Malaria Parasite Detection using Blood smear images

Yusuf Feroze
Northeastern University
akhtar.y@husky.neu.edu

Nayan Patel
Northeastern University
patel.nay@husky.neu.edu

Nikilesh Kumar
Northeastern University
kumar.ni@husky.neu.edu

Project undertaken as part of the course completion requirements for Neural Networks and Deep Learning (IE7615) *by Prof. Jerome Braun, Northeastern University*

ABSTRACT

In this report, we propose a method to classify human blood smear images infected with malaria parasite or uninfected using a deep learning architecture. This paper introduces a Convolutional Neural Network framework, comparing two different architectures and, performing hyperparameter tuning along with using data augmentation with the help of some of the image preprocessing functionalities of the "Keras", "opencv" and "Tensorflow" packages in python.

INTRODUCTION

Malaria is a serious problem in the developing world with over 216 Million reported cases just in 2016.[1] This disease is spread by mosquitoes and is caused by single celled parasitic organisms called plasmodium. Till date, diagnosis of malaria is done by microscopic examination, which is held as the "gold standard" for laboratory confirmation of malaria. Blood specimen collected from the patient is spread on a thick or thin blood smear, stained and examined with a 100X oil immersion objective[2]. Microscopy is an established and relatively simple technique that is familiar to most laboratory technicians. Any laboratory that can perform routine hematology tests is capable of performing analysis on a thin and thick malaria smear. Within a few hours of collecting the blood, the microscopy test can provide valuable information on the patient's condition. The accuracy of the diagnosis of malaria using microscopy depends on the parasitologists' skills. This procedure can be tedious and possibly erroneous, with some subjectivity leading to inconsistent or incorrect diagnosis which may cause incorrect or untimely treatment or even result in the death of the patient.

Due to the inconsistency of microscopy, this study is done, in similar fashion as pre-existing research, to potentially develop a malaria detection model that can improve upon or replicate the accuracy of the conventional gold standard of malaria detection, increase the pace of diagnosis and reduce the burden on parasitologists. This will result in improving detection in rural areas and places with a lack of highly skilled parasitologists improving the overall efficiency of the diagnosis process.

LITERATURE REVIEW

Owing to the abundance of malaria cases and diagnosis around the world, this disease has been well studied by researchers for a long time in an attempt to eradicate it and treat affected individuals promptly.

A study by Rosado et al. (2016) used a 194 cell images captured on mobile devices and a SVM based classifier to achieve a 98.2% sensitivity and 72.1% specificity[3]. Their research however did not count all possible combinations of the malarial plasmodium that could infect humans.

A subsequent approach by N.A Khan et al. (2017) used a K-means based, unsupervised classifier to classify and diagnose infected cells[4]. This was picked up and then followed by S.Kunwar et al. (2018) where they applied image preprocessing to account for a more robust classification system.

While the pre-existing approaches worked on increasing the model robustness or classification accuracy, we aim to replicate the accuracy and/or improve upon the speed of classification by taking features obtained by passing our image data through a Convolutional Neural Network. In effect, we will be training our classifier on the features extracted from the neural network with the goal of optimizing classification accuracy and speed.

DATASET

The dataset was sourced from the US National Laboratory of Medicine and was hosted on their website.

METHODS

1. Data Collection and Preprocessing

A mobile application that runs on a standard Android smartphone attached to a conventional light microscope was used to acquire the images. 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 dataset contains a total of 27,558 cell images with equal instances of parasitized and uninfected cells[5].

The images were resized to 64x64x3 and the labels converted to numerical using one-hot encoding. Scikit-learn's splitting function was used to create the train and test datasets with 80% train and 20% test set. The training data was further split into 80% training and 20% validation set to use during training.

A sample of the training data for parasitized images -

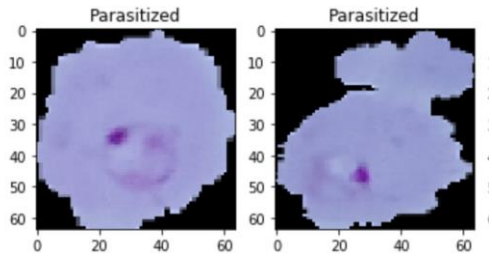


Fig. 1: Sample of parasitized cells

A sample of the training data for uninfected images -

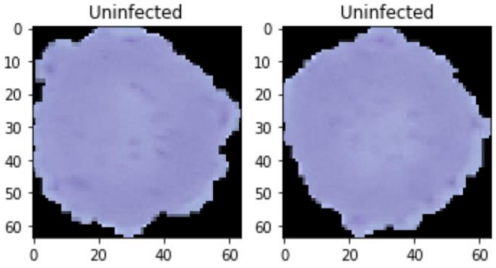


Fig. 2: Sample of non-parasitized cells

2. Data Augmentation

We experiment with different types of data augmentation techniques such as:

1) Histogram Equalization[6] - This technique increases the contrast values between pixels in the image by applying a normalization probability function on the histogram of pixel values generated. The histogram equalized image is given by the following formula

$$g_{i,j} = \text{floor}((L-1)\sum_{n=0}^{f_{i,j}} p_n),$$

On applying histogram equalization we obtain the following images -

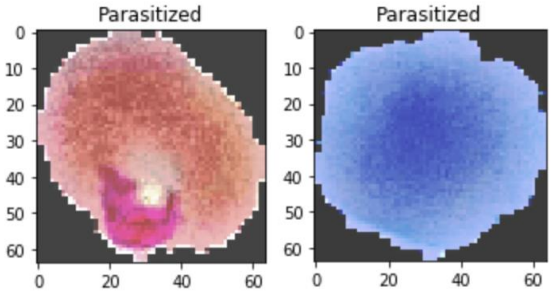


Fig. 3: Images after Histogram Equalization

2) Affine Augmentation techniques - pixel shifts and rotations.

3) ZCA Whitening - a whitening technique that works by dividing principal components by the square root of PCA eigenvalues. The function is explained as follows. When you have N data points in R^n then the covariance matrix $\Sigma \in R^{n \times n}$ is estimated to be:

$$\hat{\Sigma}_{jk} = \frac{1}{N-1}\sum_{i=1}^{N}(x_{ij} - \bar{x}_j) \cdot (x_{ik} - \bar{x}_k)$$

Where x_j denotes the j^{th} component of the estimated mean of the samples x. Any matrix W $\in R^{n \times n}$ that satisfies the condition

$$W^T W = C^{-1}$$

whitens the data.

A sample of images obtained using ZCA whitening -

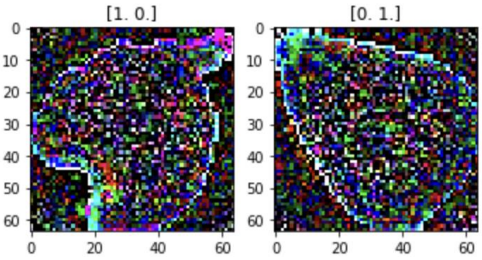


Fig. 4: Images after ZCA Whitening

For our model we will be using affine augmentation techniques (pixel shifts and rotations)

3. Model

First Architecture

The first architecture was a Conv-Pool-Conv architecture with 3 convolutional layers, 3 pooling layers and 3 dropout layers. We used 'ReLU' activation function for input and hidden layers while a 'softmax' activation function was used at the output fully connected layer. Max pooling was used for pooling and batch normalization was used to normalize the result of each conv-pool layer. The default stride of 1 with a 3x3 kernel was used for the convolutions with padding done to keep the dimensions same. Compilation was done using optimizer 'Adam' and a learning rate of 0.001 and decay rate 0.001/batch size where the mini batch size was set at 64. These were later tuned in the hyperparameter tuning experiments.

We trained this model for 30 epochs and used augmented data to add number of augmented images for the same. The loss function used was the categorical cross entropy function which was minimized during training.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 64, 64, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 32, 32, 32)	0
batch_normalization_1 (Batch	(None, 32, 32, 32)	128
dropout_1 (Dropout)	(None, 32, 32, 32)	0
conv2d_2 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_2 (MaxPooling2	(None, 16, 16, 32)	0
batch_normalization_2 (Batch	(None, 16, 16, 32)	128
dropout_2 (Dropout)	(None, 16, 16, 32)	0
conv2d_3 (Conv2D)	(None, 16, 16, 64)	18496
max_pooling2d_3 (MaxPooling2	(None, 8, 8, 64)	0
batch_normalization_3 (Batch	(None, 8, 8, 64)	256
dropout_3 (Dropout)	(None, 8, 8, 64)	0
flatten_1 (Flatten)	(None, 4096)	0
dense_1 (Dense)	(None, 4096)	16781312
batch_normalization_4 (Batch	(None, 4096)	16384
dropout_4 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 2)	8194

Total params: 16,835,042
Trainable params: 16,826,594
Non-trainable params: 8,448

Second Architecture

The second architecture employed a Conv-Conv-Pool architecture with two strides and a 'sigmoid' output activation function. The output layer wasn't changed a fully connected layer as earlier was used for the purpose. We also utilized the binary cross entropy loss function considering this was a two class problem as a better choice. The kernels were (3,3) and had stride of 2. "ReLU' activation function for the hidden layer , 'Sigmoid' for the outer layer was used.

The model was optimized using the stochastic gradient descent optimizer (SGD). The parameters, strides, kernel dimensions, activation functions, learning rate, momentum, decay and dropout layers were added and tuned to the best model.

RESULTS

Geometric transformations like rotation, translation, and shifting pixels horizontally and vertically were used to produce augmented images. This increased the variations in the training images the model trains on.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 64, 64, 32)	896
conv2d_2 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_1 (MaxPooling2	(None, 16, 16, 32)	0
batch_normalization_1 (Batch	(None, 16, 16, 32)	128
conv2d_3 (Conv2D)	(None, 8, 8, 64)	18496
conv2d_4 (Conv2D)	(None, 4, 4, 64)	36928
max_pooling2d_2 (MaxPooling2	(None, 2, 2, 64)	0
dropout_1 (Dropout)	(None, 2, 2, 64)	0
batch_normalization_2 (Batch	(None, 2, 2, 64)	256
dropout_2 (Dropout)	(None, 2, 2, 64)	0
flatten_1 (Flatten)	(None, 256)	0
dense_1 (Dense)	(None, 4096)	1052672
dense_2 (Dense)	(None, 2)	8194

Total params: 1,126,818
Trainable params: 1,126,626
Non-trainable params: 192

In this project, we test three different approaches: training data volume, model complexity and depth, and hyperparameter tuning. Accuracy and loss curves are shown for comparing different approaches. Accuracy, specificity, recall and F1 scores are reported for the final model.

PERFORMANCE

Train Data Volume:

Deep neural networks/CNN models require large amounts of training data in order to be effective in making predictions / classification. We were interested in exploring how training data volume would affect model performance. To investigate this, we ran our model on original number of training data images (17636) and doubling

(35272) the original training images by using data augmentation. The following figure highlights our findings. While training accuracy increased slightly with the larger data set, from 0.95 with 17636 images to 0.97 in case of 35272 images, the validation accuracy stayed fairly consistent at approximately around 0.95. Hence, all further experiments were therefore conducted with 17636 images.

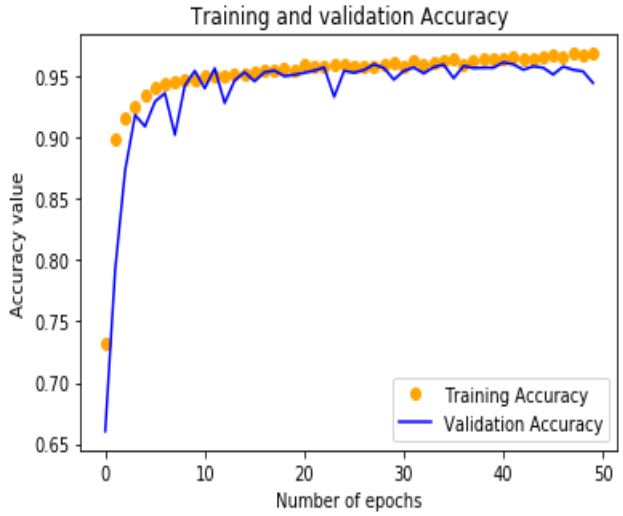


Fig. 5: Training and validation accuracy curves for 35272 training samples.

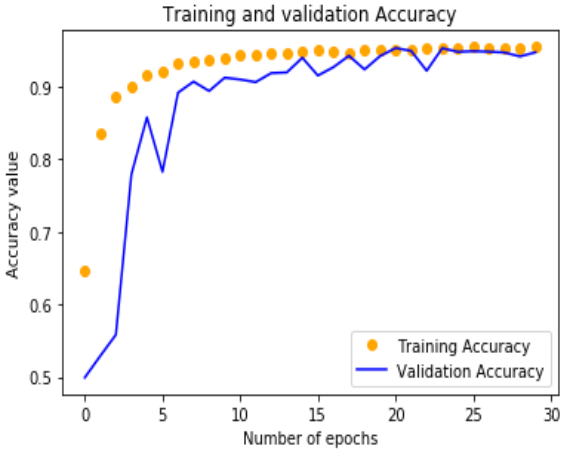


Fig. 6: Training and validation accuracy curve for 17636 training samples.

Network Architecture Comparison:

We tested out two different network architectures as described in the model sections. The following accuracy curves illustrate that the complexity of the network does not improve the validation accuracy significantly (model performance), training and validation accuracy for both models is around 0.95. But as seen in figure. 4, the complex architecture allows the model to train with fewer iterations and also the validation accuracy stays more stable compared to the conv-pool-conv architecture. Hence, we proceeded forward with the conv-conv-pool architecture.

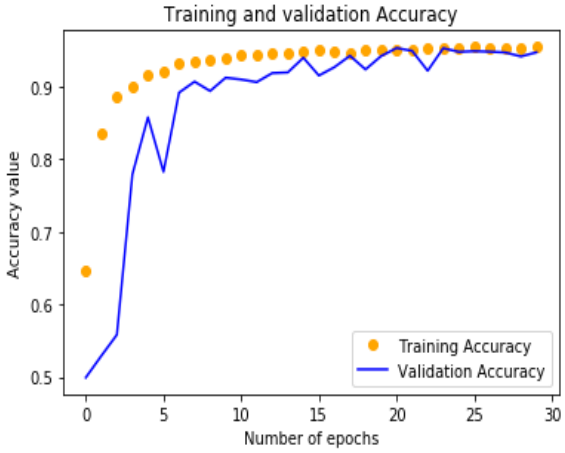


Fig. 7: Training and validation accuracy curve for conv-pool-conv Architecture.

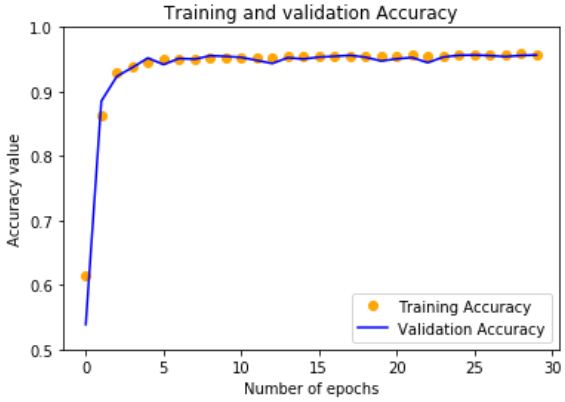


Fig. 8: Training and validation accuracy curve for conv-conv-pool Architecture (complex).

Hyperparameters:

Once the training data size and network architecture were selected, we focused on tuning hyperparameters like kernel size, learning rate, momentum, decay rate for learning rate, weight initialization and optimizers. The final model that was selected after multiple iterations of different combinations of hyperparameters has kernel size of 3 x3 with 2 strides, learning rate of 0.01 with momentum of 0.9. We were able to use a high learning rate and momentum as a result of adding high dropout of 0.5 and batch normalization in the network[7]. We applied a linear learning rate decay proportional to batch size – learning rate / batch size. We tried out two different optimizers and 'adam' and 'stochastic gradient descent' and decided on stochastic gradient descent as it allowed to use higher learning rate as compared to 'adam' and gave faster training of the model. Finally, we also used 'He uniform' weight initialization as it works best with relu activation function[8].

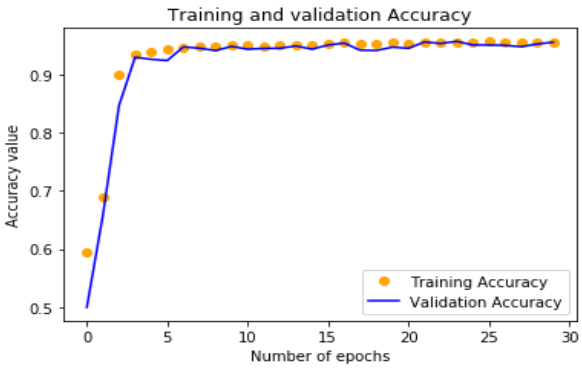


Fig. 9: Training and validation accuracy curve for final model with tuned hyperparameters.

PERFORMANCE:

	precision	recall	f1-score	support
Parasitized	0.98	0.94	0.96	2767
Uninfected	0.94	0.98	0.96	2745
Micro average	0.96	0.96	0.96	5512
Macro average	0.96	0.96	0.96	5512
Weighted average	0.96	0.96	0.96	5512

Table. 1: Classification report for the final model on Test data.

	Predicted	
Actual	Parasitized	Uninfected
Parasitized	2588	179
Uninfected	55	2690

Table 2: Confusion Matrix for the final model on Test data.

VISUALIZATIONS

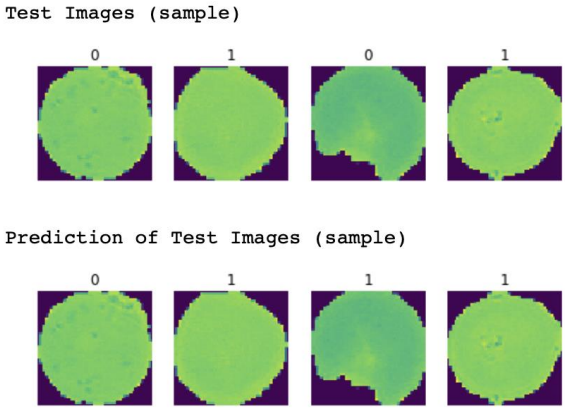


Fig. 10: Sample of Predictions Generated

CONCLUSIONS

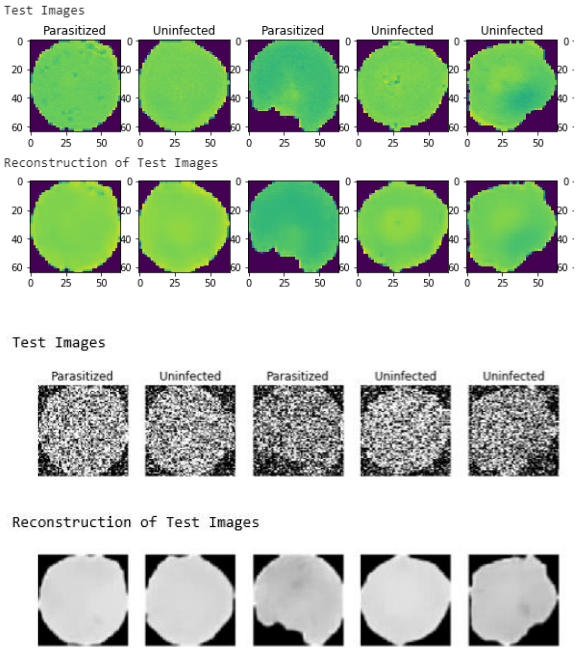
The model selected for the supervised learning task of classification of blood smear images was a stacked CNN model with two layers of convolutions and one pooling architecture. After tuning this model by changing various hyperparameters, we were able to classify the test data with 95.7% accuracy. This model is pretty robust with respect to augmented data because of training data automatically using the augmented images. We were also able to incorporate dropout layers, normalization and momentum for regularizing effects. The sensitivity of the model to predicting Parasitized images was 94% which shows the model can be used for effective scanning and elimination of Uninfected images/samples for building any automated application even if it is not directly used for diagnosis.

ADDITIONAL WORK AND SCOPE

This model in comparison to state of the art pretrained models such as ResNet 50 and other custom models for the dataset proved to be highly competitive and can be further improved with using Grid search mechanisms such as hyperopt and talos. Limitations in GPU available and project duration didn't allow us to use such mechanisms for more structured tuning.

We attempt to reduce the size of the dataset and the computation time by implementing an Convolutional Auto Encoder for feature learning and then using the encoded bottleneck layer as the input to a classification algorithm. This model however provided no real improvement over using a CNN directly for classification and was much slower in training than the supervised learning model. Also, the loss in information in this step would exacerbate the loss due to classification error, making worse assumptions than other models.

However, we were able to successfully train a denoising autoencoder for this dataset, i.e. a noisy image input can be successfully denoised and the original image reconstructed using an autoencoder. The model employed 3 Conv-Pool encoding layers and three Conv-Upsampling decoding layers during training. The activation function for the hidden layers was as earlier, 'ReLU' and output function was 'sigmoid'. This was trained on mean squared error loss and optimized using RMS Prop.



We hope to improve our methodology here or to use it for tasks where feature extraction using a CAE would be a more relevant path to take.

REFERENCES

1. World malaria report 2017. Geneva: World Health Organization; 2017. Licence: CC BY-NC-SA 3.0 IGO.
2. D. Bibin et al.: Malaria Parasite Detection From Peripheral Blood Smear Images Using DBN
3. Luis Rosado et al. Automated Detection of Malaria Parasites on Thick Blood Smears via Mobile Devices
4. Najeed Ahmed Khan et al. Unsupervised identification of malaria parasites using computer vision
5. Rajaraman S, Antani SK, Poostchi M, Silamut K, Hossain MA, Maude RJ, Jaeger S, Thoma GR. 2018. Pre-trained convolutional neural networks as feature extractors toward

improved malaria parasite detection in thin blood smear images. PeerJ 6:e4568 https://doi.org/10.7717/peerj.4568
6. Nicholas Sia et al. A Literature Review on Histogram Equalization and Its Variations for Digital Image Enhancement, 2013
7. Nitish Srivastava , Geoffrey Hinton , Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting.
8. https://towardsdatascience.com/hyper-parameters-in-action-part-ii-weight-initializers-35aee1a28404.