Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 8895429, 15 pages
https://doi.org/10.1155/2020/8895429

Research Article

Leveraging Deep Learning Techniques for Malaria Parasite Detection Using Mobile Application

Mehedi Masud,[1] Hesham Alhumyani,[1] Sultan S. Alshamrani,[1] Omar Cheikhrouhou,[1] Saleh Ibrahim,[2,3] Ghulam Muhammad,[4] M. Shamim Hossain,[5] and Mohammad Shorfuzzaman[1]

[1]*College of Computers and Information Technology, Taif University, Taif 21974, Saudi Arabia*
[2]*Electrical Engineering Department, Taif University, Saudi Arabia*
[3]*Computer Engineering Department, Cairo University, Egypt*
[4]*Department of Computer Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia*
[5]*Department of Software Engineering, College of Computer and Information Sciences, King Saud University, King Saud University, Riyadh 11543, Saudi Arabia*

Correspondence should be addressed to M. Shamim Hossain; mshossain@ksu.edu.sa

Received 26 March 2020; Revised 19 May 2020; Accepted 2 June 2020; Published 8 July 2020

Academic Editor: Yin Zhang

Malaria is a contagious disease that affects millions of lives every year. Traditional diagnosis of malaria in laboratory requires an experienced person and careful inspection to discriminate healthy and infected red blood cells (RBCs). It is also very time-consuming and may produce inaccurate reports due to human errors. Cognitive computing and deep learning algorithms simulate human intelligence to make better human decisions in applications like sentiment analysis, speech recognition, face detection, disease detection, and prediction. Due to the advancement of cognitive computing and machine learning techniques, they are now widely used to detect and predict early disease symptoms in healthcare field. With the early prediction results, healthcare professionals can provide better decisions for patient diagnosis and treatment. Machine learning algorithms also aid the humans to process huge and complex medical datasets and then analyze them into clinical insights. This paper looks for leveraging deep learning algorithms for detecting a deadly disease, malaria, for mobile healthcare solution of patients building an effective mobile system. The objective of this paper is to show how deep learning architecture such as convolutional neural network (CNN) which can be useful in real-time malaria detection effectively and accurately from input images and to reduce manual labor with a mobile application. To this end, we evaluate the performance of a custom CNN model using a cyclical stochastic gradient descent (SGD) optimizer with an automatic learning rate finder and obtain an accuracy of 97.30% in classifying healthy and infected cell images with a high degree of precision and sensitivity. This outcome of the paper will facilitate microscopy diagnosis of malaria to a mobile application so that reliability of the treatment and lack of medical expertise can be solved.

1. Introduction

Cognitive computing replicates the way humans solve problems while artificial intelligence and machine learning techniques search for creating novel ways for solving problems that humans can potentially do better. A substantial amount of research has been done during the last decades using machine learning algorithms for cost-effective solutions to support healthcare professionals in reducing diseases. Malaria disease originated from Plasmodium parasites through mosquito-borne infection. Malaria is very common over the world mainly in tropical regions. Figure 1 shows how malaria is widely spread across the globe. When infected female Anopheles mosquitoes bite a person, the parasites enter into the blood and begin damaging red blood cells (RBC) that carry oxygen. Flu virus is the malaria's first

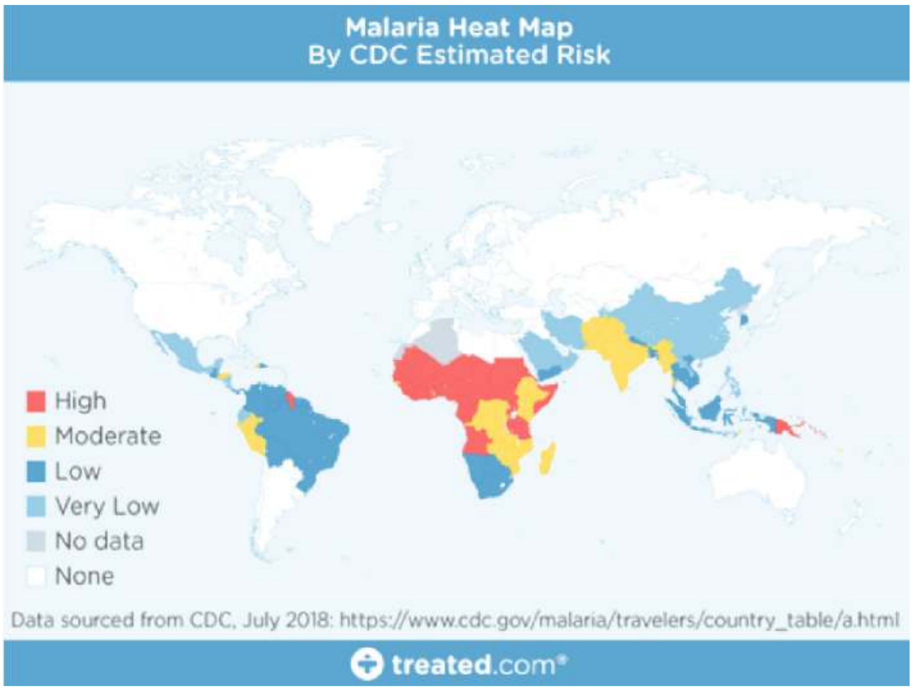
Malaria Heat Map
By CDC Estimated Risk
High
Moderate
Low
Very Low
No data
None
Data sourced from CDC, July 2018: https://www.cdc.gov/malaria/travelers/country_table/a.html
treated.com®

Figure 1: Malaria world map of estimated risk (2018 update) [3].

symptom. The symptom generally starts in few days or weeks. Most importantly, the lethal parasites can stay alive more than a year in a person's body without showing any symptoms. Therefore, a late treatment can cause complications and even death. Hence, many lives can be saved through early malaria detection. Almost 50% of the population in the world is in danger from malaria. There are more than 200 million malaria cases and 400,000 deaths reported every year due to malaria. In practice, to identify malaria, microscopists inspect blood (thick and thin) smears for disease diagnosis and calculate parasitemia. Microscopy examination is used as one of the prime standards for the diagnosis of malaria [1, 2] to identify the existence of parasites in a blood drop from thick blood smears. However, thin blood smears are used for distinguishing the species of parasite and the development of malaria stages. Examination through a microscope is commonly used since it is cheap but time-consuming. The examination accuracy relies on the quality of blood smear and a skilled person who is expert in the classification and examination of uninfected and parasitized blood cells.

Traditional approaches for malaria detection are very time-consuming, may produce inaccurate reports due to human errors, and are laborious for extensive diagnoses. This motivates us to propose an automatic detection of malaria applying deep learning techniques and using a mobile application that leads to early diagnosis which is fast, easy, and effective.

Several ideas exist to detect malaria parasites in microscopic images using convolutional neural networks (CNNs), some pretrained variants of CNN [4–8], and recurrent neural network (RNN) [9]. Moreover, authors in [10, 11] proposed approaches that consider unsupervised machine learning algorithms applying stacked autoencoders for learning the features automatically from the infected and uninfected cell images. Liang et al. [12] proposed a deep learning model for infected malaria cell classification from red blood smears. The model consists of 16-layer convolutional neural network which outperforms transfer learning-based models that use pretrained AlexNet [13].

Jane and Carpenter [14] proposed an object detection-based model using a convolutional neural network, named as Faster R-CNN. The model is first pretrained on ImageNet [15] and then fine-tuned on their dataset. Bibin et al. [16] recommended another model using deep relative attributes (DRA) [17]. Authors use CNN for epilepsy seizure detection [18]. Razzak and Naz [19] have proposed an automated process that considers the tasks of both segmentation and classification of malaria parasites. Their segmentation network consists of a Deep Aware CNN [20], and the classification network employs an extreme learning machine- (ELM-) based approach [21].

Since we are aiming to develop a mobile-based effective solution for malaria detection, we look forward to coming up with a CNN-based deep learning model which is expected to be simpler and computationally efficient in contrast to most of the state-of-the art approaches discussed before that require longer training time. In particular, we make the following contributions: (a) design and evaluation of a base CNN model with standard or no learning schedule and very less trainable parameters to classify parasitized and uninfected cell images, (b) the use of a SGD optimizer with cyclical learning rate schedule along with an automatic learning rate finder in addition to commonly applied regularization techniques in improving the model performance, and (c) deployment of our best performing model to a mobile application to facilitate simpler and faster malaria detection.

The rest of the paper is organized as follows. Related Work reviews the state-of-the-art techniques used in malaria classification. Materials and Methods provides detailed description of our model, its configuration, dataset used, and performance evaluation metrics. Results and Discussion presents the performance results obtained for our base and improved models and provides state-of-the-art comparison. Finally, Conclusions concludes the paper and outlines some potential future work.

2. Related Work

There has been a significant amount of research during the last decades using computing algorithms for cost-effective solutions to support interoperable healthcare [22] in reducing diseases. For instance, Neto et al. [23] proposed a simulator for simulating events of epidemiology in real time. Kaewkamnerd et al. [24] proposed an image analysis system consisting of five phases for malaria detection and classification. Anggraini et al. [25] developed an application applying image segmentation techniques for separating blood cells' background. Furthermore, Rajaraman et al. [4] implemented feature extractors using pretrained CNN-based deep learning models for uninfected and parasitized blood cell classification to facilitate disease identification. The research used experimental approach to identify the optimal model layers using the underlying data. The CNN model has two fully connected dense layers and three convolutional layers. The performance is measured to extract features using VGG-16, AlexNet, Xception, DenseNet-121, and ResNet-50 from the uninfected and parasitized blood cells. In contrast to [4], only CNN-based malaria classifiers are also proposed by Gopakumar et al. [26] and Liang et al. [12].

MOMALA [27] is a smartphone and microscope-based application developed to detect malaria quickly at a low cost. The MOMALA app can detect the existence of malaria parasites on a regular blood-smeared slide. A phone camera is attached to the microscope's ocular to take the photographs of the blood smear and then analyzes it. At present, the application highly depends on microscopes that are heavy, bulky, and not easily transportable.

The researchers in [28] developed a mobile app that takes photos of blood samples to detect malaria immediately. Using a cell phone app, we can analyze blood samples without involving microscope technicians. The app needs to clamp a smartphone on to a microscope's eyepiece, and the application analyzes the images of the blood sample and creates a red circle on malaria parasites. A lab worker later reviews the case. Extraction of meaningful features is the heart of success for any machine learning method. Most of the computer-used diagnosis tools that use machine learning models for image analysis are based on manual-engineered features for making decision [29–31]. The process also needs computer vision expertise in order to analyze the variability on the images in size, color, background, angle, and position of interests. Deep learning techniques can be applied with considerable success for overcoming the challenges that prevail in a hand-engineered feature extraction process [32]. Models in deep learning apply a series of sequential layers with nonlinear processing hidden units that can find out hierarchical feature relations within the raw image data. The features (low-level) that are abstracted from higher-level features assist in functions of nonlinear decision-making, learning complexity, result in end-to-end extraction of features, and classification [33]. Moreover, deep learning models show better performance compared to kernel-based algorithms such as Support Vector Machines (SVMs), in large volume of data and computational resources, building them to be greatly scalable [34].

A somewhat related pool of work in cognitive computing domain has presented similar contribution. Zhang et al. [35] proposed a protection mechanism for authentication and access control using an interactive robot while controlling private data access stored in cloud. In a subsequent effort [36], they introduced a novel paradigm of cognitive IoT using technologies of cognitive computing. A group of researchers [37] also proposed a module, called Mech-RL, for developing an agent-based literature consultant and a new channel of a meta-path learning method. Furthermore, similar to our battery-operated mobile-based application for malaria detection that can easily be deployed to edge and IoT devices, there is a handful of research [38–41] aiming at developing frameworks on mobile edge to deliver various related services such as secure in-home IoT therapy, content recommendations [42] [43], and position-based services for network amenities [44].

To summarize, the related work mentioned in the literature largely used different pretrained CNN variants such as AlexNet, VGG-16, ResNet-50, Xception, DenseNet-121, and customized CNN models as well for malaria detection in blood smear images and obtained relatively better results than using a custom CNN architecture. However, the downside is that these results are obtained through feature extraction and subsequent training that required long time in some cases [4] a little over 24 hours. In addition, size and complexity of these models make them a bit unrealistic to be used with battery-operated mobile devices. In contrast, we built a simpler and computationally efficient CNN model with considerably less trainable parameters (discussed in Model Configuration section), yet producing comparable or better results keeping in mind our model to be deployed on battery-operated edge and IoT devices such as a smart mobile phone. Moreover, techniques in the literature mostly use de facto SGD optimizer with various learning rate schedules including the adaptive learning rates which suffer from the problem of saddle point or local minima. In contrast, we have used a SGD optimizer with cyclical learning rate schedule along with an automatic optimal learning rate finder which results in faster model convergence with fewer experiments and hyperparameter updates. Finally, most of the state-of-the-art models use image augmentation to increase model generalizability at the expense of longer training time. On the other hand, our model without using data augmentation demonstrates faster convergence and generalizability to unseen data through proper hyperparameter optimization such as learning rate, regularization through batch

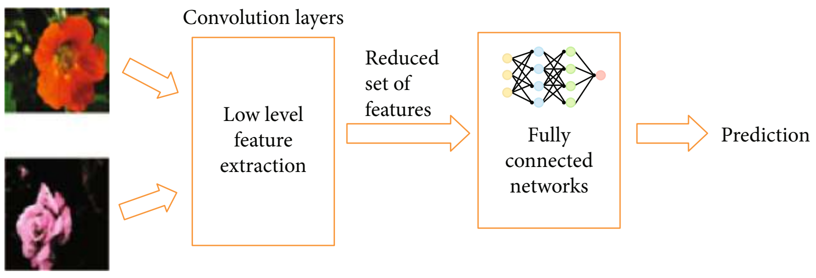

Figure 2: A general CNN model.

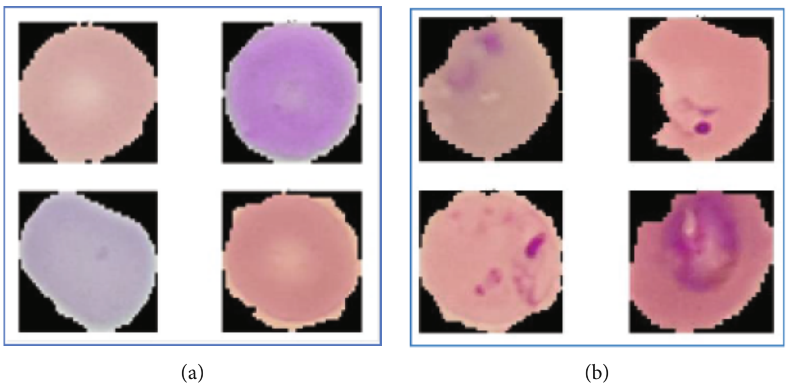

Figure 3: Sample images from NIH dataset: (a) uninfected and (b) parasitized.

normalization, and moderate dropouts in convolutional and dense layers.

Among the studied malaria detection models in the literature, the models proposed in [4, 12, 16, 26] based on custom CNN and its pretrained variants seem to be closest to our model. Hence, we performed a state-of-the-art comparison with these models to demonstrate the feasibility of using our model in a mobile-based system especially in remote disaster survival areas.

3. Materials and Methods

3.1. Deep Learning for Malaria Detection. Deep learning techniques are now widely used for image classification, video recognition, and medical image analysis. A convolutional neural network (CNN), a type of deep neural networks, is mainly considered for research in computer vision field. The deep architecture of CNN is its main power. The convolutional layer in the CNN works as an automatic feature extractor that extracts hidden and important features. Extracted features are passed to a fully connected neural network which performs classification images by maximizing the probability scores. A general CNN model is shown in Figure 2.

3.2. Dataset and Computational Resources. We have used a publicly available malaria dataset from NIH (National Institute of Health) website originally used by a group of researchers, Rajaraman et al. [4], for the detection of malaria parasites in blood smear images. There are 27,558 segmented cell images in the dataset with the same number of normal and parasitized instances. Parasitized cell images contain Plasmodium while normal cells are free of Plasmodium but can contain other staining artifacts and impurities. The data was collected by Chittagong Medical College Hospital in Bangladesh by photographing slides of Giemsa-stained thin blood smear from 200 patients where three-fourth of them were P. falciparum-infected. The manual annotation and deidentification of these collected images were performed by an expert at Mahidol-Oxford Tropical Medicine Research Unit, Thailand, and later approved and archived by Institutional Review Board, National Library of Medicine.

The images in the dataset are not of equal sizes. The minimum and maximum image resolution is 46×46 and 385×395 pixels, respectively, with 3 color channels (RGB). We plan to resize the images to 224×224 which is the standard input image size of the majority of the pretrained CNN models for faster model convergence. Figure 3 shows some sample images from both normal and parasitized categories. The infected cells seem to contain some red globular structures whereas healthy cells do not seem to contain such structures in them. The proposed deep learning model will be used to identify these patterns in cell images to effectively detect malaria parasites in a patient.

Moreover, we performed data scaling which is a crucial preprocessing task for training and evaluating deep learning models. Data from input images without scaling often hampers a steady learning process. Normalization is one of the most common data scaling techniques which rescales the original data points in the images to a range between 0 and 1. The values of data points in the original 8-bit RGB

Table 1: Training and validation data sets.

Set	Count		Percent
	Parasitized	Normal	
Training	11023	11023	80%
Validation	2756	2756	20%

color images range from 0 to 255. Therefore, using Equation (1), we rescale our input image data as follows:

$$z = \frac{x - \min(x)}{\max(x) - \min(x)} = \frac{x}{255}. \quad (1)$$

We split the data (as shown in Table 1) into (training and validation) sets randomly with the percentage of 80% and 20%, respectively. There are 22,046 images in the training set and 5512 images in the validation set having an equal number of images from both classes.

The proposed CNN model is trained and evaluated using Google Colab [45] which is a cloud-based Jupyter notebook environment available for free access. Colab provides a preconfigured system for training and evaluating deep learning applications and offers access to high-performance graphical processing units (GPUs) without any cost. Presently, it offers a single 16GB NVIDIA Tesla P100 GPU with CUDA enabled, and all the necessary packages are preinstalled which includes Python 3 with Keras 2.2.5 API and TensorFlow 1.15.0 at the backend. In addition, we have used Android Studio 3.6.1 for developing the android malaria detection app for model deployment.

4. Model Configuration and Evaluation Metrics

4.1. Model Configuration. The proposed CNN model has four convolutional blocks and two fully connected dense layers. Figure 4 shows the proposed CNN model. Each convolutional block consists of convolution, max pooling, batch normalization, and dropout layers. The first convolutional layer uses 32 filters of size 7×7 to learn larger features, and then, the filter size decreases by 2 and filter count is doubled in each subsequent convolutional layer except for the last layer. The default striding of 1 pixel is used in convolution operations in all layers. The model input consists of segmented cell images of $224 \times 224 \times 3$-pixel resolution. In addition, each convolutional layer uses a valid padding to reduce the output feature map dimension in proportionate to the filter size used. We have used nonlinear activation function called ReLU (Rectified Linear Units) in all hidden layers to introduce nonlinearity into the output of each neuron to help the model learn complex mathematical functions to better identify target classes. It removes the vanishing gradient problem and aids in faster model training and convergence [46]. Max pooling layers have a 2×2-pixel pooling window and 2-pixel stride, added after convolutional layers to down sample the feature map by summarizing the most activated existence of a feature. This means that the pooling operation reduces the size of the feature map with a factor of two. To tackle the overfitting problem and to ensure more stability of the network, we have added a batch normalization layer to be applied to pooled output. Normalization is applied on the previous activation layer by subtracting the batch mean and then dividing by standard deviation of the batch [47]. The dropout regularization (with a dropout ratio of 0.15) used in each convolutional block reduces overfitting and improves network generalization error by randomly dropping out nodes during model training [34]. A global average pooling (GAP) layer is considered right after the last block of convolution as a better replacement of flattening to reduce overfitting by minimizing the size of model parameters. The GAP layer reduces spatial dimensions of a 3-dimensional tensor having size $h \times w \times d$ to $1 \times 1 \times d$ tensor by simply taking the average of all hw pixel values of each $h \times w$ feature map to single number [48]. The output from the GAP layer followed by a dropout is passed to the first (fully connected) dense layer having 1000 neurons. The first dense layer output is then fed to a dropout and then passed to the second dense layer with two neurons and a Softmax classifier. Overall, our proposed CNN model has relatively smaller size (409 K) of trainable parameters compared to most of the pretrained transfer learning models used in the literature [4] for malaria detection or solving similar computer vision problems. This simplicity of network structure will be the first step while dealing with overfitting problem.

We consider a stochastic gradient descent (SGD) optimizer with momentum for training and optimizing the model in order to minimize binary cross-entropic loss also known as log-loss. We have optimized our custom model by tuning the learning rate. Learning rate is one of the most dominating hyperparameters in a neural network configuration. We used an automatic optimal learning rate finder in combination with cyclical learning rate (CLR) technique first introduced by Smith [49] which allows the learning rate to oscillate cyclically between a minimum and a maximum learning rate bound. The use of CLR results in faster model convergence with fewer experiments and hyperparameter updates.

4.2. Cyclical Learning Rates. The widely used learning rate schedule technique monotonically decreases learning rate after each epoch to allow the model to descend to a point of low loss. However, with this technique, the model will still be sensitive to initial choice of learning rate and the technique does not guarantee that it will land to a low loss area while decreasing the learning rate. Rather, the model may be confined to either saddle points or local minima. To better address these problems, cyclical learning rates (CLR) enable oscillation of learning rates between upper and lower bounds which in turn provide additional freedom in choosing initial learning rate and get rid of saddle points and local minima. There are three variations of the CLR based on how the oscillation of learning rate takes place: *triangular*, *triangular2*, and *exp_range*. The *triangular* policy works by starting off from a base learning rate and increasing the rate to a maximum value in half cycle and then decreasing back to the initial rate thus completing the full cycle. This whole process is repeated until the model training is finished. The *triangular2* also called triangular schedule with fixed decay is similar to the previous one except that it cuts the

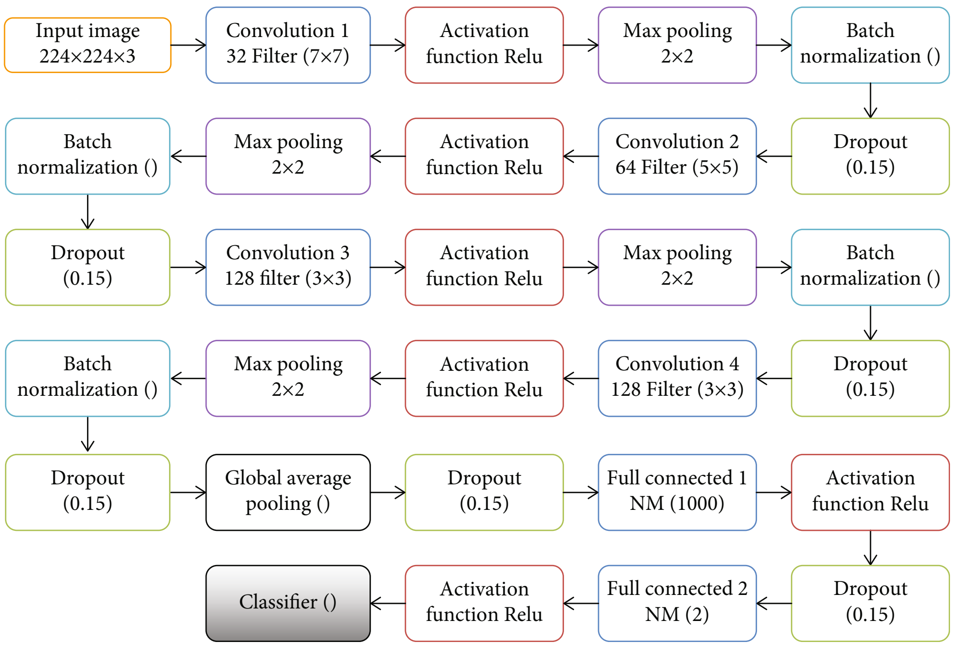

FIGURE 4: The custom CNN model.

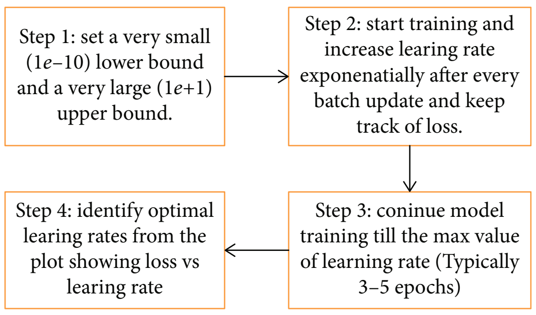
Step 1: set a very small (1e–10) lower bound and a very large (1e+1) upper bound.
Step 2: start training and increase learing rate exponenatially after every batch update and keep track of loss.
Step 3: coninue model training till the max value of learning rate (Typically 3–5 epochs)
Step 4: identify optimal learing rates from the plot showing loss vs learing rate

FIGURE 5: Steps to find optimal learning rates automatically.

upper bound of learning rate to half after every cycle. This lowering of maximum learning rate over time results in increased stability of model training. Finally, *exp_range* policy also called triangular schedule with exponential decay uses an exponential decay as the name suggests to cut down the upper bound which gives more fine-tuned control in decreasing the max learning rate over time. We have used Brad Kenstler's implementation of CLR using Keras for our model training [50].

4.3. Automatic Learning Rate Finder. Since CLR works based on a lower and upper bound of learning rate, Smith [49] also provides an automatic learning rate finder algorithm to find optimal learning rates. Various steps for obtaining the minimum and maximum values of learning rates in Figure 5. For model training, a very small ($1e-10$) value and a very large ($1e+1$) value for lower and upper bounds of learning rates are set by the algorithm. An exponential increase of learning rate after every batch update is adopted as training progresses, and at the same time, loss is also recorded. When the learning rate reaches the upper bound after a specific number of training epochs, we plot a curve showing loss and learning rate. At this point, we identify two different values for learning rates. The loss starts decreasing after the first learning rate, and the loss starts to increase from the second value of learning rate. These two values refer to the lower and upper bounds of learning rate which are used in CLR technique. Figure 6 demonstrates how the loss changes with respect to various learning rates using this automatic learning rate finder. It is apparent from the plot that loss does not change until the learning rate hits approximately $1e-6$. This indicates that our model does not start learning owing to a very low initial learning rate. Loss starts to decrease soon after the learning rate reaches approximately $1e-5$ which implies that the learning rate is large enough to enable the model to start learning. From this point, the loss keeps decreasing sharply implying that the model is learning quickly. Soon after the learning rate reaches to approximately $1e-1$, the loss starts to increase again. As such, the loss has exploded almost immediately due to the large increase in learning rate (close to $1e+1$). Hence, we select $1e-5$ and $1e-1$ as our minimum and maximum learning rates, respectively, which will be used in the CLR technique for our model training. Finally, our model configurations including hyperparameters are summarized in Table 2.

4.4. Evaluation Metrics. We have used accuracy, precision, recall, F1-score, specificity, Matthews correlation coefficient (MCC), and Area Under Curve (AUC) to evaluate the performance of our models. Since in our dataset the number of samples from each target class is equal, we consider accuracy as our primary metric. Accuracy refers to the proportion of correct predictions over all predictions made by the model. In addition, we calculated precision, recall or sensitivity, specificity, and F1-score from the confusion matrix which contains False Positives (FP), True Positives (TP), False

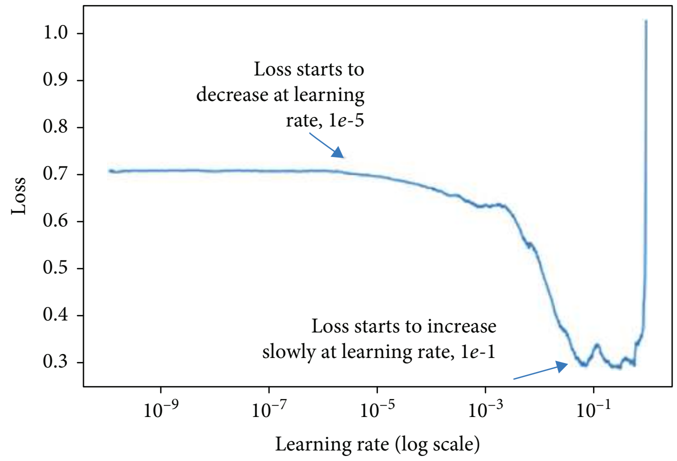

Figure 6: Model loss for various learning rates to identify optimal lower and upper bounds on learning rate.

Table 2: Model configuration summary including hyperparameters.

Parameter	Value/type
Epochs	50
Batch size	32
Optimizer	SGD with momentum 0.9
Learning rates	Min $1e-5$, max $1e-1$
Loss function	Categorical cross entropy
Input shape	224×224
Pooling	Max 2×2 (convolutional layers), GlobalAverage (flatten layer)
Activation	ReLU (convolutional layers), Softmax (final dense layer)
Dropout rate	0.15
Trainable parameters	409,146

Negatives (FN), and True Negatives (TN). Furthermore, precision and recall for each target class are calculated from a classification report. Precision measures the proportion of patients that are identified as infected really carry malaria parasites. Recall or sensitivity measures of the proportion of patients that are infected are diagnosed by the model as having malaria parasites. Specificity is the opposite of recall which measures the proportion of patients that are not infected and diagnosed by the model as not carrying any malaria parasites. F1-score is calculated as a single metric from the harmonic mean of precision and recall. MCC is computed from all four values of confusion matrix and represents the correlation coefficient between the true and predicted classes [51]. The higher the coefficient value, the better is the prediction. Equation (2) is used to calculate MCC for a binary classification problem. When all the predictions of the classifier are correct (i.e., FP = FN = 0), MCC becomes 1 implying the perfect positive correlation. On the contrary, if the predictions are always incorrect (i.e., TP = TN = 0), MCC becomes -1.

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}. \quad (2)$$

We have used binary cross entropy or log-loss, and the target is to minimize it which is equivalent to maximize the classification accuracy. The log-loss function is expressed with the following equation:

$$\mathrm{Loss} = \frac{1}{N}\sum_{i=1}^{N} y_i.\log\left(p(y_i)\right) + (1-y_i).\log\left(1-p(y_i)\right), \quad (3)$$

where y represents the target class (0 for normal cell images and 1 for parasitized cell images) and $p(y)$ is the probability of prediction of the sample being parasitized for all N images. For each parasitized image $(y=1)$, $\log(p(y))$ is added to the loss that is the log probability of its being parasitized. On the contrary, $\log(1-p(y))$ is added to the loss implying that the log probability of its being normal for each uninfected image $(y=0)$.

5. Results and Discussion

We have adopted the following approach in order to assess the performance of the proposed CNN model for the classification of uninfected and parasitized cell images. We took learning rate as one of the key hyperparameters to tune our custom CNN model for optimum classification performance. The learning rate hyperparameter is used to control the speed of learning of a deep learning model. Using a rightly constructed learning rate, a model can learn to best map input to desired output with the available resources (i.e., the number of nodes in each layer and the total number of layers) with the number of epochs passing in the training data. The SGD algorithm has long been the de facto optimizer to train deep neural networks. Moreover, some of its extensions based on adaptive learning rates have been popular for quite some time now such as Adam, RMSProp, and Adagrad. However, lately, the concept of cyclical learning rates

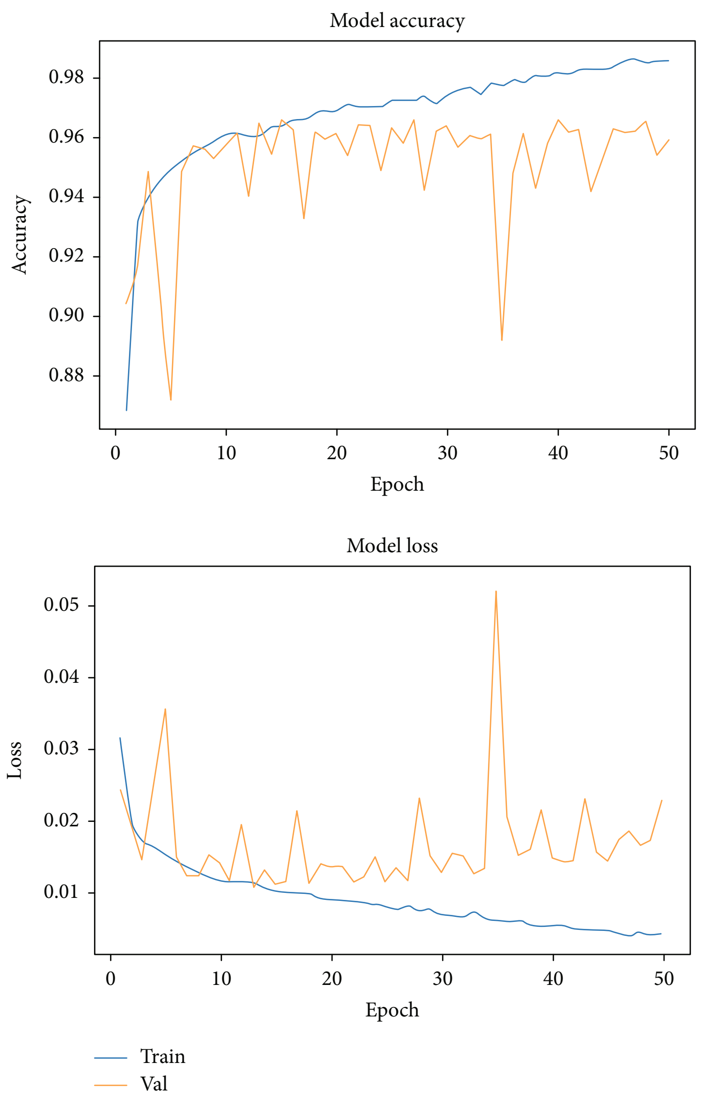

Figure 7: Loss (training and validation) and accuracy of the base model.

(CLR), originally proposed by Smith [49], attracted researchers' attention in improving deep learning model performance. In this paper, we looked at how SGD with this cyclical learning schedule holds up to the other optimizers. To this end, we built a baseline model with standard SGD and then gradually tried to improve the model's performance with CLR technique applied to SGD.

5.1. Base Model. As mentioned in the previous section, our base model represents the custom model as described before with a standard SGD optimizer. We have considered SGD with 0.9 momentum, $1e-1$ as an initial learning rate, and standard decay of *initial_learning_rate/no_of_epochs*. We saved the best model weights (i.e., the lowest validation loss) during training by using Keras's *ModelCheckpoint* library and *callback* function. We trained the model for 50 epochs. Resulted training and validation loss are shown in Figure 7 as well as accuracy over the number of epochs.

We can see that our base model does not converge well and a significant difference between the training and validation results both for loss and accuracy. In addition, a lot of fluctuation is observed in the values of loss and accuracy as the training progresses towards the end. This indicates that our base model is not trained well and might be overfitting to training data. Consequently, the model might not generalize well on unseen test data. This could be potentially attributed to the choice of learning schedule in the base model even though we have used dropout and batch normalization techniques to avoid overfitting. We aim to overcome these drawbacks by using CLR schedule with the SGD optimizer.

Table 3: Performance metrics for the base model.

Acc	AUC	Precision	Recall	F1-score	MCC
0.9646	0.9552	0.97	0.96	0.96	0.9135

Performance metrics of our base model is shown in Table 3. We have received a base accuracy of 96.46% with high precision and recall towards classifying the infected

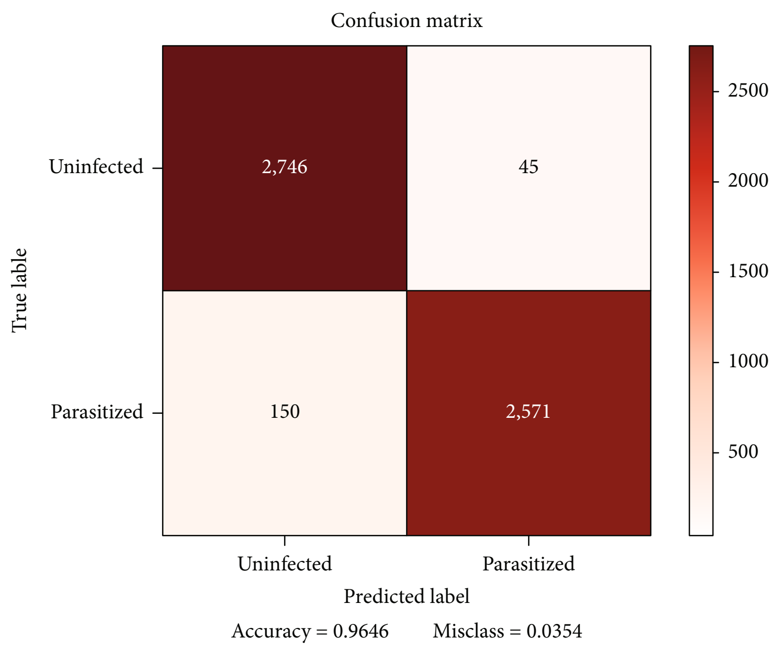

FIGURE 8: Confusion matrix for the *base model*.

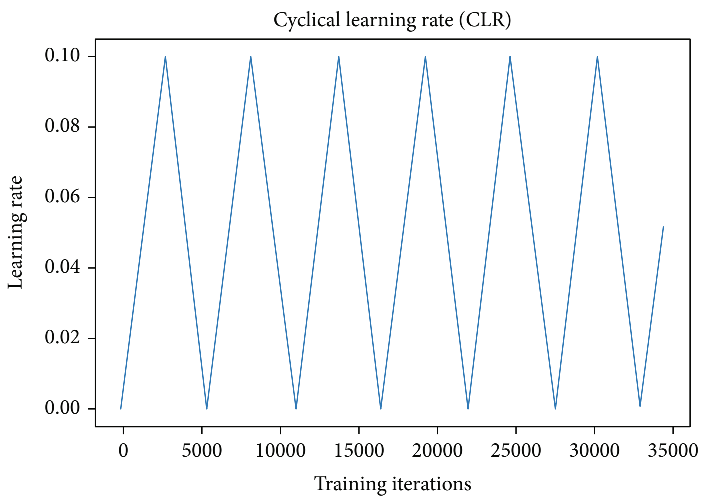

FIGURE 9: Cyclical learning rate changes using "triangular" policy. Lower and upper bounds on learning rates were calculated using an automatic learning rate finder.

and normal cells which is reasonable. By investigating the confusion matrix as shown in Figure 8, we can see that the count for False Negatives (FN) is 150 which is pretty high for a disease identification problem. FN indicates that the model declares a malaria patient to be healthy whereas the patient is parasitized. This will severely hamper the patient treatment and may result in death. Our goal is to reduce this number with the proposed improved model. A reduced number of FN will ensure that our model is effective in identifying parasitized cell images.

5.2. Improved Model with CLR. In our improved model, we used the same base model architecture with the exception that we used cyclical learning rates schedule instead of a standard one. As mentioned before, there are three variations of CLR implementation based on the policy of changing the upper bound learning rate, namely, *triangular*, *triangular2*, and *exp_range*. We have experimented with the first two variants to observe the model performance. Figure 9 shows the learning rate plot and how the learning rates oscillate between the lower and upper bounds.

More specifically, the initial learning rate of $1e-5$ increases to the maximum value of $1e-1$ in a half cycle and then decreases back to $1e-5$ in the other half cycle thus completing the full cycle. By using this triangular policy, we have obtained improved model accuracy of 97.12% as shown in Table 4 (compared to 95.25% in base model) with higher precision and recall towards classifying the infected and

Table 4: Performance metrics for the improved model with CLR schedule.

Model	Base	CLR-triangular	CLR-triangular2
Accuracy	0.9646	0.9712	**0.9730**
AUC	0.9552	0.9656	**0.9704**
Precision	0.97	0.97	**0.97**
Recall	0.96	0.97	**0.97**
F1-score	0.96	0.97	**0.97**
MCC	0.9135	0.9400	**0.9417**

normal cells. Thus, by combining cyclical learning rates with the automatic learning rate finder (discussed earlier), we are successful in obtaining a highly accuracy model.

By looking at the training history (plotted in Figure 10), we found that the gap between training and validation loss as well as accuracy reduces significantly indicating a faster and better model convergence. In addition, we observed a "wave" characteristic of our training and validation accuracy/loss curve signifying the fact that the learning rate oscillates between lower and upper bounds.

We train and evaluate our model again with "triangular2" CLR policy and found out further improvement in model accuracy which is 97.30%. Figure 11 visualizes how learning rate is adapted in a cyclic manner and, after each fully cycle, the upper bound learning rate is reduced to half and this continues till the end of model training. Compared to the first triangular CLR policy, the training and validation curves with loss and accuracy (as plotted in Figure 12) show less fluctuation and more stability. In principle, a stabilized training is less prone to the risk of overfitting.

Table 5 shows the confusion matrix for our improved model with *triangular2* CLR schedule. As mentioned earlier, our target is to reduce the FN count to make our model robust. We can see the FN count decreased to 112 compared to the base model (with FN count of 150) which makes our improved model effective in identifying parasitized cell images. This reduced count of FN is very critical because we do not expect that our model will misidentify someone as healthy while in reality the patient is carrying the malaria parasite. This will severely hamper the patient's line of treatment and even endanger life of the patient. At the same time, FP count also decreased (to 37) compared to the base model (45). A lower value FP is also expected from our model since this will prevent the patient from further undergoing unnecessary laboratory tests and treatment and will reduce financial burden on the health provider.

5.3. Mobile-Based Model Deployment. We have deployed our best improved model to a mobile application to facilitate a simple and fast detection of malaria parasite in blood cell images. We have used Google's TensorFlow Lite [52] which brings deep learning capability directly into mobile devices by running deep learning models locally. TensorFlow Lite framework supports hardware acceleration and brings low-latency inference performance to mobile devices by significantly improving model loading times. Figure 13 shows different steps of our model deployment process. Our best trained Tensorflow model is converted to a TensorFlow Lite (*.tflite*) model using the TFLite Converter. We have used the TFLite Converter from a Python API which simplifies the model conversion as part of a model deployment pipeline. Our converted *.tflite* mode size is about 22 MB. Once the converted (*.tflite*) model is deployed on the android mobile device, cell images are loaded from a cloud or device's local storage for potential malaria detection. The user opens a cell image, and the deployed model provides the prediction label. Snapshots of a few sample image predictions are displayed in Figure 14.

6. Discussion

From the performance results obtained in the previous section, we observed that the proposed custom model with *CLR-triangular2* configuration produces an optimal solution with faster convergence. This was achieved by selecting a superior combination of convolutional and dense layers in the custom CNN architecture with proper hyperparameter optimization such as learning rate, regularization through batch normalization, and moderate dropouts in convolutional and dense layers. The use of cyclical learning rate schedule with an automatic learning rate finder lowered the effect of model being overfit to training data and faster convergence to a better solution.

Our base model with no or standard learning rate schedule did not converge well and showed high variance in the values of loss and accuracy during the model training indicating a tendency to overfit to training data. We have addressed this problem by using cyclical learning rate schedule in our SGD optimizer along with implicit regularization techniques using batch normalization and dropouts. The use of two different variations of cyclical learning rate implementation, namely, triangular with no decay (triangular) and triangular with fixed decay (trianglular2), progressively improves the performance of the base model with respect to model accuracy, AUC, sensitivity (recall), and MCC. Our best improved model yields a performance accuracy of 97.30% compared to the base model's accuracy of 95.57%. A noticeable increase in the value of MCC (94.17% from the base model's MCC of 91.35%) indicates that the predicted label and the true label are strongly correlated and our improved model is competent in classifying parasitized and uninfected cell images. An increased value of AUC (97.04%) represents a high degree of separability of our improved model meaning that it can better distinguish between cell images with malaria disease and no disease. In addition, a high recall value of 97% indicates model sensitivity in predicting infected cells with malaria. Furthermore, the number of false negative (FN) cases significantly (almost half) decreased in our best improved model compared to the base model which indicates the success of our model in reducing the risk of identifying a malaria patient as healthy which is detrimental to a patient's line of treatment. It is worth mentioning here that our model did not consider using data augmentation to artificially increase the size of the training dataset largely owing to the fact that our dataset contains a decent number (27,558) of segmented cell images with the same number of normal

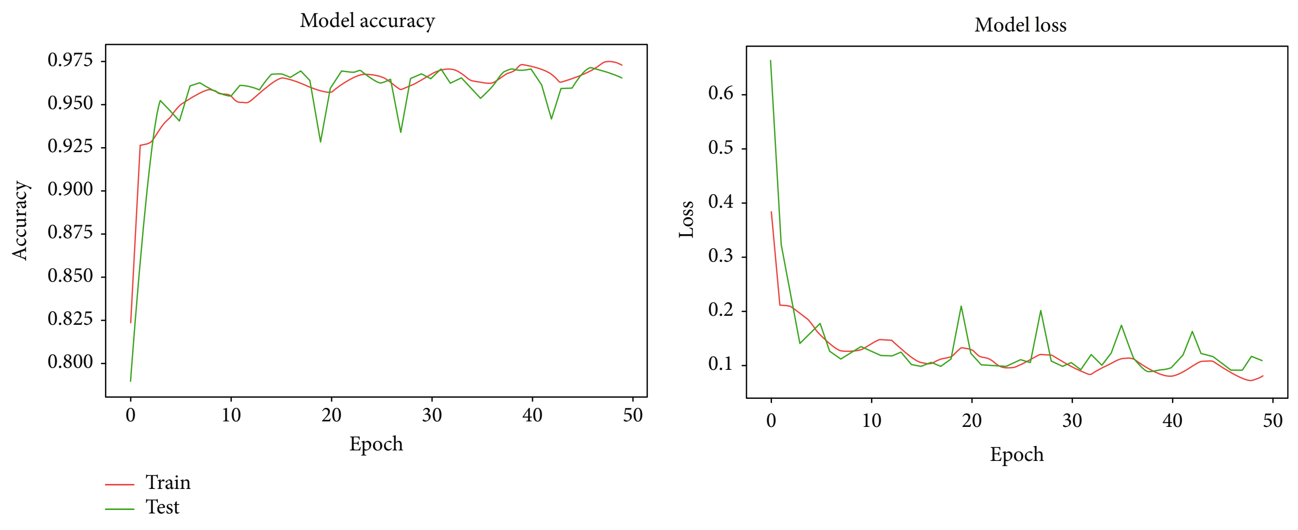

Figure 10: Training and validation loss and accuracy with *triangular* CLR policy.

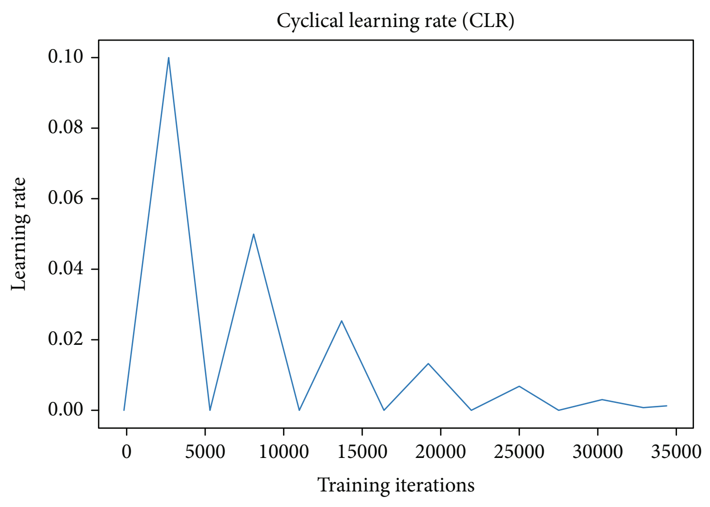

Figure 11: Cyclical learning rate changes using "triangular2" policy.

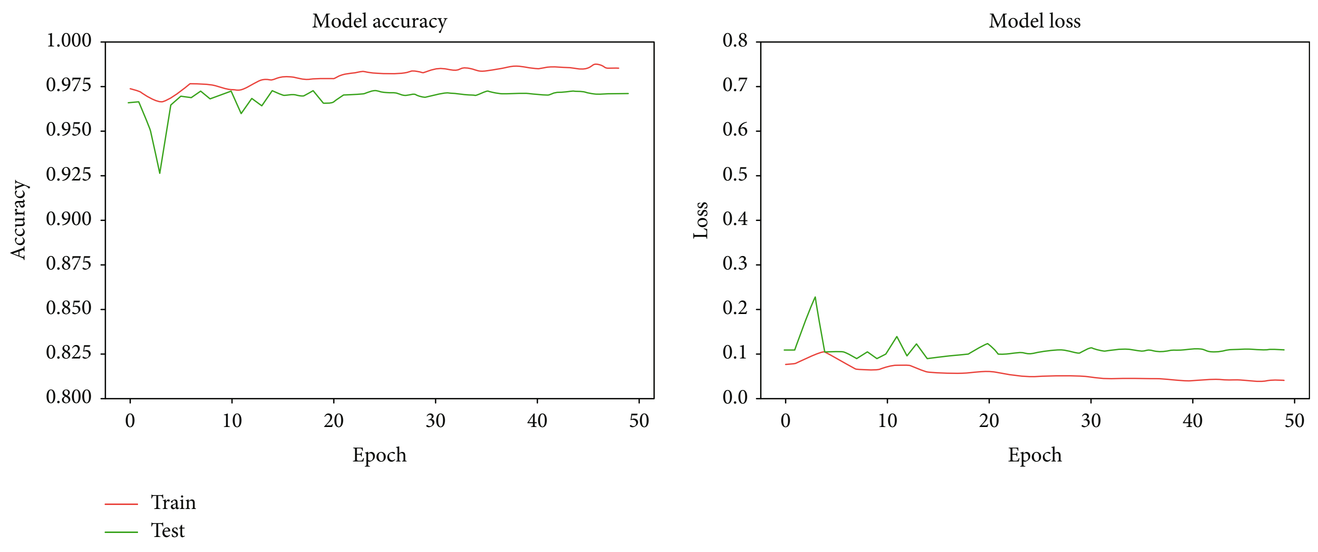

Figure 12: Training and validation loss and accuracy with *triangular2* CLR policy.

Table 5: Confusion matrix for the improved model with *triangular2* CLR schedule.

Model	TP	TN	FN	FP
Base	2746	2571	150	45
CLR-triangular	2752	2601	120	39
CLR-triangular2	2754	2609	112	37

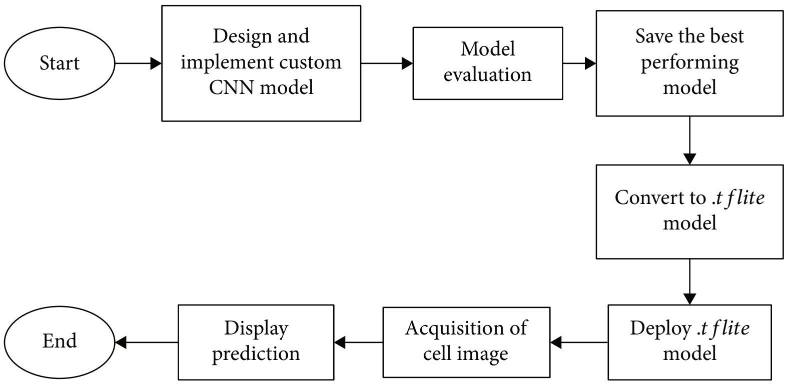
Start
Design and implement custom CNN model
Model evaluation
Save the best performing model
Convert to .*tflite* model
Deploy .*tflite* model
Acquisition of cell image
Display prediction
End

Figure 13: Process flow diagram showing different steps of model deployment in a mobile device.

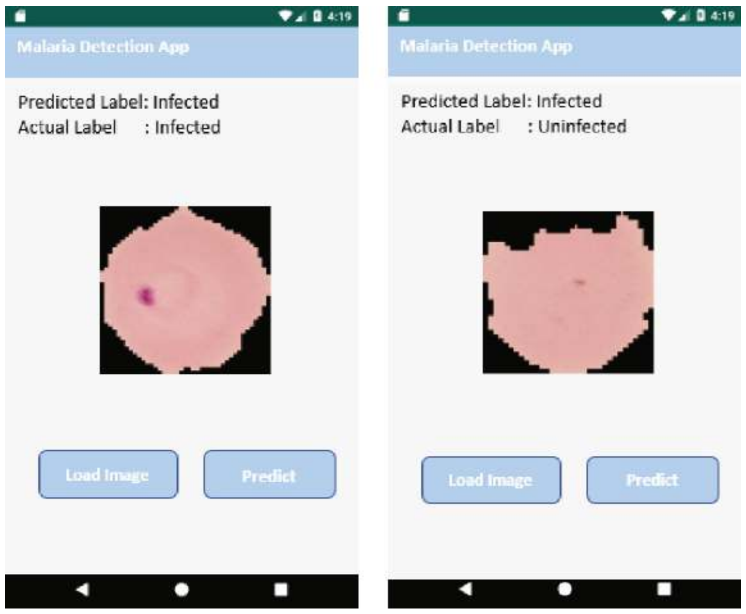

Figure 14: Snapshots of the mobile app displaying predictions on actual cell images.

and parasitized instances well enough to train a deep learning model without significantly running into overfitting problem. Hence, we trained our model without artificially augmenting our dataset and yet obtained better or comparable performance to the techniques in the literature in identifying a malaria patient.

Table 6 provides a comparison of performance metrics between our best improved model and the results of state-of-the-art approaches. We noticed that the proposed improved model is better than the customized model and other CNN models (pretrained) such as VGG-16 and ResNet-50 presented in [4] with respect to accuracy, precision, sensitivity, and MCC towards classifying healthy and infected cells with malaria. On the contrary, our proposed custom model achieved a relatively lower value for AUC as compared to ResNet-50 and VGG-16 but demonstrated similar AUC performance as the customized model proposed in [4]. Our model took about 97 min to train as compared to the training time (24 hours) of all the models proposed in [4]. We believe that this performance improvement is worth given the fact that our model is smaller in size having a relatively less number of trainable parameters and demonstrated very less training time using the SGD optimizer with a cyclical learning rate schedule.

Table 6: Comparison of performance of proposed and the state-of-the-art approaches.

Model	Accuracy	AUC	Precision	Recall (sensitivity)	F1-score	MCC
Proposed model (CLR-triangular2)	0.9730	0.9704	0.97	0.97	0.970	0.9417
Rajaraman et al. Customized model [4]	0.9400	0.9790	0.951	0.931	0.941	0.880
Rajaraman et al. ResNet-50 [4]	0.9570	0.9900	0.969	0.945	0.957	0.912
Rajaraman et al. VGG-16 [4]	0.9450	0.9810	0.951	0.939	0.945	0.887
Gopakumar et al. [26]	0.9770	—	0.985	0.971	0.977	0.731
Bibin et al. [16]	0.963	—	0.959	0.976	0.967	—
Liang et al. [12]	0.973	—	0.977	0.969	0.972	—

In contrast to the pretrained models presented in [4], the CNN-based classifier proposed by Gopakumar et al. [26] demonstrated slightly better results in classifying parasitized and uninfected cells in terms of accuracy (97.70%), precision (98.5%), and recall (97.1%) but showed a very low MCC (73.1%) value which is considered a very informative consolidated score for evaluating a binary classifier's performance representing the correlation between the predicted and true classes [51]. Liang et al. [12] have also proposed a technique for image analysis using a CNN for malaria detection. They have achieved similar accuracy (97.3%) as our improved model with a slight increase in precision (97.7%) and slightly degraded sensitivity (96.9%) as compared to our improved model. Finally, malaria parasite detection using a deep belief network done by Bibin et al. [16] did not demonstrate promising results as compared to other studies in the literature including our improved model. Based on the preceding discussion, our model is greatly specific with a large MCC value and performs pretty better than the majority of the pretrained and custom CNN models under study.

7. Conclusions and Future Work

The paper first evaluated a custom CNN-based end-to-end deep learning model to improve malaria detection on thin-blood smear images. We showed that the use of cyclical learning rate schedule with an automatic learning rate finder in addition to the use of a commonly applied regularization technique such as batch normalization and dropouts produces promising results in malaria classification. Our best model achieves an accuracy of 97.30% in classifying parasitized and uninfected cell images with a high degree of precision and sensitivity. The model also yields a high value of MCC (94.17%) compared to all other existing models under study indicating a strong correlation between predicted and true labels. We also observed that the proposed improved model showed better performance compared to the customized and other CNN models (pretrained such as VGG-16 and ResNet-50) [4] with respect to accuracy, precision, sensitivity, and MCC towards classifying healthy and infected cells with malaria. We deployed our best performing model into an android-based mobile application to facilitate simpler and faster malaria detection. Thus, we believe that the results obtained from this work will benefit towards developing valuable mobile-based solutions so that reliability of the treatment and lack of medical expertise can be solved. As an immediate extension of this work, we will consider using image augmentation on the training data with the hope to further alleviate overfitting problem and different adaptive variants of the SGD optimizer to observe their impact on the performance results. In the future, we also plan to achieve better prediction by using ensemble methods through model stacking.

Data Availability

Data set is collected from Kaggle (https://www.kaggle.com/iarunava/cell-images-for-detecting-malaria) [53].

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This study was funded by the Deanship of Scientific Research, Taif University, KSA (Research Project number: 1-440-6146).

References

[1] K. S. Makhija, S. Maloney, and R. Norton, "The utility of serial blood film testing for the diagnosis of malaria," *Pathology*, vol. 47, no. 1, pp. 68–70, 2015.

[2] WHO, *Malaria Micropscopy Quality Assurance Manual*, World Health Organization, 2016.

[3] "Our Malaria World Map of Estimated Risk (2018 update)," https://www.treated.com/malaria/world-map-risk.

[4] S. Rajaraman, S. K. Antani, M. Poostchi et al., "Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images," *PeerJ*, vol. 6, article e4568, 2018.

[5] C. Mehanian, M. Jaiswal, C. Delahunt, C. Thompson, M. Horning, and L. Hu, "Computer-automated malaria diagnosis and quantization using convolutional neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 116–125, Venice, Italy, 2017.

[6] E. Var and F. B. Tek, "Malaria parasite detection with deep transfer learning," in *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, pp. 298–302, Sarajevo, Bosnia-Herzegovina, September 2018.

[7] A. Vijayalakshmi and B. Rajesh Kanna, "Deep learning approach to detect malaria from microscopic images," *Multimedia Tools and Applications*, vol. 79, 2019.

[8] Y. Souri, E. Noury, and E. Adeli, "Deep relative attributes," in *Computer Vision – ACCV 2016*, pp. 118–123, Springer, 2016.

[9] M. I. Razzak, "Malarial parasite classification using recurrent neural network," *Journal of Image Processing (IJIP)*, vol. 9, no. 2, 2015.

[10] H. Shen, W. D. Pan, Y. Dong, and M. Alim, "Lossless compression of curated erythrocyte images using deep autoencoders for malaria infection diagnosis," in *2016 Picture Coding Symposium (PCS)*, pp. 1–5, Nuremberg, Germany, December 2016.

[11] I. Mohanty, P. A. Pattanaik, and T. Swarnkar, "Automatic detection of malaria parasites using unsupervised techniques," in *International Conference on ISMAC in Computational Vision and Bio-Engineering*, pp. 41–49, Springer, Cham, 2018.

[12] Z. Liang, A. Powell, I. Ersoy et al., "CNN-based image analysis for malaria diagnosis," in *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 493–496, Shenzhen, China, December 2016.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, no. 2, pp. 1097–1105, 2012.

[14] H. Jane and A. Carpenter, "Applying faster R-CNN for object detection on malaria images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 56–61, Honolulu, HI, USA, 2017.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: a large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, Miami, FL, USA, June 2009.

[16] D. Bibin, M. S. Nair, and P. Punitha, "Malaria parasite detection from peripheral blood smear images using deep belief networks," *IEEE Access*, vol. 5, pp. 9099–9108, 2017.

[17] X. Yang, T. Zhang, C. Xu, S. Yan, M. S. Hossain, and A. Ghoneim, "Deep relative attributes," *IEEE Transactions on Multimedia*, vol. 18, no. 9, pp. 1832–1842, 2016.

[18] M. S. Hossain, S. U. Amin, M. Alsulaiman, and G. Muhammad, "Applying deep learning for epilepsy seizure detection and brain mapping visualization," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 15, no. 1s, pp. 1–17, 2019.

[19] M. I. Razzak and S. Naz, "Microscopic blood smear segmentation and classification using deep contour aware CNN and extreme machine learning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 801–807, Honolulu, HI, USA, July 2017.

[20] V. Kantorov, M. Oquab, M. Cho, and I. Laptev, "Contextlocnet: context-aware deep network models for weakly supervised localization," in *Computer Vision – ECCV 2016*, pp. 350–365, Springer, 2016.

[21] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[22] M. Masud, M. S. Hossain, and A. Alamri, "Data interoperability and multimedia content management in e-health systems," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 6, pp. 1015–1023, 2012.

[23] O. B. Leal Neto, C. M. Albuquerque, J. O. Albuquerque, and C. S. Barbosa, "The schisto track: a system for gathering and monitoring epidemiological surveys by connecting geographical information systems in real time," *JMIR Mhealth Uhealth*, vol. 2, no. 1, article e10, 2014.

[24] S. Kaewkamnerd, C. Uthaipibull, A. Intarapanich, M. Pannarut, S. Chaotheing, and S. Tongsima, "An automatic device for detection and classification of malaria parasite species in thick blood film," *BMC Bioinformatics*, vol. 13, Supplement 17, p. S18, 2012.

[25] D. Anggraini, A. S. Nugroho, C. Pratama, I. E. Rozi, A. A. Iskandar, and R. N. Hartono, "Automated status identification of microscopic images obtained from malaria thin blood smears," in *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, Bandung, Indonesia, July 2011.

[26] G. P. Gopakumar, M. Swetha, G. S. Siva, and G. R. K. Sai Subrahmanyam, "Convolutional neural network-based malaria diagnosis from focus stack of blood smear images acquired using custom-built slide scanner," *Journal of Biophotonics*, vol. 11, no. 3, 2018.

[27] "MOMALA," https://momala.org/malaria-diagnosis/.

[28] "This New App Helps Doctors Diagnose Malaria in Just 2 Minutes," https://www.globalcitizen.org/en/content/app-diagnose-malaria-uganda.

[29] N. E. Ross, C. J. Pritchard, D. M. Rubin, and A. G. Dusé, "Automated image processing method for the diagnosis and classification of malaria on thin blood smears," *Medical & Biological Engineering & Computing*, vol. 44, no. 5, pp. 427–436, 2006.

[30] D. K. Das, M. Ghosh, M. Pal, A. K. Maiti, and C. Chakraborty, "Machine learning approach for automated screening of malaria parasite using light microscopic images," *Micron*, vol. 45, pp. 97–106, 2013.

[31] M. Poostchi, K. Silamut, R. J. Maude, S. Jaeger, and G. R. Thoma, "Image analysis and machine learning for detecting malaria," *Translational Research*, vol. 194, pp. 36–55, 2018.

[32] M. S. Hossain, M. Al-Hammadi, and G. Muhammad, "Automatic fruit classification using deep learning for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1027–1034, 2019.

[33] M. Usama, B. Ahmad, J. Wan, M. S. Hossain, M. F. Alhamid, and M. A. Hossain, "Deep feature learning for disease risk assessment based on convolutional neural network with intra-layer recurrent connection by using hospital big data," *IEEE Access*, vol. 6, pp. 67927–67939, 2018.

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[35] Y. Zhang, Y. Qian, D. Wu, M. Shamim Hossain, A. Ghoneim, and M. Chen, "Emotion-aware multimedia Systems security," *IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 617–624, 2019.

[36] Y. Zhang, X. Ma, J. Zhang, M. S. Hossain, G. Muhammad, and S. U. Amin, "Edge intelligence in the cognitive internet of things: improving sensitivity and interactivity," *IEEE Network*, vol. 33, no. 3, pp. 58–64, 2019.

[37] X. Ma, R. Wang, Y. Zhang, C. Jiang, and H. Abbas, "A name disambiguation module for intelligent robotic consultant in

industrial internet of things," *Mechanical Systems and Signal Processing*, vol. 136, article 106413, 2020.

[38] Y. Zhang, M. S. Hossain, A. Ghoneim, and M. Guizani, "COCME: content-oriented caching on the mobile edge for wireless communications," *IEEE Wireless Communication*, vol. 26, no. 3, pp. 26–31, 2019.

[39] J. Wang, Y. Miao, P. Zhou, M. S. Hossain, and S. M. M. Rahman, "A software defined network routing in wireless multihop network," *Journal of Network and Computer Applications*, vol. 85, pp. 76–83, 2017.

[40] A. K. Sangaiah, D. V. Medhane, T. Han, M. S. Hossain, and G. Muhammad, "Enforcing position-based confidentiality with machine learning paradigm through mobile edge computing in real-time industrial informatics," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4189–4196, 2019.

[41] M. A. Rahman, M. M. Rashid, M. Shamim Hossain, E. Hassanain, M. F. Alhamid, and M. Guizani, "Blockchain and IoT-Based Cognitive Edge Framework for Sharing Economy Services in a Smart City," *IEEE Access*, vol. 7, pp. 18611–18621, 2019.

[42] M. F. Alhamid, M. Rawashdeh, H. Al Osman, M. S. Hossain, and A. El Saddik, "Towards context-sensitive collaborative media recommender system," *Multimedia Tools and Applications*, vol. 74, no. 24, pp. 11399–11428, 2015.

[43] Y. Zhang, Y. Li, R. Wang, M. S. Hossain, and H. Lu, "Multi-Aspect Aware session-based recommendation for intelligent transportation services," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2020.

[44] Y. Zhang, R. Wang, M. S. Hossain, M. F. Alhamid, and M. Guizani, "Heterogeneous information network-based content caching in the internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 10216–10226, 2019.

[45] "Google Colab," https://colab.research.google.com/.

[46] W. Shang, K. Sohn, D. Almeida, and H. Lee, "Understanding and improving convolutional neural networks via concatenated rectified linear units," in *Proc of 33rd international conference on machine learning (ICML2016)*, vol. 48, pp. 2217–2225, New York, USA, 2016.

[47] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proc. of the 32nd International Conference on Machine Learning*, vol. 37, pp. 448–456, Euralille Lille, France, 2015.

[48] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proc. of International Conference on Learning Representations (ICLR)*, Scottsdale, AZ, USA, 2013https://arxiv.org/abs/1312.4400.

[49] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472, Santa Rosa, CA, USA, March 2017.

[50] B. Kenstler, "Cyclical Learning Rates Implementation," https://github.com/bckenstler/CLR.

[51] B. W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.

[52] "TensorFlow Lite- Deploy machine learning models on mobile and IoT devices," https://www.tensorflow.org/lite.

[53] "Malaria Cell Images Dataset," https://www.kaggle.com/iarunava/cell-images-for-detecting-malaria.