Malaria Detection using Machine Learning

Harshit Goyal
harshit20203@iiitd.ac.in

Madhava Krishna
madhava20217@iiitd.ac.in

Shreya Bhatia
shreya20542@iiitd.ac.in

Srishti Singh
srishti20409@iiitd.ac.in

Abstract

Malaria is a life-threatening spread by infected Anopheles mosquito bites. Existing means of diagnosis include light microscopy and rapid diagnostic tests, which are used in conjuction to provide accurate results. However, the costs associated with them, in terms of human capital and time required, are immense.

We seek to provide a complementing approach to infection classification using machine learning, which is fast and inexpensive. We examine the performance of algorithms like logistic regression, boosted decision trees, support vector machines and convolutional neural networks on cellular images, and the effect of using image transformation and data augmentation approaches.

Finally, we provide a GUI-based tool with a CNN model at its core to predict whether a given cell is parsitized or not.

The GitHub repository containing the code can be found here. The Google Drive link contains the major code files.

1. Introduction

Malaria is an infectious disease caused by 5 species of the Plasmodium parasite: *Plasmodium falciparum*, *Plasmodium vivax*, *Plasmodium malariae*, *Plasmodium ovale* and *Plasmodium knowlesi*, spread by bites of the Anopheles mosquito. An estimated 241 million infections and 627,000 deaths occurred in 2020-21 [1].

1.1. Testing Methods

The infection can be detected using microscopy tests, Rapid Diagnostic Tests (RDTs) and serological tests.[2]

Microscopy tests involve collecting and dyeing a thin or thick blood specimen with Giemsa or Wright's stain to detect infections visually and ascertain the percentage of infected to uninfected cells.

RDTs indicate whether the patient is infected with one of the species of the malaria-causing *Plasmodium* and provide results in about 15 minutes. However, they fail to indicate a premature infection and negative RDT results need further evaluation. Using microscopy is also advised with positive results, so that the proportion of parasitized to uninfected cells can be determined.

Serological tests examine whether antibodies for the infection are present. They are mostly used for screening blood donors, testing for questionable diagnosis accompanied with treatment.

1.2. Role of Machine Learning

Numerous machine learning models have been proposed which segment a Whole Slide Image to identify red blood cells (RBCs) and classify these RBCs with a secondary trained model using deep neural network architectures and boosted trees. We observe how image transformations into the HSV mode can help isolate the irregularities better and propose models which can perform as well or better than existing ones.

1.3. Colour Models

Colour models are ways to represent colour images as tuples[3]. A number of colour models have been proposed, each trying to capture a specific aspect of sight. The RGB colour model captures colours in Red, Green and Blue. The HSV colour space captures Hue, Saturation and Brightness. Saturation captures how pure the colour is[4]. Converting from RGB to HSV uses the following formula:

$$R' = R/255$$
$$G' = G/255$$
$$B' = B/255$$

$$C_{max} = \max(R', G', B')$$
$$C_{min} = \min(R', G', B')$$
$$\Delta = C_{max} - Cmin$$

$$H = \begin{cases} 0 & \Delta = 0 \\ 60 \times (\frac{G'-B'}{\Delta} \mod 6) & C_{max} = R' \\ 60 \times (\frac{B'-R'}{\Delta} + 2) & C_{max} = G' \\ 60 \times (\frac{R'-G'}{\Delta} + 4) & C_{max} = B' \end{cases} \quad (1)$$

$$S = \begin{cases} 0 & C_{max} = 0 \\ \frac{\Delta}{C_{max}} & C_{max} \neq 0 \end{cases} \quad (2)$$

$$V = C_{max} \quad (3)$$

2. Literature Survey

Poostchi *et al*. created datasets, processed them, and tested a variety of algorithms like Naive Bayes, Logistic Regression, Decision Tree, Adaboost, SVMs, Neural Networks and Deep Neural Networks (DNNs). They also considered deployment of ML-based systems to diagnose Malaria [5].

Liang *et al*. compared a 16-layer Convolutional Neural Network (CNN) model with transfer learning for classifying single infected cells. They noted that the CNN achieved greater accuracy, sensitivity, specificity, f1-score and Matthew's correlation coefficient over the latter [6].

Pan *et al*. explores preprocessing of images and segmentation to isolate single cells from wholeslide images. They used encoders and discussed encoder architectures for feature extraction, and discussed spatial and feature-space interpolation for enriching the dataset. They consistently noted higher performance of models trained on augmented datasets [7].

Fuhad *et al*. implemented a CNN-based model and implemented data augmentation techniques like random rotations, zoom, translations, shear and horizontal flips. They used CNNs as an autoencoder to extract and reduce features and used SVM and KNN algorithms to classify the images. They conducted knowledge distillation in order to prune the trained model and reduce its complexity, and deployed the resulting commpressed model to mobile and web-based applications. They also conducted analyses if common mobile phones could utilize the model to classify cells. A final accuracy of 99.23% was obtained by their efforts with a validation loss of 0.02 with the log loss function[8].

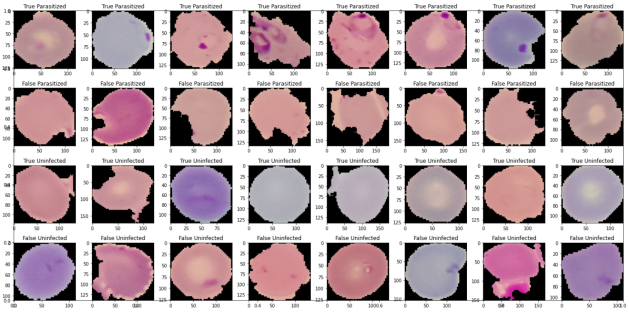

Figure 1. True and false parasitized and uninfected images.

3. Dataset

The dataset used was publicly available, courtesy of images were taken at Chittagong Medical College Hospital, Bangladesh.[9]

3.1. Dataset Description

The dataset contains 13,799 parasitized and 13,799 uninfected image samples containing 3 colour dimensions for a total of 27,588 images. The images are of varying sizes. The maximum height and width was 385 and 394 pixels respectively. The minimum height are width was 40 and 46 pixels respectively. The mean height and width was 133 and 132 pixels respectively. The median height and width was 130 pixels. The mean aspect ratio of the images is 1.0138.

Out of the 27,588 images, 647 parasitized and 750 unparasitized images were misclassified [8].

4. Methodology

We adopted a modular approach and created modules designed for a specific task: data download and labelling, test-train-validation splits, model evaluation and data

4.1. Exploratory Data Analysis

In order to determine which colour channel was the clearest with respect to the identification of the chromatin dot characteristic to the parasitized cell, we plotted the images in different colour channels and in grayscale.

Out of the plotted images, the green channel showed the maximum isolation of the chromatin dot. We also visualised inverted images and noticed that the green channel had isolated the chromatin dot the most.

We experimented with colour model transformations, and noticed that some models applying non-linear transformations (like HSV, HLS) captured the chromatin dot in parasitized cells better.

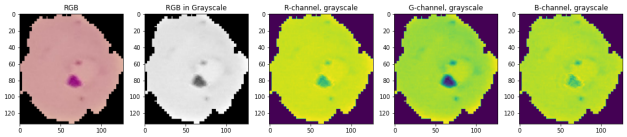

Figure 2. Comparison between various colour channels for a true parasitized cell.

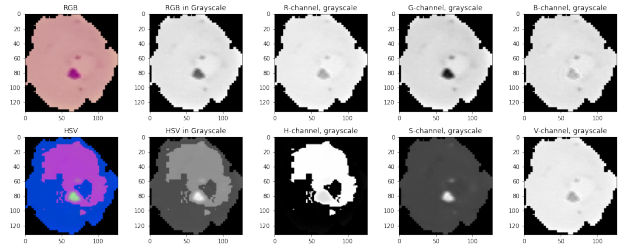

Figure 3. Conversion to HSV space from RGB.

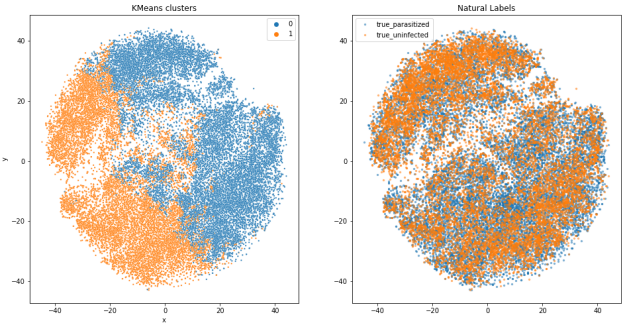

Figure 4. KMeans and Natural Labels. The dataset was reduced to 2 dimensions using t-SNE.

4.1.1 Cluster Visualisation

To visualise them, the images were first resized to a 50x50 colour format, each pixel value rescaled by 1/255, and each image finally flattened to a 50*50*3 length array. We used Euclidean distance as the distance metric and used the t-SNE algorithm to reduce dimensions to 2. We also used KMeans clustering to determine similarity clusters, but there was no clear relation between the natural clusters and the clusters output by K-Means (figure 4).

4.2. Preprocessing

The image data was read using Scikit-Image and each pixel value was scaled to a value between 0 and 1. The dimensions of each image were standardized to 25×25 to reduce computational complexity and VRAM requirements. Models were run using the standard RGB data which the datasets came in by default, and data transformed into the HSV format using Scikit-Image's $rgb2hsv$ function. Separate datasets were created and saved using Pickle for use. The datasets can be found here.

4.3. Data Augmentation

We augmented images using the Albumentations package [10] and applied the following transformations on the training data:

1. Rotation (range : -90°to +90°)
2. Scaling: (range: 0.8 times to 1 times the original image size)
3. Vertical Flip (probability = 0.5)
4. Horizontal Flip (probability = 0.5)
5. Shear (range: -7.5°to 7.5°)
6. Gaussian Noise (Mean = 0, variance between 0.01 and 0.05)

The augmentations were applied after rescaling to 25×25 image dimensions. The image was padded with zeroes after the transformations.

4.4. Models

The dataset was split in a stratified manner into training, validation and test sets. 80% of the samples were used for the training, 10% for validation and the last 10% for testing. A random seed value was used while splitting for reproducible results.

Data augmentation was carried out by taking every image from the training set and generating two two randomly operated images, storing them in a separate array.

The training dataset had 20,601 images, the augmented testing had 41,202 images, validation set had 2,943 images and the testing set had 2,617 images. The same augmented dataset was used for training all the augmented data models.

HSV conversion was carried out on all four datasets independently. As per the exploration and to reduce dimensionality, only the Saturation channel was used.

Hyperparameters were tuned on the validation set and the model parameters were unchanged between the unaugmented and augmented datasets. Early-stopping and checkpointing was used in TensorFlow-based models. The best-performing weights were loaded after the training completed.

4.4.1 Naive Bayes

We used Gaussian Naive Bayes without prior weight initialisation. Naive Bayes served as a baseline for classification, and as will be observed later, performed the worst consistently.

4.4.2 Logistic regression

Logistic Regression was implemented in TensorFlow using the Keras API. The Adam optimiser was used with learning rate tuned as per the performance on the validation set.

4.4.3 Decision trees

Decision trees provide explainable modelling and are very fast to inference with. The maximum depth was adjusted depending on the performance on the validation set.

4.4.4 XGBoost

XGBoost with GPU-acceleration was used and a forest of 20 trees was created. The maximum depth was a hyperparmeter and tuned according to performance on the validation set.

4.4.5 SVM

SVM was implemented using Scikit-Learn's implementation. Polynomial kernel with a degree of 3 was used for classification (default parameters).

4.4.6 CNNs

A convolutional neural network was implemented in TensorFlow. The model architecture for the RGB and HSV datasets were different. The architecture of the CNN used for classifying the HSV images is shown in 5

4.4.7 Transfer Learning

We used the Xception model, trained on ImageNetperformed reasonably well. A resizing layer was added to the model which upscaled the dimension of the image to 72x72 from 25x25 to be compatible with the model. Unfortunately, the model required 3 input channels and it was not possible to use it for the HSV data.

5. Results and Analysis

The results are available in tables 1, 2, 3 and 4.

We notice that among all models, the SVM model performs among the worst in all cases, followed by the Naive Bayes Classifier. It can be attributed to the classifier being unable to find a separating plane when the high dimensinal image data was compressed to the relatively low 3 dimensions.

Another striking thing we notice is that besides the logistic regression classifier, none of the models performed well on the RGB augmented data. This could have been a result of the nature of transformations used. Many attempts were made to regenerate the augmentations, but to no avail.

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 25, 25, 64)	640
max_pooling2d_8 (MaxPooling 2D)	(None, 9, 9, 64)	0
up_sampling2d_2 (UpSampling 2D)	(None, 36, 36, 64)	0
conv2d_9 (Conv2D)	(None, 36, 36, 32)	18464
max_pooling2d_9 (MaxPooling 2D)	(None, 12, 12, 32)	0
conv2d_10 (Conv2D)	(None, 12, 12, 32)	4128
max_pooling2d_10 (MaxPoolin g2D)	(None, 6, 6, 32)	0
flatten_3 (Flatten)	(None, 1152)	0
dense_9 (Dense)	(None, 512)	590336
dropout_6 (Dropout)	(None, 512)	0
dense_10 (Dense)	(None, 512)	262656
dropout_7 (Dropout)	(None, 512)	0
dense_11 (Dense)	(None, 2)	1026

Total params: 877,250
Trainable params: 877,250
Non-trainable params: 0

Figure 5. CNN model architecture

We attempted to recreate the scenario using TensorFlow's ImageDataGenerator using the same data as inputs and the same transformations, but there was no difference in results in training the CNN model. The model failed to converge yet again and resulted in bad performance overall. We hypothesize that it is caused due to the selection of augmentation functions, but that remains to be tested.

The Saturation channel in the HSV conversion certainly made a noticeable impact. For the unaugmented dataset, the performance of the CNN and XGBoost increased, while that of Naive Bayes, Logistic Regression, Decision Tree, and SVM decreased. This is particularly interesting, as this gave us an almost 10% performance increase for the XGBoost model and allowed the CNN model to match the accuracy of the top-performing models in this space.

Going from the unaugmented to augmented HSV dataset, we observe a general decline in performance, while that of the CNN increased (with the same model parame-

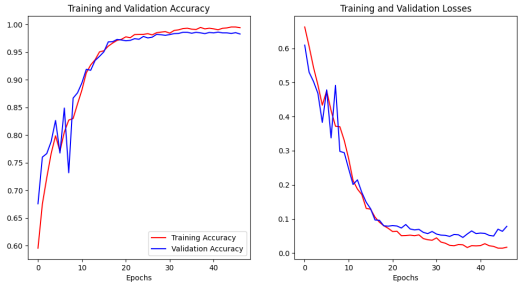

Figure 6. Loss-accuracy curve of the CNN model trained on unaugmented RGB data.

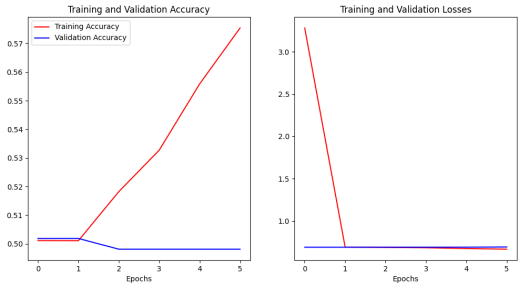

Figure 7. Loss-accuracy curve of the CNN model trained on augmented RGB data.

RGB Unaugmented (Test Data)				
Model	Acc.	Prec.	Rec.	F1
Naive Bayes	0.643	0.653	0.643	0.638
Logistic Regression	0.694	0.694	0.694	0.694
Decision Trees	0.684	0.687	0.684	0.682
XGBoost	0.850	0.850	0.850	0.850
SVM	0.518	0.550	0.516	0.419
Transfer Learning	0.955	0.956	0.955	0.955
CNN	0.982	0.982	0.982	0.982

Table 1. Performance of models trained on RGB Unaugmented training data

ters). This surpasses previously proposed models with respect to the precision, recall and f1 scores, and is almost able to match them in terms of model accuracy.

The loss-curves for the CNN model can be seen in figures 6, 7, 8, 9. We notice some overfitting in 6, 8, and 9, but the checkpointing and model reloading saves and reloads the best model, providing optimal results. Similar graphs were observed for the logistic learning and transfer-learning loss-curves.

Finally, we created a GUI-based interface which uses trained models for inferencing the class of a cell image. The video can be found **here**.

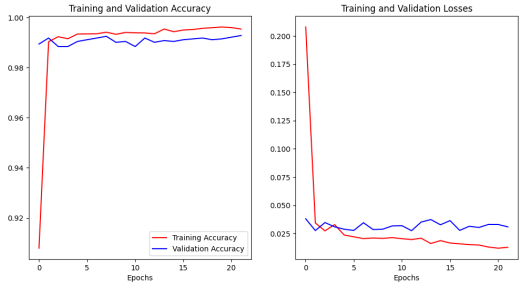

Figure 8. Loss-accuracy curve of the CNN model trained on unaugmented HSV data.

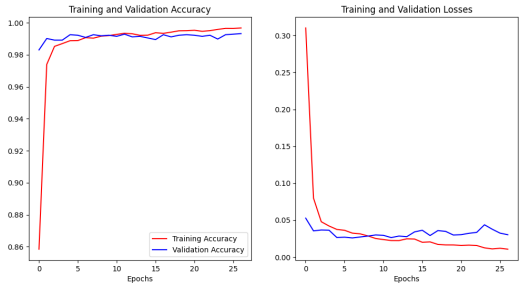

Figure 9. Loss-accuracy curve of the CNN model trained on augmented HSV data.

RGB Augmented (Test Data)				
Model	Acc.	Prec.	Recall	F1
Naive Bayes	0.502	0.251	0.500	0.334
Logistic Regression	0.588	0.665	0.587	0.532
Decision Trees	0.502	0.251	0.500	0.334
XGBoost	0.498	0.249	0.500	0.332
SVM	0.498	0.249	0.500	0.332
Transfer Learning	0.502	0.251	0.500	0.334
CNN	0.502	0.251	0.500	0.334

Table 2. Performance of models trained on augmented training data in the RGB domain.

HSV Unaugmented (Test Data)				
Model	Acc.	Prec.	Recall	F1
Naive Bayes	0.624	0.624	0.623	0.623
Logistic Regression	0.677	0.677	0.677	0.677
Decision Trees	0.675	0.678	0.675	0.674
XGBoost	0.938	0.939	0.938	0.938
SVM	0.502	0.251	0.500	0.334
CNN	0.993	0.993	0.993	0.993

Table 3. Performance of models trained on the saturation dimension training data converted into the HSV domain.

HSV Augmented (Test Data)				
Model	Acc.	Prec.	Recall	F1
Naive Bayes	0.618	0.624	0.617	0.611
Logistic Regression	0.625	0.627	0.625	0.623
Decision Trees	0.789	0.819	0.789	0.784
XGBoost	0.930	0.931	0.930	0.930
SVM	0.501	0.251	0.499	0.334
CNN	0.994	0.994	0.994	0.994

Table 4. Performance of models trained on the saturation domain of augmented images converted to the HSV domain.

6. Conclusion

We first notice the sheer strength of the CNN-based models. They are consistently at the top of the charts (excepting the outlier RGB-augmented data, on which no model could get properly trained).

We also notice that the HSV colour transformation helps with the CNN and XGBoost models, both of which shot up in accuracy significantly. The XGBoost model is said to be a powerful tool and it made it clear here.

The role of augmentations remains vague with the amount of testing done so far, with CNN models performing slightly better than before using them, but not by a significant margin. The values and kinds of transformations used for augmenting the data must be carefully determined so as to prohibit scenarios like the lack of convergence.

References

[1] W. H. Organization, *World malaria report 2021*. World Health Organization, 2021. p. 23 Death and Infection Statistics.

[2] C. for Disease Control and Prevention, "Cdc - malaria - diagnostic tools," 2020. Malaria Testing.

[3] Wikipedia contributors, "Color model — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/w/index.php?title=Color_model&oldid=1091166634`, 2022. [Online; accessed 4-December-2022].

[4] I. Kurniastuti, E. N. I. Yuliati, F. Yudianto, and T. D. Wulan, "Determination of hue saturation value (hsv) color feature in kidney histology image," *Journal of Physics: Conference Series*, vol. 2157, p. 012020, jan 2022.

[5] M. Poostchi, K. Silamut, R. J. Maude, S. Jaeger, and G. Thoma, "Image analysis and machine learning for detecting malaria," *Translational Research*, vol. 194, pp. 36–55, 2018. In-Depth Review: Diagnostic Medical Imaging.

[6] Z. Liang, A. Powell, I. Ersoy, M. Poostchi, K. Silamut, K. Palaniappan, P. Guo, M. A. Hossain, A. Sameer, R. J. Maude, J. X. Huang, S. Jaeger, and G. Thoma, "Cnn-based image analysis for malaria diagnosis," in *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 493–496, 2016.

[7] W. D. Pan, Y. Dong, and D. Wu, "Classification of malaria-infected cells using deep convolutional neural networks," in *Machine Learning* (H. Farhadi, ed.), ch. 8, Rijeka: IntechOpen, 2018.

[8] F. KMF, T. JF, S. MRA, M. S, M. N, and R. T., "Deep learning based automatic malaria parasite detection from blood smear and its smartphone based application," *Diagnostics (Basel)*, vol. 10, no. 5, p. 329, 2020. Misclassified images.

[9] N. L. of Medicine, "Thin smear single-cell dataset." Link 1; Link 2; Link 3. Dataset.

[10] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and flexible image augmentations," *Information*, vol. 11, no. 2, 2020.