Automated Complete Blood Cell Count and Malaria Pathogen Detection Using Convolution Neural Network

Arindam B. Chowdhury[1,2] Jeremy Roberson[2], Ajat Hukkoo[2], Srinivas Bodapati[2], and David J. Cappelleri[1]

***Abstract*—Complete blood cell count, which indicates the density of different blood cells in the human body is extremely important for evaluating the overall health of a person and also for detecting a wide range of disorders, including anemia, infection and leukemia. Hence, automating this task will not only increase the speed of diagnosis, but also lower the overall treatment cost. In this paper, we focus on using a convolution neural network to perform this complete blood cell count on blood smear images. The network is also trained to detect malarial pathogens in the blood, if present. Experiments show that the overall performance of the system has a mean average precision of over 0.95 when compared with the ground-truth. Furthermore, the system predicts the images containing malarial parasites as infected 100% of the time. The software is also ported to a low cost microcomputer for rapid prototyping.**

***Index Terms*—Automation in Life Sciences: Biotechnology, Pharmaceutical and Health Care, Medical Robots and Systems, Deep Learning in Robotics and Automation**

I. INTRODUCTION

A Blood smear under a microscope contains several types of information for disease diagnosis. Blood cells are mainly of three types: (1) White Blood Cells (WBC), whose concentration is $4,500$ to $10,000$ cells per microliter (cells/mcl), (2) Red Blood Cells (RBC), which have a concentration of 4.1 million to 5.1 million cells/mcl, and (3) Platelets or Thrombocytes (THR) with a concentration of $150,000$ to $450,000$ cells/mcl, for a normal healthy individual. The THRs also sometimes appears in clumps as well. A measure of the total count of these three types of blood cells ina a sample is referred to as a complete blood cell (CBC) count. WBCs again are divided into five categories, namely: Eosinophil $(1-5\%)$, Basophil $(0-1\%)$, Neutrophil $(50-70\%)$, Lymphocyte $(20-45\%)$ and Monocyte $(2-10\%)$. Their concentrations are an important indicator of the state of the immune system of a person. For example, a patient with Leukemia will have a much higher concentration of Lymphocytes than normal. Therefore, many diseases can be detected simply by having a count of RBCs, THRs and different types of WBCs in a blood smear. Additionally, a blood smear can also indicate if any foreign pathogens exits in the blood. For example, a person suffering from Malaria will have pathogens in the RBCs which can be visible in the blood smear. An example of a labelled blood smear image is shown in Fig. 1.

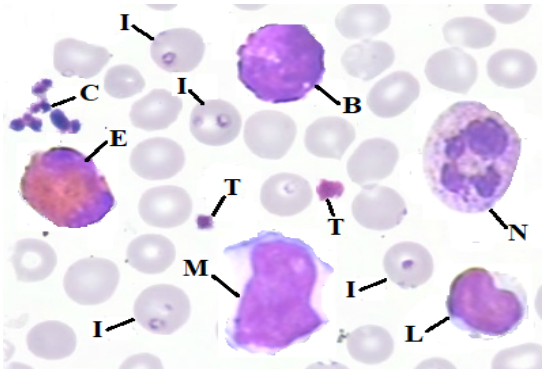

Fig. 1. Blood smear image showing different blood cells. E: Eosinophil, B: Basophil, N: Neutrophil, L: Lymphocyte, M: Monocyte, T: Thrombocytes or platelets, C: Platelet clumps, I: RBC infected with Malarial parasite. All the other objects are normal healthy RBCs.

Several different methods are used in medical clinics for counting blood cells and detecting pathogens. This can then be done manually by an expert pathologist, but is very tedious and time consuming, or by some commercial automated systems like Cella-Vision [1], Hema-CAM [2], MEDICA EasyCell Assistant [3] etc. These automated systems employ pre-processing, image segmentation, feature extraction and classification techniques. The pre-processing stage includes noise removal, color correction and image enhancement processes. The segmentation stage separates the cell components, like nuclei or cytoplasm, which is especially useful for WBC identification. However, different types of blood cells have different shapes and structures (as seen in Fig. 1). Therefore, it is difficult to generalize a segmentation algorithm for all types of cells. The feature extraction stage is used to extract the features to be used for classification. This stage sometimes involves extracting pre-defined hand-crafted features, which again may suffer from lack of generalization. The limitations of these automated systems are discussed in detail in [4]. Furthermore, these systems can also be expensive and may not be affordable for many clinics.

Recent advancements in deep learning, have shown that a convolutional neural network (CNN) can perform image classification more accurately than humans on a wide range

Manuscript received: September, 18, 2019; Revised December, 14, 2019; Accepted January, 8, 2020.

This paper was recommended for publication by Editor Youngjin Choi upon evaluation of the Associate Editor and Reviewers' comments. *Intel Corporation

[1] A. Chowdhury and D. Cappelleri are with the Multi-Scale Robotics & Automation Lab, School of Mechanical Engineering, Purdue University, West Lafayette, IN. USA. {abhanjac, dcappell}@purdue.edu

[2] A. Chowdhury, J. Roberson, A. Hukkoo, and S. Bodapati are with Intel Corporation, Santa Clara, CA USA. {jeremy.roberson, ajat.hukkoo, srinivas.bodapati}@intel.com

Digital Object Identifier (DOI): see top of this page.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2020.2967290, IEEE Robotics and Automation Letters

of objects [5]. Thus, we are interested in investigating a deep learning approach for this task. There has been some previous work on this topic as well. In [6], the authors have found a good mean average precision for detecting different WBC types, although they have ignored the Basophils due to a lack of samples in the dataset. In [7], an iterative circle detection algorithm is used to find the WBCs and RBCs. However, their system does not detect the WBCs types. A cost-effective method for obtaining a CBC count as well as WBC classification is shown in [8] using traditional computer vision techniques. Such techniques though, have hand-crafted algorithms which are difficult to scale to more diverse datasets. The work in [9] shows an attempt to use a deep belief network for Malaria detection, but CNNs do perform better when it comes to classifying images. The U-Net deep learning framework shown in [10] uses semantic segmentation for segmenting neural structures. Theoretically, this method could be used here for classifying and counting the different types of cells. To count the number of cells of a particular type with U-Net, it will include counting the number of pixels representing an object in the predicted segment map and then dividing this number by the average number of pixels that represents such objects in ground truth maps. This process can be prone to errors due to noisy pixels and the size ranges in the number of pixels making up a particular type of cell.

Considering these previous attempts, the main focus of this project is to create a low-cost CNN based automated system to identify and count RBCs, THRs, different WBC types and also detect Malarial infection from a digitized blood smear image. To the best of our knowledge, a combined system of such kind has not been designed or implemented yet. Our objective is not to come up with a new kind of deep learning architecture or technique, but to optimize existing architectures for the required digital pathology application and finally to implement the overall software on a low-cost embedded electronic micro-computer. The rest of the paper is organized as follows. Section II describes how the dataset is prepared for training the CNN. Section III describes the CNN architectures used for the system. Experimental results are provided in Section IV. The contributions and future work of this project are given in Section V.

II. DATASET PREPARATION

A. Original Medical Databases

The images used for creating the training, testing and validation datasets are obtained from four different databases: (1) Leukocyte Images for Segmentation and Classification (LISC) database [11]. This contains images of five types of WBCs on a background of RBCs. The images are labeled by the type of WBC in them, and each image also has a binary mask that indicates the pixels representing the WBC region; (2) Isfahan University of Medical Science (IUMC) database [12]. This has labeled images of individual WBCs with their binary masks. However, this database does not have Basophil images; (3) MAMIC database [13]. It has large blood smear images of healthy RBCs, THRs, Platelet clumps and Malaria infected RBCs. Occasionally, WBCs also

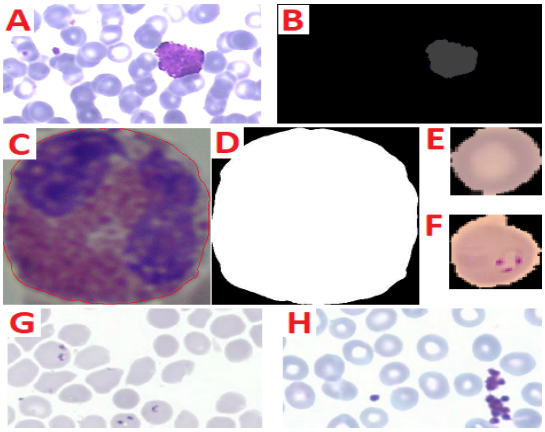

Fig. 2. Sample images from different databases. A, B: Basophil from LISC and its binary mask, C, D: Eosinophil from IUMC and its binary mask, E: Healthy RBC from KAGGLE, F: Malaria infected RBC from KAGGLE, G: Some infected and healthy RBCs from MAMIC, H: Some healthy RBCs, THR and Platelet clumps from MAMIC.

appear in the MAMIC images, but they are not labelled. Every image contains multiple cells, without any binary masks to separate them; (4) KAGGLE database [14]. This contains images of individual healthy and infected RBCs, but without any binary masks. All the Malarial infection images in the last two databases are with Plasmodium Falciparum pathogen. Fig. 2 shows sample images from each of these databases. The main reason to combine all these different databases is the unavailability of a single annotated database that contains all types of blood cells (mentioned earlier) along with malaria infected RBCs.

B. Preprocessing Image Samples

For a robust training of the CNN, the training dataset should have a wide variety of combinations of the different blood cells. For example, there should be images with an Eosinophil and a Basophil with healthy RBCs in the background, images with a Monocyte and Platelet clumps on a background containing both healthy and infected RBCs, images containing only Lymphocytes on a background of infected RBCs, etc. None of the databases mentioned earlier has this much variety. Additionally, total number of WBC images over all the databases is around 391, which is not sufficient for a good training. Hence, a fresh dataset was created which has the desired variations, using images from the previously mentioned databases as building blocks.

As a first step, a set of images is created that has only one kind of cell in them along with their binary masks. This is done for the LISC, KAGGLE, and MAMIC images. IUMC images are already in this format. The region of WBCs in the LISC images are cropped off using their masks to create individual images of WBCs. LISC and IUMC provides all the required WBC samples. One set of infected and healthy RBCs are obtained from KAGGLE. THRs, Platelet clumps and another set of infected and healthy RBCs are cropped out manually from several MAMIC images. The binary masks of the samples obtained from KAGGLE and MAMIC are created using simple image thresholding technique. Finally, all these newly created samples are resized such that they are similar in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2020.2967290, IEEE Robotics and Automation Letters

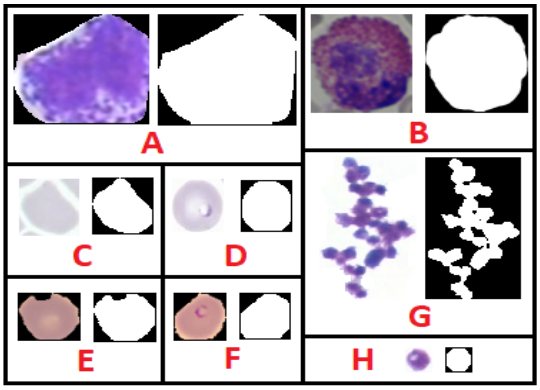

Fig. 3. Resized image samples and their binary masks used for creating training, testing and validation datasets. A: Basophil cropped from an LISC image, B: Eosinophil from IUMC, C: Healthy RBC cropped from a MAMIC image, D: Infected RBC cropped from a MAMIC image, E: Healthy RBC from KAGGLE, F: Infected RBC from KAGGLE, G: Platelet clump cropped from a MAMIC image, H: THR cropped from a MAMIC image.

size to cells seen under a microscope with $40\times$ magnification. Some of these final samples are shown in Fig. 3. The total number of samples obtained in this manner for different cells is given in Table I.

TABLE I
NUMBER OF SAMPLES CREATED FROM THE DIFFERENT DATABASES

Cell Types	LISC	IUMC	MAMIC	KAGGLE
Eosinophil	37	42	-	-
Basophil	50	-	-	-
Neutrophil	47	38	-	-
Lymphocyte	45	32	-	-
Monocyte	48	36	-	-
Thrombocyte	-	-	82	-
Platelet clump	-	-	36	-
Infected RBC	-	-	407	13779
Healthy RBC	-	-	3539	13779

C. Synthetic Dataset Creation

Now the actual images for training, testing and validation are created. First, all of the different types of image samples shown in Table I are separated into three groups namely: *training samples* (comprising 80% of all the samples), *testing samples* (comprising 10% of all the samples) and *validation samples* (comprising 10% of all the samples). Only images from the training samples set are used to create the synthetic training dataset. Similarly, only images from the testing and validation samples sets are used to create the images for testing and validation datasets, respectively. This is done so that there are no common samples between the three datasets created and the neural networks never see any testing samples during training.

Fig. 4 shows a flowchart of how the datasets are created. The size of the images in these datasets are 224×224 pixels. At first, some 1000×1000 background images are created that contain only RBCs in them. This is done by affixing randomly selected RBC samples on a blank image at random places. These locations are also recorded in a separate list. Altogether, 1500 such background images are created. 500 of these have only infected RBCs, 500 have only healthy RBCs, and 500 have a mix of both. Then, 224×224 blocks are cropped out

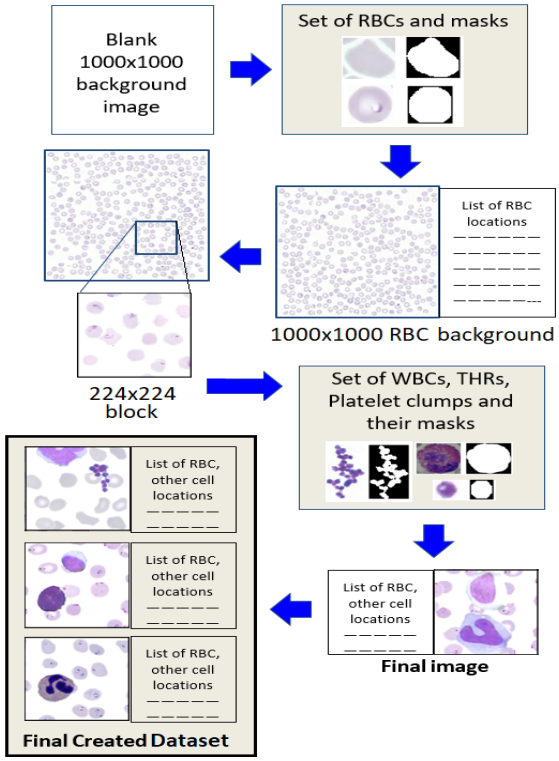

Fig. 4. Flowchart showing how the training, testing and validation datasets are created.

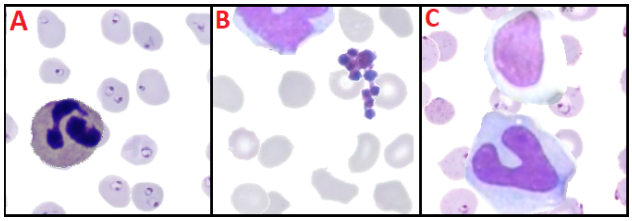

Fig. 5. Sample images from the final dataset. A: Neutrophil on Infected RBC background, B: Platelet clump and a Partial WBC on Healthy RBC background, C: Two Monocytes on Infected RBC background.

of these images from random locations and WBC, THR and Platelet clump samples are affixed in them randomly. For each such image, the class names of the objects and the position and size of their bounding boxes are recorded in a separate list. The samples are also rotated at random angles while affixing them. Some sample images obtained are shown in Fig. 5.

D. Partially Visible Cells

A completely digitized blood smear image from a microscope at $40\times$ magnification, is huge in size and cannot fit into 224×224 pixels. Such an image has to be broken down into segments and sent into the CNN as a batch of 224×224 images. The CNN produces the inference output on this batch of images which are then stitched back to the original size. While creating the batch of 224×224 images, it may happen that some WBCs may fall at the boundary of two adjacent segments and only be partly visible in the image. Therefore, there may not be enough features visible to identify such WBCs correctly. Hence, while creating the training dataset, some images are deliberately created with the WBC only partially visible in them, as shown in Fig. 5. In some of these

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2020.2967290, IEEE Robotics and Automation Letters

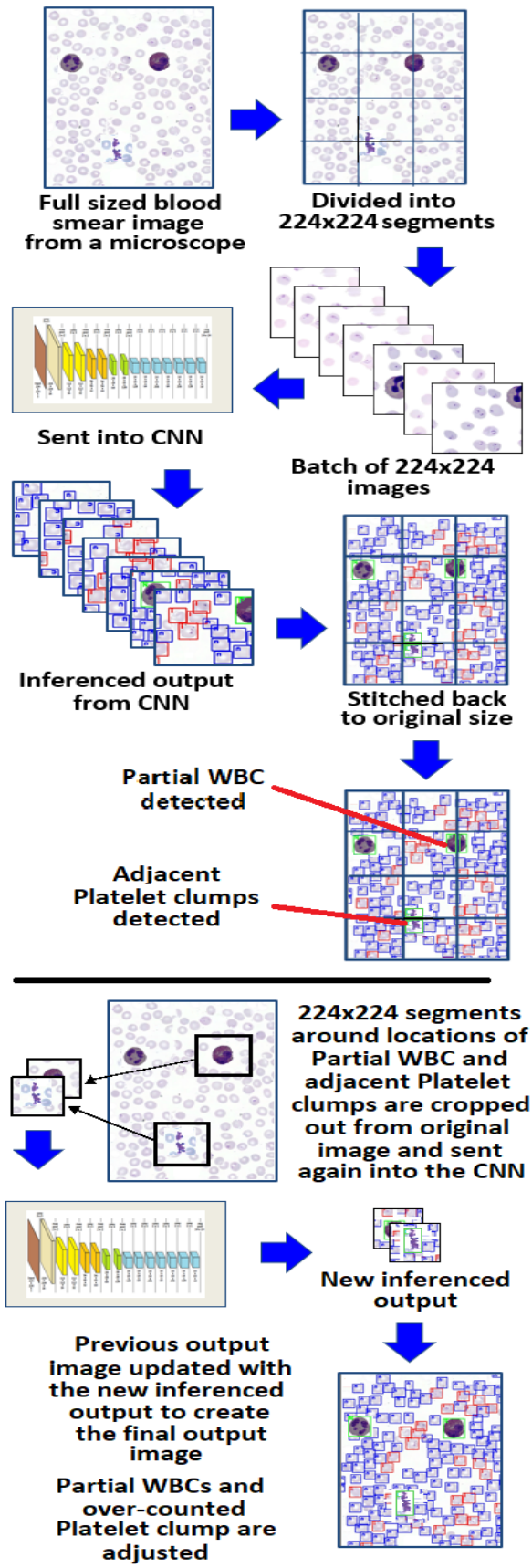

Fig. 6. Flowchart showing how the inference will be calculated on a complete blood smear image and how the case of Partial WBC and partially visible Platelet clump or THR will be handled.

images only half of the cell is visible and in some others only one quarter may be visible. These WBCs are classified as a separate class called **Partial WBC** and the type of WBC is not specified. After stitching back the batch of output images from the CNN, if any Partial WBC appears in it, then an inference is carried out for a second time on a 224×224 square around the location of the Partial WBC in the original image. This time the WBC is fully included in the image and hence a proper detection result is obtained. The Partial WBC is then replaced with the new result to improve detection accuracy. No image is created where THR and Platelet clumps are partially visible because their partial and complete form has similar visual features and they can be classified correctly regardless how much is visible. However, if any THR or Platelet clump becomes partially visible in a 224×224 image, then there will be two of these objects detected adjacent to each other when the images are stitched together for final detection. Thus, to prevent over-counting, the same approach of a second inference around these adjacent objects is taken, just like the case of Partial WBCs. A flowchart of this entire process is shown in Fig. 6

In total, there are 10 classes in the entire dataset. Their names (along with short-forms) are: **(1) Eosinophil (Eos), (2) Basophil (Bas), (3) Neutrophil (Neu), (4) Lymphocyte (Lym), (5) Monocyte (Mon), (6) Partial WBC (pWBC), (7) Thrombocytes (THR), (8) Platelet Clumps (Plc), (9) Infected RBC (iRBC), and (10) Healthy RBC (hRBC)**.

E. Final Overview of the Created Dataset

The 224×224 images consist of a wide variety of combinations of different cells. The number of images for each kind of combination in the training set is given in Table II. All types of WBCs were used to create the Partial WBC class so the CNN does not get biased towards any particular type. The kind of WBC sample used to create a partial WBC is also mentioned in Table II. For example, pWBC(Bas) means that the partial WBC seen in that image is a Basophil. But in the final created dataset, all of them are referred to as *Partial WBC* in the ground truth label. The type of WBC is not specified. Many images have more than one kind of cell in them. For example, Eos-Plc means that image contains a Eosinophil and a Platelet clump.

In addition to the images listed in Table II, there are 4000 more images which contains only a mix of infected and healthy RBCs. The testing and validation datasets are approximately $1/10^{th}$ of the size of the training dataset, but they have the same distribution of image types. The total number of images in the final training, testing and validation sets are 65350, 6560, and 6560 respectively. The overall number of Basophil images are less than other types because the LISC database did not have any Basophil images. The Basophil detection still works well, as will be shown in the *Experimental Results* section.

III. CNN ARCHITECTURE AND TRAINING

A. CNN Model

A stripped down version of the Yolo neural network model given in [15] and [16] is considered for training. Yolo has

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2020.2967290, IEEE Robotics and Automation Letters

TABLE II
NUMBER OF DIFFERENT TYPES OF IMAGES IN THE TRAINING DATASET

Type(s) of blood cell in the image	Type of background RBC	
	Infected	Healthy
only RBC	4000	4000
only Bas	750	750
only Eos, only Neu, only Lym, only Mon, only THR, only Plc	1500 (of each type)	1500 (of each type)
Eos-Bas, Bas-Bas, Bas-Neu, Bas-Lym, Bas-Mon, Bas-THR, Bas-Plc	250 (of each type)	250 (of each type)
Eos-Eos, Eos-Neu, Eos-Lym, Eos-Mon, Eos-THR, Eos-Plc, Neu-Neu, Neu-Lym, Neu-Mon, Neu-THR, Neu-Plc, Lym-Lym, Lym-Mon, Lym-THR, Lym-Plc, Mon-Mon, Mon-THR, Mon-Plc, THR-THR, THR-Plc, THR-Plc,	500 (of each type)	500 (of each type)
Eos-pWBC(Bas), Bas-pWBC(Eos), Bas-pWBC(Bas), Bas-pWBC(Neu), Bas-pWBC(Lym), Bas-pWBC(Mon), Neu-pWBC(Bas), Lym-pWBC(Bas), Mon-pWBC(Bas), THR-pWBC(Bas), Plc-pWBC(Bas)	50 (of each type)	50 (of each type)
Eos-pWBC(Eos), Eos-pWBC(Lym), Eos-pWBC(Mon), Eos-pWBC(Neu), Neu-pWBC(Eos), Neu-pWBC(Neu), Neu-pWBC(Lym), Neu-pWBC(Mon), Lym-pWBC(Eos), Lym-pWBC(Neu), Lym-pWBC(Lym), Lym-pWBC(Mon), Mon-pWBC(Eos), Mon-pWBC(Neu), Mon-pWBC(Lym), Mon-pWBC(Mon), THR-pWBC(Eos), THR-pWBC(Neu), THR-pWBC(Lym), THR-pWBC(Mon), Plc-pWBC(Eos), Plc-pWBC(Neu), Plc-pWBC(Lym), Plc-pWBC(Mon)	100 (of each type)	100 (of each type)
pWBC(Bas)	150	150
pWBC(Eos), pWBC(Neu), pWBC(Lym), pWBC(Mon)	300 (of each type)	300 (of each type)
pWBC(Eos)-pWBC(Bas), pWBC(Bas)-pWBC(Bas), pWBC(Bas)-pWBC(Neu), pWBC(Bas)-pWBC(Lym), pWBC(Bas)-pWBC(Mon)	15 (of each type)	15 (of each type)
pWBC(Eos)-pWBC(Eos), pWBC(Eos)-pWBC(Neu), pWBC(Eos)-pWBC(Lym), pWBC(Eos)-pWBC(Mon), pWBC(Neu)-pWBC(Neu), pWBC(Neu)-pWBC(Lym), pWBC(Neu)-pWBC(Mon), pWBC(Lym)-pWBC(Lym), pWBC(Lym)-pWBC(Mon), pWBC(Mon)-pWBC(Mon)	30 (of each type)	30 (of each type)

been used for images detection with good detection accuracy, but the network is too large for our task. Since our eventual goal is implementation on an embedded computer, the overall model is reduced in size and the resulting architecture is shown in Table III. This model will be henceforth referred to as **ModifiedYolo**. All the convolution layers (conv) have a leaky Relu activation (with $\alpha = 0.1$) followed by a batch normalization layer at their output. These are not shown in the table. The last convolution layer opens into a global average pooling layer with a sigmoid activation, since an image may have several different types of cells which are not mutually exclusive. The network is first trained only for classifying the cells in the images. At this stage, it outputs a multi-hot vector of 10 elements, one for each class.

It took approximately 2 hrs to train the network for classification on an NVIDIA Tesla P100 GPU for 13 epochs to reach a validation accuracy of 97.16%. The batch size was 100 and initial learning rate was 0.001 with Adam optimizer which was reduced to 0.0001 after the 11^{th} epoch.

TABLE III
ARCHITECTURE OF MODIFIEDYOLO

Layer (Stride)	Filters	Input Shape	Output Shape
3×3 conv (1)	16	$224 \times 224 \times 3$	$224 \times 224 \times 16$
2×2 maxpool (2)	-	$224 \times 224 \times 16$	$112 \times 112 \times 16$
3×3 conv (1)	32	$112 \times 112 \times 16$	$112 \times 112 \times 32$
2×2 maxpool (2)	-	$112 \times 112 \times 32$	$56 \times 56 \times 32$
1×1 conv (1)	16	$56 \times 56 \times 32$	$56 \times 56 \times 16$
3×3 conv (1)	128	$56 \times 56 \times 16$	$56 \times 56 \times 128$
1×1 conv (1)	16	$56 \times 56 \times 128$	$56 \times 56 \times 16$
3×3 conv (1)	128	$56 \times 56 \times 16$	$56 \times 56 \times 128$
2×2 maxpool (2)	-	$56 \times 56 \times 128$	$28 \times 28 \times 128$
1×1 conv (1)	32	$28 \times 28 \times 128$	$28 \times 28 \times 32$
3×3 conv (1)	256	$28 \times 28 \times 32$	$28 \times 28 \times 256$
1×1 conv (1)	32	$28 \times 28 \times 256$	$28 \times 28 \times 32$
3×3 conv (1)	256	$28 \times 28 \times 32$	$28 \times 28 \times 256$
2×2 maxpool (2)	-	$28 \times 28 \times 256$	$14 \times 14 \times 256$
1×1 conv (1)	64	$14 \times 14 \times 256$	$14 \times 14 \times 64$
3×3 conv (1)	512	$14 \times 14 \times 64$	$14 \times 14 \times 512$
1×1 conv (1)	64	$14 \times 14 \times 512$	$14 \times 14 \times 64$
3×3 conv (1)	512	$14 \times 14 \times 64$	$14 \times 14 \times 512$
1×1 conv (1)	128	$14 \times 14 \times 512$	$14 \times 14 \times 128$
1×1 conv (1)	10	$14 \times 14 \times 128$	$14 \times 14 \times 10$
14×14 avgpool (1)	-	$14 \times 14 \times 10$	10
Last two layers are replaced by next four during detection training			
3×3 conv (1)	128	$14 \times 14 \times 128$	$14 \times 14 \times 128$
3×3 conv (1)	128	$14 \times 14 \times 128$	$14 \times 14 \times 128$
3×3 conv (1)	128	$14 \times 14 \times 128$	$14 \times 14 \times 128$
1×1 conv (1)	90	$14 \times 14 \times 128$	$14 \times 14 \times 90$

If the network was trained directly for the object detection, then the overall training error in bounding box coordinates might have overwhelmed the classification error and the network would have focused more on reducing the bounding box location error, thereby neglecting the classification error altogether. The bounding boxes would have been created correctly but the predicted class names might have been wrong.

B. Modification of the Model for Detection

After the classification training, the same ModifiedYolo model is trained for object detection to predict the location of the cells and create bounding boxes around them. In this stage, the last convolution layer is replaced by 4 other fresh convolution layers to make the network more powerful and to reshape the last layer into the required number of output elements needed.

The first layer of the network has a height and width of 224×224. This is reduced to 14×14 in the last few layers. Each of these 14×14 grid cells is a potential location to find an object. For each grid cell, 6 anchor boxes are predefined, whose dimensions are modified by the network to suit the shape of the detected objects. Each anchor box has 15 elements for object detection. Two of them are offsets for the x, y coordinates of the center of the bounding box. The next two are its height and width. The 5^{th} one is a confidence score indicating the probability of the presence of an object in that bounding box. The last 10 are a 10 element one-hot vector for the 10 classes. Therefore the last convolution layer has a depth of $6 \times (5 + 10) = 90$. These metrics are defined in details in [16]. The detection training results are given in the Experimental Results section. Non-maximum suppression (NMS) was used to remove redundant bounding boxes based on their intersection over union (IOU) score in the same manner as done in [15].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2020.2967290, IEEE Robotics and Automation Letters

C. Modification of the Loss Function

Initially, the same detection loss function used for the Yolo model [15] was used for this project. A simplified version of that is shown in (1):

$$\begin{aligned} loss \;=\; & \lambda_{coord} * (x_{err} + y_{err} + w_{err} + h_{err}) + \\ & C_{err} + \lambda_{noobj} * C'_{err} + Class_{err} \end{aligned} \tag{1}$$

$x_{err}, y_{err}, w_{err}, h_{err}$ are the errors between true and predicted values of the center x, y coordinates and width and height of the bounding box. C_{err} is the error in the confidence (that an object is present in the box) and C'_{err} is the error in the confidence (that an object is not present in the box). $Class_{err}$ is the classification error. λ_{coord} and λ_{noobj} are multiplying factors that tells the network how much importance should be emphasized on the corresponding errors. These parameters are explained in more detail in [15].

With this, the network was able to create the bounding boxes properly but the class name prediction was not very accurate. The overall mean average precision (mAP) obtained was 89%. Hence, to improve this, an extra parameter $\lambda_{\mathbf{class}}$, was included in the equation (1) so that the network puts more importance on the class name prediction as well. The modified function is shown in (2):

$$\begin{aligned} loss \;=\; & \lambda_{coord} * (x_{err} + y_{err} + w_{err} + h_{err}) + \\ & C_{err} + \lambda_{noobj} * C'_{err} + \lambda_{class} * Class_{err} \end{aligned} \tag{2}$$

The values of the parameters and size of the anchor boxes used in detection training are as follows:

- λ_{coord} = 5.0, λ_{noobj} = 0.5, λ_{class} = 1.5
- IOU threshold for NMS: 0.7
- Number of anchor boxes used: 6
- Anchor box sizes [W×H]: [3.1×3.1], [6.125×3.1], [3.1×6.125], [6.125×6.125], [5.32×5.32], [2.7×2.7]

D. Optimization of the Model Architecture

After getting a good detection result from the ModifiedYolo model, the CNN architecture is modified several times to finally come up with a smaller and more optimized structure without compromising too much on the detection performance. The architecture of this model is described in Table IV. The same datasets, GPUs, and loss functions are used to train this model as well. It took around 2.5 hrs to complete the classification training. After 31 epochs a validation accuracy of 97.07% is reached. A batch size of 100 and the initial learning rate of 0.001 was used for all the epochs. During detection training, the last convolution layer is replaced by 2 other fresh convolution layers. The results of this detection are given in *Experimental Results* section. The same parameter values and NMS algorithm are used here as well, as done for the ModifiedYolo.

E. Modified Resnet-18 Architecture

To compare the performances of the ModifiedYolo and optimized CNN with other network architectures, the Resnet-18 model is chosen [5]. This network was selected since it has a different structure than the ModifiedYolo and optimized CNN network models due to its skip connections and strided convolutions instead of max-pool layers. The Resnet-18 network is also trained in two stages. The classification training achieved an accuracy of 94.68% after 11 epochs with a batch size of 100. The learning rate was set at 0.001 and then reduced to 0.0001 in the last epoch. A few modifications were done to the original Resnet-18 architecture given in [5] to fit this particular task:

- The max-pool layer at the beginning and fully connected layer in the end were removed.
- Height and width of all the feature maps in the residual blocks were made twice the original size. Hence, the smallest feature map size is now 14×14.
- The same loss function is used for detection training as used in ModifiedYolo.
- During detection training, the final average-pool layer is replaced with a 3×3 convolution layer with depth of 128, followed by a 1×1 convolution layer with depth of 90 for predicting bounding box parameters properly.

The detection performance of Resnet-18 is given in the *Experimental Results* section.

TABLE IV
OPTIMIZED CNN ARCHITECTURE

Layer (Stride)	Filters	Input Shape	Output Shape
3×3 conv (1)	16	$224 \times 224 \times 3$	$224 \times 224 \times 16$
2×2 maxpool (2)	-	$224 \times 224 \times 16$	$112 \times 112 \times 16$
3×3 conv (1)	32	$112 \times 112 \times 16$	$112 \times 112 \times 32$
2×2 maxpool (2)	-	$112 \times 112 \times 32$	$56 \times 56 \times 32$
3×3 conv (1)	64	$56 \times 56 \times 32$	$56 \times 56 \times 64$
2×2 maxpool (2)	-	$56 \times 56 \times 64$	$28 \times 28 \times 64$
3×3 conv (1)	128	$28 \times 28 \times 64$	$28 \times 28 \times 128$
2×2 maxpool (2)	-	$28 \times 28 \times 128$	$14 \times 14 \times 128$
1×1 conv (1)	64	$14 \times 14 \times 128$	$14 \times 14 \times 64$
3×3 conv (1)	256	$14 \times 14 \times 32$	$14 \times 14 \times 256$
1×1 conv (1)	64	$14 \times 14 \times 256$	$14 \times 14 \times 64$
3×3 conv (1)	256	$14 \times 14 \times 32$	$14 \times 14 \times 256$
1×1 conv (1)	128	$14 \times 14 \times 256$	$14 \times 14 \times 128$
1×1 conv (1)	10	$14 \times 14 \times 128$	$14 \times 14 \times 10$
14×14 avgpool (1)	-	$14 \times 14 \times 10$	10
Last two layers are replaced by next two during detection training			
3×3 conv (1)	128	$14 \times 14 \times 128$	$14 \times 14 \times 128$
1×1 conv (1)	90	$14 \times 14 \times 128$	$14 \times 14 \times 90$

IV. EXPERIMENTAL RESULTS

The experimental results presented here are compared to the ground truth data of the testing and validation datasets that is known to be 100% accurate. It is assumed that a human pathologist is also 100% accurate. Therefore, the results reported here can also be considered as a comparison with the performance of a human pathologist.

A. Detection Performance

Detection training of the ModifiedYolo network took about 4 hrs on the NVIDIA Tesla P100 GPU for 29 epochs with a learning rate of 0.001 for the first 26 epochs and then changed to 0.0001 for the last 3. The optimized CNN took about 6 hrs and 55 epochs to get trained. The learning rate was kept at 0.001 for the first 52 epochs and then reduced by $1/10^{th}$ in each of the next 3 epochs. Resnet-18 was trained for 10 hrs

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2020.2967290, IEEE Robotics and Automation Letters

and 38 epochs with learning rates of 0.001 and 0.0001. But performance did not improve after 12 epochs.

The overall mean average precision (mAP), which is the most popular metric for determining the detection performance of a CNN, is calculated over the testing dataset for both the networks, as shown in Table V along with the performance of Resnet-18 for comparison.

TABLE V
AVERAGE PRECISION ON TESTING DATASET

Class Name	ModifiedYolo	Optimized CNN	Resnet-18
Eosinophil	0.97773	0.96782	0.94524
Basophil	0.96333	0.97833	0.92000
Neutrophil	0.98878	0.99562	0.96397
Lymphocytes	0.88194	0.87365	0.87875
Monocytes	0.98626	0.98765	0.93619
Partial WBC	0.97947	0.98008	0.97682
Thrombocyte	0.95270	0.94641	0.95926
Platelet Clumps	0.86686	0.83572	0.85214
Infected RBC	0.98091	0.97760	0.97849
Healthy RBC	0.98162	0.98045	0.98202
MEAN (mAP)	**0.95596**	**0.95233**	**0.93929**

For Malaria detection, the true detection and false detection rates are more important than mAP because the detection of pathogen in even a few RBC cells is sufficient to infer that the blood is infected. Hence, these metrics are also calculated for all the images of testing dataset. It is checked if both the CNNs are able to find the trace of malaria in the images which contain infected RBCs. The resulting true and false detection rates as shown in Table VI along with the performance of Resnet-18 for comparison.

TABLE VI
TRUE AND FALSE DETECTION MATRICES OF IMAGES WITH INFECTED RBCS FROM TESTING DATASET

	ModifiedYolo		Optimized CNN		Resnet-18	
Testing Dataset	**Predicted as Infected**	**Predicted as Healthy**	**Predicted as Infected**	**Predicted as Healthy**	**Predicted as Infected**	**Predicted as Healthy**
Truly Infected	100.00 %	0.00 %	100.00 %	0.00 %	100.00 %	0.00 %
Truly Healthy	2.53 %	97.47 %	4.48 %	95.52 %	2.08 %	97.92 %

Table VI indicates that out of all the images in the testing dataset containing infected RBCs, the ModifiedYolo, the optimized CNN, and the Resnet-18 all have detected infections correctly, with true detection rates of 100% for each of them. Out of all the images with healthy RBCs, the ModifiedYolo and Resnet-18 have falsely predicted 2.53% and 2.08% to be infected, respectively (i.e., false detection rates). Similarly, the false detection rate for the optimized CNN is 4.48%. The mAP for the Modified Yolo and optimized CNNs are very close in magnitude, with both of them better than the mAP produce with Resnet-18. Therefore, even though the optimized CNN is significantly smaller than the ModifiedYolo and Resnet-18 networks, it's performance is on par or better than the performance of each of them.

Some examples of the cell detection are shown in Fig. 7. The inference on the overall big images which is done by segmenting it into 224×224 parts and then stitching those back after doing inference on them, is also shown. The big image is actually 1000×1000 in size, but it has been reduced to fit in the paper. The total number of cells detected and the types are also listed out by the algorithm which is used to obtain a complete blood cell count in the blood smear.

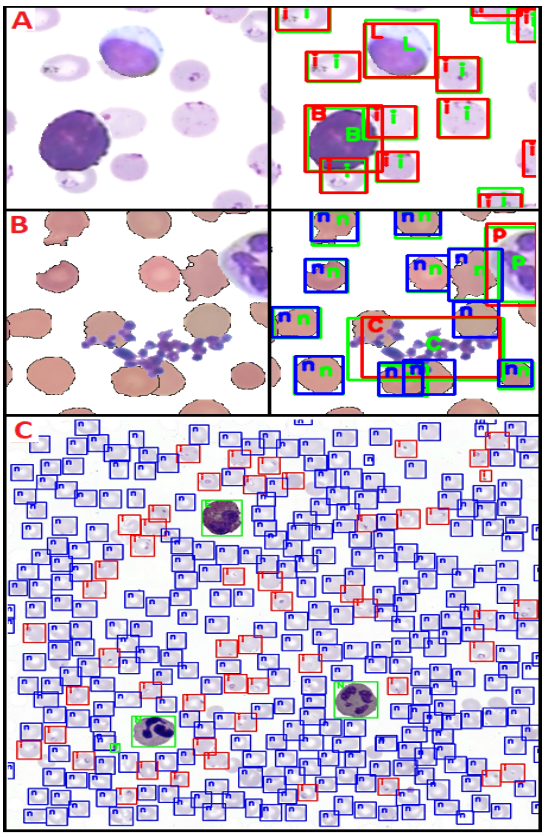

Fig. 7. Detection of different cells. Ground truth bounding boxes are in Green and Predicted boxes are in Red and Blue. A: (Left) Basophil and Lymphocyte on infected RBC background. (Right) Detection result. i indicates infected RBCs. B: (Left) Platelet clump and a Partial WBC on healthy RBC background. (Right) Detection result. n indicates healthy RBCs. P indicates Partial WBC. C: Prediction result on a 1000×1000 image with an Eosinophil, two Neutrophils, infected and healthy RBCs.

B. "Real" Blood Smear Image Results

The overall experimental results shows that both networks have good performances on the testing dataset whose cell samples and images were completely unknown to the networks during training. However, it can be argued that the arrangements of the cells in the images of the automatically synthesized datasets may be different from those in a real blood smear image. Therefore, the performances of the networks were tested on an additional set of images that closely resemble actual blood smear images that a pathologist may encounter. As mentioned earlier in Section II, an annotated database of real blood smear images with all the different kinds of cells (including the malaria infected RBCs) is not available. The most comprehensive databases with real blood smear images are the LISC database, having annotated WBCs, and the MAMIC database, which contains THR, platelet clumps, infected, and healthy RBCs. The MAMIC images do not have a good collection of WBCs. Therefore, a test-set of images

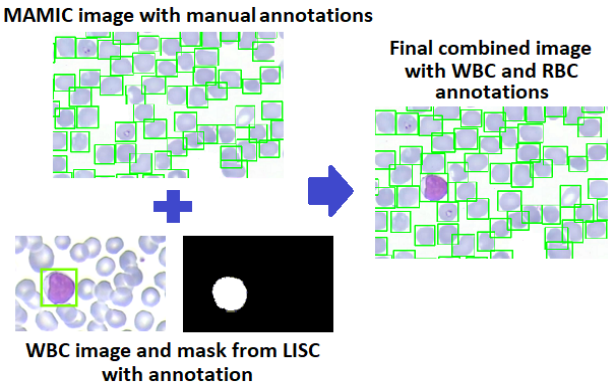

Fig. 8. Manually annotated real blood smear MAMIC images and annotated WBC images of LISC are combined to make blood smear image with annotated cells of all kinds. Ground truth bounding boxes are shown in green.

was created by manually annotating the cells in some MAMIC blood smear images and WBCs extracted from the LISC images using the appropriate masks are added to them. Care has been taken so that the RBC background of the original LISC image closely matches that of the MAMIC image it is combined with to ensure that the final image has as little distortion as possible. Fig. 8 illustrates this process. None of the MAMIC images used to create this new test-set were used to create the synthetic datasets and all have manually annotated ground truth bounding boxes for each type of cell in the image. The WBC sample used were also unseen to the networks during training. Similar to the results in Table V, the mAP for detecting all classes of cells reported by the ModifiedYolo and optimized CNN on these "real" blood smear images are **0.93656** and **0.92776**, respectively.

C. Hardware Implementation

One of the main objectives of this project is a prove that a low cost, compact system can be created using off the shelf hardware for rapid medical diagnosis. Hence, the overall software was implemented on a *Raspberry Pi 3* single board microcomputer without any change in the algorithm or the python program scripts. The only difference was in the inference time. The approximate inference time on a 224×224 image, for an Intel core i5 processor (in a laptop) and the Raspberry Pi 3 are shown as follows:

- ModifiedYolo on Core i5: 4 sec.
- ModifiedYolo on Raspberry Pi 3: 14 sec.
- Optimized CNN on Core i5: 2.9 sec.
- Optimized CNN on Raspberry Pi 3: 11 sec.

V. CONCLUSION

In this paper, we presented a low cost digital pathology system capable of detecting Malaria, providing a CBC, and WBC count from a blood smear image. The overall contributions and future work are discussed as follows.

A. Contributions

- Created a dataset of blood smear images with a wide variation in the combination of blood cells.
- Complete blood cell count, distribution of WBCs, and Malaria detection, all done in the same system.
- The case of partially visible WBCs is handled separately to improve the accuracy of detection.
- Performance of the system is also tested on a compact off-the-shelf embedded platform like Raspberry Pi 3.
- Overall system is cost-effective and has a performance relatively close to a human pathologist. So, its use may enable cheaper and faster treatment of patients.

B. Future Work

Using proper datasets, this work can be further extended to identify more types of infections in blood. With sufficient data, the different stages of Malarial infection can also be estimated. An overall self contained hardware can also be created comprising of a digital microscope connected to a Raspberry Pi 3 and a blood sample collecting device. A different version of this approach can also be used to find fractures and defects in bone structures from X-ray images.

REFERENCES

[1] "Cellavision.com." CellaVision, 2019. [Online]. Available: https://www.cellavision.com/
[2] "Visionhema.com." Vision Hema, 2019. [Online]. Available: http://visionhema.com/
[3] "medicacorp.com." Sysmex, 2019. [Online]. Available: http://www.medicacorp.com/products/hematology-imaging-analyzers/
[4] A. Shahin, Y. Guo, K. M. Amin, and A. A. Sharawi, "White blood cells identification system based on convolutional deep neural learning networks," *Computer methods and programs in biomedicine*, 2017.
[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[6] S. N. T. Persson, "White blood cell differential counting in blood smears via tiny yolo."
[7] Y. M. Alomari, S. Abdullah, S. N. Huda, R. Zaharatul Azma, and K. Omar, "Automatic detection and quantification of wbcs and rbcs using iterative structured circle detection algorithm," *Computational and mathematical methods in medicine*, vol. 2014, 2014.
[8] K. Prabakaran, I. Khan, M. B. Abrar, . MohammedAbbas, and S. Khurshid, "A smart sensing and quantification of platelets , red blood cells (rbc) , white blood cells (wbc) and classification of wbc ' s using microscopic blood image," 2015.
[9] D. Bibin, M. S. Nair, and P. Punitha, "Malaria parasite detection from peripheral blood smear images using deep belief networks," *IEEE Access*, vol. 5, pp. 9099–9108, 2017.
[10] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
[11] S. H. Rezatofighi, K. Khaksari, and H. Soltanian-Zadeh, "Automatic recognition of five types of white blood cells in peripheral blood," in *International Conference Image Analysis and Recognition*. Springer, 2010, pp. 161–172.
[12] O. Sarrafzadeh, H. Rabbani, A. Talebi, and H. U. Banaem, "Selection of the best features for leukocytes classification in blood smear microscopic images," in *Medical Imaging 2014: Digital Pathology*, vol. 9041, 2014, p. 90410P.
[13] "The mamic image database." WebMicroscope, 2019. [Online]. Available: http://fimm.webmicroscope.net/Research/Momic/mamic
[14] "Malaria cell images dataset." Kaggle, 2019. [Online]. Available: https://www.kaggle.com/iarunava/cell-images-for-detecting-malaria
[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
[16] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.