Blood Cell Image Classification for Detecting Malaria using CNN

Debabrata Mallick[1]

Student Dept. of Computer Science and Engineering, IIT Bombay, Maharashtra, India

---

Abstract - *Artificial intelligence combined with open source tools can improve the diagnosis of the fatal disease malaria. Malaria detection is not an easy procedure, and the availability of qualified personnel around the globe is a serious concern in the diagnosis and treatment of cases. We looked at an interesting real-world medical imaging case study of malaria detection. Easy-to-build, open source techniques leveraging AI can give us state-of-the-art accuracy in detecting malaria, thus enabling AI for social good. AI can improve the diagnosis of Malaria. In our Project, we have used CNN, Max Pooling, Dropout, Rectified Linear Unit (ReLU), and Adam Optimizer for detecting Malaria.*

Key Words: Deep learning, Convolutional Neural Network, image detection, Convolutional layers, Max Pooling, Dropout.

1. INTRODUCTION

Malaria is a serious and life-threatening disease caused by Plasmodium parasites which can travel to the liver by entering the person's bloodstream. It infects the red blood cells, which results in fatal symptoms. However, this deadly parasite can live in our body for one year without causing any symptoms that lead to death. So, it is really necessary to detect malaria disease earlier to save lives. Artificial intelligence (AI) combined with open-source tools can be used to improve the diagnosis of malaria disease in a fast, easy, and accurate way. This is our motivation to make a model using CNN (convolutional neural network) that helps us to detect malaria by classifying blood cell images.

2. Dataset

2.1 Source of the Dataset

For our project, the Dataset we have used is collected from "The Lister Hill National Center for Biomedical Communications (LHNCBC), part of the National Library of Medicine (NLM), USA." In the dataset, there are 27558 cell images which are divided into two parts Parasitized cell images which contain 13779 images, and Uninfected cell images, which contain 13779. The cell images contained in the dataset were collected from 201 people, and among them, 151 people were infected, and 50 people were healthy.

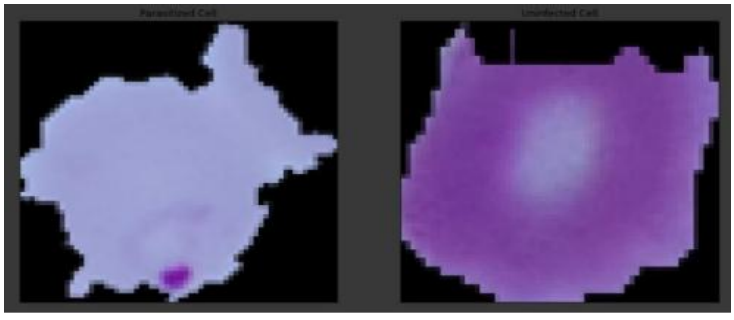

Figure 1: Sample Images of Blood Cell.

3. Methodology

3.1 Environment Setup and Library Import:

For this project, we have used Google Colab from our browser, which helps us to write and execute python code. We have imported different libraries that work as follows:

- NumPy: It is a Python library used for working with arrays, matrices, and image processing by using concepts such as vectorization and advanced indexing.

- Pandas: It is a software library that is used for data manipulation and analysis.
- TensorFlow: It is a Python-friendly open-source library for numerical computation. It also enables the user to implement a deep neural network that is used for solving image recognition/classification tasks.
- OpenCV: It is a library of Python that is used for solving computer vision problems.
- Matplotlib: It is a plotting library for data visualization
- Seaborn: It is a Python data visualization library based on matplotlib, which is used to give a high-level interface for statistical graphics.

3.2 Data Processing:

We have resized our images by 64*64 and rotated our images by 40 degrees to change the position of the images. We have also shuffled our image cell by changing the position to detect in which position there is infected cell can be shown. We have also normalized our images by 255 to increase the intensity and to count the pixels accurately. We have also labeled our uninfected images as 0 and infected images as 1. Here we have three color channels in our images that are red, green blue.

After processing the data, we divided the data into train and test sets. At first, the dataset is divided into Training and Test set (Train 80%, Test 20%). Then the test set is divided into validation and test set (Validation 50%, Test 50%)

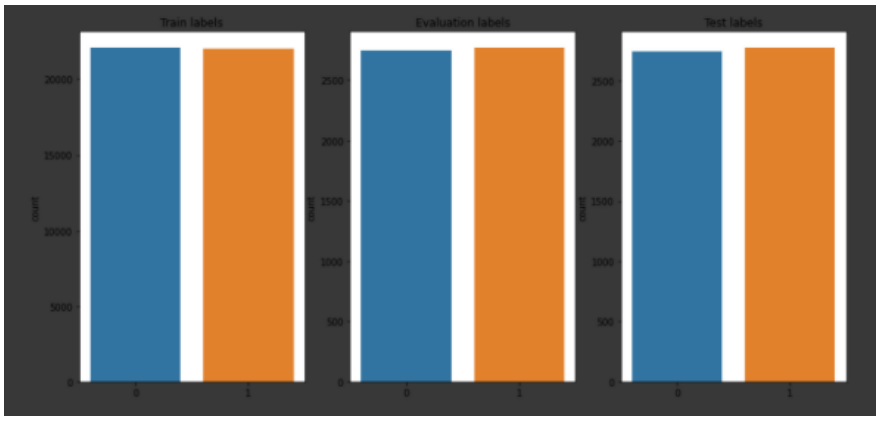

Figure 2: Dataset Divided into Training, Validation, and Testing.

3.3 The architecture of our Model:

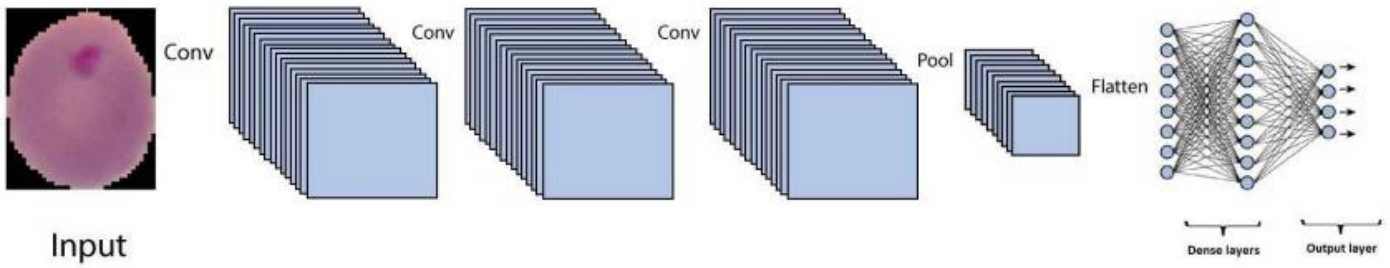

Figure 3: CNN Model Architecture

We have used CNN to train our dataset, which is so far been the most popularly used network for analyzing images. However, image analysis has been the most widespread use of CNN's that can also be used for other data analysis or classification problems as well.

- **Convolutional layers**: Here, we have used convolutional neural network layers to detect spatial patterns from data. This pattern detection makes the CNN model useful for our image analysis. CNN has hidden layers called

convolutional layers, and in our project, we have used three convolutional layers. Basically, one convolutional layer receives input. After that, it transforms the input in some way where the outputs of the transform input go to the next convolutional layer. In our project, our first convolutional layer is used to learn small and local patterns, such as edges and corners. We have taken 40 filters and the kernel size as [5,5]. Our second convolutional layer learns larger patterns based on the features from the first layer, where there are 70 filters, and the kernel size is [3,3]. And the last layer logit is used for calculating the accuracy where we have taken 15 filters. Here the kernel size is the same as the second convolutional layer.

- Max Pooling: After that, we have used Max pooling which helps us to downsample and dimension reduction. It also reduces the computational cost by reducing the number of parameters.
- **Dropout**: We have used dropout to remove some layers. By preventing complex co-adaptations on training data, it reduces overfitting in artificial neural networks.
- **Rectified Linear Unit (ReLU):** For high-performance analysis, we used the rectified linear activation function. It runs faster and performs better by vanishing the gradient problems. It will give output if the input is positive. Otherwise, it gives the output as 0.

4. Results/Analysis:

- For evaluating the model, we used the confusion matrix, classification report, and accuracy score functions of scikit-learn. The performance of our model with relevant classification metrics is given below:

```
[[2653   89]
 [ 289 2481]]
              precision    recall  f1-score   support

           0       0.90      0.97      0.93      2742
           1       0.97      0.90      0.93      2770

    accuracy                           0.93      5512
   macro avg       0.93      0.93      0.93      5512
weighted avg       0.93      0.93      0.93      5512

0.931422351233672
```

Figure 4: Final Classification Report for Test data

- It looks like our models perform well on the test dataset. After testing the dataset, we have got a good model accuracy. For evaluating the model, we used the confusion matrix, classification report, and accuracy score functions of scikit-learn. We got a training loss of 14.13% using the model, while the evaluation stage accuracy is 93.39% and the loss of 18.23%.
- For the test set the classification report, we got 90% precision and 97% recall when predicting healthy cells. And there is 97% precision and 90% recall when predicting infected cells.

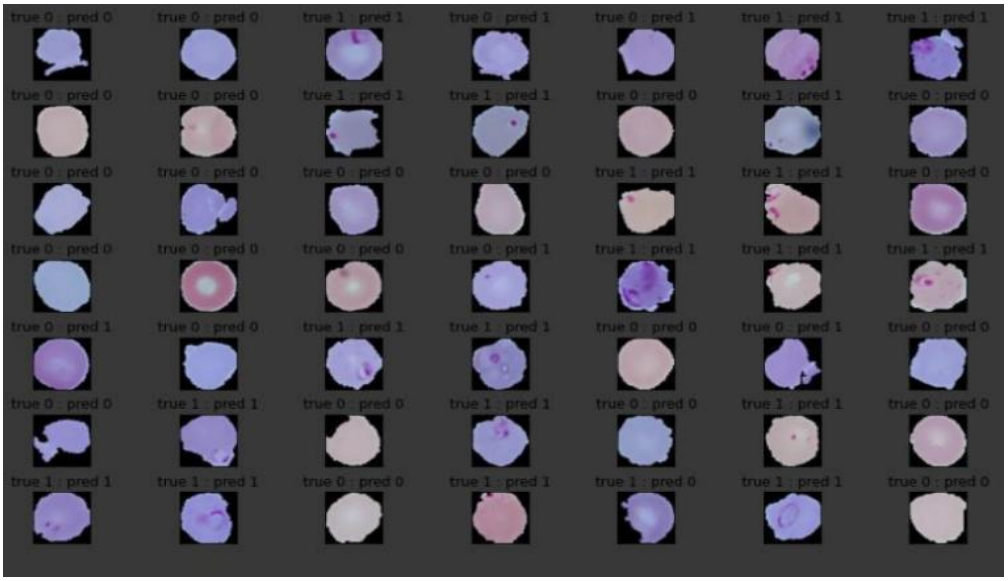

Figure 5: Some Random Predictions and Actual Value

- Some Random Predictions and Actual Value: (Infected Cell = 1, Uninfected Cell = 0)
- If we see the outputs of our model, we get some false positive and false negative predictions. F1-score is 93%. So there is some scope to improve the model.

5. CONCLUSIONS

Malaria detection is a complex procedure and needs the proper guidance of qualified personnel around us to diagnose the disease. We tried to make the process easy and accessible for everyone using Machine Learning models. We achieved an accuracy of 93.39 percent using Convolutional neural networks. This model can be used for detecting malaria. In the future, we will try to improve the accuracy of our model for better results and develop a mobile application so that everyone can diagnose Malaria in the quickest possible time.

REFERENCES

[1] Convolutional neural network - wikipedia," https://en.wikipedia.org/ wiki/Convolutional_neural_network, (Accessed on 08/14/2022).

[2] Detecting malaria with deep learning | opensource.com," https://opensource.com/article/19/4/detecting-malaria-deep-learning, (Accessed on 08/14/2022).

[3] Dilution (neural networks - wikipedia," https://en.wikipedia.org/wiki/Dilution_(neural_networks, (Accessed on 08/14/2022).

[4] Gentle introduction to the adam optimization algorithm for deep learning," https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/, (Accessed on 08/14/2022).

[5] Google colab," https://colab.research.google.com/github/Giffy/CarCrashDetector/blob/master/, (Accessed on 08/14/2022).

[6] Matplotlib - wikipedia," https://en.wikipedia.org/wiki/Matplotlib, (Accessed on 08/14/2022).

[7] Max-pooling / pooling - computer science wiki," https:// computersciencewiki.org/index.php/Max-pooling_/_Pooling#:~:text=Max, (Accessed on 08/14/2022).

[8] Numpy - wikipedia," https://en.wikipedia.org/wiki/NumPy, (Accessed on 08/14/2022).

BIOGRAPHIES

Complected M.Tech CSE at IIT Bombay, Maharashtra.