Project Report: Malaria Parasite Detection

Ajay Ramesh
IMT2017502

Amogh Johri
IMT2017003

Arjun Verma
IMT2017008

***Abstract*—Malaria is a lethal disease caused by a parasite known as Plasmodium. The current common technique of medical diagnosis for the disease includes manually identifying and counting the parasitized cells by performing light microscopy of thick and thin stained blood smears. Since this procedure is carried out manually, there is always a scope of misdiagnosis either due to poor familiarity with the procedure by a diagnostician or problems in accessing and acquiring the proper equipments required for diagnosis. In this project, we explore different techniques for the automation and unsupervised detection of malarial parasites.**

I. Introduction

Malaria is an infectious disease and causes serious health problems. Half of the world's population, particularly in the developing countries is at risk of malaria. According to the World Health Organization (WHO), malaria causes approximately nearly million deaths and over 250 million infections every year and is caused by parasites of the genus Plasmodium, of which Plasmodium falciparum contributes 98% of deaths. The diagnosis of the infections due to P. falciparum is still carried out via manual procedures especially in developing countries. Although there are advanced methods of diagnosis, manual microscopy of blood films on slides is still considered to be the gold standard. Manual microscopy has advantage over other techniques in that it is both sensitive and specific. One of the disadvantages of diagnosis using manual microscopy methods is that it requires extensive human intervention during the diagnostic process which can often lead to late and sometimes erroneous diagnosis. The microscopist requires extensive training to gain expertise in the diagnosis, and because of the sheer volume of the samples that need to be analysed, the method is not consistent and is dependent upon blood smear and stain quality, microscope quality and the expertise of the microscopist. Some of the problems of manual microscopy can be overcome by exploring computer based, specifically image-based, diagnostic methods. This project aims at providing a diagnosis method based on image processing and one that provides a reliable and consistent solution. Considering the high fatality rate and huge volumes of samples that need to be analysed we need a sensitive, practical and robust method with minimum human intervention. In this context, computer based diagnosis can help in the rapid, accurate and consistent identification of true malaria cases, ensuring that only those patients with malaria are treated.

The primary objective of this project was to apply the machine learning techniques comprehended in our course to the problem of detecting parasitized cells in thin-blood smear images. As the problem statement suggested, we have tackled the problem using two different methods:-

- Traditional image processing techniques

  These techniques include solely using various python libraries to extract features deemed important by eyeballing the dataset provided to us. Different permutations of the various features extracted were then combined and tested to check which concatenation of features gave best results.
- The CNN approach

  CNN's are primarily the state-of-the-art approach to majority of tasks in the domain of image recognition. We have used CNN for the end-to-end classification task at hand. An image is passed through a series of convolutional, nonlinear, pooling layers and fully connected layers to generate more abstract concepts about the image.

The document elaborates our efforts on the above chosen methodologies and then proposes the best models that we came up with using the two methodologies based on their performace on the test dataset provided.

II. Dataset

The dataset consists of images of segmented cells from the thin blood smear slide from the Malaria Screener research activity. To reduce the burden for microscopists in resource-constrained regions and improve diagnostic accuracy, researchers at the Lister Hill National Center for Biomedical Communications (LHNCBC), part of National Library of Medicine (NLM), have developed a mobile application that runs on a standard Android smartphone attached to a conventional light microscope. Giemsa-stained thin blood smear slides from 150 P. falciparum-infected and 50 healthy patients were collected and photographed at Chittagong Medical College Hospital, Bangladesh. The smartphone's built-in camera acquired images of slides for each microscopic field of view. The images were manually annotated by an expert slide reader at the Mahidol-Oxford Tropical Medicine Research Unit in Bangkok, Thailand. The de-identified images and annotations are archived at NLM (IRB12972). They then applied a level-set based algorithm to detect and segment the red blood cells. The dataset contains a total of 27,558 cell images with equal

instances of parasitized and uninfected cells. An instance of how the patient-ID is encoded into the cell name is shown herewith: “P1” denotes the patient-ID for the cell labeled “C33P1thinF_IMG_20150619_114756a_cell_179.png”.

III. Methodology

A. Traditional Approach

We have primarily focused on extracting 3 set of features:

- Blob features
- Contour features
- Bag of Visual Words model

Each methodology is associated with it’s own unique data pre-processing and extraction techniques.

1) Blob Detection: The primary (and probably the most intuitive) approach towards finding characteristic differences between the Uninfected and Parasitized red-blood cell images was the presence of a blob/blotch in the latter. This is due to the presence of trophozoite in the blood cell [REFERENCE - Poostchi M, Silamut K, Maude RJ, Jaeger S, Thoma G. Image analysis and machine learning for detecting malaria. Transl Res. 2018;194:36–55. doi:10.1016/j.trsl.2017.12.004]. Hence, the most intuitive approach towards finding these was to look for blobs in the cell. Laplacian of Gaussian(LoG), Difference of Gaussian(DoG) and Determinant of Hessian(DoH) were applied however, the results obtained were below-par. Finally, the SimpleBlobDetector, a class provided by OpenCV was used which seemed to give good results. The goodness of results were based on two criterias :
1) Number of parasitized images in which a blob was detected
2) Ratio of the parasitized images to uninfected images in which the blob was detected
The SimpleBlobDetector Class by OpenCV allows to detect/not-detect blobs on the basis of a number of parameters. These parameters include filtering by color, area, circularity, convexity, ratio of minimum inertia to maximum inertia, etc. Through a number of systematic observations, two sets of ideal parameters were finalized.
1) One set provided for the best ratio of detecting blob(s) in parasitized images to uninfected images. It detected a blob in 8295/11024 parasitized images and 80/11024 uninfected images. 2) The second provided for a good accuracy in detecting blobs for parasitized images while giving a fairly decent performance in terms of uninfected images. It detected a blob in 9567/11024 parasitized images and 649/11024 uninfected images.
Our initial idea involved using the 1^{st} set of parameters in order to use it as a threshold. For the first set of parameters,

$$P(image\ being\ parasitized \mid detection\ of\ blob\ ) = 0.99044 \quad (1)$$

The probability was very optimistic and hence, our initial attempt was to use it as a threshold while adding with classification techniques for the images in which no blob was detected. The second set of parameters were used to model a feature-vector which contained of the following 4 fields :
1) Number of blobs detected
2) Area of blob1 (0 if no blob detected)
3) Area of blob2 (0 if no blob detected)
3) Area of blob3 (0 if no blob detected)
This was used in addition to a number of other features to train the classification model.

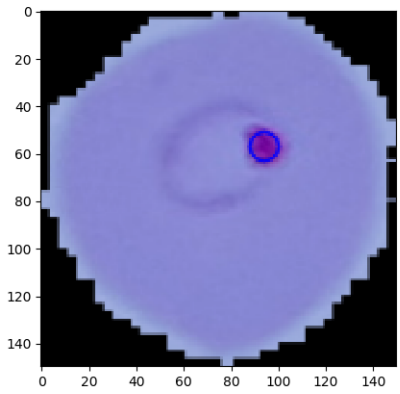

Fig. 1. Parasitized Red-Blood Cell with the trophozoit detected as a blob

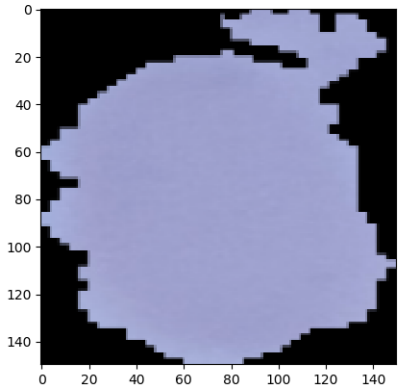

Fig. 2. Uninfected Red-Blood Cell

In order to improve upon results, a number of other techniques were applied:

Morphological Transformation
The images were dilated after passing them through 4x4 ellipsoid filter, before attempting for the blob detection. A number of other filters were also tried however, this seemed to work the best. For the second set of parameters, it brought down the number of parasitized images to 9542, however also bringing down the number of uninfected images to 259, in which blob(s) were detected. The following figures represent the same. The same image where a blob was being detected

before the transformation, gave no detection after being convoluted with the 4x4 ellipsoid filter followed by dilation.

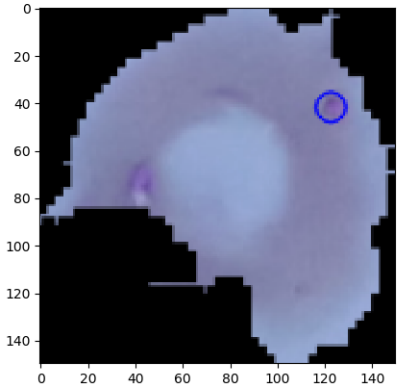

Fig. 3. Uninfected Red-Blood Cell (no filter)

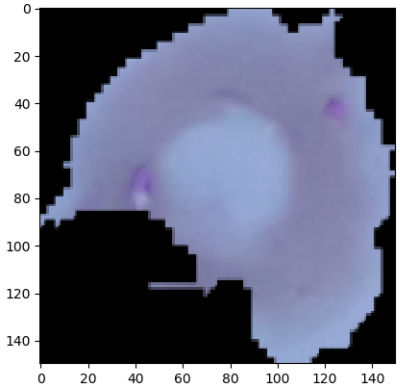

Fig. 4. Uninfected Red-Blood Cell (2x2 Ellipsoid Dilating filter)

Gamma Correction

This is a popular technique to tone both the brightness and the contrast for the image which affects (majorly) the midtones in an image. This was used with the 2nd set of parameters as even though lowering gamma increased the number of images (parasitized and uninfected) to be detected for blobs, it increased the number of parasitized images by a good margin over the uninfected images.

Other Methods

Methods such as gaussian blurring, change of color scheme (Grayscale, RGB, HSV and LAB), etc were also incorporated however, the best results were found on a combination of dilation and gamma-correction over BGR images.

The exact parameters, etc have been clearly shown in the jupyter notebook.

2) Bag of Visual Words Model: The Bag of Visual Words model is an important concept in computer vision and is commonly used in image classification. The method has been adapted from the Bag of Words technique of Text processing, where the frequency of each word is computed, and a set of keywords are identified. In the case of images, image 'features' are used instead of words. These image features comprise of various unique patterns that can be identified from the image. The final idea of the Bag of Visual Words model is to represent every image with a set of features which can then be used for image classification. The features consists of two parts -

1) Keypoints - They are the unique stand-out points of the image. The keypoints can be detected using various algorithms such as SIFT, SURF, FAST etc. The detected keypoints for the image of an infected blood sample using the FAST algorithm is shown in Figure 5. A sufficient number of keypoints can be detected in the infected area. Since the keypoints are stand-out points of an image, they do not change even if the image is rotated, shrunk or expanded.

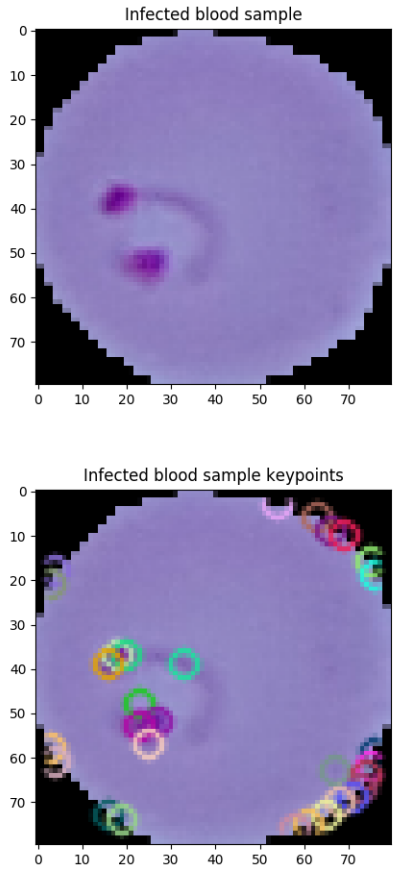

Fig. 5. (a) Infected blood sample, (b) Keypoints of the infected blood sample

2) Descriptors - They are a description of the keypoint and can be used to interpret the difference or similarity between various keypoints.

The keypoints and descriptors together were used to construct vocabularies. The vocabulary for an image represents each image as a frequency histogram of the features of the image. These vocabularies were then used as feature vectors for classification. The vocabulary was constructed by clustering the keypoints and using the centre of these clusters as the vocabularies. The clustering was performed using the K-Means algorithm with 16 means. The vocabulary obtained from the infected sample is shown in Figure 6. The FAST algorithm was preferred for keypoint detection instead of SIFT because of its faster computation. The LUCID algorithm was used for extracting the descriptors for the keypoints. The LUCID algorithm is capable of efficiently generating descriptors for colored images.

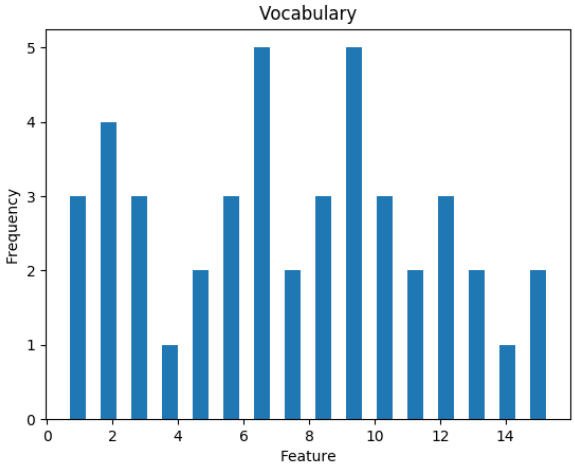

Fig. 6. Vocabulary

3) Contour Detection: The next set of features that we focused upon was trying to mark contours around blobs/blotches in the infected cells. Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. Contours are a useful tool for shape analysis and object detection and recognition. After marking the contours, the number of contours and the area of contours were the set of features that we extracted. The top five areas among the varying contours were taken along with the total number of contours formed as the contour features. Applying a Gaussian Blur over images and converting the the images to a grayscale image were the data pre-processing techniques used to obtain the area features. Gaussian Blur was applied as a noise reduction technique. In Gaussian Blur operation, the image is convolved with a Gaussian filter. The Gaussian filter is a low-pass filter that removes the high-frequency components. A 5x5 kernel with a standard deviation in X direction as 2 gave us the best results. The image was converted to a grayscale to perform image thresholding

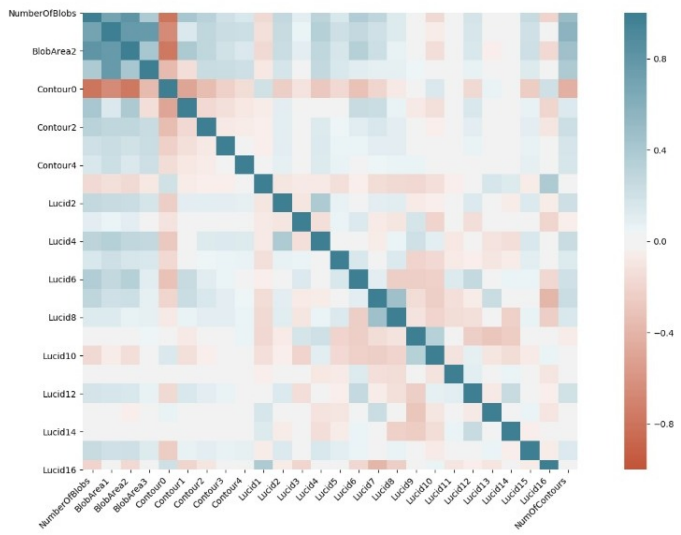

Fig. 7. hmap

on it. Image thresholding is a simple, yet effective, way of partitioning an image into a foreground and background. This image analysis technique is a type of image segmentation that isolates objects by converting grayscale images into binary images. Image thresholding is most effective in images with high levels of contrast. We set 127 as the threshold value which is used classify the pixel values. If the pixel value is more than this, it is replaced by 255. Before using findContours() to obtain the contours, one last thing to apply is the Canny() function. Canny is an algorithm made for edge detection. This is the base algorithm for any line ,edge or contour detection for getting a higher accuracy. We then use the findContours() and contourArea() functions to obtain the contours and their areas.

B. Convolution Neural Networks

Convolution Neural Networks (CNN) were employed to perform end-to-end classification of the images. The ever-increasing computational power in the modern times has resulted in a surreal growth in the domain of Deep Learning. Especially in the domain of Image processing, since AlexNet, CNNs have dominated the majority of the realm, providing state-of-the-art performances for a wide range of problem statements. Once such area where CNNs have been immensely successful is Image Classification.
We have utilized a CNN model for an end-to-end classification of the images of red-blood cells. CNNs, although immensely successful, lack explain-ability for their behavior hence, tasks such as hyperparameter tweaking, choosing the architecture, etc can at best be based upon heuristics and prior research literature.

1) Architecture: The general strategy followed in order to build the model was that, we start with small number of activation maps corresponding to a kernel in the shallow

Fig. 8. Ensemble Model Architecture

layers, and keep doubling the size as we move forward. As even attempting an exhaustive number of such combinations is far from possible, the number of activation maps were created were in the powers of 2. A number of different values were tried before arriving at the present architecture. The current architecture has been described in the figure below. A uniform kernel size was used for all the filters i.e. 2x2 and for all the convolution layers the inputs were padded just enough to maintain the size. Hence, all the activation maps originating from all convolutions in the network have the same spatial dimensions as that of the input. The Rectified Linear-Unit (ReLU) activation function was used for all the layers alike, except for the output layer where softmax activation function was used in order to get the output in terms of probabilities. Pooling was also done with the same size (2x2) throughout the architecture. The loss measure was chosen to be cross-entropy and adam optimizer was used. As shown in Figure 12, the accuracy of a number of models was in the and around the same range. This accuracy was calculated on fitting the model over a 80-20 split over 3 epochs and the validation accuracy was averaged over 10 such instances. The model chosen as the final one gave a validation accuracy of 96.04%. Even though a number of other models gave accuracy in the same range, we chose this model as it was simpler than most ("if having to make a choice between two hypothesis with similar outcomes, chose the one which is simpler"). The table describes the architecture and performances of all these models. There were max pooling layers (as described above) used between every 2 (or 4 layers for when the image dimensions became too small to pool) layers.

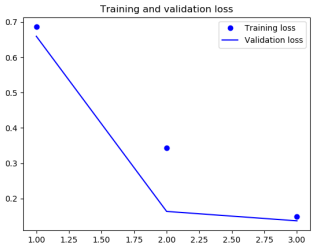

Fig. 9. Training Loss vs Validation Loss for an 80-20 split over 3 epochs

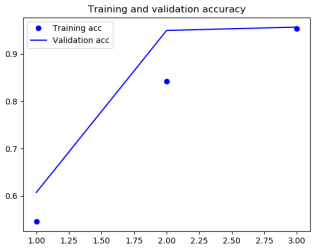

Fig. 10. Training Accuracy vs Validation Accuracy for an 80-20 split over 3 epochs

2) Data augmentation: A number of data-augmentation techniques were used in order to increase the number of data-points as well as for the model to generalize better. A parasitized image retains its property of being parasitized irrespective of its orientations. This allowed us to perform a number of rotation and orientation changing augmentations. The images were flipped horizontally and vertically and rotated at all possible right-angles. Since, the RAM soon became a bottleneck in this process, at a time, a single type of augmentation was performed and the model was trained. The trained models weights were then saved and in the next iteration, the weights were loaded and the model was retrained over the existing weights with a different type of augmentation. This process was carried out a number of times in order to avoid overfitting.

In addition, data augmentation also involved attempting changing of color spaces and morphological transformations. A number of different color spaces was searched such as Grayscale, LAB, HSV, RGB, out of which HSV seemed to work well. Apart from just working well, it seemed to capture features different from those which were being captured in BGR images. This was analyzed through checking for discrepancies between the models predictions for identical data-points which were represented in different color spaces.

3) Ensemble Model: The current architecture provided for validation accuracy in the range of 95.*%. As this stagnated around this value, we took to a new approach for creating ensemble models. Our motivation was loosely linked to the creation of random forests which utilizes a number of overfitted decision trees in order to make predictions. We took similar architectures and created a number of models. Each CNN model was then overfitted with one "type" of augmented image, for example, one CNN model was repeatedly trained over HSV images while one was trained repeatedly over BGR images, etc. The models were overfitted till their training accuracy reached in the ranges of 99.* %. Various color space transformations were utilized as well as gamma corrected images, gaussian blurred images, dilated images, etc were used as well. Once we had a number of overfitted CNN models, the predictions were run on the validation set using all these models simultaneously. Each of the model gave an accuracy in the range of 93.% on the validation set (when run in isolation). Hence, we could conclude that each model in itself was doing a fairly decent job in correctly classifying the images.

Now, this lead for us to making an assumption that the classifications which are incorrect would likely be incorrect only with a small probability gap (p(incorrect class) - p(correct class)) where as the points which get correctly identified must occur with a decent gap in their probabilities (p(correct class) - p(incorrect class)). Hence, we can expect this gap to cancel out if we sum up the probabilities through a number of different models, since if any one of the model classifies a certain data point into a given class with a good probability, since the individual performances of models are 93% +, such a classification is most likely correct. Using this logic, an ensemble of models were created and predictions were

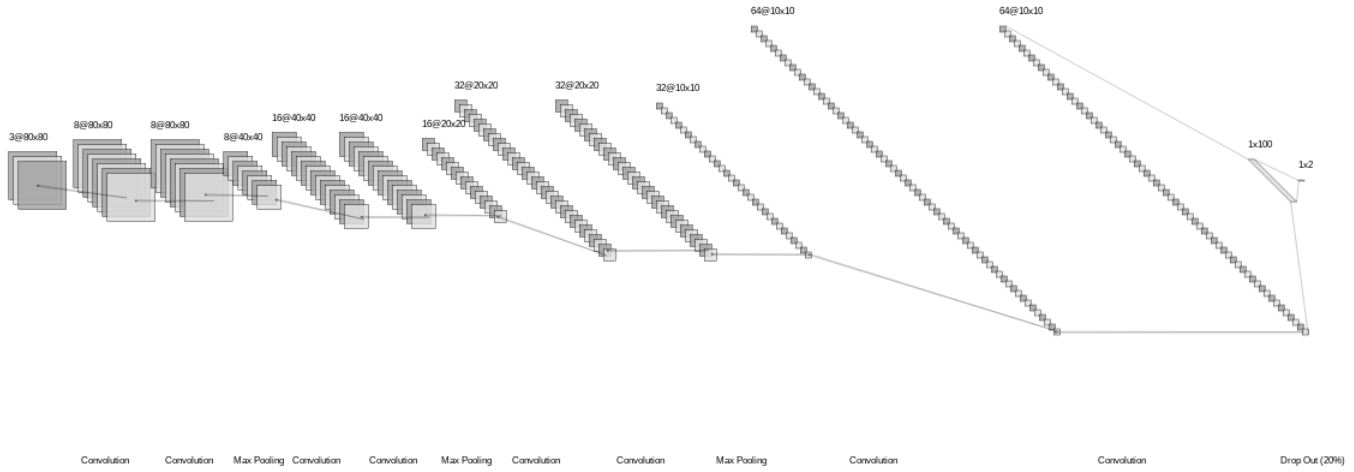

Fig. 11. Uninfected Red-Blood Cell (2x2 Ellipsoid Dilating filter)

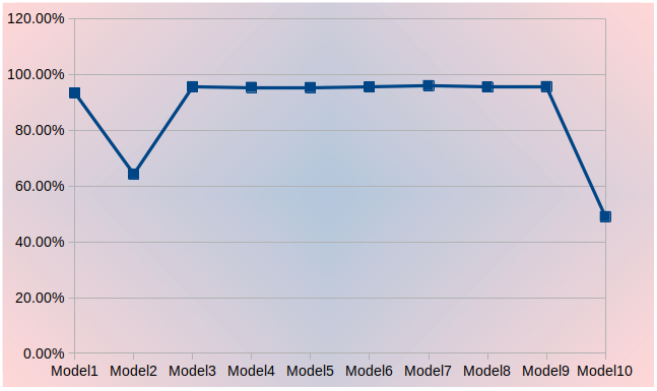

Fig. 12. Accuracy for different model architectures

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 80, 80, 8)	104
conv2d_2 (Conv2D)	(None, 80, 80, 8)	264
max_pooling2d_1 (MaxPooling2	(None, 40, 40, 8)	0
conv2d_3 (Conv2D)	(None, 40, 40, 16)	528
conv2d_4 (Conv2D)	(None, 40, 40, 16)	1040
max_pooling2d_2 (MaxPooling2	(None, 20, 20, 16)	0
conv2d_5 (Conv2D)	(None, 20, 20, 32)	2080
conv2d_6 (Conv2D)	(None, 20, 20, 32)	4128
max_pooling2d_3 (MaxPooling2	(None, 10, 10, 32)	0
conv2d_7 (Conv2D)	(None, 10, 10, 64)	8256
conv2d_8 (Conv2D)	(None, 10, 10, 64)	16448
dropout_1 (Dropout)	(None, 10, 10, 64)	0
flatten_1 (Flatten)	(None, 6400)	0
dense_1 (Dense)	(None, 100)	640100
dropout_2 (Dropout)	(None, 100)	0
dense_2 (Dense)	(None, 2)	202

Total params: 673,150
Trainable params: 673,150
Non-trainable params: 0

Fig. 13. Model Summary

carried out on the basis of summed-up probability scores. This increased the validation accuracy to 97.%. This was used for the final model that was built for our current project. It was an ensemble of 4 models:

1) Using non-transformed images in BGR color-space
2) Using non-transformed images in HSV color-space
3) Using various rotated and flipped versions of images in BGR color-space
4) Using various rotated and flipped versions of images in HSV color-space

The accuracy on the test-set for the current model was 96.84% over 5572 images.

4) Miscellaneous: a) We tried to use blob detection with the first set of parameters (which gave 99% true positives/false positives ratio for parasite classification) and tried to train the CNN on the remaining 3000 parasitized images (after increasing that number through data augmentation) and the 11000 uninfected images, where no blob was detected. This was thought of an attempt to pipeline the problems into two steps, if a blob is detected -¿ declare it as parasitized, else run the CNN model, give the output as the output of the CNN. However, this did not lead to massive increase in accuracy and was a very slow process (as blob detection is rather slow). To find what went wrong, we tried to compare the results of CNN to blob detection, exclusively on the images on which blobs were detected. Turned out that this number was very small (only 25 images out of 8000+ images where the blob was

Model Name	Architecture	Validation Accuracy
Model1	8x8x16x16	93.42%
Model2	16x16x32x32	64.32%
Model3	8x8x16x16x32x32	95.64%
Model4	16x16x32x32x64x64	95.41%
Model5	8x8x16x16x32x32x64	95.35%
Model6	8x16x16x32x32x64x64	95.62%
Model7	8x8x16x16x32x32x64x64	96.04%
Model8	8x8x16x16x32x32x64x64x128x128	95.46%
Model9	8x8x16x16x32x32x64x64x128x128x256x256	95.69%
Model10	8x8x16x16x32x32x64x64x128x128x256x256x512x512	49.04%

1.png

Fig. 14. Accuracy for different model architectures - Table

detected, was characterized by CNN wrongly). This leads us to believe that our architecture could have learned something similar to blob detection itself. We also attempted to mask the images where a blob was detected and send the masked images as input to the model, but this also did not give any better results, for what we believe are the same reasons.
b) The technique to overfit models based on one-type of data augmentation was tried for a number of augmentations (a number of different color spaces, morphological transformations, etc) however, these four gave the best results.
c) For normalization, every value in the image-tensor has been divided by 255. in order to get them in the range of [0,1] so that at no point should the values blow-up. In addition, two drop out layers have been added before the fully connected layers both with a dropping out of 20% of neurons. This is a widely used technique in CNN to increase the generalizability of the model.

IV. Results

The results using the traditional approach and CNN's have been shown seperately below. In the traditional approach, we tried different models to obtain RandomForest as the best classifier as it outperforms the rest of them. Among the others that were tried include logistic regression and SVM with a linear kernel. Their accuracies are as follows:

A. Traditional Approach

- Using RandomForest

Model	**RandomForest**
Accuracy	94.99092833

- Using Logistic Regression

Model	**Logistic Regression**
Accuracy	93.50771091

- Using SVM with Linear Kernel

Model	**SVM(Linear)**
Accuracy	92.69428485

B. Convolution Neural Network

Model	**CNN**
Accuracy	96.84

V. Conclusion

Malaria is a widespread disease that claims many lives all over the world. Automation of the diagnosis process can facilitate accurate diagnosis of the disease. We have described the workflow of classification of infected blood samples using both traditional image processing techniques as well as Convolution Neural Networks. The traditional image feature extraction techniques involved blob detection, contour feature extraction and the bag of visual words model. These techniques produced a modest 94.99% classification accuracy using the Random Forest classifier. The CNN, which is nowadays commonly used for image classification tasks was used to achieve a higher accuracy of 96.84%.

VI. Link to the model files

https://drive.google.com/open?id=
16qtnlChDxsOf0lFDcCKIoF_gnvm3DYus

VII. References

1) https://www.intechopen.com/books/machine-learning-advanced-techniques-and-emerging-applications/classification-of-malaria-infected-cells-using-deep-convolutional-neural-networks
2) https://lhncbc.nlm.nih.gov/system/files/pub9624.pdf
3) https://ieeexplore.ieee.org/document/7437798