Malaria Detection from Image Cells using Machine Learning Models

Hanzalah Firdausi
Harsh Vardhan Bhadauriya
Lavanya Adhikari

Abstract

Malaria is a disease caused by a plasmodium parasite and transmitted by the bite of an infected female Anopheles mosquito. Each year, more than 400,000 people die of malaria. The standard method for malaria detection involves preparing a blood smear and examining the stained blood smear using a microscope to detect the parasite genus Plasmodium, which heavily relies on the expertise of trained experts. In this paper, we have compared various feature extraction and classification methods on the red blood cell smears of the sampled cells.

The dataset used in the paper was taken from the National Institute of Health. The evaluation metrics such as accuracy, recall, precision, F1 score and Area under curve (AUC) were used to compare and select the best performing architecture. Our best model achieves an accuracy of 95.76%.

1 Literature Survey

Presently, numerous assessment papers are dispersed related to our problem statement and the methodologies we used. In Rahman et al. (2019), their proposed strategy utilizes deep learning, rather than the conventional strategies that use tedious hand engineering feature extraction, in an end-to-end arrangement that performs both feature extraction and classification directly from the raw segmented patches of the red blood smears. On the other hand, in Saiprasath et al., shallow machine learning algorithms are utilized against the conventional strategy, like AdaBoost, Random Forest, Decision Tree, KNN. Their proposed methodology determines the malarial infection with the help of captured images of patients without staining the blood or the need of experts.

In Poostchi et al. (2018), they give an overview and comparison of various techniques used in image analysis and machine learning for microscopic malaria diagnoses, such as imaging, image preprocessing, parasite detection and cell segmentation, feature computation, and automatic cell classification.

In Kumar and Bhatia (2014), they compared various types of features, feature extraction techniques and why it is significant. We additionally became more acquainted with under what situation, which features extraction technique, will be better and more valuable, and Deivanayagampillai et al. (2017), global and local texture feature extraction is done using different algorithms. The importance of texture as a feature in an image is highlighted. The global texture features for an image such as homogeneity, correlation, contrast, dissimilarity, maximum probability are computed.

Further, Vijayalakshmi and Kanna (2019) proposed CNN models (VGG16, VGG19) Simonyan and Zisserman (2015) with help vector machines (SVM) to decide the phases of parasite contamination and further developed the preparation time by utilizing pre-trained CNN models and the transfer learning techniques. The contribution of the research was aimed at fostering a CNN model to fine-tune the hyperparameter of the pre-trained model by using transfer learning. We referred to these articles throughout the making of this proposal.

2 Introduction

Malaria, a mosquito-borne deadly disease, is caused by protozoan parasites of the genus Plasmodium transmitted through the bites of infected female Anopheles mosquitoes, infecting red blood cells. The symptoms of an infected person are similar to the flu and can also include other symptoms, including high fever, fatigue, chills, septicemia, pneumonia, gastritis, enteritis, nausea, vomiting, headaches, and, in severe cases, seizures and coma, leading to death. Though it cannot spread from

person to person, it can pass on from mother to fetus as well as patients may be infected with malaria through blood transfusions or through sharing syringes. Malaria is most common in regions with warm, moist environments near natural water resources, close to the territory of Anopheles mosquitoes. Malaria is commonly diagnosed by microscopic examination of blood cells using blood films. Hundreds of millions of blood films are examined every year for malaria, which involves manual counting of parasites and infected red blood cells by a trained microscopist.

Accurate parasite counts are essential not only for malaria diagnosis but also for testing for drug resistance, measuring drug effectiveness, and classifying disease severity. For false-negative cases, this leads to unnecessary use of antibiotics, a second consultation, lost days of work, and in some cases, progression into severe malaria. For false-positive cases, a misdiagnosis entails the unnecessary use of anti-malaria drugs and suffering from their potential side effects, such as nausea, abdominal pain, diarrhoea, and sometimes severe complications. Hence, we will primarily use the F1 Score as the evaluation metric for our model. F1 Score is the weighted average of Precision and Recall. Therefore, this Score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more beneficial than accuracy in such cases.

We are trying to automate the diagnostic system using various machine learning techniques that would aid human specialists in making the most accurate diagnosis. We conducted extensive preprocessing on the NIH malaria dataset and utilized a few baseline models: Decision tree, Logistic Regression, and Support Vector Machine as our classifiers. Furthermore, several preprocessing methods such as various filters, global feature extraction, and local feature extraction were applied to the dataset to maximize performance in an unbiased test set.

Further, we delved into Deep learning to increase our accuracy. Deep Learning is a machine learning method made to impersonate the capacity of information processing and decision making in the human mind. Elements of the human mind are a lot more extensive than current profound learning capacities and incorporate association, mindfulness, personality, etc. We started with using ANN(Artificial neural network) as our baseline model for delving into deep learning, which is a computational network based on biological neural networks that construct the structure of the human brain.

We then moved on to convolutional neural network (CNN), which is a class of deep neural networks characterized by shared-weights architecture (i.e., which can take in an input image, assign importance to various aspects/objects in the image, and differentiate one from the other), and translation invariance characteristics. We applied different pre-trained VGG models, such as VGG-11, VGG-13, VGG-16 and VGG-19, with ReLU(Rectified linear unit) as our activation function. We used several evaluation metrics such as accuracy, recall, precision, F1 Score and Area under curve (AUC) to compare the baseline models. Our best model performed 95.76% for these different test sets.

3 Dataset

The dataset was taken from the National Institute of Health(NIH), named NIH Malaria Dataset. Our dataset consists of 27558 labelled red blood cell stained smears of the sampled cells. The dataset is equally distributed in 13779 uninfected images and 13779 parasitized images. The patches of segmented red blood cells are 3-channels (RGB) with a size variation of 110-150 pixels and a channel depth of 3.

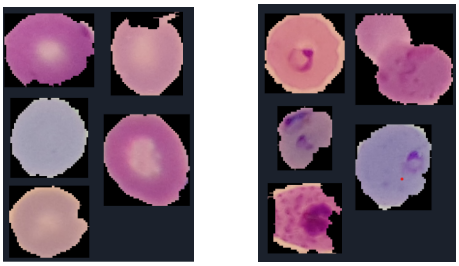
Positive samples contained plasmodium, and negative samples contained no plasmodium but could contain other objects, including staining impurities.

The images were later re-sampled to 64 x 64 output dimensions, with a channel depth of 3. We then calculated the greyscale channel as well. After that, we created a list named 'labels' to label the uninfected as 0 and parasitized as 1.

The Dataset is split into two sets: training test and testing set having the ratio of 75:25, after shuffling the Dataset.

4 Methodology

We have used various filters such as edge detection algorithms, namely, Canny, Sobel and Scharr, and Adaptive Histogram Equalization, also known as CLAHE. We have also used Global feature extraction, which finds features based on shape, texture and, colour histogram, and local feature extraction techniques such as SIFT (Scale-Invariant Feature Transform) and KAZE. We have then applied our



(a) Uninfected Cells (b) Parasitized Cells

Figure 1: Raw Images in Dataset

three baseline machine learning models to our processed data, namely, Decision Tree, Logistic Regression and, Support Vector Machine.

We then performed Ensembling on the best performing models, using different bagging and boosting techniques such as Random Forest and Adaboost for hyperparameter tuning of our models.

Further, we delved into Deep Learning, where we first used ANN(Artificial Neural Network). We then applied CNN (Convolution Neural Network) on our dataset using different pre-trained VGG models, such as VGG-11, VGG-13, VGG-16 and VGG-19, with ReLU(Rectified linear unit) as our activation function.

4.1 *Preprocessing*

The raw formatted pixel data in image patches won't be directly helpful in the classification task. Instead, we use a representation, which won't be affected by translation, rotation, and constant offsets in the intensity. The shape of objects in the input patches is the primary concern in the Plasmodium detection problem. We need to scale the images collected with different sizes and require a representation invariant to translation, intensity, and rotation.

We have used three edge detection filters, namely Canny, Sobel, and Scharr. We used edge detection for preprocessing in our dataset as the Parasitized Cell Images contained stained purple patches. Hence we decided to use edge detection to get the enhanced image of the infected cell, which would further help us in feature extraction.

Histogram Equalisation is used for the segmentation of images. Still, simple histogram equalisation induces a lot of noise in the image because it considers the global contrast of the image, not just the local contrast. Therefore performing global equalisation might not work very well on our image. Hence we have used Adaptive Histogram Equalization, also known as CLAHE. It does histogram equalisation in small patches or small tiles with high accuracy and limiting contrast.

4.2 *Global Feature Extraction*

We then used Global Feature Extraction on our filtered as well as non-filtered images, which extracted features based on shape, texture and, colour histogram. We used OpenCV's Humoments to find out the Hu Moments of the shapes present in the input image since we need to calculate moments that are invariant to translation, scale, and rotation. Haralick texture features are derived from the co-occurrence matrix, which contains information about how image intensities in pixels with a specific position in relation to each other occur together. Color histogram is a representation of the distribution of colours in an image. For digital images, a color histogram represents the number of pixels with colours in each of a fixed list of color ranges that span the image's color space, the set of all possible colours.

4.3 *Local Feature Extraction*

We then used two Local Feature Extraction techniques on filtered and non-filtered images, namely SIFT and KAZE. SIFT detector is based on the Difference-of-Gaussians (DoG) operator, an approximation of Laplacian-of-Gaussian (LoG)(Tareen and Saleem, 2018). Feature points are detected by searching local maxima using DoG at various scales of the subject images. The description method extracts a 16x16 neighbourhood around each detected feature and further segments the region into sub-blocks, rendering a total of 128 bin values. SIFT is robustly invariant to image rotations, scale, and limited affine variations.

KAZE features are also invariant to rotation, scale, limited affine and have more distinctiveness at varying scales with the cost of a moderate increase in computational time. KAZE features exploit non-linear scale-space through non-linear diffusion filtering, making blurring images locally adaptive to feature points, thus reducing noise and retaining regions' boundaries in subject photos. KAZE detector is based on scale normalized determinant of Hessian Matrix, computed at multiple scale levels. The maxima of detector response are picked up as feature points using a moving window. Feature description introduces the rotation invariance property by finding dominant orientation in a

circular neighbourhood around each detected feature.

4.4 *Normalisation*

After converting images into feature vectors, normalization was applied to the feature vectors to decrease the intensity of each feature before sending it to the various classifier models. Normalization gives new values that maintain the general distribution and ratios in the original feature vectors. Thus, making the computation easier and helping in training models faster. It also further reduces the chances of getting stuck in local optima. We have used basic standard scaling features by removing the mean and scaling to unit variance.

5 Baseline Systems

We used three baseline machine learning models: Decision Tree, Logistic Regression, and Support Vector Machine(SVM) on our filtered and non-filtered raw images with basic preprocessing. The results of our models on raw data were terrible, with maximum accuracy achieved with the SVM model, capping at 73.5%, which can be attributed to the low contrast of the image, the high number of features and significant information not given the due importance because of no feature extraction.

Model	Accur.	Precision	Recall	F1
DT	0.694	0.706	0.675	0.690
LR	0.634	0.645	0.611	0.628
SVM	0.735	0.739	0.736	0.738

Table 1: Models Performance Metrics on Raw Data

On applying models on global features extracted from raw data, we got better accuracy compared to previous models, with maximum capping at 87.1% with SVM.

When we applied the classifiers on the images with Canny Filter, we maxed out 86.1% with SVM. Edge Detection Filter Canny filter performs better than previous models because the infected patch is easily detectable. However, it performs worse than other Edge Detection techniques since it reduces noise by using a Gaussian filter and then uses Edge Detection on the potential image, which reduces the quality of features.

Applying our baseline models on our CLAHE images, we get maximum accuracy at 84.8% with SVM. Simple histogram equalisation induces a lot of noise in the image because it considers the global contrast of the image, not just the local contrast. Therefore performing global equalisation might not work very well on our image. In those cases, we can use Adaptive Histogram Equalization, also known as CLAHE. It does histogram equalisation in small patches or small tiles with high accuracy and limiting contrast.

With Sobel and Scharr filtered images, we get the best results at 89.5% and 90.8%, respectively. These Edge Detection Filters do not use the noise detection technique and give better-parasitised patch edges detection quality.

Decision trees give lower accuracy as they are greedy, grown deep, and not pruned. Thus each tree has a high variance(tend to overfit) but has lower bias. They can be susceptible to small perturbations in the data: a slight change can result in a drastically different tree. Thus, we didn't get good results with the decision tree during our baseline model implementations. Hence they were not a good baseline model.

Methods	Logistic Regression	SVM
Raw Images	0.862554	0.870682
Canny	0.845718	0.860958
CLAHE	0.824964	0.847896
Scharr	0.894194	0.895065
Sobel	0.915383	0.908128

Table 2: Models Performance on Various Filters with Global Features.

6 After Baseline Models

After seeing the results of the baseline models, we decided to try further to improve our results. Hence, we used two Local feature extraction techniques: SIFT and KAZE, on our filtered and non-filtered images. When applied on raw images with an SVM classifier, SIFT gave the best results, with 90.9% accuracy.

Techniques	Logistic Regression	SVM
SIFT	0.902808	0.908984
KAZE	0.672812	0.685441

Table 3: Models Performance on Raw Images with Local Features.

The results were underwhelming when we applied both Local feature extraction techniques on the filtered images. KAZE again performed poorly compared to SIFT, but even the latter performed

poorly compared to its performance on raw data. KAZE, we found out, was performing poorly because it could not capture many features from the data.

Method	Logistic Regression	SVM
Canny	0.753249	0.753438
CLAHE	0.793614	0.801597
Scharr	0.678066	0.692528
Sobel	0.685096	0.693510

Table 4: Models Performance on Various Filters with KAZE

Method	Logistic Regression	SVM
Canny	0.819784	0.820886
CLAHE	0.861313	0.876852
Scharr	0.795337	0.805873
Sobel	0.796865	0.801313

Table 5: Models Performance on Various Filters with SIFT

Next, we performed Grid Search on our best performing models, decidedly Sobel and Scharr filtered images, Global Feature Extraction on SVM classifier, and Local Feature extracted images from raw images on SVM classifier. We used polynomial and RBF kernels to perform the grid search. We got the max accuracy of 91.7% on Sobel Filter with Global Feature extraction.

Method	Accur.	Precision	Recall	F1
SIFT	0.907	0.910	0.902	0.906
Scharr	0.904	0.915	0.891	0.903
Sobel	0.917	0.930	0.903	0.916

Table 6: Grid Search results on our Best performing Models.

Then we decided to perform further ensembling techniques, such as Bagging and Boosting, to determine whether we could improve our accuracy on the same models. For Bagging, we used Random Tree Classifier, which fits multiple decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Using Bagging, the accuracy of SIFT data remained the same, but Scharr and Sobel deteriorated.

Then we performed Boosting using Adaboost with both Logistic Regression and the Decision Tree. AdaBoost classifier fits a classifier on the original dataset. It then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on complex cases. Using Boosting did not turn out fruitful since we did not get better results on both types of classifiers than the Grid Search on Sobel.

Method	Accur.	Precision	Recall	F1
SIFT	0.907	0.901	0.912	0.906
Scharr	0.873	0.870	0.880	0.875
Sobel	0.879	0.878	0.883	0.880

Table 7: Bagging (Random Tree Classifier) results on our Best performing Models.

Method	Accur.	Precision	Recall	F1
SIFT	0.875	0.916	0.826	0.869
Scharr	0.838	0.831	0.853	0.842
Sobel	0.872	0.876	0.870	0.873

Table 8: Adaboost with Logistic Regression results on our Best performing Models.

Method	Accur.	Precision	Recall	F1
SIFT	0.874	0.943	0.796	0.863
Scharr	0.847	0.827	0.881	0.841
Sobel	0.855	0.846	0.871	0.873

Table 9: Adaboost with Decision Tree results on our Best performing Models.

7 Final Models

After applying traditional models like SVM, RandomTree, AdaBoost, etc., we achieved an accuracy of 91.7%, which is pretty good, but it requires a lot of preprocessing and additional hyperparameter tuning. Hence, we decided to move towards Deep Learning models to have a robust system that utilises the properties of images well rather than relying on flattened output. We have leveraged the PyTorch framework to implement neural networks models. We decided to use CNN since it is built for image classification tasks, and it leverages the convolution operation and works wells on high dimensional images.

Each convolution task can pick up smart information like detecting an edge or a particular shape. Pooling helps to reduce dimensionality further. Onwards, it has a fully connected layer that behaves similarly to an ANN and helps in classification.

To implement CNN, we have first to prepare our dataset in a certain way. We resized each image to 32x32 with three channels, normalised the data, divided it into an 80:20 ratio for training and testing, respectively. We finally created DataLoaders with a batch size of 256.

We decided to use Visual Geometry Group(VGG) as our CNN architecture since it leverages the "deep" in deep learning to a greater extent than its predecessors. It replaced kernels of considerable size by stacking multiple small kernels. This way, we could get the same receptive field with fewer parameters, allowing us to go deeper.

We have used the lite version of the famous VGG architecture by decreasing the number of neurons in fully connected layers to reduce training overhead. Several convolutional layers are used along with pooling and batch normalisation to extract the image features. After which, a few fully connected layers are used to classify the model. Relu activation is used to introduce the non-linearity and solves the exploding/vanishing gradient problem. Dropout is used to reduce the overfitting and generalise the model. We implemented four different variations of VGG and made a comparison between them.

Type	Accur.	Precision	Recall	F1
VGG11	0.950	0.942	0.960	0.950
VGG13	0.953	0.947	0.960	0.953
VGG16	0.957	0.969	0.945	0.957
VGG19	0.958	0.959	0.956	0.958

Table 10: Performance metrics of various VGG architectures

8 Results and Analysis

In our project, we saw how traditional machine learning models could output high accuracy with various feature extraction techniques and hyperparameter tuning. From our traditional models, we got the best F1 score of 0.917 when we applied Sobel Filter with Global Feature Extraction on SVM Model. Further, we saw how CNN architecture could be used to get even better results, with an F1 score of 0.958 on VGG19 CNN architecture.

9 Conclusion

We can see that both traditional and deep learning models perform well on the cell images for malaria detection. However, traditional models need a lot of feature extraction, processing, and engineering to give good results. On the other hand, deep learning models perform exceptionally well. There is always a tradeoff between accuracy and power consumption. Deep networks require a lot of computational power. Still, traditional architecture, if coupled with sound engineering behind the scenes, traditional architecture can generate good results, if not better than deep learning models.

10 Contribution of each Member

Hanzalah Firdausi: Literature Survey, Preprocessing, Local Feature Extraction, Deep Learning
Lavanya Adhikari: Literature Survey, Preprocessing, Hyper Parameter Tuning, Deep Learning
Harsh Vardhan Bhadhauriya: Literature Survey, Preprocessing, Global Feature Extraction, Deep Learning

Complete Code along with Trained Models

References

Nagarajan Deivanayagampillai, Suruliandi A, and Kavitha Jc. 2017. Melanoma detection in dermoscopic images using global and local feature extraction. *International Journal of Multimedia and Ubiquitous Engineering,scopus*, 12:19–27.

Gaurav Kumar and Pradeep Kumar Bhatia. 2014. A detailed review of feature extraction in image processing systems. In *2014 Fourth International Conference on Advanced Computing Communication Technologies*, pages 5–12.

Mahdieh Poostchi, Kamolrat Silamut, Richard J. Maude, Stefan Jaeger, and George Thoma. 2018. Image analysis and machine learning for detecting malaria. *Translational Research*, 194:36–55. In-Depth Review: Diagnostic Medical Imaging.

Aimon Rahman, Hasib Zunair, M Sohel Rahman, Jesia Quader Yuki, Sabyasachi Biswas, Md Ashraful Alam, Nabila Binte Alam, and M. R. C. Mahdy. 2019. Improving malaria parasite detection from red blood cell using deep convolutional neural networks.

G.B. Saiprasath, N. Babu, J. ArunPriyan, R. Vinayakumar, V. Sowmya, and Dr Soman K. P. Performance comparison of machine learning algorithms for malaria detection using microscopic images". *IJRAR19RP014 International Journal of Research and Analytical Reviews*, 6(1).

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition.

Shaharyar Ahmed Khan Tareen and Zahra Saleem. 2018. A comparative analysis of sift, surf, kaze, akaze, orb, and brisk.

VGG. [link].

Vijayalakshmi and Rajesh Kanna. 2019. Deep learning approach to detect malaria from microscopic images. *Multimedia Tools and Applications*, 79(21-22):15297–15317.