Malaria Identification Using a Deep Learning Approach Convolution Neural Network

Mashrur Hossain Khan
2016-1-60-006
Department of CSE
2016-1-60-006@std.ewubd.edu

Sadia Tasneem
2016-1-60-074
Department of CSE
2016-1-60-074@std.ewu.bd

Nishat Jahan Nishi
2016-1-60-015
Department of CSE
2016-1-60-015@std.ewu.bd

***Abstract*—Malaria is a serious and sometimes fatal disease caused by a parasite that commonly infects a certain type of mosquito which feeds on humans. It is a life-threatening disease and is a concern of global health threat. Malaria parasites can be identified by examining under the microscope a drop of the patient's blood and spread out as a "blood smear" on a microscope slide. In this paper we try to propose a custom convolutional Neural Network model for detection of malaria on blood smear slide images. The images are available on website of kaggle.The training and testing were performed on the 13515 single cell images.One of the model achieve 95.01 percent accuracy in training and 91.95 percent accuracy on the test data.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Half of the world's population are at risk for contracting malaria, with an estimated 212 million cases in 2015[1]. Malaria causes symptoms that typically include fever, tiredness, vomiting, and headaches. In severe cases, it can cause yellow skin, seizures, coma, or death[2]. The World Health Organization (WHO) encourages regular monitoring of antimalarial efficacy in malaria-endemic countries. A majority of the 429,000 deaths from malaria in 2015, mostly of young children[3]. For the identification of the state, the proposed malaria identification method is based on morphological processing. Inside the initial convolution layers, features like irregular shape and size are extracted. All research studies are based on image statistics obtained from digital cameras combined with high-end mild lenses. Microscopy can detect low density infections if enough blood is scanned, but this is time-consuming, difficult, and tedious due to the low density and small size of parasites as well as the abundance of similar non-parasite objects[4].

A. Objectives

a. Examine the difference between infected and uninfected malaria cell b. Analyze malaria image using CNN and find out the cell is infected or not c. Evaluate the CNN model with unseen images to the model

B. Motivation

In most cases, only manual examination of the microscopic slide can diagnose malaria. Whole slide imaging, which scans the conventional glass slides in order to produce digital slides, is the most recent imaging modality being employed by pathology departments worldwide [5]. In order to provide a reliable diagnosis, necessary training and specialized human resource are required. Unfortunately, these resources are far from being adequate and frequently often unavailable in underdeveloped areas where malaria has a marked predominance. Therefore, an automated diagnostic system can provide an efficient solution to this problem. Machine learning algorithms have recently gained growing attention from researchers for their excellent ability to develop an automated malaria diagnostic system. SVM and Naive Bayes Classifier were utilized to achieve accuracies of 84%nd 83.5%espectively. In contrast to the supervised learning, unsupervised learning, for instance, K-Nearest Neighbor (KNN) has also been proposed to recognize malaria infected cells[6].

II. Related Work

A study by Rajaraman applied pre-trained deep learning models on the classification task of differentiating parasitised and uninfected red blood cells in thin blood smear images. The study compared different models including AlexNet, VGG-16, Xception, ResNet-50 and DenseNet-121 for the classification task. A similar work of classifying parasitised and uninfected red blood cells in thin blood smears using pre-training deep models like LeNet, AlexNet and GoogleNet is done by Dong Y. Another work of comparing the best suited model from faster R-CNN, SSD, RetinaNet on a dataset of thick blood smear images is done by Nakasi. Delgado used a novel three-stage pipeline involving two neural networks, a SNN and a CNN, to detect malaria in digital images of whole-slide peripheral blood smears. The three stage pipeline was RBC segmentation, RBC cropping and masking and the binary classification of RBCs into infected or non-infected.

III. Methodology

In this section we describe our working process of recognizing nurse care activities. Our method consists of preprocessing, imputation, data segmentation, standardization, training and testing with a CNN model.

A. Processing

As raw data was given a lot of preprocessing needed before final recognition process. Dataset was given such a way, that

Identify applicable funding agency here. If none, delete this.

labels of the activities need to be identified manually. As a result a lot of data remain unlabeled. We followed a basic imputation strategy to label the unlabeled data.

B. Segmentation

As raw data was given a lot of preprocessing needed before final recognition process. Dataset was given such a way, that labels of the activities need to be identified manually. As a result a lot of data remain unlabeled. We followed a basic imputation strategy to label the unlabeled data.

C. Standarization

After the segmentation we standardized each window. So that our CNN model can converge easily. We standardized the dataset using equation,

$$z = x - u\sigma \tag{1}$$

where x is the given value of an axis from the accelerometer data, u is mean of that axis and sigma is the standard deviation.

D. Convolution Neural Network

In this section, we discuss about the overall structure of convolutional neural networks and later we describe our proposed system architecture. The idea of CNN is basically inspired by biological visual system where the structure follows hierarchical feed-forward network. CNN is constructed with the help of multiple layers, mainly include convolutional and pooling layers which are lately followed by fully connected layers. Working mechanism and overall construction of CNN is demonstrated below:

1) Convolution layer: Convolutional layer is made up with a numbers of filters whose parameters have to be trained. Filters height and weight are smaller than input shape. Filters are sliding through the entire input region and at each step.If the input size is n×n and f×f kernel is used, then the output is the following equation.

$$(n \times n)(f \times f) = (n - f + 1)(n - f + 1) \tag{2}$$

2) Pooling Layer: Pooling layer is also known as down-sampling layer which reduces the size of the activation maps. Usually convolutional layer is followed by pooling layers in order to lessen the computational requirements through the entire network. Max pooling and Average pooling are considered as two most commonly used methods for reduction. Max pooling basically finds out the maximum value in the particular window region and rest of the values are lessened. On the contrary, mean values within the region are used in average pooling.

3) Activation Function (ReLU): Rectified Linear Unit is a nonlinear activation function which is commonly used in neural networks, especially in CNN. Convolutional layer is usually followed by ReLU function for learning the non-linear decision boundaries. For all positive values, it is linear and zero for the negative values. Equation 2 represents relu activation function. Hyperbolic tangent and sigmoidal activation function can also be used as the substitute of ReLU.

$$\sigma(z) = 11 + e - (z) \tag{3}$$

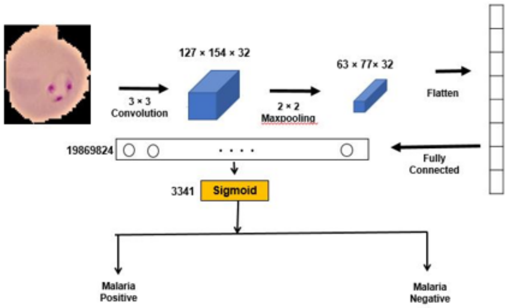


Fig. 1.

4) Flatten layer: Flatten layer exists in between the convolution and the fully connected layer, which actually transforms the multidimensional matrix into a vector. This vector is then passed to the fully connected layer

5) Fully Connected Layer: Finally, fully connected layers, also known as dense layers processes the output of previous layer results for classification. An activation function called SoftMax is used to calculate the probability distribution of the predicted classes. The softMax function is described in equation (3)

$$\sigma(z)i = ezi \ln j = 1ezj \tag{4}$$

Where n = number of classes, z = input vector,

$$\sigma(z)i = classprobability \tag{5}$$

The value of the vector component can be negative or greater than 1 and it is not possible to get the sum of 1. SoftMax function returns all classes probability distribution where each class probability lies between 0 and 1 and the output probability sum is 1. SoftMax plays a big role in multiclass classification because it helps to converge in the fastest way.

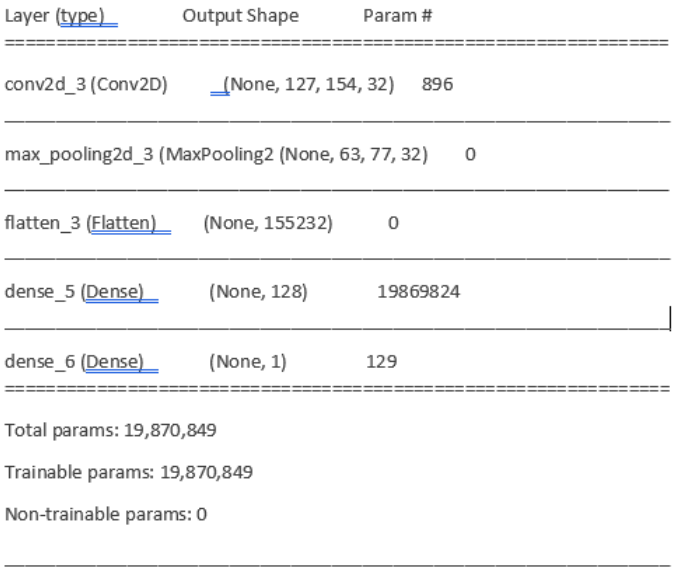
IV. PROPOSED ARCHITECTURE

In this section, first we present the proposed architectures to classify images to detect malaria .Secondly, we give a brief outline of the dataset used in this paper and the various steps which have been applied before training the proposed architectures.

A. System Architecture

In this paper, we have used two-dimensional convolutional neural networks to implement the model. We have used 32 convolutional layers and the kernel size was 3×3. 2×2 maxpooling layer is used in the next layer. We have used flatten followed by fully connected layers. We have used relu function in the layers and sigmoid activation function in the dense layer. We have used adam as optimizer and binary crossentropy as loss function.

We have proposed two models of CNN. One of our proposed system architecture of Convolutional Neural Networks is displayed in figure 1 .

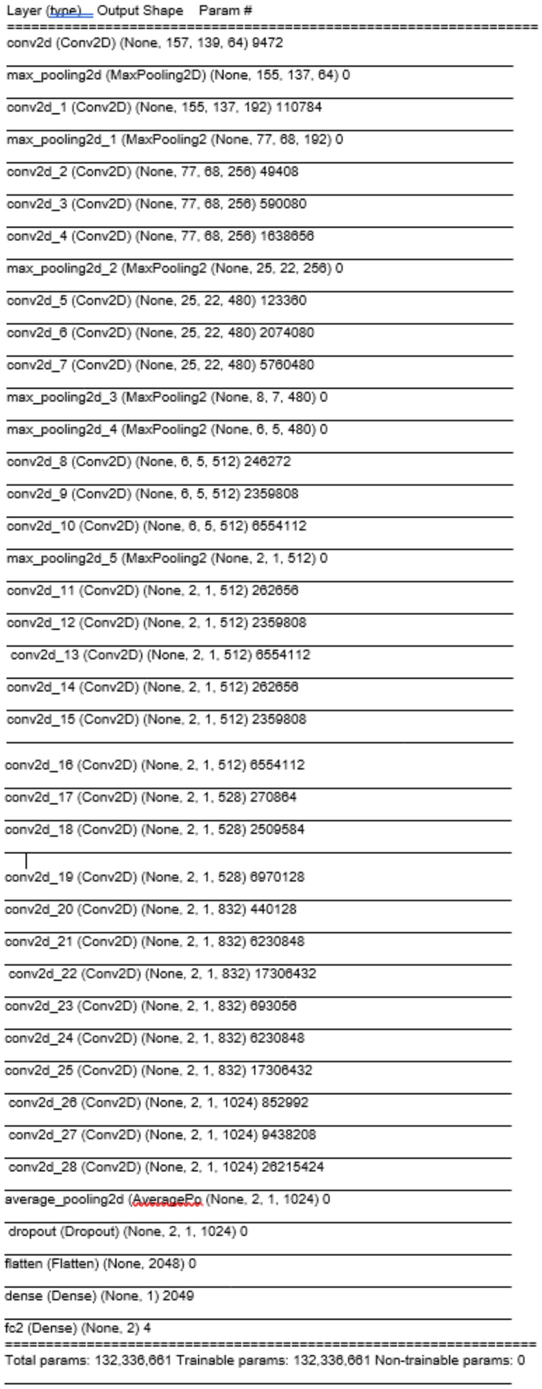


```
Layer (type)            Output Shape            Param #
===============================================================
conv2d_3 (Conv2D)       (None, 127, 154, 32)    896
_______________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 63, 77, 32)     0
_______________________________________________________________
flatten_3 (Flatten)        (None, 155232)       0
_______________________________________________________________
dense_5 (Dense)            (None, 128)          19869824
_______________________________________________________________
dense_6 (Dense)            (None, 1)            129
===============================================================
Total params: 19,870,849

Trainable params: 19,870,849

Non-trainable params: 0

_______________________________________________________________
```

Fig. 2. Model Summary of CNN

Another model is built with a well-known model of CNN named Googlenet. Although, we have modified some of its layer to get better result. The modified model summary of Googlenet is following figure: 03

GoogleNet was proposed by researchers of google which was the winner at the ILVRC 2014 image classification challenge. The model consists with several numbers of layers of CNN. Although, it has inception module and global average pooling.



```
Layer (type)   Output Shape    Param #
================================================================
conv2d (Conv2D) (None, 157, 139, 64) 9472
______________________________________________________________
max_pooling2d (MaxPooling2D) (None, 155, 137, 64) 0
______________________________________________________________
conv2d_1 (Conv2D) (None, 155, 137, 192) 110784
______________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 77, 68, 192) 0
______________________________________________________________
conv2d_2 (Conv2D) (None, 77, 68, 256) 49408
______________________________________________________________
conv2d_3 (Conv2D) (None, 77, 68, 256) 590080
______________________________________________________________
conv2d_4 (Conv2D) (None, 77, 68, 256) 1638656
______________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 25, 22, 256) 0
______________________________________________________________
conv2d_5 (Conv2D) (None, 25, 22, 480) 123360
______________________________________________________________
conv2d_6 (Conv2D) (None, 25, 22, 480) 2074080
______________________________________________________________
conv2d_7 (Conv2D) (None, 25, 22, 480) 5760480
______________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 7, 480) 0
______________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 6, 5, 480) 0
______________________________________________________________
conv2d_8 (Conv2D) (None, 6, 5, 512) 246272
______________________________________________________________
conv2d_9 (Conv2D) (None, 6, 5, 512) 2359808
______________________________________________________________
conv2d_10 (Conv2D) (None, 6, 5, 512) 6554112
______________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 2, 1, 512) 0
______________________________________________________________
conv2d_11 (Conv2D) (None, 2, 1, 512) 262656
______________________________________________________________
conv2d_12 (Conv2D) (None, 2, 1, 512) 2359808
______________________________________________________________
 conv2d_13 (Conv2D) (None, 2, 1, 512) 6554112
______________________________________________________________
conv2d_14 (Conv2D) (None, 2, 1, 512) 262656
______________________________________________________________
conv2d_15 (Conv2D) (None, 2, 1, 512) 2359808
______________________________________________________________

conv2d_16 (Conv2D) (None, 2, 1, 512) 6554112
______________________________________________________________
conv2d_17 (Conv2D) (None, 2, 1, 528) 270864
______________________________________________________________
conv2d_18 (Conv2D) (None, 2, 1, 528) 2509584
______________________________________________________________

conv2d_19 (Conv2D) (None, 2, 1, 528) 6970128
______________________________________________________________
conv2d_20 (Conv2D) (None, 2, 1, 832) 440128
______________________________________________________________
conv2d_21 (Conv2D) (None, 2, 1, 832) 6230848
______________________________________________________________
 conv2d_22 (Conv2D) (None, 2, 1, 832) 17306432
______________________________________________________________
conv2d_23 (Conv2D) (None, 2, 1, 832) 693056
______________________________________________________________
conv2d_24 (Conv2D) (None, 2, 1, 832) 6230848
______________________________________________________________
conv2d_25 (Conv2D) (None, 2, 1, 832) 17306432
______________________________________________________________
 conv2d_26 (Conv2D) (None, 2, 1, 1024) 852992
______________________________________________________________
 conv2d_27 (Conv2D) (None, 2, 1, 1024) 9438208
______________________________________________________________
 conv2d_28 (Conv2D) (None, 2, 1, 1024) 26215424
______________________________________________________________
average_pooling2d (AveragePo (None, 2, 1, 1024) 0
______________________________________________________________
 dropout (Dropout) (None, 2, 1, 1024) 0
______________________________________________________________
flatten (Flatten) (None, 2048) 0
______________________________________________________________
dense (Dense) (None, 1) 2049
______________________________________________________________
fc2 (Dense) (None, 2) 4
================================================================
Total params: 132,336,661 Trainable params: 132,336,661 Non-trainable params: 0
______________________________________________________________
```

Fig. 3. Model Summary of CNN

1) Inception Model: The inception module is different from previous architectures such as AlexNet, ZF-Net. In this architecture, there is a fixed convolution size for each layer. In the Inception module 1×1, 3×3, 5×5 convolution and 3×3 max pooling performed in a parallel way at the input and the output of these are stacked together to generated final output. The idea behind that convolution filters of different sizes will handle objects at multiple scale better.

2) inception Model: In other architecture such as AlexNet, the fully connected layers are used at the end of the network. These fully connected layers contain the majority of parameters of many architectures that causes an increase in computation cost. In GoogLeNet architecture, there is a method called global average pooling is used at the end of the network. This layer takes a feature map of 7×7.We have removed some of the pooling layers from the Googlenet model to run the code using the malaria dataset.

B. DATASET

The dataset was collected from kaggle (https://www.kaggle.com/iarunava/cell-images-for-detecting-malaria). This particular disease detection dataset is all about blood test. So, we will be having different kind of scanned images or microscopic images of blood where will be samples of malaria infected bloods. With the help of this kind of images we tried to detect with the help of machine learning the particular person is basically having malaria or not. The dataset we downloaded from the Kaggle.com webpage. This dataset contains Parasitized and uninfected malaria cell images . This all are basically microscopic images.This dataset contains infected and uninfected malaria cell images which contains a total of 27,558 images where each section has 13.8k images The images were collected on June 2015 to December 2015 which shows several stages of infected and uninfected malaria cell images . Where parasitized basically from those persons whose blood samples found of malaria disease and second one cells from a healthy human.

C. Processing

Each of the dataset had 13002 images and we have divided into test and train dataset. Parasitized and uninfected directory was in each folder. For, Googlenet, we have divided each dataset by 10.

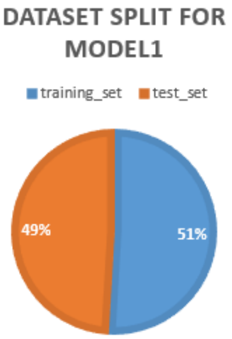


Fig. 4. Dataset split for CNN Model

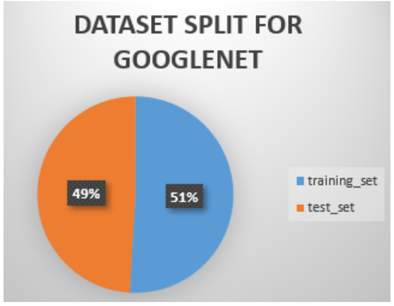


Fig. 5. Dataset split for GoogleNet Model

V. EXPERIMENT RESULT

The dataset was divided into two subsets. One set of the dataset was used for train and another was as a test dataset. There are 14029 images in the train dataset. On the other hand, 13515 images were in the test dataset. We have got 95% accuracy using the train dataset and the validation accuracy was 91.9%. Using the 1st model, we achieved a good accuracy for both training and validation dataset after only one epoch. So, the model will work well for unseen data too.

We have used another well-known model named Googlenet. We have not completed epochs using all of the images due to time limitations. Although, we had 14029 images in the train dataset and 13515 images in the test dataset, we have used 1403 images for train and 1351 images for test. Using googlenet, we got 40% accuracy, 40% precision and 60% for validation accuracy and 60% for validation recall and validation precision. If we had enough time to train and test all of the images, the accuracy and other parameters can be better.

TABLE I
THE RESULT OF CNN MODEL

Accuracy	Val accuracy	Loss	Val loss
95.01%	91.95%	16.12%	35.10%

VI. CHALLENGES

As we are working with a deep learning model, we had to have a clear concept about the layers and dataset as well. We had to check by changing the parameters of the layers which is time consuming. Again, each epoch takes up to 5-10 hours to complete in the 1st model and in the googlenet, it asked more than 120 hours to complete. So, we had to decrease the number of images .

VII. LIMITATIONS

We had a lot of confusion during the learning stage. If we had GPU in our computer, that would be more helpful to train.

VIII. CONCLUSION AND FUTURE WORK

Convolutional neural network shows a good result on the dataset by using model 1. We will use more preprocess method on the dataset as well as use more epochs to get more accuracy and other parameters as well. We will try to implement models like Vgg-16, vgg-19, resnet, alexnet and others.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first . . ."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

REFERENCES

[1] Rajaraman S,K,S, Antani SM, Silamut K, Hossain MA, Richard J, Maude R, Jaeger S, Thomas GR (2018) Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. PeerJ 6:e4568.

[2] Dong Y, Jiang Z, Shen H, David PW, Williams LA, Reddy V, Benjamin W, Bryan A (2017) Evaluations of deep convolutional neural networks for automatic identification of malaria infected cells. In: IEEE EMBS international conference on biomedical and health informatics, BHI 2017. IEEE, Piscataway, pp 101–104

[3] Nakasi, R., Mwebaze, E., Zawedde, A. et al. A new approach for microscopic diagnosis of malaria parasites in thick blood smears using pre-trained deep learning models. SN Appl. Sci. 2, 1255 (2020).

[4] Delgado-Ortet, M.; Molina, A.; Alférez, S.; Rodellar, J.; Merino, A. A Deep Learning Approach for Segmentation of Red Blood Cell Images and Malaria Detection. Entropy 2020, 22, 657.

[5] Yuhang Dong1, Zhuocheng Jiang1, Hongda Shen1, W. David Pan 1 Lance A. Williams2, Vishnu V. B. Reddy2, William H. Benjamin, Jr.2, Allen W. Bryan, Jr.2; Evaluations of Deep Convolutional Neural Networks for Automatic Identification of Malaria Infected Cells

[6] Courosh Mehanian1, Mayoore Jaiswal1,2, Charles Delahunt1,2, Clay Thompson3, Matt Horning1, Liming Hu1, Shawn McGuire1, Travis Ostbye1, Martha Mehanian1, Ben Wilson1, Cary Champlin1, Earl Long4, Stephane Proux5, Dionicia Gamboa6, Peter Chiodini7, Jane Carter8, Mehul Dhorda9, David Isaboke8, Bernhards Ogutu10, Wellington Oyibo11, Elizabeth Villasis6, Kyaw Myo Tun12, Christine Bachman13, David Bell13 ;Computer-Automated Malaria Diagnosis and Quantitation Using Convolutional Neural Networks