KANTIPUR ENGINEERING COLLEGE

(Affiliated to Tribhuvan University)

Dhapakhel, Lalitpur

[Subject Code: CT755]

A MAJOR PROJECT REPORT ON

MALARIA DETECTION USING CNN

Submitted by:

Bijaya Kumar Sharma	**[41416]**
Manas Adhikari	**[41434]**
Monsoon Bikram Thapa	**[41437]**
Neetish Paudel	**[41440]**

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

March, 2024

MALARIA DETECTION USING CNN

Submitted by:

Bijaya Kumar Sharma	**[41416]**
Manas Adhikari	**[41434]**
Monsoon Bikram Thapa	**[41437]**
Neetish Paudel	**[41440]**

Supervised by:

Er. Mahesh Singh Kathayat

Associate Professor

Kathmandu Engineering College

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

Kantipur Engineering College

Dhapakhel, Lalitpur

March, 2024

COPYRIGHT

KANTIPUR ENGINEERING COLLEGE
DEPARTMENT OF COMPUTER AND ELECTRONICS ENGINEERING

APPROVAL LETTER

The undersigned certify that they have read and recommended to the Institute of Engineering for acceptance, a project report entitled ”MALARIA DETECTION USING CNN” submitted by

Bijaya Kumar Sharma	[41416]
Manas Adhikari	[41434]
Monsoon Bikram Thapa	[41437]
Neetish Paudel	[41440]

in partial fulfillment for the degree of Bachelor in Computer Engineering.

....
Supervisor
Er. Mahesh Singh Kathayat
Associate Professor
Kathmandu Engineering College

....
External Examiner
Er. Bharat Bhatta
Senior Lecturer
Sagarmatha Engineering College

....
Head of Department
Er. Rabindra Khati
Associate Professor
Department of Computer and Electronics Engineering

Date: March 6, 2024

ACKNOWLEDGMENT

Firstly, we would like to express our deepest gratitude towards our project supervisor, **Er. Mahesh Singh Kathayat** for his constant guidance, encouragement, timely check-ins and valuable suggestions during the course of our project. Moreover, we would like to thank the Department of Computer and Electronics of Kantipur Engineering College as well as our project manager **Er. Bishal Thapa** for providing us with the suitable facilities and timely checks for the completion of the project. Also we would like to express our gratitude towards medical lab technician of EDCD, **Mr. Dabal BC** for helping us with data cleaning process and providing insights on malaria detection technology and processes used in Nepal. Finally, we would like to acknowledge and thank everyone who helped us directly or indirectly in this project.

Bijaya Kumar Sharma	[41416]
Manas Adhikari	[41434]
Monsoon Bikram Thapa	[41437]
Neetish Paudel	[41440]

ABSTRACT

Malaria is a life-threatening disease that is spread by the Plasmodium parasites. It is detected by trained microscopists who analyze microscopic blood smear images. Modern deep learning techniques may be used to do this analysis automatically. CNNs have been prominently used for identification, classification, and feature extraction tasks, and they have delivered a great performance at these tasks. We propose a CNN model for identifying malaria parasitic red blood cells and distinguishing them from healthy red blood cells which would assist to get an immediate information on patient's health. The dataset is taken from the official National Institutes of Health website. Images of infected and non-infected erythrocytes are gathered and fed into the CNN model from the dataset. A model was developed which makes it easier for microscopists in areas with limited resources to diagnose maria more accurately, capable of detecting and isolating red blood cells in the images. Our model can detect malarial parasites from microscopic images with an accuracy of 99.31%. For practical validation of model efficiency, we have deployed the model in a web application. It shows that our model is able to perform better along with low computational requirements. Therefore, it can be used more efficiently and can be easily deployed for detecting malaria cells.

Keywords−*CNN (Convolutional Neural Network), Deep Learning,Feature Extraction.*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI: Artificial Intelligence

CNN: Convolutional Neural Network

ML: Machine Learning

RBCs: Red Blood Cells

ReLU: Rectified Linear Unit

ROC: Receiver Operating Characteristics

CHAPTER 1
INTRODUCTION

1.1 Background

Malaria is a severe disease caused by parasites of the genus Plasmodium, which is transmitted to humans by a bite of an infected female mosquito of the species Anopheles. Malaria remains a significant contributor to global mortality, particularly prevalent in Africa and certain Asian countries. In 2017, approximately 219 million individuals were impacted by malaria, resulting in 435,000 deaths worldwide. The predominant species in the Americas and Europe are P.vivax and P.malariae, while P.falciparum prevails in Africa. It is responsible for over 1 million deaths annually, with 90% occurring in African children. Historical treatments include the use of sweet sagewort in ancient China and later, quinine. Despite available management strategies, malaria cases continue to rise due to various factors. Hence, timely and efficient diagnostic methods are imperative for its management and containment [1].

Timely and accurate diagnosis is crucial for effective management and treatment of the disease. However, manual microscopy-based diagnosis is often challenging in these settings due to limited access to skilled health care professionals and adequate laboratory facilities. To overcome these limitations, this project aims to leverage machine learning, specifically deep convolutional neural networks (CNNs), for automated malaria detection. By developing an automated system, we seek to improve the accuracy and efficiency of malaria diagnosis, particularly in areas with limited resources.

The project involves image processing techniques for segmenting malaria-infected cells from whole slide images and the creation of a curated dataset of annotated images for training the deep CNNs. Data augmentation methods are employed to enhance the dataset and address overfitting. The performance of the deep CNN model is evaluated using various combination of datasets. Successful implementation of this project has the potential to provide reliable healthcare solutions for malaria diagnosis in resource-scarce areas, facilitating timely interventions and reducing the burden of the disease.

1.2 Problem Statement

Time consumption: Using a microscope to detect whether the RBC is infected or not, for this a continuous human effort is required to focus on parasites which is time consuming.

Lack of Expert Technician: In rural areas, it is hard to find an expert technician, which may lead to the critical condition of the patient. In such a case, we can use such a digital diagnosis procedure, where even a health assistant can click the photo of RBC using a microscopy camera and upload in the system, and diagnose malaria.

1.3 Objectives

1. To develop an automated malaria detection system using CNN for accurate identification of malaria-infected cells.

1.4 Application Scope

The application scope of this project is broad, encompassing various areas where automated malaria diagnosis can have a significant impact. It primarily targets resource-limited areas and remote healthcare settings where access to skilled professionals and laboratory facilities is limited. By automating the diagnostic process, the project aims to overcome the challenges associated with manual microscopy-based diagnosis in these regions. Additionally, the system can be integrated into screening and surveillance programs to efficiently detect malaria-infected individuals within a population, aiding in early detection, treatment, and prevention. Furthermore, the project holds potential for integration into telemedicine and mobile health applications, enabling healthcare providers to remotely access the automated system and facilitate accurate malaria diagnosis in underserved regions. Overall, the application scope of the project extends to any healthcare setting where automated malaria detection can address the need for accurate and timely diagnosis, ultimately improving healthcare outcomes.

1.5 Features

1. Web application functionality
2. Low computational requirements
3. High accuracy in malaria detection achieved
4. User-friendly interface for ease of use

1.6 System Requirements

The system requirements for the project are as follows:

1.6.1 Development Requirments

1.6.1.1 Hardware Requirments

The listed below are the recommended hardware requirements for the development of the system:

- CPU: i7 9750H
- RAM: 16 GB DDR4
- GPU: GTX 1660 ti

1.6.1.2 Software Requirement(Minimum)

The listed below are the software requirements for the development of the system:

- Operating System: Windows 8 or above
- Web Technologies: Python
- Jupyter Notebook with libraries

1.6.2 Deployment Requirments

1.6.2.1 Hardware Requirments

- More than 1.5 GHz clock speed
- Minimum 4GB RAM

1.6.2.2 Software Requirement(Minimum)

- Operating System: Windows 8 or above / Linux / MacOS
- Web Browser

1.7 Feasibility Study

Conducting a feasibility study is crucial for any project. It involves carefully looking into different aspects to see if the project can actually work. Here are some important things we need to study to make sure our project is feasible.

1.7.1 Economic Feasibility

Since this is an undergraduate project, we didn't focus on making money or calculating returns on investment. Instead, we checked if we could afford to develop the project. The costs for libraries, software tools, and other components were all within our budget. So, economically, our project is economically feasible.

1.7.2 Technical Feasibility

Our project uses freely available software tools and doesn't require advanced technical skills, making it technically feasible. With numerous free machine learning libraries and plenty of sufficient documentation and quality courses, it's evident that our project can be successfully implemented from a technical standpoint.

1.7.3 Operational Feasibility

The project is highly useful for different types of users, and with technology becoming more advanced, using system software or applications isn't difficult anymore. Users just need some basic familiarity with the software, aided by simple visual explanations that become clearer with time. Opting for a higher level of specification would have added more features, but it might have limited the number of potential users. Overall, the project is practical and feasible to operate.

1.7.4 Schedule Feasibility

Schedule Feasibility is defined as the probability of a project to be completed within its scheduled time limits, by a planned due date. If a project has a high probability to be completed on-time, then its schedule feasibility is appraised as high. Schedule feasibility ensures that a project can be completed before the technology becomes unnecessary. Given the numerous features in our project, it has been completed on time.

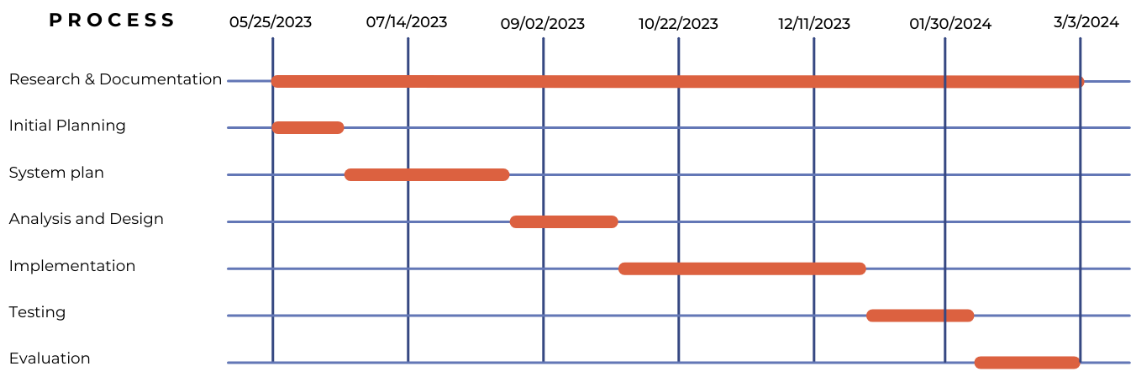


Figure 1.1: Gantt Chart

CHAPTER 2
LITERATURE REVIEW

2.1 Related Projects

Malaria detection using Deep Convolutional Neural Network:

The objective of this study was to utilize a CNN-based network for classifying cell images into parasitized and uninfected categories. A labeled and preprocessed dataset of cell images was available for training and evaluating the model. The dataset, sourced from the Malaria Screener research activity, consisted of 27,558 cell images, equally divided between parasitized and uninfected cells. The images were in color and had variable sizes, and they were stored in two folders labeled ”Infected” and ”Uninfected”, to handle the color images, the LeNet-5 architecture was selected as the initial model, originally designed for grayscale images. Modifications were made to adapt LeNet-5 for multi-channel images, specifically RGB scale, enabling it to work with three layers of images. Hyperparameter tuning was performed to enhance the model's performance, aiming for an efficiency target of 95% or higher on the test dataset. The training process was terminated once the validation accuracy reached 95%. Subsequently, the trained model was evaluated by calculating the testing error on the test dataset. The achieved testing accuracy was approximately 95.4%, indicating a satisfactory performance of the model in distinguishing between parasitized and uninfected cell images [2].

2.2 Related research

Malaria Diagnosis Using a Lightweight Deep Convolutional Neural Network:

In their research paper, the authors propose a lightweight CNN specifically designed for identifying malaria parasitic red blood cells and distinguishing them from healthy red blood cells. To compare the accuracy of their model, they conducted transfer learning experiments using two well-known models, VGG-19 and Inceptionv3. The dataset utilized in this comparative study consisted of microscopic images of red blood cells (RBCs), accessible through the National Library of Medicine (NLM), USA. This dataset comprised two classes: parasitic RBC images and healthy RBC cell images, each containing 13,794 images. For experimentation, the dataset was partitioned into three different proportions: 30:70, 50:50, and 70:30, for training and validation samples. The authors performed separate experiments for each partition to evaluate model performance under varying data proportions. To ensure consistency, the images were resampled to a resolution of 134x134, given their initial differing resolutions. Among all the models evaluated, the authors' proposed CNN achieved the highest training and validation accuracy across all data variations. Specifically, the VGG-19 model achieved a maximum training accuracy of 93.34% and a validation accuracy of 91.64% when the data variation was set to 70:30. These results indicate that the proposed CNN model outperformed the VGG-19 model in terms of accuracy on the given dataset [3].

Malaria detection using deep learning and transfer learning:

This paper discusses the use of Convolutional Neural Networks (CNNs) and deep learning techniques, particularly image processing, to improve the accuracy of diagnosing parasitemia in microscopic blood slides. The researchers utilized the intensity characteristics of Plasmodium parasites and erythrocytes, which are known to vary. They collected images of infected and noninfected erythrocytes and fed them into four CNN models (ResNet50, ResNet34, VGG-16, and VGG-19) that were trained on the same dataset. They employed transfer learning and fine-tuning techniques and compared the outcomes. To conduct their investigation, they had a dataset comprising 27,558 segmented red blood cells (RBCs) with an equal proportion of infected and uninfected cells. The dataset included both training and validation sets, which complemented each other. About 80 percent of the training set was used for actual training, while the remaining 20 percent was reserved for validation. The results revealed that VGG-19 outperformed the other pretrained models, achieving an accuracy of 0.972055. Model accuracy, in this context, refers to the percentage of correct predictions made by the model overall [4].

Malaria detection using deep learning and transfer learning:Intelligent diagnostic model for malaria parasite detection and classification using imperative inception-based capsule neural networks:

This study introduces a deep-learning approach that combines the imperative capsule neural network with the inception neural network to effectively differentiate between malaria-parasitized and uninfected cells in blood cell images. The aim is to enhance the classification accuracy of identifying malaria parasites from photographs of blood cells. The research utilizes images of thin blood smears containing two distinct strains of malaria—one infected and the other uninfected—obtained from the National Institutes of Health (NIH) repository, which is publicly available for study. The dataset comprises 13,779 images of parasites and an equal number of images of uninfected cells, totaling 27,558 images. The proposed network is evaluated against established classification models developed by various researchers, which were pretrained using NIH malaria datasets and other private datasets, to determine the presence of parasitized red blood cells. Notably, the proposed network achieves higher classification accuracy compared to existing deep learning methods when analyzing blood cell images for malaria. In the worst-case scenario of a 50/50 split, the model achieves an accuracy of 98.10% on the test dataset, while on a 20% split, it achieves an accuracy of 99.355%. These experimental results underscore the robustness and flexibility of the developed model, which outperforms competing models in malaria classification tasks [5].

Deep Learning Based Automatic Malaria Parasite Detection from Blood Smear and Its Smartphone Based Application:

This paper presents an automated Convolutional Neural Network (CNN) model for diagnosing malaria from microscopic blood smear images. Various techniques, including knowledge distillation, data augmentation, Autoencoder, and CNN-based feature extraction, followed by classification with Support Vector Machine (SVM) or K-Nearest Neighbors (KNN), are explored across three training procedures: general training, distillation training, and autoencoder training. The malaria dataset comprises 27,558 cell images divided into parasitized and uninfected cells, each with an equal number of instances, sourced from patients at Chittagong Medical College Hospital, Bangladesh.

During annotation, suspicious and falsely labeled data were excluded, reducing the dataset to 26,161 instances. An autoencoder-based architecture is proposed for malaria parasite detection from blood smears. The deep learning model achieves 99.23% accuracy when trained on 32×32 images. Model efficiency is practically validated by deployment on different mobile phones and a server-backed web application, tested across various computational capacities [6].

CHAPTER 3
THEORETICAL FRAMEWORK

Before we can figure out how this project works and identify its research components, we need to learn about the theories behind the topics discussed in Chapter 4 methodology.A detailed explanation of the topics has been presented below.

3.1 Convolutional Layer

A convolutional layer is a fundamental building block of a convolutional neural network (CNN). It applies convolution operations to the input data, typically an image, using learnable filters or kernels to extract various features and patterns. These features are learned hierarchically through multiple layers, enabling the network to capture increasingly abstract representations of the input data.

3.2 Max Pooling Layer

A max pooling layer is used in convolutional neural networks to downsample the feature maps obtained from convolutional layers. It partitions the input into a set of non-overlapping regions and outputs the maximum value within each region. By selecting the maximum value, max pooling helps reduce the spatial dimensions of the feature maps while preserving the most important features, making the representations more compact and computationally efficient.

3.3 Dense Layer

A dense layer, also known as a fully connected layer, is a type of neural network layer where each neuron is connected to every neuron in the preceding layer. In a dense layer, each neuron receives input from all neurons in the previous layer and applies a set of learnable weights and biases followed by an activation function. Dense layers are commonly used for high-level feature extraction and classification in neural networks.

3.4 Dropout Layer

A dropout layer is a regularization technique used to prevent overfitting in neural networks. During training, dropout randomly drops out a fraction of the neurons in the preceding layer by setting their outputs to zero with a specified probability. This helps in reducing the interdependence among neurons, forcing the network to learn more robust features and improving its generalization capability.

3.5 Flatten Layer

A flatten layer is used to convert the two-dimensional or multi-dimensional output of the preceding layer into a one-dimensional vector. It collapses all dimensions except for the first one, which corresponds to the batch size. Flatten layers are typically used to transition from convolutional layers to fully connected layers in neural network architectures.

3.6 Output Layer

The output layer is the final layer in a neural network architecture, responsible for producing the model's predictions or outputs. In classification tasks, the output layer typically consists of neurons corresponding to each class, with the predicted class being the one with the highest activation value. In regression tasks, the output layer may consist of a single neuron representing the predicted numerical value.

3.7 Loss Calculation and Optimization

Loss calculation and optimization refers to the process of measuring the discrepancy between the model's predictions and the actual targets (labels) and adjusting the model's parameters to minimize this discrepancy. Common loss functions include mean squared error for regression tasks and cross-entropy loss for classification tasks. Optimization algorithms like gradient descent or its variants (e.g., Adam optimizer) are then used to update the model's parameters iteratively based on the computed loss, facilitating learning and convergence.

3.8 Training and Evaluation

Training and evaluation are the two main phases of building a machine learning model. During training, the model is exposed to labeled training data, and its parameters are adjusted to minimize the loss function. Once training is complete, the model is evaluated using separate validation or test data to assess its performance on unseen examples. Training and evaluation are iterative processes, with adjustments made to the model's architecture, hyperparameters, and training procedures to optimize performance

3.9 Sigmoid Function

The sigmoid function, also known as the logistic function, is a mathematical function that maps input values to a range between 0 and 1. It is commonly used as an activation function in neural networks, particularly in binary classification tasks, where it squashes the network's output to produce probabilities. The sigmoid function is defined as :

$\sigma(x) = \frac{1}{1+ e^{-x}}$, where e is the base of the natural logarithm.

3.10 ReLU (Rectified Linear Unit)

ReLU is an activation function commonly used in neural networks due to its simplicity and effectiveness. It computes the output as the maximum of zero and the input value, i.e., $f(x) = max(0, x)$

ReLU introduces non-linearity into the network, enabling it to learn complex patterns and preventing the vanishing gradient problem commonly encountered with activation functions like sigmoid or tanh.

3.11 Batch Normalization

Batch normalization is a technique used to normalize the activations of each layer across mini-batches during training. It aims to stabilize and accelerate the training process by reducing the internal covariate shift, i.e., the change in the distribution of the activations of each layer. By normalizing the inputs to each layer, batch normalization helps in improving the convergence speed, enabling the use of higher learning rates, and regularizing the network.

3.12 Adam Optimizer

Adam (Adaptive Moment Estimation) is an optimization algorithm commonly used to update the parameters of a neural network based on the gradients of the loss function with respect to the parameters. Adam combines the advantages of both momentum optimization and RMSProp by computing adaptive learning rates for each parameter. It maintains two moving averages of gradients and squared gradients, allowing it to adaptively adjust the learning rate for each parameter and converge efficiently in a wide range of scenarios.

3.13 Image Augmentation

Image augmentation is a technique used to artificially increase the size of a training dataset for machine learning models by applying various transformations to the existing images. These transformations include but are not limited to rotation, flipping, scaling, translation, shearing, brightness and contrast adjustment, noise injection, and color space transformation. By introducing variations in the training data, image augmentation helps in increasing the diversity of the dataset, reducing overfitting, and improving the model's ability to handle real-world scenarios.

3.14 Tools and Libraries:

Python:

- Python is a high-level, interpreted programming language known for its simplicity and readability.
- It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.
- Python has a vast ecosystem of libraries and frameworks for various purposes, including data analysis, machine learning, and more.

Jupyter Notebook:

- Jupyter Notebook is an open-source web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text.
- It supports various programming languages, but it is particularly popular in the data science community for Python.
- Jupyter Notebooks are interactive and can be used for data cleaning, exploration, visualization, and analysis, making them an essential tool for data scientists and researchers.

OpenCV:

- OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library.
- OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products.

Streamlit:

- Streamlit is an open-source Python library used for building interactive web applications for machine learning and data science projects.

- It allows developers to create user-friendly interfaces directly from Python scripts, without requiring knowledge of web development languages like HTML, CSS, or JavaScript.
- Streamlit applications can be quickly developed and deployed, making them suitable for prototyping, sharing insights, and showcasing machine learning models.

TensorFlow:

- TensorFlow is an open-source deep learning framework developed by Google.
- It provides a comprehensive ecosystem of tools, libraries, and resources for building and deploying our model.

Keras:

- Keras is a high-level neural networks API written in Python.
- It provides a user-friendly interface for building, training, and deploying deep learning models, with support for TensorFlow and other backends.

NumPy:

- NumPy is a fundamental package for numerical computing in Python.
- It provides support for multi-dimensional arrays, matrices, and mathematical functions, making it essential for scientific and numerical computations.

Pandas:

- Pandas is a Python library used for data manipulation and analysis.
- It provides data structures like Data Frame and Series, along with functions for data cleaning, manipulation, and exploration.

Matplotlib:

- Matplotlib is a plotting library for Python.
- It enables the creation of a wide variety of plots, including line plots, bar plots,

histograms, scatter plots, etc., making it useful for data visualization.

Seaborn:

- Seaborn is a Python visualization library based on Matplotlib.
- It provides a high-level interface for creating attractive statistical graphics, facilitating the visualization of complex datasets.

Scikit-learn (sklearn):

- Scikit-learn is a machine learning library for Python.
- It provides a wide range of supervised and unsupervised learning algorithms, along with tools for data preprocessing, model selection, evaluation, and more.
- It also provides Label Encoder which is a utility used to convert categorical labels into numerical values by assigning a unique integer to each category, facilitating the training of machine learning models.

CHAPTER 4
METHODOLOGY

4.1 Working Mechanism

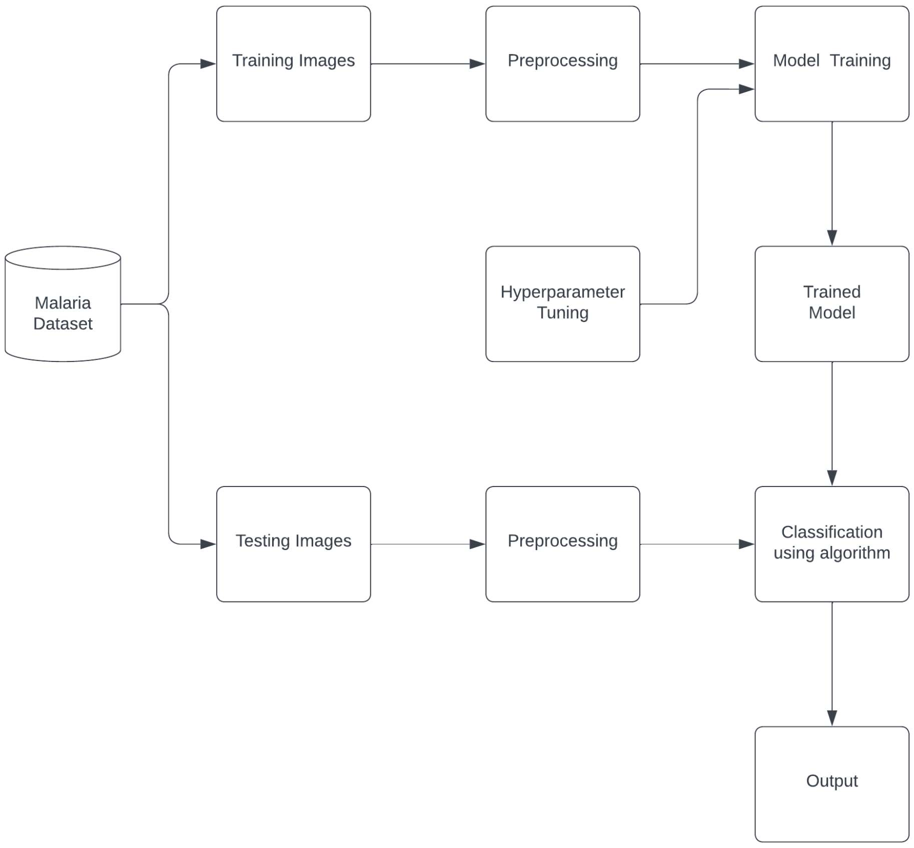


Figure 4.1: Block diagram of the system

1. **Malaria Dataset:** The foundation of our project lies in a comprehensive dataset sourced from the National Institutes of Health (NIH) Malaria Database. This valuable resource comprises a collection of labeled 27,558 cell images with 13779 being parasitized and 13779 being uninfected meticulously annotated for classification out of which 1344 images were removed from total dataset after data cleaning process which reduced the dataset to 26,214 images with 13,200 being parasitized and 13,014 being uninfected. The dataset's significance stems from its role in training and evaluating our model, paving the way for accurate malaria detection.
2. **Training images:** We allocated 90% of the total dataset for training. Within the training set, we further divided 81% for actual training and 9% for validation.

This ensured a sufficient amount of data for model learning while allowing for validation to fine-tune hyperparameters.

3. **Testing images:** The remaining 10% of the dataset was reserved exclusively for testing the final model's performance, ensuring an unbiased evaluation on unseen data.
4. **Preprocessing:** The raw input images are pre-processed to enhance image data and features. The pre-processing steps include image resizing, train-test validation split, normalization and image augmentation.
5. **Training:** The model is trained using the designated 81% training set images and their corresponding ground truths. The model architecture consists of a convolutional neural network with different layers that work together to recognize and classify the input images. The training process involves iterative updates of the model weights to minimize the loss function.
6. **Classification using Algorithm:** In the classification process using algorithms, we trained a model on a labeled dataset, teaching it to associate input images of RBC with specific predefined categories. This training phase involved adjusting the model's parameters iteratively to minimize the discrepancy between its predictions and the ground truth labels. Once trained, the model was used to classify new data, both in the training and testing phases. In both cases, the model took the input data and applied learned transformations to predict the corresponding class labels. During testing, the model's predictions were compared to the actual labels to assess its performance. By analyzing metrics such as accuracy, precision, recall, and F1-score, we evaluated how effectively the model generalized to unseen data and whether it accurately discriminated between different classes.
7. **Hyperparameter Tuning:** The training process did not happen overnight. After completing each stage of training evaluation, we collected and analyzed the model's bad cases. Based on this analysis, we made targeted adjustments to the proportion of training data and added synthetic data as needed. We then repeated this process of training and evaluation through multiple iterations, which allowed us to continually optimize the effectiveness of our model. As a result of this approach, we were able to improve the model's performance over time and achieve better results.

In hyperparameter tuning we explored learning rates, batch sizes, dropout rates, kernel sizes, filter numbers, and activation functions. Then we evaluated the effects on training stability, convergence, and model generalization. And selected optimal configurations maximizing performance that led us to the proposed model.

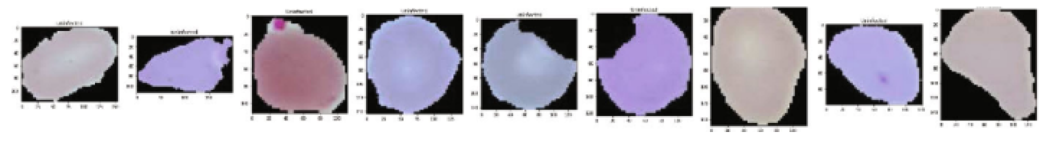
8. **Output:** Our output is a very accurate model which can classify RBC to differentiate between infected and uninfected cells.

4.1.1 Data Cleaning

In our project, data cleaning primarily focused on removing mislabeled images. This process was essential to ensure the accuracy and reliability of our dataset, as mislabeled images could potentially introduce errors and biases into our analysis. By meticulously reviewing and correcting mislabeled instances, we aimed to improve the quality of our data and enhance the performance of our classification model. Additionally, this approach helped maintain the integrity of our research findings and facilitated more reliable insights into malaria detection.

4.1.2 Image Augmentation

For image augmentation, we used a tool called ImageDataGenerator, which is part of the Keras library, to apply different transformations to our images. These transformations include things like rotating the images, shifting them horizontally and vertically, and flipping them horizontally. By doing this, we created more variations of our images, which helps our model learn better and make more accurate predictions. This technique is important because it helps prevent our model from becoming too specialized and allows it to perform well on different types of images.


(a)

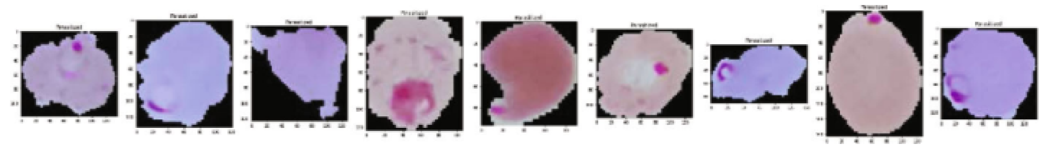
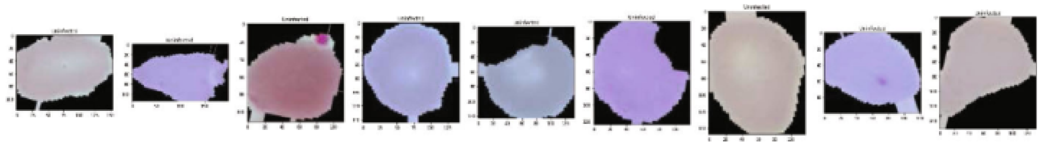

(b)

Figure 4.2: Sample image of RBCs without augmentation: (a) uninfected and (b) parasitized


(a)

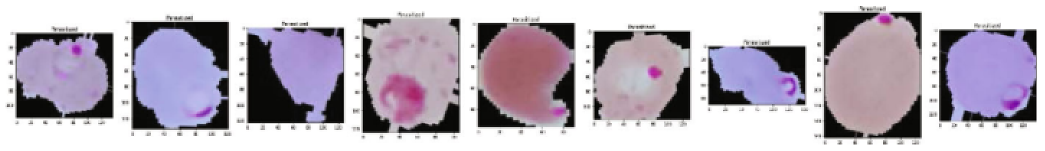

(b)

Figure 4.3: Sample image of RBCs with augmentation: (a) uninfected and (b) parasitized

4.1.3 Training CNN Model

In our project, we have opted to use Convolutional Neural Networks (CNNs) due to their well-documented effectiveness in handling image data and their ability to learn hierarchical representations directly from raw pixels.

By utilizing a CNN model with 20 layers and parameters like filters, activation functions, learning rate, batch size, dropout rate, optimizer, loss function, etc, our project aims to leverage the inherent ability of CNNs to automatically learn relevant features directly from raw image data, enabling accurate and efficient detection of malaria-infected cells.

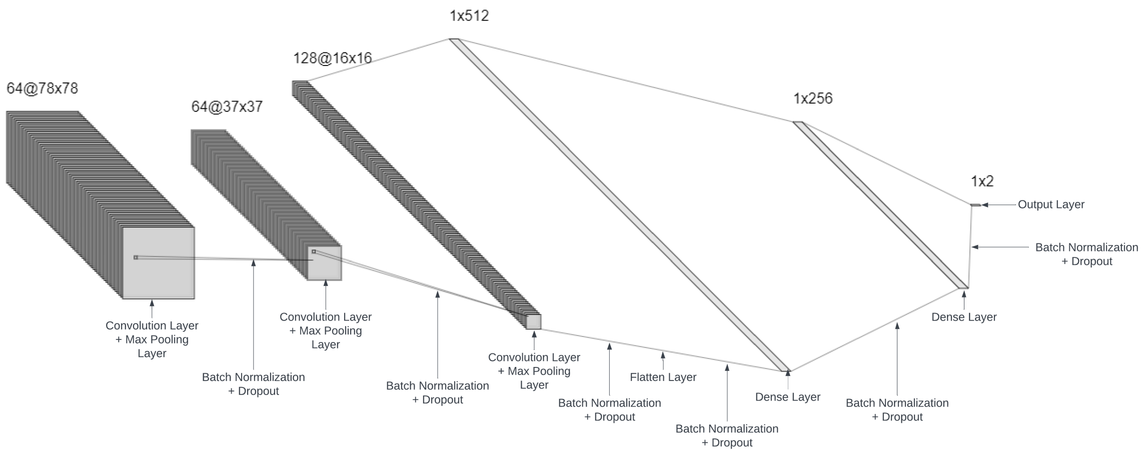

1x512
128@16x16
64@78x78
64@37x37
1x256
1x2
Output Layer
Batch Normalization + Dropout
Dense Layer
Convolution Layer + Max Pooling Layer
Convolution Layer + Max Pooling Layer
Batch Normalization + Dropout
Batch Normalization + Dropout
Convolution Layer + Max Pooling Layer
Flatten Layer
Batch Normalization + Dropout
Dense Layer
Batch Normalization + Dropout
Batch Normalization + Dropout

Figure 4.4: Model Architecture

We have a total of 4,443,330 parameters from which 4,441,282 are trainable parameters and 2048 are non-trainable parameters.

- Convolutional Layer:

  In our project, convolutional layers have been employed to extract important features from the input images. With 3 convolutional layers using 3x3 filters, the network can effectively capture various patterns and textures associated with malaria-infected cells. These layers are crucial for detecting intricate details that distinguish between infected and uninfected cells.

  We have used a total of three convolutional layers with 3x3 filters in our architecture. In the first convolutional layer, we employed 64 filters to capture local patterns and features present in the input images. This initial layer acts as a feature extractor, detecting basic patterns such as edges and textures.

  Following the first convolutional layer, we applied another convolutional layer with 64 filters of size 3x3. This additional layer further refines the extracted features, capturing more complex patterns and structures relevant to malaria detection.

  In the third convolutional layer, we increased the number of filters to 128, each with a size of 3x3. This deeper layer allows the network to learn even more abstract representations of the input data, facilitating the discrimination between malaria-infected and uninfected cells.

- Max Pooling Layer:

  Max pooling layers are utilized after each convolutional layer to downsample the feature maps, reducing their spatial dimensions while retaining the most relevant information. This process helps in preserving important features while decreasing the computational burden and controlling overfitting. In our project, max pooling is applied to condense the information obtained from the convolutional layers, making it more manageable for subsequent layers.

- Batch Normalization:

  Batch normalization is applied to normalize the activations of each layer across mini-batches during training. By reducing internal covariate shift, batch normalization helps in stabilizing and accelerating the training process. In our project, batch normalization is utilized to ensure that the network learns more efficiently and effectively, leading to improved convergence, generalization performance and reduced overfitting.

- Dropout Layer:

  Dropout layers with a dropout rate of 0.25 are incorporated after each block in our project. With a dropout rate of 0.25, we're effectively regularizing the network, improving its ability to generalize to unseen data and reducing the risk of overfitting.

- Flatten Layer:

  Flatten layers have been used to transform the output of the preceding layers into a one-dimensional vector, preparing it for input into a fully connected layer (dense layer).

- Dense Layer:

  After flattening the feature maps from the convolutional layers, dense layers are employed for high-level feature extraction and classification. In our project, we've opted for a dense layer architecture comprising 512 neurons after the first flatten layer, followed by 256 neurons in a subsequent block. Finally, 2 neurons are used in the output layer for binary classification, where one neuron represents each class (infected or uninfected). This configuration enables the network to learn complex patterns and make accurate predictions based on the extracted features.

- Adam optimizer:

  Adam optimizers have been used to efficiently update the model's parameters based on adaptive estimates of the first and second moments of the gradients, leading to faster convergence and improved generalization performance.

- Training and evaluation:

  Training and evaluation procedures have been conducted iteratively to optimize the neural network's performance on the given task, ensuring robustness and generalization capability.

- ReLU (Rectified Linear Unit):

  ReLU activation functions are employed in every convolutional layer and dense layer except for the final dense layer in our project. ReLU is chosen due to its simplicity and effectiveness in overcoming the vanishing gradient problem, which can occur with activation functions like sigmoid or tanh. By setting all negative values to zero and leaving positive values unchanged. In our project, ReLU is used to introduce non-linearity after each convolutional operation and dense layer, facilitating the network's ability to capture complicated features and patterns associated with malaria-infected cells.

- Sigmoid Activation Function:

  In the final dense layer, which serves as the output layer for binary classification (infected or uninfected), a sigmoid activation function is used. The sigmoid function squashes the output of the network between 0 and 1, effectively converting it into a probability score. In our Malaria detection project, the sigmoid activation function is applied to the output layer to produce the probability of each input image being infected with malaria. This allows for easy interpretation of the model's predictions and facilitates decision-making based on the likelihood of infection.

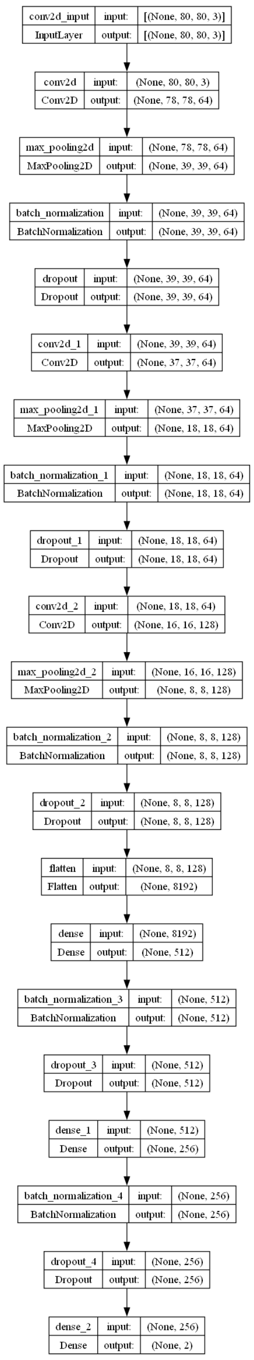

conv2d_input | InputLayer | input: [(None, 80, 80, 3)] | output: [(None, 80, 80, 3)]
conv2d | Conv2D | input: (None, 80, 80, 3) | output: (None, 78, 78, 64)
max_pooling2d | MaxPooling2D | input: (None, 78, 78, 64) | output: (None, 39, 39, 64)
batch_normalization | BatchNormalization | input: (None, 39, 39, 64) | output: (None, 39, 39, 64)
dropout | Dropout | input: (None, 39, 39, 64) | output: (None, 39, 39, 64)
conv2d_1 | Conv2D | input: (None, 39, 39, 64) | output: (None, 37, 37, 64)
max_pooling2d_1 | MaxPooling2D | input: (None, 37, 37, 64) | output: (None, 18, 18, 64)
batch_normalization_1 | BatchNormalization | input: (None, 18, 18, 64) | output: (None, 18, 18, 64)
dropout_1 | Dropout | input: (None, 18, 18, 64) | output: (None, 18, 18, 64)
conv2d_2 | Conv2D | input: (None, 18, 18, 64) | output: (None, 16, 16, 128)
max_pooling2d_2 | MaxPooling2D | input: (None, 16, 16, 128) | output: (None, 8, 8, 128)
batch_normalization_2 | BatchNormalization | input: (None, 8, 8, 128) | output: (None, 8, 8, 128)
dropout_2 | Dropout | input: (None, 8, 8, 128) | output: (None, 8, 8, 128)
flatten | Flatten | input: (None, 8, 8, 128) | output: (None, 8192)
dense | Dense | input: (None, 8192) | output: (None, 512)
batch_normalization_3 | BatchNormalization | input: (None, 512) | output: (None, 512)
dropout_3 | Dropout | input: (None, 512) | output: (None, 512)
dense_1 | Dense | input: (None, 512) | output: (None, 256)
batch_normalization_4 | BatchNormalization | input: (None, 256) | output: (None, 256)
dropout_4 | Dropout | input: (None, 256) | output: (None, 256)
dense_2 | Dense | input: (None, 256) | output: (None, 2)

Figure 4.5: Summary of the model

4.2 UML Diagrams

4.2.1 Use Case Diagram

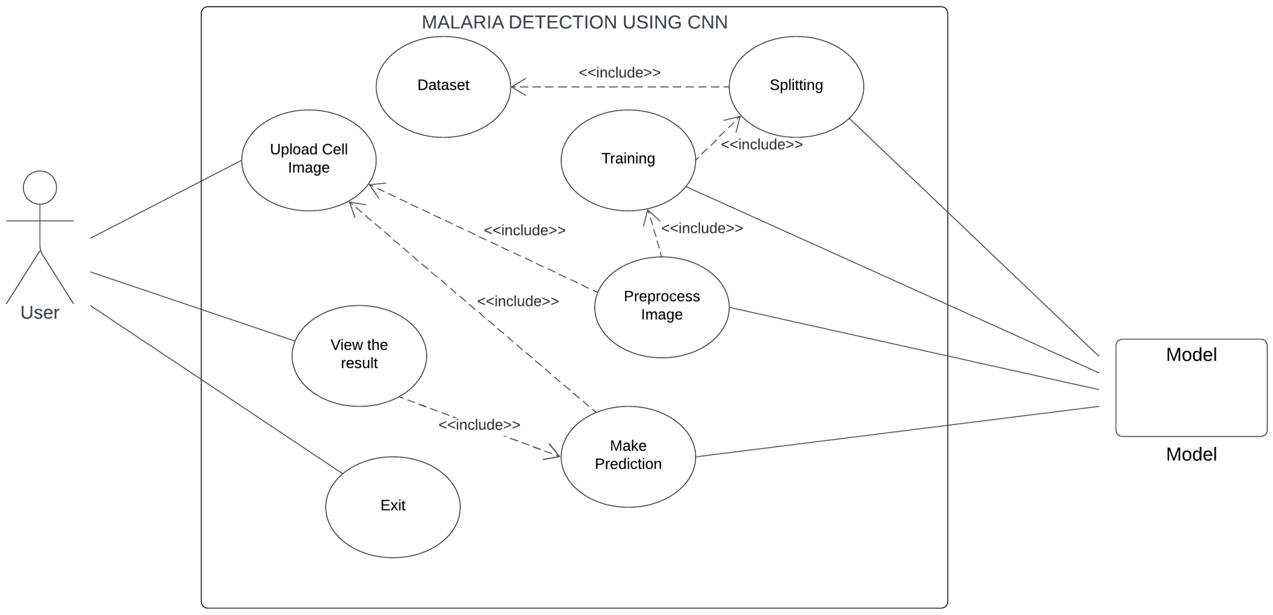


Figure 4.6: Use Case Model of the System

4.2.2 DFD level 0

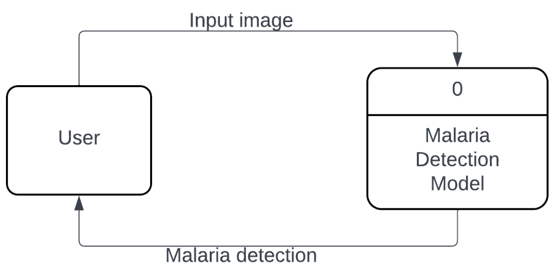


Figure 4.7: DFD level 0 Diagram

4.2.3 DFD level 1

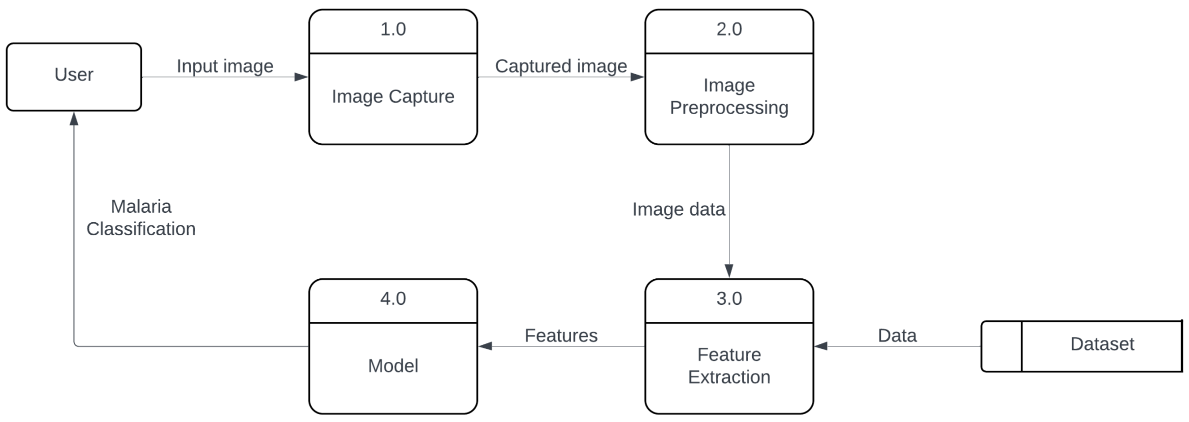


Figure 4.8: DFD level 1 Diagram

4.2.4 Activity Diagram

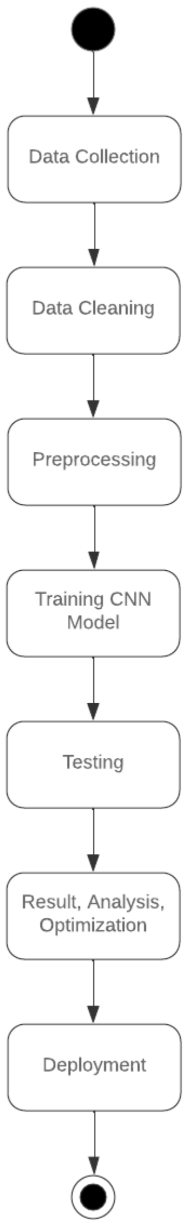


Figure 4.9: Activity Diagram of the System

4.3 Software Development Model

For the development of our system, we have implemented iterative and incremental model of Software Development. The software development life cycle follows a pattern of planning, developing, testing and rebuilding over and over to evolve the system through several stable intermediates until a final finished product is achieved. The development process that we have used will also focus on developing rapid prototypes over the entire cycle to verify the system outputs to the actual software requirements.

Initial Planning: In the foremost stage, we have analyzed the feasibility of the project based on several factors including requirements of our course of study, our domain expertise and knowledge, literature reviews and scope analysis, availability of dataset, practicality of concept and validate it for approval by the proper guidance of our project supervisor.

System plan: Once the project approval is received, we created a concrete plan for development of the system. To begin with, we started analyzing the use cases and several scenarios possible in the system and develop every possible alternative to the main success scenario that could likely occur in the system.

Analysis and Design: We designed the system models for proper visualization and understanding of tasks to be completed in each iterative cycle.

Implementation: After making the system models, we developed the codes and equivalent program sections required for the implementation of different parts of the system. The developed system code will be compiled to create a prototype of the system to depict expected output.

Testing: The developed prototype is then tested after implementation to check whether the goal of the iteration is achieved or not.

Evaluation: The system at last will be revised with required modifications during the testing phase until the system gradually evolves into a desired system.

CHAPTER 5
RESULT AND DISCUSSION

5.1 Results

This project has culminated in the successful development of a robust and efficient system for automated malaria diagnosis. Employing a custom CNN model tailored specifically for this task, the project has achieved remarkable accuracy in identifying malaria-infected red blood cells (RBCs). A user-friendly interface was meticulously designed, facilitating easy image uploads for analysis. Upon uploading an image of an RBC, the system promptly determines whether the cell is parasitized with malaria or not. With an accuracy rate of 99.315% the system demonstrates outstanding performance, underscoring its reliability and effectiveness in malaria detection. This achievement marks a significant advancement in medical diagnostics, providing a promising tool for early and accurate malaria diagnosis.

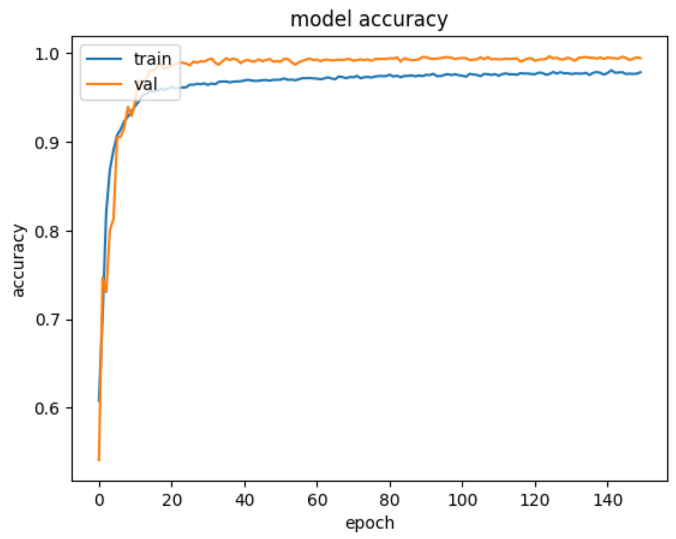


Figure 5.1: Accuracy graph of the system

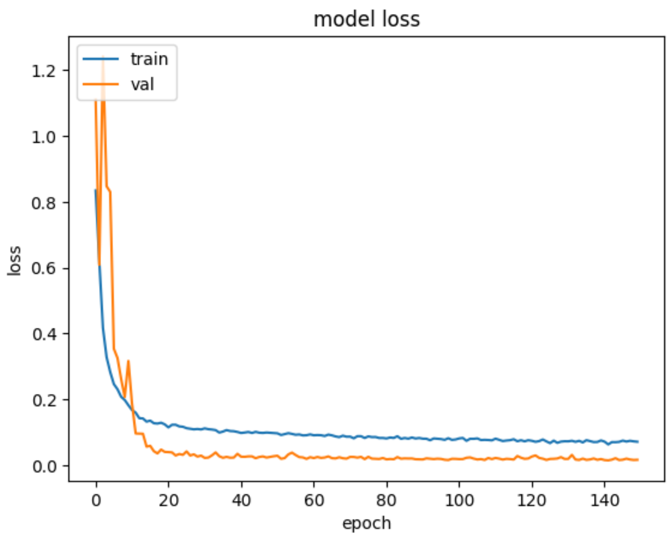


Figure 5.2: Loss graph of the system

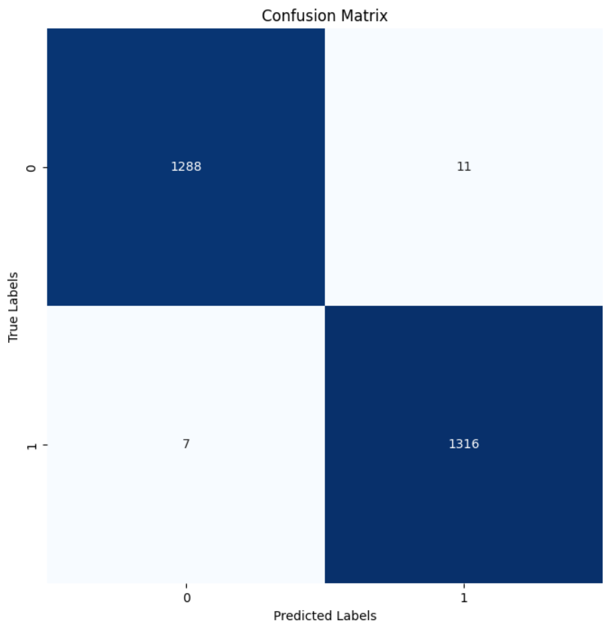


Figure 5.3: Confusion Matrix

In a set of 2622 test images comprising 1299 malaria-infected cells (positive) and 1323 uninfected cells (negative), our model achieved a true positive count of 1288, a false negative count of 11, a false positive count of 7, and a true negative count of 1316. The model demonstrated an accuracy of 99.3135%, precision of 99.4594%, sensitivity of 99.1531%, specificity of 99.4708%, and an F1 Score of 0.991903.

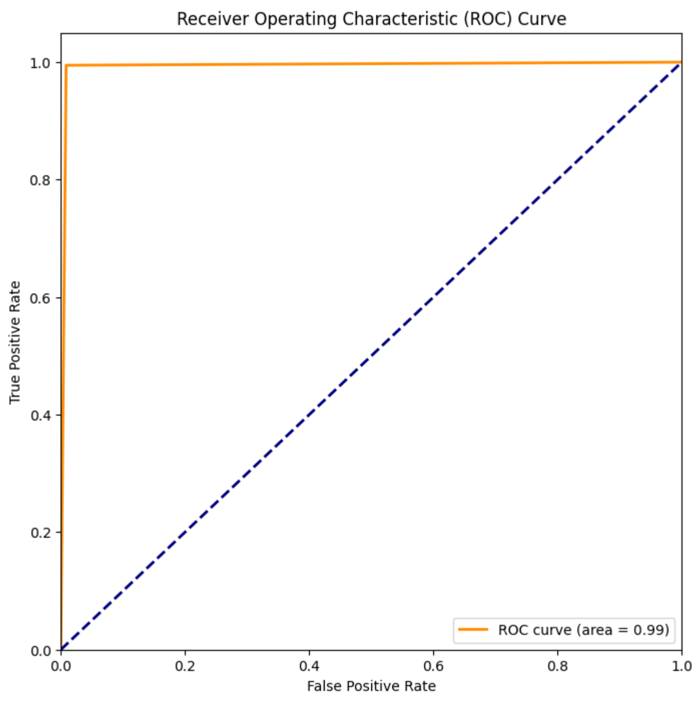


Figure 5.4: ROC curve

5.2 Discussion

In this project, we developed a web-based application that can predict malaria with high accuracy. Our system was trained using a dataset sourced from the official NIH website, ensuring the reliability and quality of the data. The malaria prediction accuracy of our project is notably high, reflecting the effectiveness of our approach in identifying malaria-infected red blood cells.

However, there are some limitations to our approach that warrant consideration in future work. One limitation is that our project focuses solely on malaria diagnosis without providing specific information regarding the type of malaria present. While malaria encompasses various types, our project's primary objective is to classify malaria based on the presence of parasites in red blood cells.

Additionally, while there are existing projects and research works on malaria prediction, many of them come with high costs. In contrast, our project offers a more cost-effective solution that can be implemented with greater accessibility.

Overall, our developed system has demonstrated promising results in predicting malaria by analyzing red blood cell images. However, further research and development are needed to address the limitations identified and improve the robustness and generalizability of the system.

CHAPTER 6
CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion

In conclusion, our project has made significant strides in improving malaria diagnosis. By creating a web based application and an easy-to-use interface, we've built a system that can accurately detect malaria in blood cells. The high accuracy and reliability of our system mean it could be a valuable tool for healthcare workers, helping them predict and diagnose malaria quickly and accurately. This project highlights the potential of technology to make a positive impact on healthcare, and we're excited about its future implications for improving medical diagnostics.

6.2 Limitations

- Our project is dedicated to the singular classification of individual cells. To accurately assess the severity of malaria, it is essential to determine the malaria density, which requires the examination of multiple cells in a person, while our project allows for the processing of one cell image at a time.
- Our project focuses on malaria diagnosis without providing specific information regarding the type of malaria present. While malaria encompasses various types, our project's primary objective is to classify malaria.
- For implementation of our model, the availability of red blood cell smears image is a must.So, microscope is needed for microscopic image of red blood cells which may not be available everywhere.

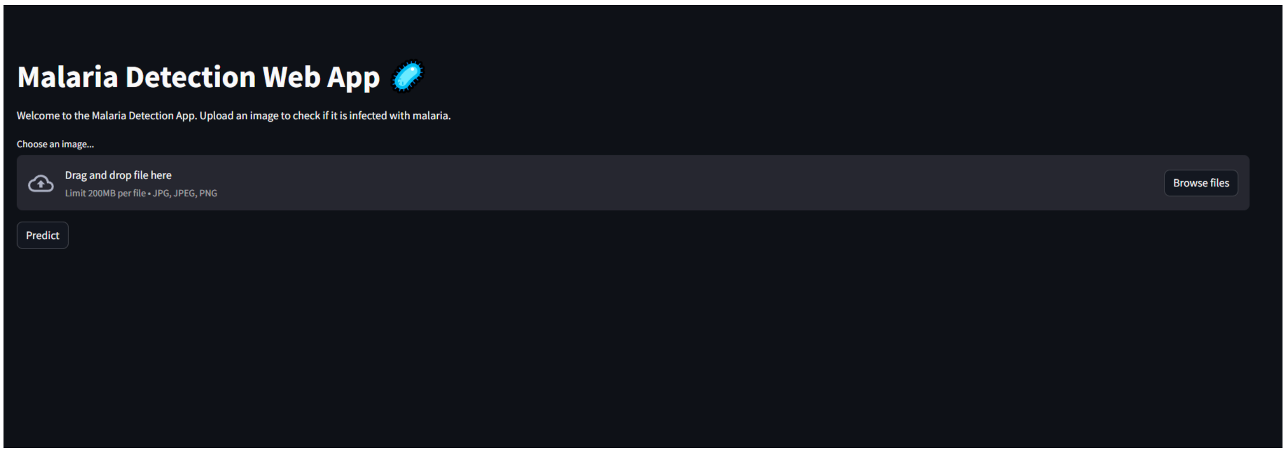
6.3 Future Enhancement

- While malaria comprises various strains, our project's primary objective is to establish a robust classification system. Future iterations of our project will incorporate the capability to specify the type of malaria detected.
- While our current framework enables the processing of one cell image at a time, future iterations of our project aim to expand capabilities to assess malaria sever-

ity and determine malaria density by analyzing multiple cells simultaneously advancing diagnostic methodologies and improving precision of malaria diagnosis.

- An embedded system incorporating an AI model, microscope, and blood smear images as input holds promise for enabling rapid malaria diagnosis in the near future.

CHAPTER 7
ANNEX



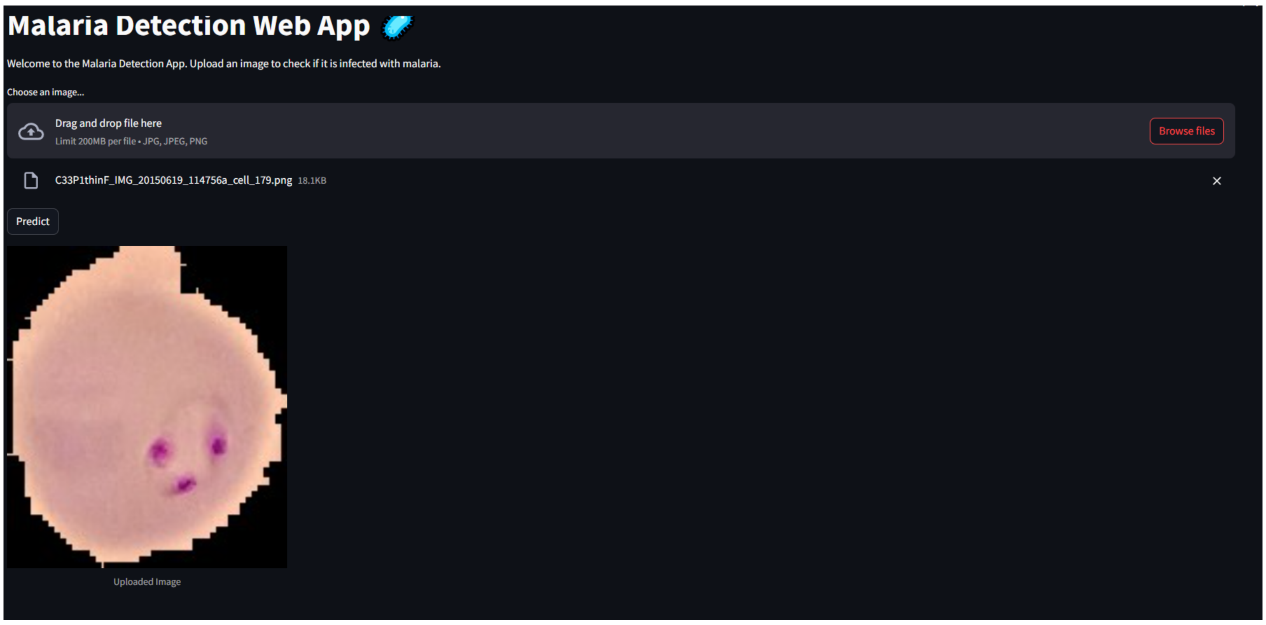
Malaria Detection Web App

Welcome to the Malaria Detection App. Upload an image to check if it is infected with malaria.

Choose an image...

Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG



Malaria Detection Web App

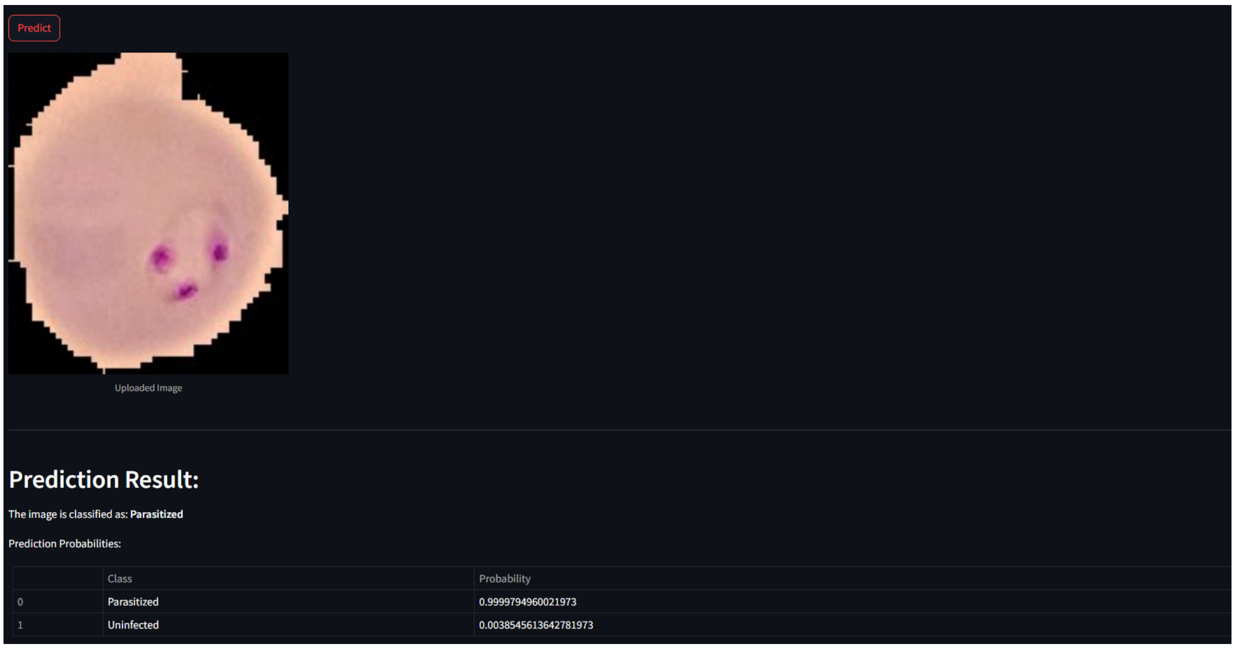
Welcome to the Malaria Detection App. Upload an image to check if it is infected with malaria.

Choose an image...

Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG

C33P1thinF_IMG_20150619_114756a_cell_179.png 18.1KB



Prediction Result:

The image is classified as: **Parasitized**

Prediction Probabilities:

	Class	Probability
0	Parasitized	0.9999794960021973
1	Uninfected	0.0038545613642781973

REFERENCES

[1] I. Talapko, T. Škrlec, T. Alebić, M. Jukić, and V. A., “Malaria: The past and the present,” *Multidisciplinary Digital Publishing Institute (MDPI)*, june 2019.

[2] S. Kumar, S. Priya, and A. Kumar, “Malaria detection using deep convolution neural network,” *Georgia State University*, pp. 3–5, 2022.

[3] V. Magotra and M. Kumar Rohil, “Malaria diagnosis using a lightweight deep convolutional neural network,” pp. 5–7, March 2022.

[4] T. Jameela, K. Athotha, N. Singh, V. Kumar Gunjan, and S. Kahali, “Deep learning and transfer learning for malaria detection,” June 2022.

[5] G. Madhu, A. Wagdy, S. Kautish, and M. Asif Shah, “Intelligent diagnostic model for malaria parasite detection and classification using imperative inception-based capsule neural networks,” August 2023.

[6] K. Fuhad, M. Tuba, J.F. Sarker, S. Momen, N. Mohammed, and T. Rahman, “Deep learning based automatic malaria parasite detection from blood smear and its smartphone based application,” *MDPI, Basel*, no. 5-8, May 2020.

[7] S. Shambhu, D. Koundal, P. Das, V. Hoang, K. Tran-Trung, and H. Turabieh, “Computational methods for automated analysis of malaria parasite using blood smear images: Recent advances,” *Computational Intelligence and Neuroscience*, vol. 2022, p. 18, April 2022.

[8] N. Tangpukdee, C. Duangdee, P. Wilairatana, and S. Krudsood, “Malaria diagnosis: A brief review,” *The Korean Society for Parasitology and Tropical Medicine*, june 2009.