Deep Learning for Identifying Malaria Parasites in Images. *Master of Science thesis - University of Edinburgh, School of Informatics*.

Sengur, A. and Turkoglu, I. (2004). Parasite egg cell classification using invariant moments. In *Proceedings of 4th international symposium on intelligent manufacturing systems*, pages 98–106.

Turk, M. A. and Pentland, A. P. (1991). Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 586–591.

van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Yang, Y. S., Park, D. K., Kim, H. C., Choi, M.-H., and Chai, J.-Y. (2001). Automatic identification of human helminth eggs on microscopic fecal specimens using digital image processing and an artificial neural network. *IEEE Transactions on Biomedical Engineering*, 48(6):718–730.

Zhang, J., Lin, Y., Liu, Y., Li, Z., Li, Z., Hu, S., Liu, Z., Lin, D., and Wu, Z. (2014). Cascaded-Automatic Segmentation for Schistosoma japonicum eggs in images of fecal samples. *Computers in Biology and Medicine*, 52:18 – 27.

Zuiderveld, K. (1994). Contrast Limited Adaptive Histograph Equalization. *Graphic Gems IV. San Diego: Academic Press Professional*, pages 474–485.