Learning-based parasite segmentation in the context of the MOVABLE project

Roberto Rigamonti and Magali Fröhlich and Yann Thoma
REDS institute, HEIG-VD — School of Business and Engineering Vaud
CH-1400 Yverdon-les-Bains, Switzerland
name.surname@heig-vd.ch

***Abstract*—This document details the learning process adopted to segment parasites from the Plasmodium family in the context of the MOVABLE project[1], which is aimed at computing the parasitemia from optical microscope images of blood samples in the presence of a suspected malaria infection.**

I. Introduction

Malaria is a disease where parasites from the Plasmodium family attack the erythrocytes of the vertebrate host and propagate until the host dies, usually because of complications occurred due to organ failure. The transmission between hosts is performed by mosquitoes, and indeed the use of bed-nets is a very effective strategy to hinder the spreading of the disease. Nonetheless, the World Health Organization (WHO) estimates that more than 400’000 people die each year because of malaria, and reported infections are above 200 millions [11]. These numbers were significantly higher before a joint effort of the developed countries boosted the quality of education, prevention, and therapies.

The current major concern relates to the abuse of antibiotics, mostly related to ineffective detection techniques adopted in third world’s countries. For instance, only 12-20% of all malaria cases in Tanzania are confirmed parasitologically [5]. Beside the economic impact that unneeded expensive treatments have on the small budgets that low-income countries devote to health care, this phenomenon favours parasite resistance, alarmingly weakening the weapons we have to counter the disease.

In this document we analyze in detail a learning-based approach to segment parasites from the Plasmodium family in blood samples. While learning-based systems traditionally require more computational resources than their image processing-based counterparts, they are better suited to a context where imaging conditions undergo extreme variations. Nonetheless, the two approaches must not be considered as competing but as complementary: image processing is indeed a powerful tool to accomplish some of the tasks required for computing the parasitemia value — for instance, segmenting red blood cells —, and can be used to incorporate in the learning process prior knowledge that would otherwise be impossible for the learning algorithm to acquire from data [8].

The solution we explore is grounded on the technique presented in [2], [3], [9] and called KernelBoost. In the following sections we will examine how this technique works and detail the characteristics of the built system.

[1]*MicrOscopic VisuAlization of BLood cElls for the Detection of Malaria and CD4+*, http://reds.heig-vd.ch/rad/projets/movable

II. Segmenting parasites with KernelBoost

In this section we will illustrate the inner workings of the KernelBoost algorithm, showing how it has been adapted to our purposes. Precision will be sacrificed for the sake of clarity; for a mathematically-sound overview of the proposed approach, please refer to [2], [3].

A. Gradient Boosting

The core component of the system presented in [2], [3], [9] is an algorithm called Gradient Boosting [10], which can be seen as a generalization of the well-known AdaBoost algorithm where weak-learners are real-valued and different loss functions can be used. In essence, what Gradient Boosting does is the following:

1) The algorithm starts by taking a set of binary-classified samples and assigning to each sample the same weight.
2) On a randomly-sampled subset of these samples — where the weight of each sample influenced the sampling process — a simple classifier (called **weak learner**) is learned. The classifier has to be simple to trade accuracy for speed. As there are two classes only, random guessing would have given us roughly 50% accuracy, so we expect the classifier to be a bit better than this (but not much).
3) We include the classifier in the set of classifiers learned so far, giving it a weight proportional to its accuracy on a validation set that has been kept apart.
4) We evaluate how the overall set of weak learners performs on the data by adding the weighted predictions of each weak learner in the set. The weight of each sample is kept into account too, as the algorithm is trying to minimize the overall misclassification penalty.
5) If we are satisfied with the result, or if the number of weak learners in the set has reached a predefined limit, then stop.
6) Otherwise, samples are reweighted according to a specific rule (in AdaBoost it is an exponential), giving more weight to samples that have been misclassified and less weight to the correctly classified ones. The rationale is that we want the algorithm to focus on the mistakes it has made. The algorithm then jumps back to step (2).

This technique has proven to be resistant to overfitting, and has been applied in several disparate tasks [3].

B. Learning in KernelBoost

KernelBoost differs from a typical boosting algorithm in that the features (that is, the characteristics) on which the weak learners are learned are not hand-crafted — for instance, gradients, SIFT features, …— but are extracted by a set of kernels that are learned on the data itself and are applied in a convolutional way on the input images. While hand-crafted features have their advantages — notably, they are mathematically sound, they incorporate prior knowledge on the domain, and they are typically fast to compute —, devising the best feature type for a given dataset is very difficult and biased towards what we *assume* the learning algorithm prefers as input.

Insights from the neurosciences [1] showed that living beings are not pre-wired with a set of feature extractors, but learn them directly from the data they acquire. However, learning both the features and the classifier on the same dataset increases the risk of overfitting, therefore care has to be taken while manipulating sample sets.

In particular, KernelBoost operates on three sets while learning each weak learner:

- Given a set of samples A, one third of the set is used to learn a set of N candidate kernels.
- The kernels are then used to learn a regression tree on a disjoint sample set (taking another third of the overall set of samples), which will discard all but M kernels (typically, for a regression tree of depth 4, only 15-25 kernels are retained while N could be in the order of thousands).
- Finally, the whole set of samples (thus including 1/3 of the set that has never been seen before) is used for scoring the performance of the learned classifier and attributing it a weight using line search.

The algorithm used to learn each individual kernel takes as input a set of few thousands samples from the two considered classes, and uses Regularized Least Squares to compute the hyperplane that best splits the two sets. In particular, given a set of S samples (where S is 1/3 of the overall number of samples), R positive and R negative samples are extracted from the images to create a subset $\mathcal{P}$. To discriminate among positive and negative samples, the label of the central pixel of the sample is assumed to be the label of the sample and all its subsamples. As shown in Fig. 1, a random center point inside the sample area and a random size are chosen. Then, all corresponding subsamples in $\mathcal{P}$ are taken and used to compute the kernel by Regularized Least Squares minimization.

The original dataset, from which the samples are taken, can be artificially balanced by activating a flag in the program's configuration. This prevents a label class to overcome by more than 20% the number of samples of the other class, thus avoiding an initial bias in the class label given to the samples.

The most basic implementation of the algorithm uses, as input data, a grayscale version of the input images, and learns a fixed number of kernels on them for each weak learner.

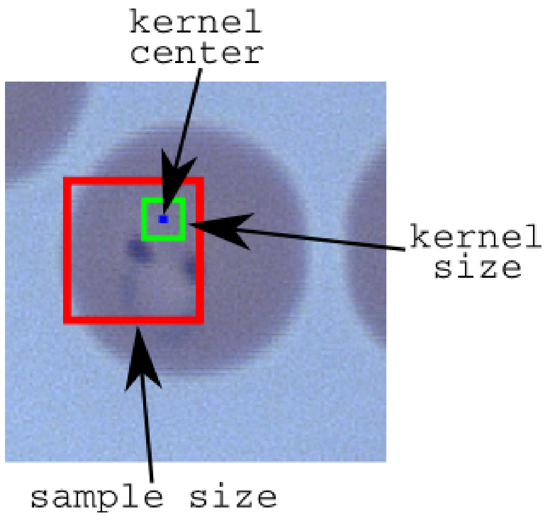

Fig. 1. Sampling mechanism for kernel learning: Samples of the specified size are extracted from the image set, then a random center point is chosen inside the sample and a subsample of random size is considered. The corresponding subsample is taken from all the samples in the sample set, and kernels are learned on this collection of subsamples. The label of each subsample is the label inherited from the sample it was taken from, which corresponds to the label of the center of the sample.

However, multiple different channels are possible: indeed, one can learn a separate set of kernels on each individual channel, and then let the regression tree learning algorithm pick which kernels work best. We have considered 8 input channels that can be easily computed starting from the original RGB input images:

- grayscale version of the input image;
- green component of the input image;
- red component of the input image;
- grayscale version of the input image filtered with a median filter;
- grayscale version of the input image filtered with a Laplacian filter;
- grayscale version of the input image filtered with a Gaussian filter;
- Sobel derivative in the X-direction of the grayscale version of the input image;
- Sobel derivative in the Y-direction of the grayscale version of the input image.

Additional channels, incorporating for instance some prior knowledge or extracted using image processing techniques, can be easily added to the system, monotonically increasing the performances. Tab. I shows the repartition of the kernels over the channels in a test run (where each Boosted Classifier was composed by up to 500 weak learners).

The dataset we consider is constituted by images collected at the CHUV medical center in Lausanne, Switzerland. The images contain mainly parasites in their ring-shaped phase, but other stages of the parasite's life might appear. As the clinically-relevant stage is the ring one, we created the ground-truth images so as to mark with different colours the background, the parasites in the ring stage, and the parasites in the other stages of development, as shown in Fig. 2. Indeed, trying to solve the problem of finding the ring-shaped parasites

TABLE I
KERNEL REPARTITION OVER THE CHANNELS IN A TEST RUN WHERE EACH BOOSTED CLASSIFIER WAS COMPOSED BY UP TO 500 WEAK LEARNERS

Classifier	Grayscale	Green	Red	Median	Laplacian	Gaussian	Sobel-X	Sobel-Y	BC(0-127)	BC(0-255)	BC(127-255)
BC(0-127)	20.19%	11.52%	21.60%	12.39%	6.18%	13.59%	7.08%	7.44%	-	-	-
BC(0-255)	19.04%	11.13%	18.84%	13.06%	7.39%	14.95%	8.29%	7.30%	-	-	-
BC(127-255)	19.74%	11.98%	26.66%	12.49%	3.10%	13.06%	5.86%	7.10%	-	-	-
BC(FINAL)	14.03%	7.29%	8.44%	7.25%	6.03%	7.29%	6.44%	6.09%	10.38%	14.03%	12.73%

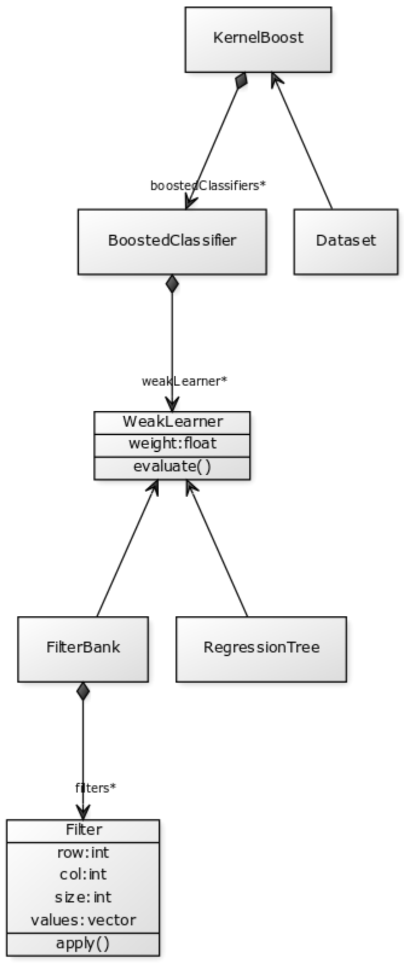

Fig. 4. UML diagram of the developed system. In the code the kernels are dubbed *filters* given their structure and that they are applied in a convolutional way to images.

directly proved to be very difficult. However, introducing a third class poses a problem too, as KernelBoost is capable of dealing with binary problems only. To allow for the three different ground-truth values, the problem is split over three parallel learning problems where each deals with a separate value pair. The three classifiers are then applied on the initial images, giving their predictions that are then used in a final KernelBoost classifier solving the initial binary problem. This technique of reusing the results of previous stages is known in literature as Auto-Context [12].

The full architecture of the system is depicted by Fig. 3, while the corresponding UML diagram is shown in Fig. 4.

C. Computing the segmented image

Once the KernelBoost classifier produces its prediction, we still have to post-process the image to obtain a binary classification than can then be fed to the GUI for being revised by the technician and that can be used to estimate the parasitemia.

We have, at first, implemented a strategy for automatically devising the threshold used to binarize the classifier's prediction from the data by finding the threshold that maximized the Precision-Recall measure. This approach led to images which were too noisy to be used as final segmentation results. We have, however, realized that the desired threshold is stable across multiple images and could be safely hand-determined. We have therefore adopted this approach, leaving the automatic adaptation of this value — possibly based on ROC curves — as future work.

Once the image has been binarized, the obtained regions could still present small holes and spurious pixels affecting image quality. We have thus removed all regions whose area was below a fixed threshold and performed a morphological close operation that filled small gaps. Then we computed the convex hull of each blob and filled it. Finally, we added to the image the binary inverse of the flood-fill operation, to ensure that even big holes inside detections are filled.

D. Automatic incorporation of user's feedback

Boosting is known to exhibit a certain degree of robustness against errors in the training data. However, having high quality data is crucial to reduce misclassification error. Once an image is segmented and presented to the technician via the GUI, he can manually fix the segmentation errors and return back the image, along with the fixed ground-truth, as feedback for retraining the system. This feedback is very valuable, so we heavily weight samples that belongs to it. These feedback images are distinguished from initial ones by a marking prepended to the image file name in the configuration files. This initial reweighting makes the samples from those images more likely to be considered and makes mistakes on them more serious, reducing the impact of the errors in the initial training data.

III. CONCLUSION AND FUTURE WORK

In this document we have presented a bird's-eye view of the principles upon which the learning system used in MOVABLE has been developed.

The system is agnostic with respect to the object to detect, requiring only minor tuning in the sample and kernel size, and can therefore be used for tasks different than segmenting

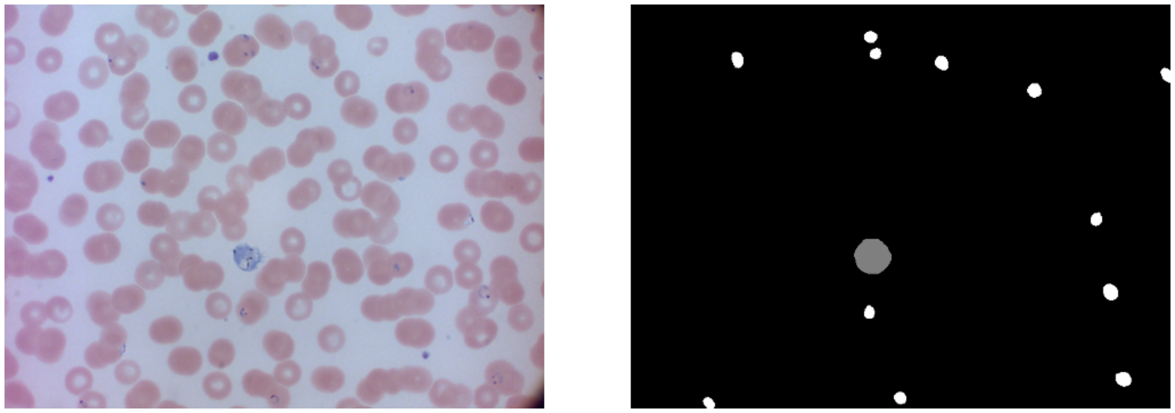

Fig. 2. Example image from the dataset, along with the corresponding ground-truth. In the ground-truth the ring-shaped parasites are marked in white, while the parasites in other stages of their life are marked in gray.

Plasmodium parasites — for instance, it can be used to segment CD4+ cells (indeed, it has been shown to be very successfully in segmenting Jurkat cells in [9]).

For further information on the algorithm, please refer to [2], [3], [9].

The system has several aspects that can be further improved:

- The algorithm might incorporate the image processing-based solution proposed by [4]; as mentioned in the introduction, this presents several important advantages, namely a fast and precise erythrocyte segmentation and an initial parasite detection, both of which can be fed to the system as additional channels significantly improving the quality of the final segmentation.
- The image segmentation technique currently adopted is called "pixel-wise segmentation", as it gives a label to each individual pixel ignoring the labels of the neighboring pixels. The size of the kernels and the kernel learning technique partially reduce this restriction, but still the number of computations is extremely large: a weak learner, for a reasonable parametrization, has indeed on average around 20 kernels, and if we consider 300 weak learners (on average) for each boosted classifier and three ground truth classes, we need to perform $4 \times 300 \times 20 = 24'000$ convolutions with the input channels, and then individually classify each of the pixels based on these 24'000 features. Given that parasites are chemically stained and must be located inside a red blood cell, the vast majority of this effort is wasted. Indeed, erythrocytes cover at most 50% of the image area, and only few of them present marks that has to be inspected to asses whether they correspond to parasite or not. Restricting the algorithm to operate on these regions, for example by using a cascaded classifier in the spirit of the one proposed by Viola-Jones [6], could boost the performance — as finer-resolution images and more weak learners could be used — while at the same time reducing the computational time by several orders of magnitude.
- Additional input channels should be explored.
- The threshold used to binarize the result proved to be sufficiently stable across the dataset we have explored, but having an adaptive thredshold could increase the quality of the results presented to the user.
- The 2D kernels could be decomposed in 1D kernels by using the technique explored in [7], considerably reducing the cost of the convolutions required to extract the features.

References

[1] B.A. Olshausen and D.J. Fields. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 1996.

[2] C.J. Becker and R. Rigamonti and V. Lepetit and P. Fua. KernelBoost: Supervised Learning of Image Features For Classification. In *MICCAI*, 2013.

[3] C.J. Becker and R. Rigamonti and V. Lepetit and P. Fua. KernelBoost: Supervised Learning of Image Features For Classification. Technical Report 183586, EPFL, 2013.

[4] G. Burri. Parasitémie automatisée de la malaria à partir d'images microscopiques. Master's thesis, Haute École Spécialisée de Suisse Occidentale, 2016.

[5] J. Kahama-Maro and V. D'Acremont and G.D. Mtasiwa and C. Lengeler. Low quality of routine microscopy for malaria at different levels of the health system in Dar es Salaam. *Malaria Journal*, 2011.

[6] P. Viola and M. Jones. Robust Real-time Object Detection. *International Journal of Computer Vision*, 2001.

[7] R. Rigamonti and A. Sironi and V. Lepetit and P. Fua. Learning Separable Filters. In *CVPR*, 2013.

[8] R. Rigamonti and V. Lepetit. Accurate and Efficient Linear Structure Segmentation by Leveraging Ad Hoc Features with Learned Filters. In *MICCAI*, 2012.

[9] R. Rigamonti and V. Lepetit and P. Fua. Beyond KernelBoost. Technical Report 200378, EPFL, 2014.

[10] T. Hastie and R. Tibshirani and J. Friedman. *The Elements of Statistical Learning*. Springer New York Inc., 2001.

[11] World Health Organization. World Malaria Report. Technical report, 2015.

[12] Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3D brain image segmentation. *IEEE TPAMI*, 2010.

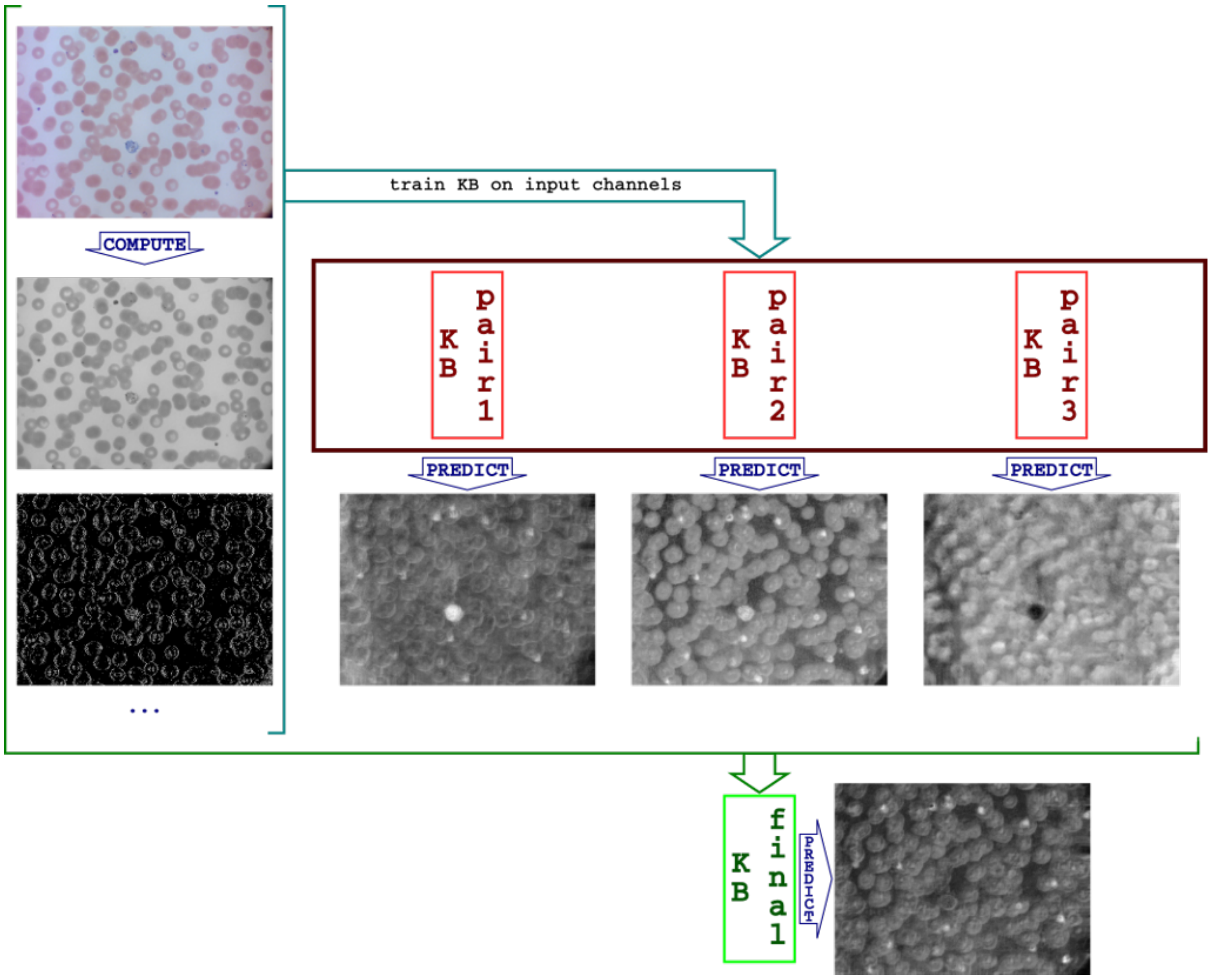

Fig. 3. Architecture of the learning system. The input image is used to compute the different channels that will be used during the learning process; These channels are then fed to the three separate KernelBoost classifiers (one for each ground-truth pair) that output a prediction of the input image. These predictions are finally used as supplemental channels to train a final KernelBoost predictor on the desired ground-truth pair.