Malaria Cell Classification Using Convolutional Neural Networks

Student Name: **Debasmita Chatterjee**

Pradyumna Kumar Rout

Rhitam Mondal

Tomin Jacob

Group: **G-9**

Mentor Name: **Dr. Madhuchhanda Dasgupta**

Period of Internship: 25th August2025 –31st October 2025

Report submitted to: **IDEAS – Institute of Data Engineering, Analytics and Science Foundation, ISI Kolkata**

Abstract

Malaria continues to be a major global health concern, transmitted primarily through the bites of infected female Anopheles mosquitoes. Traditional diagnostic methods based on microscopic examination of blood samples are accurate but time-consuming, labour-intensive and heavily dependent on expert interpretation. This project proposes the development of an automated malaria detection system using a Convolutional Neural Network (CNN) to classify blood cell images as infected or uninfected. The CNN model efficiently extracts hierarchical image features, learning to recognize the distinct features of *Plasmodium*-infected cells without the need for manual feature selection. A publicly available dataset of labelled cell images was used for training, validation, and testing. The model demonstrated high accuracy, precision, and recall, confirming its ability to generalize effectively across unseen samples. By significantly reducing diagnostic time, this system offers a rapid and reliable alternative to manual testing. The study highlights the potential of deep learning in transforming medical diagnostics and emphasizes its application in improving healthcare accessibility, particularly in resource limited regions

Contents

Introduction

Malaria is a mosquito-borne, life-threatening disease that spreads among human beings, primarily through the bites of infected female Anopheles mosquitoes. It can also be transmitted through blood transfusions and the use of contaminated needles. The disease is caused by a deadly parasite known as Plasmodium. If left untreated, malaria can progress rapidly to severe illness and death within 24 hours. According to a recent study, there are an estimated 263 million malaria cases and 597,000 malaria-related deaths across 83 countries. Early diagnosis and prompt treatment are crucial to reducing disease severity, preventing deaths, and minimizing transmission. Traditionally, malaria diagnosis is confirmed through parasite-based microscopic testing. However, manually examining blood samples under a microscope is both time-consuming and labour-intensive, making it difficult to deliver rapid results essential for timely treatment.

This project aims to build a Convolutional Neural Network (CNN) model capable of classifying infected and uninfected cell images. The deep learning model learns to identify unique features of infected cells and trains itself to distinguish them from uninfected ones. A Convolutional Neural Network is a type of deep learning model particularly effective for image classification and feature extraction. It operates by applying multiple layers of convolutional filters to detect features — beginning with simple patterns such as edges and progressing to more complex structures as data passes through deeper layers of the network. Deep learning models can automate the entire diagnostic process, from image analysis to parasite identification, eliminating the need for manual feature extraction. This automation significantly reduces detection time, which can save millions of lives that are at risk due to delays in treatment. Furthermore, automated systems are highly accessible and portable, making them especially valuable in remote areas where healthcare facilities are limited and the risk of malaria is high.

The successful implementation of this project could greatly enhance the efficiency of detecting malaria-infected cells in pathology laboratories. It would assist hospitals, clinics, and doctors by enabling faster diagnosis and treatment. Additionally, this system can serve as a valuable educational tool in medical colleges, helping students understand the morphology of malaria parasites and supporting further research into disease prevention and control.

Project Objective

The main objectives of doing this project are:

- To build a deep learning model to classify between a malaria infected and an uninfected cell.
- To build and test a Convolutional Neural Network model for image classification.
- To evaluate the efficiency of the CNN based model for malaria cell classification task.
- To demonstrate how a deep learning model can contribute to efficient diagnosis of malaria rather than manual diagnosis.
- To construct a basic building block that can contribute in the construction of a fully automated malaria diagnosis system.

Methodology

The project methodology includes the descriptive explanation of the steps incorporated to build the model and evaluate its efficiency. A Deep-Learning based Convolutional Neural Network model has been build to efficiently distinguish between an infected and an uninfected cell.

The python script for the project is attached herewith:

https://github.com/Debasmita-Chatterjee/Malaria-Cell-Classification.git

1. The CNN Architecture (The Model's Structure)

This section focuses on building a **Convolutional Neural Network (CNN)** to classify images of malaria-infected cells (Parasitized) from healthy ones (Uninfected).

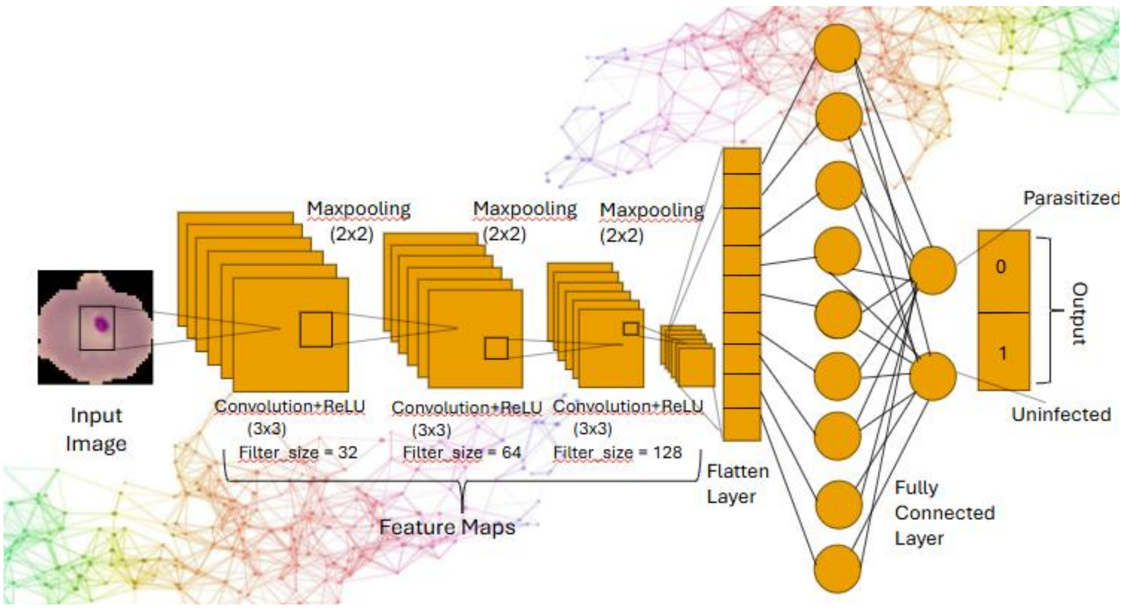


Fig.1. Model Architecture

The CNN is designed with several layers to process the 64x64 colour images:

a) **Convolutional Blocks:** The model started with three blocks, each containing a **Conv2D** layer followed by a **MaxPooling2D** layer.

- **Conv2D (The Feature Detector):** These layers use small filters to scan the image and detect features like edges, textures and patterns. We started with **32 filters**, then **64** and finally **128**, all employing a (3, 3) kernel size which got more complex with each layer. We used the **ReLU** activation function to introduce non-linearity.

- **MaxPooling2D (The Shrinker):** This layer reduces the size of the image data, keeping only the most important features. This makes the model faster and less prone to getting fixated on tiny details (overfitting).

b) **Classification Layers:**

- **Flatten:** The 3D feature maps were unwrapped into a single long list of numbers.

- **Dense Layer:** A standard neural network layer with **128 neurons** processed this list. We added a **Dropout layer (0.5)**, which randomly ignores half the neurons during training. This forces the remaining neurons to be more robust and is a powerful way to **prevent overfitting**.

- **Output Layer:** The final layer had **2 neurons** (one for each class) with a **Softmax** activation function for class probability prediction.

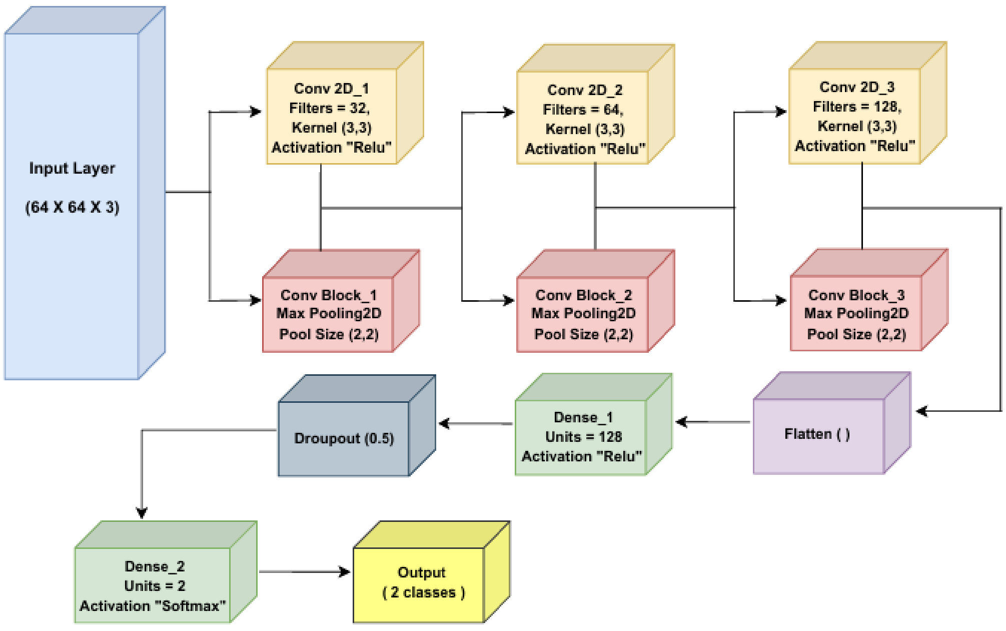


Fig.2. CNN Basic Blocks

2. Training and Visualization

The model was **compiled** using the **Adam optimizer** (an efficient way to adjust the model's weights) and the **categorical cross-entropy loss** (a measure of how wrong the predictions are) was evaluated. The model was then **trained** for **10 epochs** using a batch_size of 32. (10 full passes through the training data).

The learning progress was tracked by plotting two graphs:

i. **Accuracy Plot:** Shows how often the model correctly classified images on both the training set and a hidden **validation set** (20% of the training data).
ii. **Loss Plot:** Shows how much error the model made.

The training process generated performance metrics over the epochs. The **Data Visualization** step utilized **Matplotlib** to plot the **Training Accuracy vs. Epochs** and **Training Loss vs. Epochs** . Monitoring the gap between the Training and Validation curves helped us ensure that the model wasn't just memorizing the training data (overfitting). Finally, the trained model was **saved** as malaria_cnn_model.h5 for practical application.

3. Model Testing and Validation

Model validation represents the final and one of the most crucial stages of the project. It ensures that the developed model performs accurately on unseen data and can be reliably applied to real-world situations. In healthcare-related AI systems, such as malaria detection, model validation is essential to prevent diagnostic errors that could adversely affect patient safety.

3.1 Loading the model

The pre-trained CNN model (malaria_cnn_model.h5) is loaded using Keras. A quick check confirmed that the model was successfully loaded and ready for testing.

3.2 Preparing the data

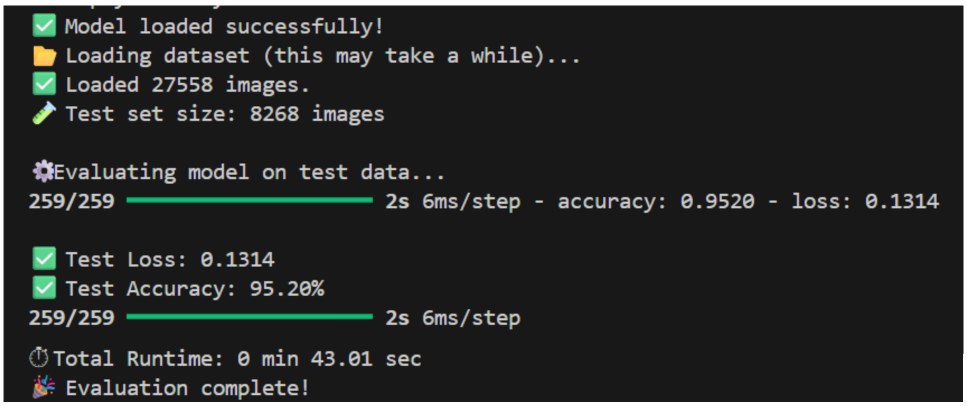
This stage undergoes the same preprocessing procedures as referred in section 3.1. The unseen datasets are normalized and encoded into one-hot encoding.

3.3 Testing and Visualization

The dataset is finally loaded into the model. The dataset is processed with 10 epochs and batch_size 32 in the model. To assess the model's performance on unseen datasets, the loss and accuracy graphs are plotted.

3.4 Assessing the Runtime

In this stage, the full evaluation process was timed using Python's time module.



```
✅ Model loaded successfully!
📁 Loading dataset (this may take a while)...
✅ Loaded 27558 images.
🧪 Test set size: 8268 images

⚙️Evaluating model on test data...
259/259 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.9520 - loss: 0.1314

✅ Test Loss: 0.1314
✅ Test Accuracy: 95.20%
259/259 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step

⏱️Total Runtime: 0 min 43.01 sec
🎉 Evaluation complete!
```

Fig.3. Final Output Panel

4. Model Evaluation

Model evaluation plays a vital role in assessing a model's overall performance, efficiency, and reliability. They provide standardized, quantitative measures that help in understanding how well the model's predictions align with the actual outcomes. Through these metrics, we can ensure that the model meets the desired accuracy and robustness requirements.

The following steps were followed during the model evaluation phase:

- The test dataset was loaded into the trained Convolutional Neural Network (CNN) model.

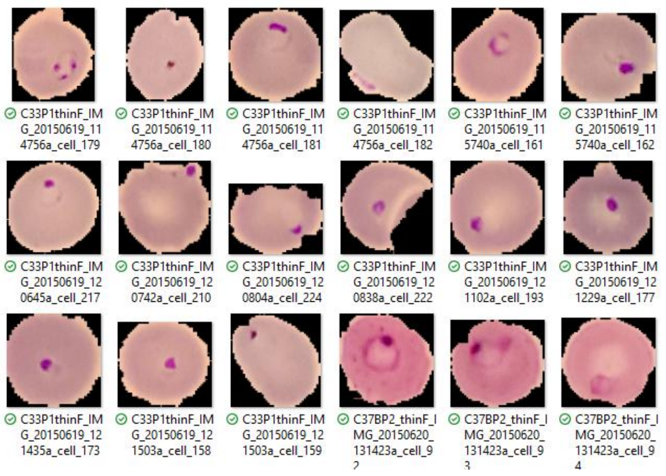
- The model generated predictions, which were stored and compared with the actual ground truth labels.

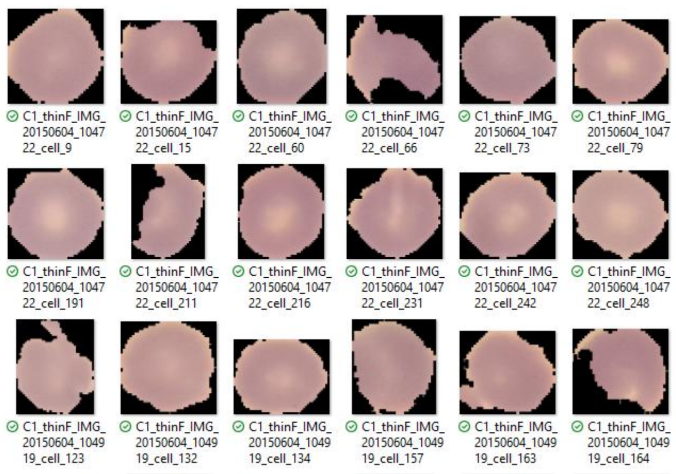
- Various evaluation metrics were calculated to measure the performance of the model, and a confusion matrix was constructed to visualize the classification results.

Data Analysis and Results

1. <u>Dataset Collection</u>

The dataset consists of a large set of parasitized and uninfected cell images observed under microscope. The dataset has a total of 27,558 images. The dataset is originated from the official NIH Website : https://ceb.nlm.nih.gov/repositories/malaria-datasets/ and has been extracted from https://www.kaggle.com/datasets/iarunava/cell-images-for-detecting-malaria . The parasitized images demonstrate *P.falciparum* parasite infected cell extracted from the malaria infected patients. The examples of the sample cells are demonstrated below.


C33P1thinF_IMG_20150619_114756a_cell_179
C33P1thinF_IMG_20150619_114756a_cell_180
C33P1thinF_IMG_20150619_114756a_cell_181
C33P1thinF_IMG_20150619_114756a_cell_182
C33P1thinF_IMG_20150619_115740a_cell_161
C33P1thinF_IMG_20150619_115740a_cell_162
C33P1thinF_IMG_20150619_120645a_cell_217
C33P1thinF_IMG_20150619_120742a_cell_210
C33P1thinF_IMG_20150619_120804a_cell_224
C33P1thinF_IMG_20150619_120838a_cell_222
C33P1thinF_IMG_20150619_121102a_cell_193
C33P1thinF_IMG_20150619_121229a_cell_177
C33P1thinF_IMG_20150619_121435a_cell_173
C33P1thinF_IMG_20150619_121503a_cell_158
C33P1thinF_IMG_20150619_121503a_cell_159
C37BP2_thinF_IMG_20150620_131423a_cell_92
C37BP2_thinF_IMG_20150620_131423a_cell_93
C37BP2_thinF_IMG_20150620_131423a_cell_94


C1_thinF_IMG_20150604_104722_cell_9
C1_thinF_IMG_20150604_104722_cell_15
C1_thinF_IMG_20150604_104722_cell_60
C1_thinF_IMG_20150604_104722_cell_66
C1_thinF_IMG_20150604_104722_cell_73
C1_thinF_IMG_20150604_104722_cell_79
C1_thinF_IMG_20150604_104722_cell_191
C1_thinF_IMG_20150604_104722_cell_211
C1_thinF_IMG_20150604_104722_cell_216
C1_thinF_IMG_20150604_104722_cell_231
C1_thinF_IMG_20150604_104722_cell_242
C1_thinF_IMG_20150604_104722_cell_248
C1_thinF_IMG_20150604_104919_cell_123
C1_thinF_IMG_20150604_104919_cell_132
C1_thinF_IMG_20150604_104919_cell_134
C1_thinF_IMG_20150604_104919_cell_157
C1_thinF_IMG_20150604_104919_cell_163
C1_thinF_IMG_20150604_104919_cell_164

Parasitized Cell Sample Uninfected Cell Sample

Fig.4.Dataset

2. <u>Data Transformation and Labeling</u>

The data transformation task converts the raw image dataset to a structured format for loading it into the model. The data labeling task adds labels to the input dataset to efficiently train the model to predict the desired outcomes. The dataset images undergo the following transformation tasks:

- The images are read and resized to 64 X 64 pixels for efficient deep learning analysis.
- The parasitized and uninfected images are assigned labels namely ‘0’ for infected and ‘1’ for uninfected.
- The images and their corresponding labels are stored in separate lists and are converted to numpy arrays for loading them into the CNN model.

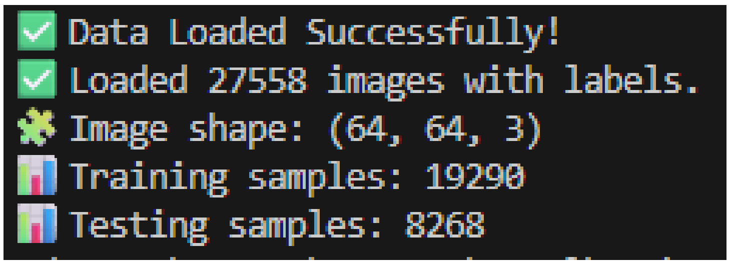
3. Getting the Data Ready (Preprocessing)

We performed the following steps in this stage:

a) **Normalization:** Image pixels are represented by numbers, typically from 0 to 255. We divided all these numbers by **255.0** to scale them down to a small range between **0 and 1**.

b) **One-Hot Encoding:** The labels were simple: **0** for Uninfected, **1** for Parasitized. These labels are converted into a "one-hot" format (e.g., 1 becomes [0, 1], 0 becomes [1, 0]). This format is necessary for the model's final layer.

c) **Splitting the Data:** The entire dataset is divided into two main parts: **70% for Training** (the material the model learns from) and **30% for Testing** (unseen material to check how well it learned). We used a **stratified split**, which means that we made sure the same proportion of infected and uninfected cells were in both the training and testing datasets.



```
✅ Data Loaded Successfully!
✅ Loaded 27558 images with labels.
🧩 Image shape: (64, 64, 3)
📊 Training samples: 19290
📊 Testing samples: 8268
```

Fig.5.Data Preprocessing and Loading

4. System Configuration

This section describes the system configurations, including the programming languages, libraries, and packages used to build the model. The entire implementation of this model has been done in Python, an efficient programming language that provides a wide range of libraries and packages for developing machine learning models.

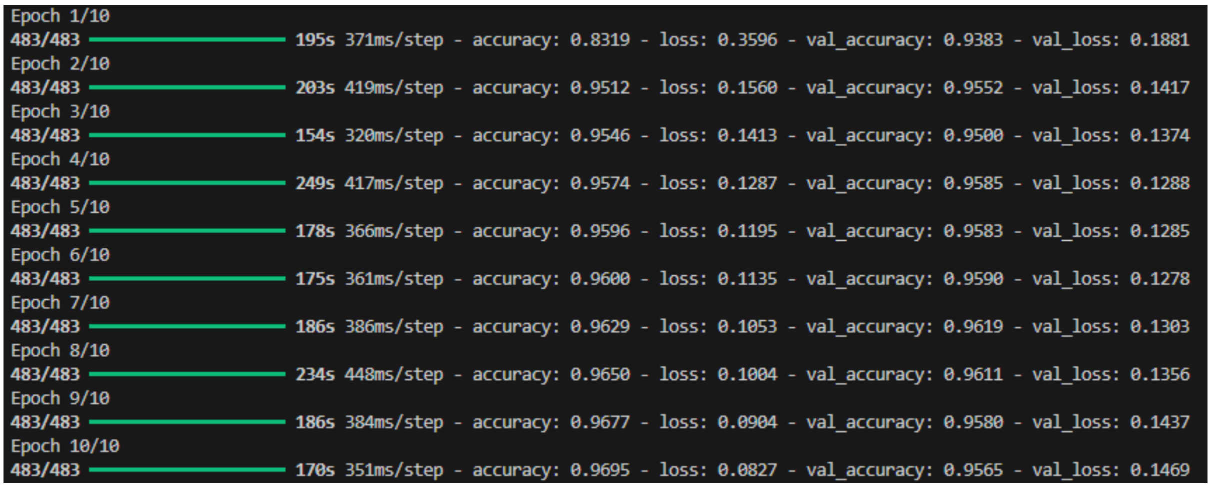
The primary libraries used at various stages of this project include: NumPy**,** Pandas**,** OpenCV (cv2)**,** Glob**,** Matplotlib**,** TensorFlow**,** Keras**,** Time**,** and Scikit learn**.**

The final model was compiled using the **Adam optimizer**, and the **evaluation metrics** were implemented using the **Scikit learn** library.

5. Data Visualization

This section demonstrates the data visualizations performed to observe the model performance and also analyzes the results obtained from those observations.

I. Visualization of the Epochs and Losses



```
Epoch 1/10
483/483 ━━━━━━━━━━━━━━━━━━━━ 195s 371ms/step - accuracy: 0.8319 - loss: 0.3596 - val_accuracy: 0.9383 - val_loss: 0.1881
Epoch 2/10
483/483 ━━━━━━━━━━━━━━━━━━━━ 203s 419ms/step - accuracy: 0.9512 - loss: 0.1560 - val_accuracy: 0.9552 - val_loss: 0.1417
Epoch 3/10
483/483 ━━━━━━━━━━━━━━━━━━━━ 154s 320ms/step - accuracy: 0.9546 - loss: 0.1413 - val_accuracy: 0.9500 - val_loss: 0.1374
Epoch 4/10
483/483 ━━━━━━━━━━━━━━━━━━━━ 249s 417ms/step - accuracy: 0.9574 - loss: 0.1287 - val_accuracy: 0.9585 - val_loss: 0.1288
Epoch 5/10
483/483 ━━━━━━━━━━━━━━━━━━━━ 178s 366ms/step - accuracy: 0.9596 - loss: 0.1195 - val_accuracy: 0.9583 - val_loss: 0.1285
Epoch 6/10
483/483 ━━━━━━━━━━━━━━━━━━━━ 175s 361ms/step - accuracy: 0.9600 - loss: 0.1135 - val_accuracy: 0.9590 - val_loss: 0.1278
Epoch 7/10
483/483 ━━━━━━━━━━━━━━━━━━━━ 186s 386ms/step - accuracy: 0.9629 - loss: 0.1053 - val_accuracy: 0.9619 - val_loss: 0.1303
Epoch 8/10
483/483 ━━━━━━━━━━━━━━━━━━━━ 234s 448ms/step - accuracy: 0.9650 - loss: 0.1004 - val_accuracy: 0.9611 - val_loss: 0.1356
Epoch 9/10
483/483 ━━━━━━━━━━━━━━━━━━━━ 186s 384ms/step - accuracy: 0.9677 - loss: 0.0904 - val_accuracy: 0.9580 - val_loss: 0.1437
Epoch 10/10
483/483 ━━━━━━━━━━━━━━━━━━━━ 170s 351ms/step - accuracy: 0.9695 - loss: 0.0827 - val_accuracy: 0.9565 - val_loss: 0.1469
```

Fig.6.Epoch Visualization

From the above analysis we can observe that with the increase in the number of epochs the accuracy rate increases gradually and there is a significant decrease in the loss rate.

II. Training Accuracy and Loss Graph

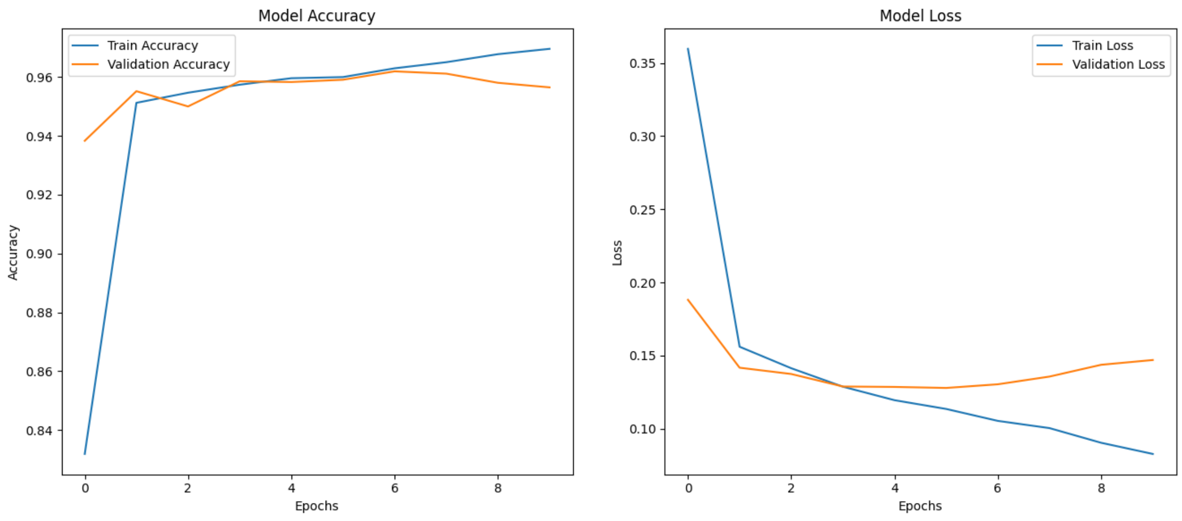
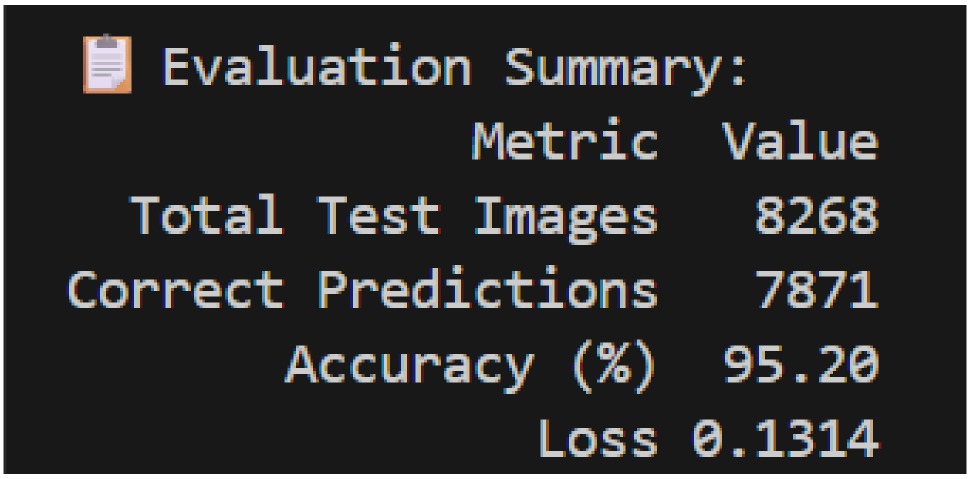


Fig.7.Training Accuracy and Loss Graphs

III. Test Evaluation Summary



```
📋 Evaluation Summary:
                Metric   Value
     Total Test Images    8268
   Correct Predictions    7871
          Accuracy (%)   95.20
                  Loss  0.1314
```

Fig.8. Test Evaluation Summary

The CNN model performed significantly well on unseen data, showing that it effectively learned to identify infected and uninfected cells. This confirms that the trained model is reliable and suitable for real-world malaria detection.

IV. Confusion Matrix

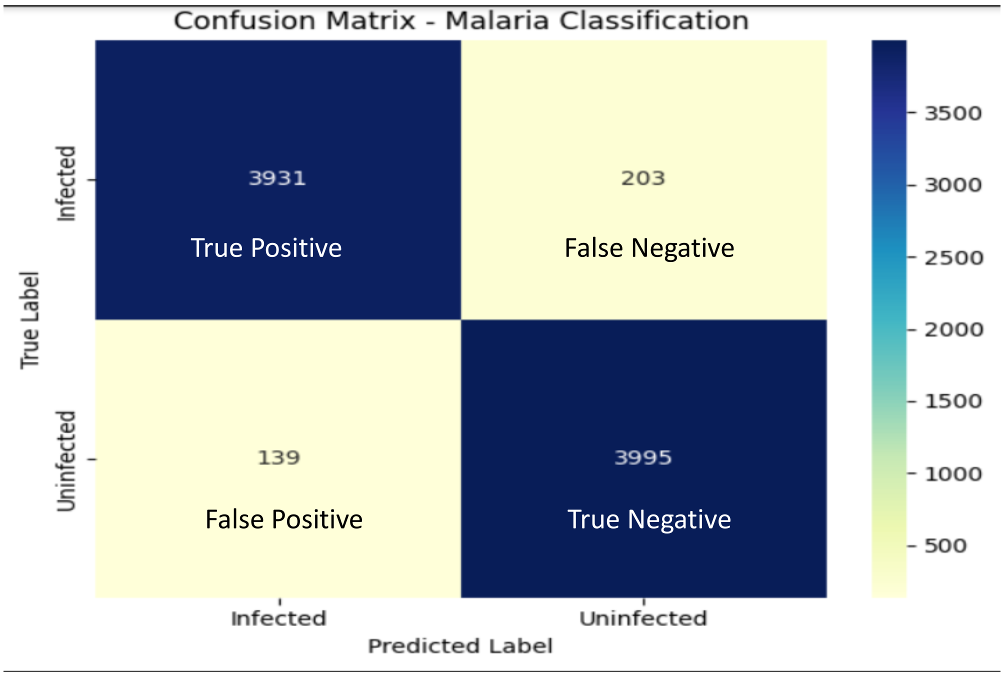


Fig.9. Confusion Matrix

The model correctly identified 3931 infected and 3995 uninfected cells, with only minor misclassifications. This indicates strong classification ability and high predictive reliability.

V. Evaluation Metrics

The main performance metrics used to assess the CNN model are Accuracy, Precision, Recall, and F1 Score.

The formulas for calculating the Accuracy, Precision, Recall, and F1 Score are as follows:

Note: True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN).

Accuracy (A) = $\frac{TP + TN}{TP + TN + FP + FN}$

Precision (P) = $\frac{TP}{TP + FP}$

Recall (R) = $\frac{TP}{TP + FN}$

F1 Score (F) = $\frac{2 \times P \times R}{P + R}$

The values obtained are shown below:

Table.1. Evaluation Metrics

Metric	Result (%)
Accuracy	95.86
Precision	95.16
Recall (Sensitivity)	96.64
F1 Score	95.90

These results confirm that the model has achieved a high level of accuracy and precision. The recall score further demonstrates the model's effectiveness in correctly identifying malaria-infected cells, minimizing false negatives—a crucial aspect in medical image classification.

Conclusion

The CNN model effectively learns image features—from basic patterns to complex cellular structures—allowing accurate detection of malaria-infected cells without manual feature engineering. It achieves high accuracy, precision, and recall, confirming its reliability in classifying unseen samples. This automation speeds up diagnosis, ensures consistency, and reduces reliance on human expertise—especially valuable in resource-limited settings. By enabling faster treatment decisions, such systems can help lower mortality rates. The study also highlights the broader potential of AI in healthcare. The model offers a scalable framework adaptable to other blood disorders or parasitic infections and serves as a practical educational tool. In conclusion, the project showcases how CNN-based systems can revolutionize medical diagnostics with rapid, accurate, and cost-effective results. With further development and integration into clinical workflows, such technology can greatly support global health efforts in disease detection and control.

Appendices

[1] Github Link for the project: https://github.com/Debasmita-Chatterjee/Malaria-Cell-Classification.git

[2] Dataset: https://www.kaggle.com/datasets/iarunava/cell-images-for-detecting-malaria

References

[1] Chima, J. S., Shah, A., Shah, K., & Ramesh, R. (2020). Malaria cell image classification using deep learning. International Journal of Recent Technology and Engineering, 8(6), 5553-59.

[2] Sivaramakrishnan, R., Antani, S., & Jaeger, S. (2017, October). Visualizing deep learning activations for improved malaria cell classification. In Medical informatics and healthcare (pp. 40-47). PMLR.

[3] Narayanan, B. N., Ali, R., & Hardie, R. C. (2019, September). Performance analysis of machine learning and deep learning architectures for malaria detection on cell images. In Applications of Machine Learning (Vol. 11139, pp. 240-247). SPIE.

[4]https://www.geeksforgeeks.org/machine-learning/introduction-convolution-neural-network/

[5]https://eranfeit.net/how-to-classify-malaria-cells-using-convolutional-neural-network/

[6]https://www.geeksforgeeks.org/machine-learning/metrics-for-machine-learning-model/

[7] S. S, S. C and V. B, "Classification of Malaria cell images using Deep Learning Approach," *2023 Third International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, Bhilai, India, 2023, pp. 1-5, doi: 10.1109/ICAECT57570.2023.10117649.

[8] https://www.nature.com/articles/s41598-024-63831-0

[9]https://wandb.ai/mostafaibrahim17/ml-articles/reports/A-Deep-Dive-Into-Learning-Curves-in-Machine-Learning--Vmlldzo0NjA1ODY0

[10]https://developers.google.com/machine-learning/crash-course/overfitting/interpreting-loss-curves