University of Essex

Assessing blood samples with Malaria

School of Computer Science and Electronic Engineering

Eurico Pinto | Dariush Mirshekar-Syahkal

Malaria is a deadly disease, responsible for 400,000 deaths each year. The lack of medical professionals cause deficient Malaria diagnosis, where sometimes, a patient with Malaria symptoms is given a medicine without knowing wetter the dosage is correct or if its cells are actually infected.

This project appears to present a possible solution for these cases. The main goal is to deliver a mobile app that can analyse a microscopic image of blood and retrieve the ratio of infected/uninfected cells with Malaria.

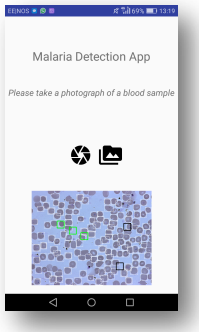

Figure 1. Mobile App Prototype

The Software used are Python, OpenCV library, Android Studio and Java. There are also external sources – images of individual cells, uninfected and infected; blood sample images. The single cell images are used to train an Haar Cascade Classifier – a key factor to determine if cells are infected. The blood sample images are mainly to check how efficient the classifier is.

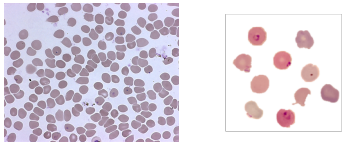

Figure 2. Blood sample and single cell images

One of the first steps is to segment the single cell images. Those have an original dark background, completely different from the background colour present on the blood sample images – light colour. This is easily achieved by thresholding the image and then turn each dark pixel to white – the background will turn to white colour. The threshold basically separates the cell from the background in binary terms, so the change of pixel colour does not interferes with the actual cell colour.

Figure 3. Single cell segmentation – change to light background

Moving forward, those segmented single cells can then be used to train an Haar Cascade Classifier. To summarise, all infected cells represent positive images and all uninfected cells represent negative images for the training. The training runs under the terminal shell using OpenCV library. This retrieves a XML file, that can be used as a Classifier in the future by Python or Java.

When the training has finished, it is time to try it out on the blood sample images. Before doing that, the cell contours need to be encountered, in that way, the classifier can interpret each cell at a time, instead of doing a complete analysis of the all image. The histograms below help finding a good threshold value to sepate each cell indiviadually

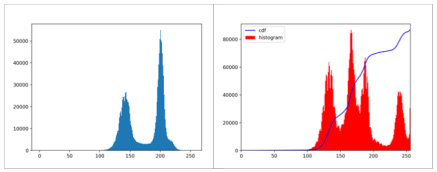

Figure 4. Histograms of blood sample image

OpenCV library offers the possibility to analyse each cell at a time. When one cell is detected, it is interpreted by the classifier – if it finds that it is infected, the cell coordinates are saved in an array, if not it just moves to the next cell. In the end, all infected cells detected should be presented with a red square and the actual infected ones with a black square.

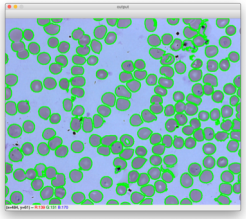

Figure 5. Cell contours using Python and OpenCV

With time, the number of right ones detected increased but also, the number of false detections decreased. The main factor for better results were better and more complete classifiers and the proper segmentation of images – single cells and blood sample images.

Figure 6. Testing classifier with its own training samples

When the testing of multiple blood sample images is done, the functionality is translated to the mobile app, running manly Java – Android App. All translations were being made throughout the project timeline to avoid overload.

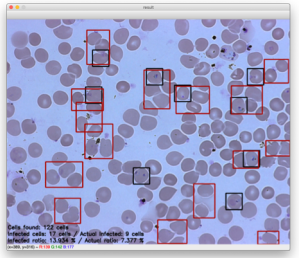

Figure 7. Actual results in Python with classifier