Developing a Streamlit App for Malaria Cell Detection

Telugu Bharadwaj

Student of Imarticus Learning, Bangalore, Karnataka, India

Abstract - This technical paper presents the development of a Streamlit app for the classification of malaria-infected cells using deep learning techniques(CNN). The app provides an intuitive web interface that allows users to upload an image of a blood cell and obtain a prediction of whether the cell is infected or uninfected. Additionally, the app highlights the hotspots in the image that contribute to the classification prediction. The paper outlines the step-by-step process of building the app, including the model training, preprocessing techniques, and the integration with Streamlit for visualization..

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

1. INTRODUCTION

Malaria, caused by Plasmodium parasites, is a severe and potentially fatal disease that affects millions of people worldwide. Detecting malaria at an early stage is crucial for effective treatment and prevention. Artificial intelligence, coupled with open-source tools, offers a promising approach to improve the diagnosis of malaria by providing fast, easy, and accurate detection of the disease. In this study, we aim to develop a Convolutional Neural Network (CNN) model that can classify blood cell images to aid in the detection of malaria. By leveraging the power of AI and utilizing open-source resources, we seek to enhance the efficiency and accuracy of malaria diagnosis, ultimately contributing to saving lives.

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.

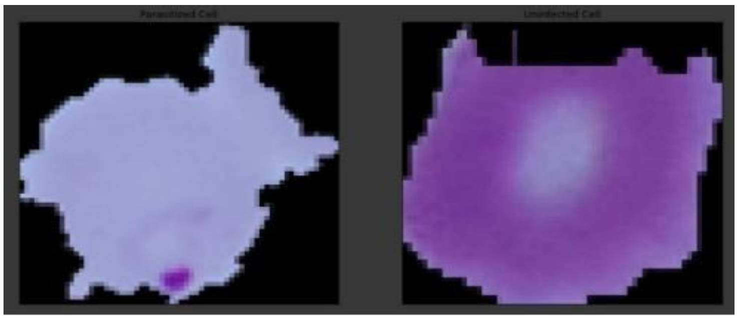

Figure 1: Sample Images of Blood Cell.

3. Methodology

3.1 Environment Setup and Library Import:

For this project, we have used Jupyter Notebook 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 width 0f 68 and height of 68(68*68). 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 1 and infected images as 0. 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. The dataset is divided into Training and Test set (Train 80%, Test 20%)

3.3 The architecture of our Model:

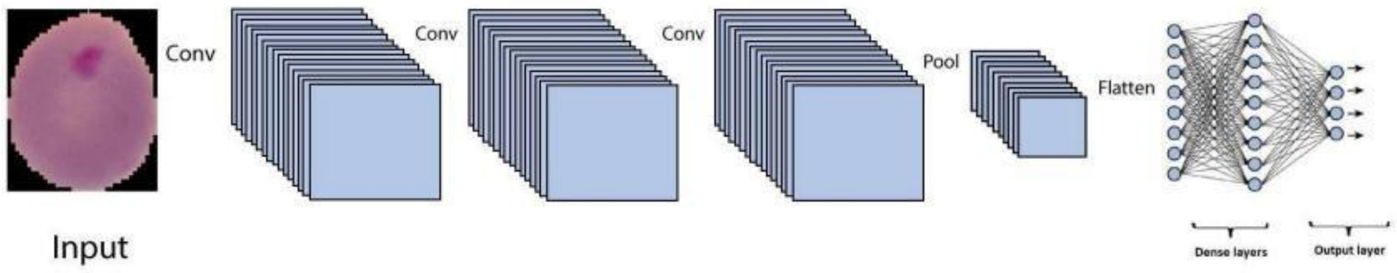

Figure 2: CNN Model Architecture

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 calledconvolutional 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 tothe 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.
- The model performed with an accuracy of around 94%

3.4 Image Preprocessing for Inference:

To classify new images, they need to undergo the same preprocessing steps as the training dataset. The uploaded image is resized to match the input size of the trained model and normalized to a suitable range.

3.5 Model Inference and Heatmap Generation

The preprocessed image is fed into the trained model for inference. The predicted class probability is obtained, indicating whether the cell is infected or uninfected. To generate a heatmap highlighting the hotspots, the activations from a selected layer of the model are extracted. The heatmap is resized to match the dimensions of the input image and then overlayed on the original image.

3.6 Streamlit Integration

Streamlit, a Python library for building interactive web applications, is used to create the user interface for the app. The Streamlit app consists of multiple sections, including image uploading, classification result display, and heatmap visualization. The app allows users to upload an image, view the original image, see the classification result (infected or uninfected), and observe the heatmap overlay on the image

3.7 **Deployment and Usage**

The developed Streamlit app can be deployed on a web server or locally on the user's machine. Users can access the app through a web browser and interact with the provided interface by uploading their own images and obtaining real-time predictions.

4.0 Conclusion

The technical paper outlines the process of developing a Streamlit app for malaria cell classification. By integrating deep learning models with Streamlit, the app provides an accessible and user-friendly tool for malaria diagnosis. The combination of image classification and heatmap visualization enhances the interpretability of the predictions, aiding in the identification of crucial features contributing to the classification decision. The developed app can contribute to efficient and accurate malaria diagnosis and promote the application of deep learning in healthcare settings.

5.0 Future Work:

Future work could involve expanding the app's capabilities to handle multiple image uploads, integrating with additional deep learning models, and further enhancing the interpretability of the heatmaps. User feedback and performance evaluation on diverse datasets can provide insights for improvement and potential application in real-world scenarios

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).