DEEP LEARNING APPROACH FOR AUTOMATED SCREENING OF MALARIA PARASITE

[1]Srujana Inturi, [2]V.S.N.Nihanth

[1]Asst prof, [2]Student
[1]Department Of Computer Science and Engineering,
[1]Chaitanya Bharathi Institute Of Technology, Gandipet, Hyderabad.

Abstract : Malaria is a life-threatening disease caused by Plasmodium parasites that infect the red blood cells (RBCs).According to the 2018 World Health Organization (WHO) report, an estimated 435,000 malaria- related deaths are reported globally. Children under 5 years of age are reported to be the most vulnerable, accounting for 61% of the estimated death counts. Early diagnosis and treatment is the most effective way to prevent the disease. Manual identification and counting of parasitized cells in microscopic thick/thin-film blood examination remains the common, but burdensome method for disease diagnosis. Its diagnostic accuracy is adversely impacted by inter/intra-observer variability, particularly in large-scale screening under resource-constrained settings.The primary aim of the paper is to overcome these inter/intra-observer variability or errors that occur as a result of these by automating this process using Deep Learning Algorithms like Convolutional Neural Network (CNN) in such a way so as to have the maximum accuracy. These CNN's can be used to detect whether a Red Blood Cell stained with a chemical Giemsa is infected with Malaria or not. The Malaria Cell Image Dataset provided by Kaggle consisting of a total of 27,558 images of the blood cells is being used to train the model.This data can be used to train various pre trained CNN models like ResNet- 152, WideResNet101-2 as they have low top-1 and top-5 errors. Instead of using the pretrained models as it is, we can unfreeze the already existing layers and modify the layers to suit our dataset. The dataset will also be tested using SGD+Nesterov+Momentum,Adam AdaGrad, AdaMax optimizers and the best combination would be used for the final model.This CNN model would then be either deployed directly in an Android Phone or can be deployed in a remote web server to which images can be sent, processed at the server and results sent back..

***IndexTerms* -,Convolutional Neural Network, Ada Grad, Ada Max optimizer.**

I. INTRODUCTION

Medical image analysis (MIA) is an interdisciplinary field at the intersection of computer science,information engineering, electrical engineering,physics,mathematics and medicine. This field develops computational and mathematical methods for solving problems pertaining to medical images and their use for biomedical research and clinical care.

The main goal of MIA is to extract clinically relevant information or knowledge from medical images. While closely related to the field of medical imaging, MIA focuses on the computational analysis of the images, not their acquisition.

Malaria is a mosquito-borne infectious disease that affects humans and other animals.Malaria causes symptoms that typically include fever, tiredness, vomiting, and headaches.In severe cases it can cause yellow skin, seizures, coma, or death.Symptoms usually begin ten to fifteen days after being bitten by an infected mosquito.If not properly treated, people may have recurrences of the disease months later.

Blood smears are used for detection of malaria where they are stained with a chemical called Giesma which stains the Red Blood Cells(RBC's) which give the RBC's a distinguishing color when there is a presence of the Malaria parasite which we make use.

II. PROPOSED SYSTEM DESIGN

A block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. They are heavily used in engineering in hardware design, electronic design, software design, and process flow diagrams.

Block diagrams are typically used for higher level, less detailed descriptions that are intended to clarify overall concepts without concern for the details of implementation. Contrast this with the schematic diagrams and layout diagrams used in electrical engineering, which show the implementation details of electrical components and physical construction.
The below block diagram gives a brief flow of the working of the project. The steps include preprocessing, augmentation and training and validation testing. This is followed by testing the model and deploying the model.

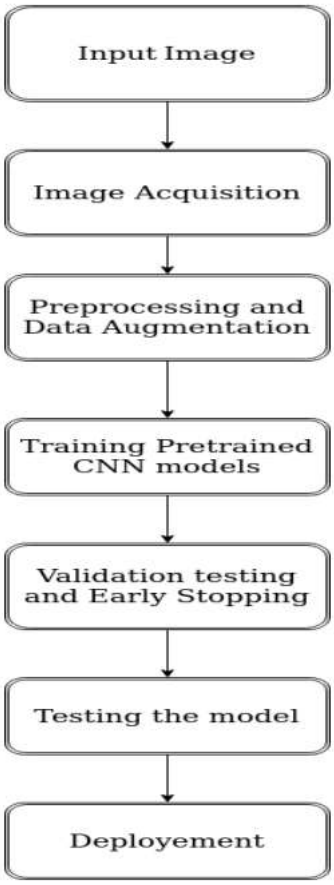

Figure 1: Block diagram of the proposed system

III. MODULES DESCRIPTION

Image Acquisition

In this system suggested approach the authors first believed that the blood cell images in the blood smears of a given patient are preprocessed with a default size of 256×256. Image is determined by using a big matrix whose entrances are numerical values between 0 and 255, where 0 represents to black and 255 to white for illustrating. All these Images are first put into an ImageFolder Object of Pytorch and are split into test,train and validation datasets and are randomly shuffled to remove any order so that the model does not learn the order.

Pre Processing and Data Augmentation

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format which can be understood by our system and be further processed by the system in a desirable way. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Preprocessing usually includes removing/replacing null values, encoding values, converting labels to numbers for textual data whereas for image data preprocessing includes converting color images to black and white images and cropping the images so as to capture the essential features of the images only and remove the unnecessary features among others.

Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data. Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks.

All the images in the test, train and validation datasets are given some transforms like rotation,flipping,random cropping,shear rotations and other transforms to provide the model with the benefit of Data Augmentation and the images are also normalized based on a standard mean and standard deviation to bring all images to a single standard.

Training Pre Trained Models

We make use of the concept of Transfer learning inorder to train our pretrained models on our dataset to create the model. Transfer learning is a deep learning method where a model developed for a task is reused as the starting point for a model on a second task. Transfer learning is an optimization that allows rapid progress or improved performance when modeling the second task.It is a popular approach in deep learning where pre- trained models are used as the starting point on computer vision and natural language processing tasks.

The pretrained models are taken and are trained on the dataset by adding some layers at the classifier layer of the pretrained model. The pretrained CNN models used are Resnet-152,DenseNet among other networks. The pretrained CNN's are trained on the training data and their weights are saved in Google Drive for future use when training is discontinued in between.

IV. RESULTS AND DISCUSSIONS

The following partial results will be present in the further sections:

- Training the model
- Getting weights of the final model
- Finding the accuracy of the model
- Deploying the model to Flask
- Giving the input image
- Predicting the class of the input image

Training the model

The images are taken and are divided into three sub datasets and one of the subdatasets the train dataset is used to train the model(the pretrained model or unfreezed model) and the validation dataset is used to check whether the validation loss is reducing and saves the model if it does else continues training.

The losses are calculated using CrossEntropyLoss or NLLLoss and are output as following.

```
epoch       1       loss        tensor(0.6111, device='cuda:0')
epoch       1       loss        tensor(0.6104, device='cuda:0')
epoch       1       loss        tensor(0.6094, device='cuda:0')
epoch       1       loss        tensor(0.6087, device='cuda:0')
epoch       1       loss        tensor(0.6074, device='cuda:0')
epoch       1       loss        tensor(0.6074, device='cuda:0')
epoch       1       loss        tensor(0.6069, device='cuda:0')
epoch       1       loss        tensor(0.6061, device='cuda:0')
epoch       1       loss        tensor(0.6059, device='cuda:0')
epoch       1       loss        tensor(0.6052, device='cuda:0')
epoch       1       loss        tensor(0.6053, device='cuda:0')
epoch       1       loss        tensor(0.6043, device='cuda:0')
```

Figure 2: Some of the training losses during training

Getting weights of the final model

We train the model till the validation loss of the model stagnates at a point and does not reduce further .After training the model till the desired number of epochs, the model is then loaded from Google drive(by loading the model we mean that we get the weights and other hyperparamters associated with the model). The final weights after training are loaded to the pretrained model(Resnet-152 model).

The weights of the saved model for each of the layers present in the model are as follows which are used by our model.

```
2, 'model_state_dict': OrderedDict([('conv1.weight', tensor([[[[ 4.7132e-07,
   2.1123e-07,  1.3036e-07],
  [ 4.8263e-07,  7.1548e-07,  7.1251e-07,  ...,  3.0581e-07,
   2.6611e-07,  2.3413e-07],
  [ 4.9888e-07,  6.3326e-07,  6.1920e-07,  ...,  1.2629e-07,
   1.8429e-07,  2.0732e-07],
  ...,
  [ 5.5013e-07,  3.1735e-07,  4.1098e-07,  ...,  3.1079e-07,
   3.4928e-07,  3.4718e-07],
  [ 6.2982e-07,  4.0325e-07,  3.4432e-07,  ...,  4.8297e-07,
   6.4529e-07,  5.4214e-07],
  [ 7.1402e-07,  5.0883e-07,  4.4785e-07,  ...,  6.2946e-07,
   6.5617e-07,  5.0979e-07]],

 [[ 5.0878e-07,  6.8802e-07,  6.1782e-07,  ...,  2.2142e-07,
   2.1541e-07,  1.8464e-07],
  [ 4.2393e-07,  6.5220e-07,  6.2894e-07,  ...,  2.8318e-07,
   2.5690e-07,  2.3177e-07],
  [ 4.6649e-07,  6.4230e-07,  6.2854e-07,  ...,  1.3226e-07,
   2.2451e-07,  2.1060e-07],
  ...,
  [ 4.9365e-07,  2.8871e-07,  3.9284e-07,  ...,  2.7530e-07,
   3.1328e-07,  3.3525e-07],
```

Figure 3: Weights of final trained model

Finding the accuracy of the model

After loading the weights of our trained model we test our model with the test dataset that we split initially and we calculate the test loss of our model either using the CrossEntropyLoss or the NLLLoss loss functions and calculate the testing accuracy of our model and we get an output as follows where we print the test loss and the test accuracy of our model which is represented in the form of percentages.

```
Test Loss: 0.348637

Test Accuracy: 84% (2336/2756)
```

Figure 4: Final model accuracy

Deploying the model to Flask

After our model is trained and is tested then we deploy this model as a web service to be accessible from a web browser using a web User Interface. For this we write a code in flask namely hello.py(along with some HTML files for the user interface) and execute the code in a terminal/python shell to start a development server(we run the web server in development mode). On successfully executing the code we get the following screen stating that our web site can be accessed from the URL http://127.0.0.1:5000/ as follows.

```
nithanth@nithanth-HP-Laptop-15-bs1xx:~/Desktop$ cd project/
nithanth@nithanth-HP-Laptop-15-bs1xx:~/Desktop/project$ FLASK_ENV=development FLASK_APP=hello.py flask run
 * Serving Flask app "hello.py" (lazy loading)
 * Environment: development
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 746-699-398
127.0.0.1 - - [22/Mar/2020 12:55:52] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [22/Mar/2020 12:55:52] "GET /favicon.ico HTTP/1.1" 404 -
```

Figure 5: Model Deployement in Flask

On visiting the URL http://127.0.0.1:5000/ on our web browser we get the following web interface with two buttons one to upload an image and the other to submit the image for evaluation.

Figure 6: Web User Interface

Giving the input image

Later after launching the web site we then click on the choose file button in the web site to open a file picker menu and then we select the image that we want to upload to our trained model residing in the flask web server to predict whether the blood cell in the image is parasitized or not.

The file picker which pops up is somewhat like this.

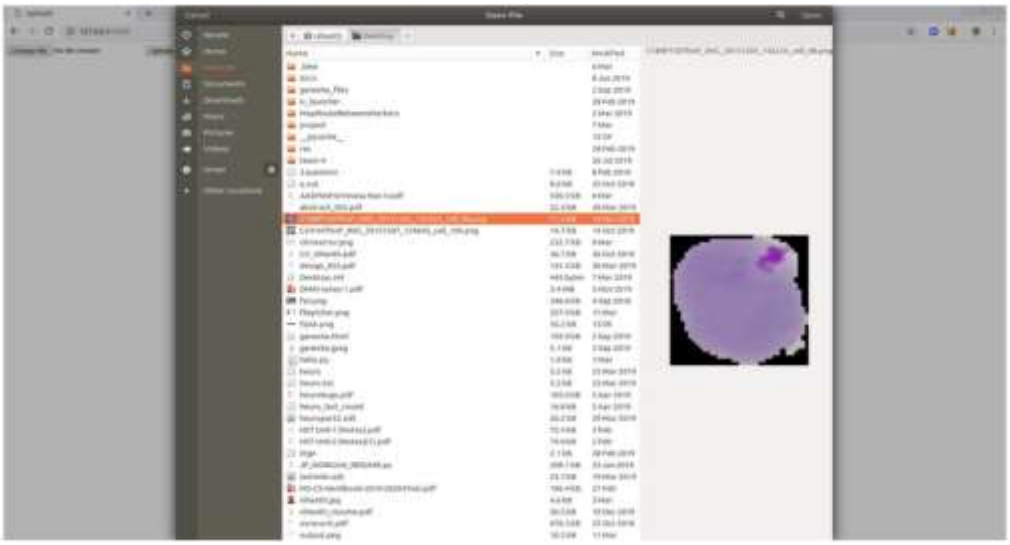

Figure 7: Providing input to web service

Predicting the class of the input image

After selecting the desired image we click on the upload button in the website User Interface to send the particular image in the form of a POST HTTP request to our web service which is hosted by flask and we are redirected to the

http://127.0.0.1:5000/predict URL wherein we receive the output of the model after the image is processed by the model in the form of JSON text which is rendered on the web browser. The field class name is the field where we get to see whether the blood cell image that we uploaded to the model is parasitised or not.

The output JSON is rendered something like this on the web browser

Figure 8: Output class of image

Performance of various different models on the dataset

The dataset has been tested on different permutaions and each of the models has tested and given different performance measures on the same dataset. The following table provides an overview about the various parameters and optimizers that have been tested and the various accuracies that have been recorded for these models.

Table 1: Performance of various models on dataset

MODEL	ACCURACY
ResNet-152 without Data Augmentation, No Unfreezing, Adam Optimizer,No Dropout	69%
ResNet-152 without Data Augmentation, No Unfreezing, SGD+Momentum+Nesterov Optimizer,No Dropout	71%
ResNet-152 with small amounts of Data Augmentation, No Unfreezing, Adam Optimizer, No Dropout	73%
ResNet-152 with small amounts of Data Augmentation, No Unfreezing, SGD+Momentum+Nesterov Optimizer,No Dropout	73%
ResNet-152 with small amounts of Data Augmentation, No Unfreezing, Adam Optimizer, Dropout present	83%
ResNet-152 with small amounts of Data Augmentation, No Unfreezing, SGD+Momentum+Nesterov Optimizer, Dropout present	83%
ResNet-152 with small amounts of Data Augmentation, No Unfreezing, Adam Optimizer, Large Dropout present	74%
ResNet-152 with small amounts of Data Augmentation, Unfreezing, Adam Optimizer, Dropout present	85%
ResNet-152 with small amounts of Data Augmentation, Unfreezing, SGD+Momentum+Nesterov Optimizer, Dropout present	85%
ResNet-152 with small amounts of Large amount of Data Augmentation, Unfreezing, SGD+Momentum+Nesterov Optimizer, Dropout present	83%

V. CONCLUSION

This paper presented a novel approach for automating Malaria Detection using Deep Learning. In this paper, Convolutional Neural Networks were applied to blood smear images to generate the result of whether the cell in the image has malaria or not. Unfreezing concept was used to reduce the generalization error of the model and thus enhance accuracy. One of the limitations of this paper is that the proposed system is that the accuracy of the model is not very high and falls under the mediocre scale. The images should also be cropped appropriately inorder to be made use by the system correctly. In the real world high accuracies are expected for which some more complex models like ensemble models can be used to satisfy the need.

.

REFERENCES

[1] Haque, Ubydul, Hans J. Overgaard, Archie CA Clements, Douglas E. Norris, Nazrul Islam, Jahirul Karim, Shyamal Roy et al. "Malaria burden and control in Bangladesh andprospects for elimination: an epidemiological and economic assessment." The LancetGlobal Health 2, no. 2 (2014): e98-e105..

[2] Wang, Haidong, Mohsen Naghavi, Christine Allen, Ryan M. Barber, Zulfiqar A. Bhutta,Austin Carter, Daniel C. Casey et al. "Global, regional, and national life expectancy, allcause mortality, and cause-specific mortality for 249 causes of death, 1980–2015: asystematic analysis for the Global Burden of Disease Study 2015." The lancet 388, no.10053 (2016): 1459-1544. **[3]** Bhatti, U. and Hanif. M. 2010. Validity of Capital Assets Pricing Model.Evidence from KSE-Pakistan.European Journal of Economics, Finance and Administrative Science, 3 (20).

[3] World Health Organization. Malaria:fact sheet. No.WHO-EM/MAC/035/E. World Health Organization. Regional Office for the Eastern Mediterranean, 2014.

[4] Var, Esra, and F. Boray Tek. "Malaria Parasite Detection with Deep TransferLearning." In 2018 3rd International Conference on Computer Science and Engineering(UBMK), pp. 298-302. IEEE, 2018.

[5] Razzak, Muhammad Imran, and Saeeda Naz. "Microscopic blood smear segmentationand classification using deep contour aware cnn and extreme machine learning." In 2017IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp.801-807. IEEE, 2017.
[6] Mehanian, Courosh, Mayoore Jaiswal, Charles Delahunt, Clay Thompson, MattHorning, Liming Hu, Travis Ostbye et al. "Computer-automated malaria diagnosis andquantitation using convolutional neural networks." In Proceedings of the IEEE International Conference on Computer Vision, pp. 116-125. 2017.
[7] Mohanty, Itishree, P. A. Pattanaik, and Tripti Swarnkar. "Automatic Detection ofMalaria Parasites Using Unsupervised Techniques." In International Conference onISMAC in Computational Vision and Bio-Engineering, pp. 41-49. Springer, Cham, 2018.
[8] Das, Dev Kumar, Madhumala Ghosh, Mallika Pal, Asok K. Maiti, and ChandanChakraborty. "Machine learning approach for automated screening of malaria parasiteusing light microscopic images." Micron 45 (2013): 97-106.
[9] Park, Han Sang, Matthew T. Rinehart, Katelyn A. Walzer, Jen-Tsan Ashley Chi, andAdam Wax. "Automated detection of P. falciparum using machine learning algorithmswith quantitative phase images of unstained cells." PloS one 11, no. 9 (2016): e0163045.
[10] . Caraballo, Hector, and Kevin King. "Emergency department management of mosquitoborne illness: malaria, dengue, and West Nile virus." Emergency medicine practice 16, no.5 (2014): 1-23.
[11] Wilson, Michael L. "Malaria rapid diagnostic tests." Clinical infectious diseases 54, no.11 (2012): 1637-1641.