Addis Ababa University

Addis Ababa Institute of Technology
Center of Biomedical Engineering

A Trilingual Android Application with Automatic Malaria Detection from Microscopic Images of Red Blood Cells

By

Berihun Nigussa Duresa

An Industrial Project Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Biomedical Engineering (Biomedical Instrumentation & Imaging)

Advisor: Dawit Assefa (PhD)

Addis Ababa, Ethiopia

December, 2023

DECLARATION

I, declare that this Industrial Project report submitted to the Center of Biomedical Engineering at the Addis Ababa Institute of Technology, Addis Ababa University in partial fulfillment of the requirements for the Degree of Master of Engineering in Biomedical Engineering (Biomedical Instrumentation & Imaging), is my own work with the exception of the paraphrased or quoted works whose sources are duly cited and acknowledged in the references.

Name: Berihun Nigussa

Signature: _____

Date: _____

This MEng. Industrial project has been submitted for examination with my approval as an advisor.

_____

Dawit Assefa Haile (PhD)

Certificate of Examination
Addis Ababa University
School of Graduate Studies

This is to certify that the Industrial Project prepared by **Berihun Nigussa** entitled:***A Trilingual Android Application with Automatic Malaria Detection from Microscopic Images of Red Blood Cells*** submitted in partial fulfillment of the requirements for the degree of Master of Engineering in Biomedical Engineering (Biomedical Instrumentation & Imaging) complies with the regulations of the University and meets the acceptance standards with respect to originality and quality.

Signed by the Examining Committee

Chairman:———————————— Signature:———————— Date:————————

Examiner:———————————— Signature:———————— Date:————————

Examiner:———————————— Signature:———————— Date:————————

———————————————————————————————————————-

Chief of Department or Graduate Program Coordinator

ACKNOWLEDGEMENT

Thanks to the Almighty God, who makes everything beautiful in its time. For his sake, this long journey with a lane full of bumps and obstacles came to accomplishment.

I am greatly thankful to my project advisor, Dawit Assefa (PhD), for his unreserved support, professional guidance in this project work and dedication to let us know the subject matter of the courses he taught us.

I also want to extend my gratitude to the Ethiopian Metrology Institute (EMI) for their full sponsorship of my MEng program and all staffs of EMI, the Center of Biomedical Engineering at AAiT and my friends who supported and encouraged me to complete the project. Special thanks go to my former staff member Mr. Wubshet Shimels (MSc) for his support in providing me reference materials during the course of the project. Finally, I owe my deepest gratitude to my parents, for their love, support, and guidance.

ABSTRACT

Malaria, which is a mosquito-borne blood disease caused by Plasmodium parasites, is one of the virulent infectious diseases affecting human beings and other animals since antiquity. Even though there were promising progresses in the reduction of malaria morbidity and mortality in the past two decades before the outbreak of COVID-19, the latest two reports of the World Health Organization (WHO) statistics indicate that malaria has been overlooked due to the COVID pandemics. Malaria is still prevalent specifically in low resource setting areas such as the sub-Saharan African countries, including Ethiopia. WHO reported that there were 229 million new cases of malaria and 409,000 deaths globally in 2019, alone. Whereas in the year 2021, the morbidity and mortality was reported to rise up to 247 million and 619,000, respectively. Timely diagnosis and treatment as well as good awareness about the disease play a major role to combat malaria. In the current project work, it was intended to design and develop a multi-lingual Android App that offers useful information about the malaria disease and is capable of automatically detecting malaria infected red blood cells (RBCs) from color microscopic images based on a deep learning approach. The Convolutional Neural Network (CNN) based deep learning model was trained, validated and tested on a publicly available dataset composed of microscopic images of RBCs taken from individuals with confirmed malaria infection as well as normal control groups. Experimental results generated from the deep learning model showed that the detection capability of the model achieved 100% training accuracy, 96% validation accuracy and 96% testing accuracy. The developed App avails useful information about malaria disease in general and tips users with fundamental information regarding its prevention and transmission mechanisms acting as an m-health system.

Key words: Malaria, Android App, Deep Learning, CNN, Automation.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AI Artificial Intelligence

AUC Area Under the ROC Curve

CNN Convolutional Neural Network

DL Deep Learning

DT Decision Tree

FN False Negative

FP False Positive

FPR False Positive Rate

GPU Graphics Processing Unit

HSV Hue-Saturation-Value

HTML HyperText Markup Language

ICTs Information and Communication Technologies

IDE Integrated Development Environment

IoT Internet of Thing

IRS Indoor Residual Spray

KNN K-Nearest Neighbor

LLIN Long Lasting Insecticide Treated Net

MCC Matthews Correlation Coefficient

MGG May Grunwald-Giemsa

ML Machine Learning

NIH National Institute of Health

NLM National Library for Medicine

NN Neural Network

PCR Polymerase Chain Reaction

RBC Red Blood Cell

RDT Rapid Diagnostic Test

RF Random Forest

RGB Red-Green-Blue

ROC Receiver Operating Characteristics

SVM Support Vector Machine

TN True Negative

TP True Positive

TPR True Positive Rate

UI User Interface

WHO World Health Organization

XML Extensible Markup Language

CHAPTER 1

INTRODUCTION

1.1 Background

Malaria, which is transmitted mainly through the bites of infected female Anopheles mosquitoes (malaria vector), is an acute febrile life-threatening disease caused by protozoan parasites of the genus *Plasmodium*. It can also be transmitted by infected blood transfusion, organ transplant, or the shared use of needles or syringes and from a mother to her unborn infant before or during delivery. It can be preventable and curable if certain measures are taken. The severity of the disease varies among individuals of different age groups and whether one has developed partial immunity or not. Infants and children under 5 years of age, pregnant women and patients with HIV/AIDS are at particular risk in malaria endemic regions and need special care.

According to a previous report by the World Health Organization (WHO), nearly 50% of the world's population is at risk of malaria and in the year 2021 alone, an estimated 247 million new malaria cases and 619,000 deaths were reported globally [3]. These figures show an increase of 51.34% morbidity and 7.86% mortality rate compared to the figures reported for the year 2019 [4]. It is more prevalent in Africa, specially in sub-Saharan regions with 90% of the global malaria burden share where 6% of this is contributed by Ethiopia alone [5] [6].

There are four major species of malaria that infect humans: *Plasmodium falciparum, Plasmodium malariae, Plasmodium ovale and Plasmodium vivax*; of these, the vast majority of malaria cases and deaths found in Africa are caused by the *Plasmodium falciparum* parasite. The *Plasmodium falciparum* parasite

infects both types of Red Blood Cells (RBCs), i.e., matured RBCs and immatured RBCs (reticulocytes). Other species such as *plasmodium vivax*, for example, infect the reticulocytes only.

Malaria has to be given a proper attention in countries found within tropical and temperate regions. For example in Ethiopia, one of the tropical region countries with 75% of its land mass found below 2,000m above sea level where 60% of the country's people dwell, is seriously affected by malaria [7]. This is because the climate of the land in this topographical region is conducive for the reproduction of malaria vector[8]. The coverage of malaria in Ethiopia is not uniform due to the topographical and climatic feature variations. Temperate regions at altitudes between 1,600m and 2,000m above sea level are also favorable habitats for mosquito [9]. Such forehand knowledge about malaria is essential to design appropriate interventions against the disease.

The main strategy for malaria control is timely, rapid and accurate diagnosis followed by effective treatment. The early and accurate diagnosis of malaria is essential for both effective disease management and surveillance. Current diagnostic methods include microscopic blood smear examination, Rapid Diagnostic Tests (RDTs), Polymerase Chain Reaction (PCR) based antigen detection, fluorescent microscopy, and new methods that rely on hemozoin (malaria pigment, by-product of malaria infection in RBC) detection [10]. Among these, microscopic blood smear image diagnosis is taken as the gold standard. Although malaria control efforts are being challenged by insecticide resistance, drug resistance and climate changes, there is an encouraging progress in the fight against malaria in the last two decades with the use of Long Lasting Insecticide Treated Nets (LLINs), Indoor Residual Spray (IRS), and through draining of small ponds where mosquitoes may lay their eggs. Malaria prevention is better and cheaper than curing. The importance of effective and practical diagnostics for malaria control is increasing, because effective diagnosis alleviates both complications and mortality from malaria [11].

The role of innovative tools becomes crucial in the fight against malaria. Several technologies have been introduced in the medical field both for detection as well as for treatment. Nowadays, the realm of Artificial Intelligence (AI) with its branches such as Machine Learning (ML) and Deep Learning (DL), is widely involved in the health sector, in spite of the trends seen some years back.

1.2 Statement of the Problem

Malaria is one of the deadliest diseases globally and more specifically in sub-Saharan Africa including Ethiopia. Countries where the disease is endemic have set specific goals to eliminate and finally eradicate the disease. One of the means of achieving these set goals is creating community awareness about the

disease. And the other means of controlling mechanism is a timely diagnosis and treatment of the disease. The current project considered both automatic malaria diagnosis and community awareness creation mechanisms and development of an android application with these functionalities. The current ICT technology allows creation of affordable, effective and efficient platforms to insure mass awareness about different diseases. Among these opportunities is use of mobile applications as a mobile health (m-health) option. Making such apps multi-lingual permits reaching out more number of people. It is equally important to have the m-health system able to automatically detect malaria given a patient's sample in a hospital setting where microscopes that could be coupled with a smart phone are available. In this regard, it was also intended in the current project to develop and test an algorithm that could take a microscopic image of blood samples and automatically detect malaria infected RBCs. Particularly, in places with inadequate number of qualified professionals that could read microscopy images for the purpose of malaria diagnosis, the automation could play a vital role.

1.3 Objectives

1.3.1 General Objective

The general objective of this project is to build an android application that gives a information about malaria in three different languages, namely: Amharic, Afaan Oromoo and English and implementing an AI model that can automatically detect if a red blood cell is infected with malaria or not.

1.3.2 Specific Objectives

The specific objectives of this project are:

- To collect, organize and compile relevant information about malaria to be included in m-health platform;
- To develop a proper AI model that is capable of distinguishing malaria infected red blood cells from microscopic images of blood samples;
- To train, validate and test the selected AI model using publicly available data sets, and
- To design a user friendly trilingual Android application.

1.4 Significance of the Project

About 60% of Ethiopians live in malarious regions and hence they need to be educated about this deadly disease to take necessary measures in order to protect themselves and contribute to the national goal of eliminating the disease from the country by 2030 [12]. Lack of awareness, inadequate availability of skilled experts for malaria diagnosis and language barrier are some of the main challenges in malaria control and prevention. In light of this, developing a android app that provides information about malaria with an intent of addressing majority of people considering languages that are largely spoken and understood is one of the primary goals of the current project work. The other significant contribution of this project is the alleviation of errors encountered among individual lab experts conducting microscopic diagnosis of malaria by introducing an android application based deep learning approach for malaria detection. By implementing an android based malaria application, it was aimed to improve people's awareness on the adverse effects of malaria on individual's health as well as its economic impacts and assist laboratory experts in their routine malaria parasite detection procedures based on blood smear microscopic images. This can lead to faster and more accurate diagnosis, enabling timely treatment of patients and potentially reducing the morbidity and mortality as a result of infection from malaria. Moreover, the implementation of this android app based awareness creation and diagnosis of malaria fosters the confidence of healthcare workers in malaria endemic areas.

1.5 Scope and Limitations of the Project

The outcome of this project is an Android application, which is intended to provide malaria related information in three popularly understood languages in Ethiopia. The App would be an invaluable tool for users as it provides useful information in a way that is easily understandable. The developed deep learning tool is able to automatically identify malaria infected RBCs with microscopic images of individual RBCs as inputs while malaria detection based on whole slide microscopic samples is beyond the scope of the current study. The developed app has three major limitations:

- The developed app is limited to the prediction of whether a cell image is malaria infected or not.
- The parasite species, developmental stage and parasitemia level determination are not considered in the current project.
- The app considered only three commonly spoken and understood languages: Amharic, Afaan Oromoo and English.

1.6 Organization of the Project

The rest of the project document has been organized into four chapters. The second chapter revises what technologies have been reported in the literature for use in automatic malaria diagnosis and available malaria information systems. Chapter three presents materials and methods used to develop the proposed deep learning model for use in automatic malaria detection as well as the complete Android application that provides useful information on malaria in general. Chapter four summarizes the overall results obtained in the project work with useful discussions. The last chapter concludes the project and gives recommendations on what could possible be improved in the future.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter reviews relevant literature obtained from open access online database, including Google Scholar, ResearchGate, PubMed®, and ScienceDirect. The literature search was focused on technologies for Android App Development, Malaria Diagnosis, AI, ML, DL and System Development methods using a combination of key search words which include '(Android Studio OR Data Analytic OR Machine Learning OR Deep Learning OR Smartphone OR Mobile Intervention) AND Malaria' written in English. For the Amharic and Afaan Oromoo information contents of the app, relevant documents were searched in the respective languages on the Internet.

2.2 Mobile Applications in Disease Prevention

Challenges facing the diversified nature of healthcare organizations vary widely around the world. In the developed countries, even though it may not be pervasive, there are sophisticated healthcare infrastructures in place. However, in the developing countries, access to healthcare facilities is not only very limited but also cumbersome due to issues such as poor road infrastructure and chaotic transportation systems. This challenges require developing countries use these limited health facilities in efficient ways to provide effective and equitable health services to their communities. This basically needs sound management that relies on reliable, relevant, comprehensive and timely information, which is recognized as an essential foundation for any public health intervention. A health information system can be defined

as a digital information that comes from different sources and that is ethically used through effective Information and Communication Technologies (ICTs) tools to generate strategic information for the benefit of public health [13]. Such systems are expected to provide health workers and health managers with systematic tools for decision making.

In recent years, the wide application of ICTs in the medical field has played tremendous roles in the healthcare delivery systems, such as in promoting patient-centered healthcare, improving quality of care, and educating health professionals and patients. It is believed that smartphone technology, which is an element of ICTs, including apps and app integrated wearable sensors, offers useful disease management potential [14]. Popularity, availability, increased ownership as well as technological capacities are making the smartphones attractive tools to assist patient self-management, continuous symptoms and vital sign monitoring, and ease of communication between patients and physicians [15]. Among the diseases that require such interventions in developing countries, malaria is one that claims the lives of many children under the age of five in sub-Saharan Africa [16].

2.2.1 Malaria Awareness Creation

Mankind has suffered from malaria since antiquity. As it has been narrated in [17], medical literature written 2,700BC back in China and India described about a disease what is most likely to be malaria. This disease also has got the power to grab the attentions of the ancient known philosophers such as the Greek Poet Homer (circa 750 BC), Aristophanes (445-385 BC), Aristotle (384-322 BC), Plato (428-347 BC) and Sophocles (496-406 BC) [18].

Known this long, malaria, a disease which can be cured, prevented, eliminated and eradicated, has been perceived differently in varies cultures. The disease was attributed to be caused by demons as a diety's punishment to humans. Until the first causative parasites evidence was found, it has been commonly known as *"bad air", "evil air"* or *"corrupted air"* due to the belief that it was supposed to be a miasmatic illness [19]. Later, the notion of the cause of the illness began to be believed as not all polluted air brings the ailment, but it was the night air that arises from certain kinds of ground, more specifically low, marshal areas that cause the disease. Such accusations of the physical nature to be a contagium is at least a few inches away from what really is causing malaria than that blames a god or a devil [20].

Such perceptions and cultural norms related to malaria often influence the effectiveness of the control practices being undertaken against the disease. The understanding of the probable causes, ways of transmission, and measures taken for the prevention and control differs in various communities and among

individuals. Studies show that fostering the public awareness on the true causes, ways of transmission, prevention and treatment of malaria have great positive impacts on the fight against malaria and recommend for its continuous improvement [21]. In such cases, community engagement (working with group of people who are geographically connected, share similar interest or are found in similar situations with respect to matters that affect their well being) has been adopted especially by countries with low and middle resource settings in a pursue to reach elimination of malaria by 2030, in accordance with WHO Global Malaria Strategy 2016-2030. The community engagement can play a vital role for malaria prevention, control and elimination by involving them in the formation of community leadership groups comprising local decision-makers, participation in drama campaigns and health education programs conducted in local languages and delivered in schools, mosques and churches; house-to-house visits by community health volunteers to improve early detection and timely treatment in rural areas [22]. Health interventionists use community engagement to trap communities in health promotion practices, research and policy-related decision making to advance knowledge and promote behavioral and environmental changes to improve health outcomes.

2.2.2 Mobile Application Development

Mobile application is a software that is developed for use on small computing devices, such as smartphones and tablets. Mobile application development is the process of making software for smartphones, tablets, and personal digital assistants (PDAs), most commonly for Android and iOS operating systems. Mobile app development requires at least basic knowledge of Java and other light weight programming languages such as JavaScript and markup languages such as HyperText Markup Language (HTML) and Extensible Markup Language (XML).

Since great portion of smartphones run on android, estimated to be 70%, and Google Play Store has fewer restrictions than the App store, many independent application development teams choose to build their apps for Android [23].

2.3 Malaria Diagnosis Methods

Early diagnosis is imperative in the combat of malaria prevention and control. Currently, there are many malaria diagnostic approaches being implemented which includes: slide blood smear microscopy examination, clinical diagnosis, RDTs, PCR, fluerescent microscopy, antigen detection and others [10]. Three of these commonly used malaria diagnosis methods are reviewed below.

2.3.1 Clinical Diagnosis

This method is traditional and most commonly practiced among medical professionals, where laboratory diagnosis is not available. For a patient suspected to be infected with a malaria, health workers dealing with the case have to decide what, how much and how frequent antimalarial drug to be administered after the confirmation of the parasitemia. Microscopic blood smear examination will be the first line choice, in the areas where resource settings allow the use of microscopy with required reagents and experienced professionals. But unfortunately, in most malaria endemic places, microscopic examination is rarely found and the only option available is the use of clinical signs and symptoms in the identification of a patient who really needs the treatment for malaria. In such practices, only common symptoms of malaria such as fever are checked and that could result in wrong diagnosis results.

The adverse side effect of such practice is that, patients with high fever are not properly diagnosed for other treatable diseases that show similar symptoms [24]. This type of diagnosis, which is undertaken based on the clinical signs and symptoms alone, is error-prone and leads to high mortality, drug resistance, and economic burden as a result of buying unnecessary medicines.

2.3.2 Slide Blood Smear Microscopy Examination

Slide blood smear examination under microscopy is the *gold standard* for malaria diagnosis [25]. The nonspecific nature of the clinical signs and symptoms of malaria may result in over-treatment of malaria or non treatment of other diseases in malaria endemic areas and misdiagnosis in non-endemic areas. Microscopic examination of malaria, which is one of the laboratory diagnosis approaches of malaria, uses May Grunwald-Giemsa (MGG) staining of thin and thick peripheral blood smears on microscope slides for contrasting purpose so that infected RBCs can be identified better.

The two types of peripheral blood smears (thick and thin smears) are used depending on the purpose of the analysis. Thick blood smear is a drop of blood on a glass slide. They are most useful for detecting the presence of malaria parasites, as they examine a large sample of blood. A thin blood smear is a drop of blood that is spread across a large area of the slide. It helps lab technicians to identify what species of malaria is causing the infection [26].

Even though this technique is considered as the *gold standard* in malaria diagnosis according to WHO, it has many limitations related to sensitivity, specificity, accuracy, precision, time consumed, labor intensiveness and the need for skilled microscopy technician. Microscopic malaria diagnosis involves visual examination of blood smears on glass slide if malaria parasite infected blood cells or not. This leads to

a subjective decision making regarding the presence or absence of the parasite in the blood cells among different clinicians.

2.3.3 Rapid Diagnostic Tests

This method is effective in areas where there are challenges of electricity and no other medical utensils are required. The problem with this method is that, some of the products are specific to only certain species of malaria parasites. RDT method is suitable in rural areas where electricity is a major problem to use other tools and requires only a general knowledge to use the product. Despite these benefits, the method has a number of shortcomings that include lack of sensitivity, inability to quantify parasitaemia and differentiate among *P. vivax, P. ovale and P. malariae*, higher cost compared to microscopy, and susceptibility to damage by heat and humidity. Microscopic systems do not suffer from these shortcomings and are considered to be effective for malaria parasite detection, provided that it is operated by a well trained microscopist.

2.4 Malaria Diagnosis Automation

The manual identification of malaria parasites may be difficult in a resource limited countries due to the lack of well trained technicians and quality microscopes with sufficient reagents. In this visual examination of the blood smears, the accuracy of the diagnosis is also significantly affected by the experience of the examiner and quality of the smear [27]. Currently, the field of computer vision has significantly advanced to the extent that object recognition tasks can be automated with help of AI technologies. Specially, its ML and DL branches are being widely utilized in medical image processing. What makes these branches of AI interesting is their ease of implementation in a web based, android application based and Internet of Things (IoTs) based (such as Raspberry Pis) settings. Fields of ML related to the classifications of datasets and some of the DL methods are reviewed in the following sections.

2.4.1 Machine Learning Malaria Diagnosis

Arthur Samuel, perhaps the first person to coin and define machine learning, defined machine learning (ML) as *"the field of study that gives computers the ability to learn without being explicitly programmed"* [28]. Stated in other terms, it is a sub-field of computer science which extracts information

from data applying computational and statistical methods to make predictions. There are three major categories of ML: **supervised, unsupervised**, and **reinforcement**.

Supervised machine learning works with labeled datasets and learns by connecting relationship between variables and known outcomes. It deciphers rules and patterns of data to create a model, which will be applied to a real world data after passing the training and testing stages. Decision trees, KNNs, Random forest, naive Bayes, Multi-layer perceptron and Support vector machines are some of supervised machine learning algorithms most commonly used in data classifications. On the other hand, unsupervised learning tries to find hidden patterns, groups and interpret the data solely based on the input. The third category, reinforcement learning, was some thing that was proposed to directing unsupervised machine learning and functions through trial and error based on a set rewarding/punishing mechanism. The current project work checks the applicability of a supervised ML as well as a CNN deep learning (to be discussed later) approach to automate malaria diagnosis based on microscope images of blood samples. While the final app was developed entirely on a deep learning platform. In the process of automating malaria diagnosis using ML algorithms for the dataset selected in this project, we have to pass through three common steps: preprocessing, feature selection and classification [29]. The input images are assumed to contain a single RBC and hence could be assumed segmented. Had the image dataset not already been segmented, the process would have included the image segmentation process, as a second step in the sequence, before final classification.

2.4.1.1 Support Vector Machine

SVM is one of the supervised machine learning algorithms that has gained significant attention and proven to be effective in medical image analysis. It can handle both linearly and non-linearly separable data. This characteristics of SVM allows it to handle complex and overlapping classes which makes the algorithm to be chosen for distinguishing between healthy and infected RBC images in automatic malaria diagnosis. The project under consideration is a binary classification problem, while SVM can also be used for multiclass classification problems, as discussed in [30]. As mentioned earlier, SVM can efficiently perform non-linear data using kernel trick in biomedical and biological applications. Common kernel functions are sigmoidal, polynomial and radial basis functions where kernel parameters have direct impact on decision boundary of SVM, a fictional border between classes.

The fictional border between classes is known as hyperplane. SVM performs classification of data by using the best possible hyperplane that segregates one group of data points of a class from data points of another class. The hyperplane of an SVM will be called best if it has larger margin between the two

linearly separate classes, considering a binary classification. The word margin stands for the maximal breadth of the plane parallel to the hyperplane that is void of interior data points (see also Fig. 2.1). In most practical cases, the algorithm optimizes the soft margin tolerating a few of the crossing elements of the opponent class [1].

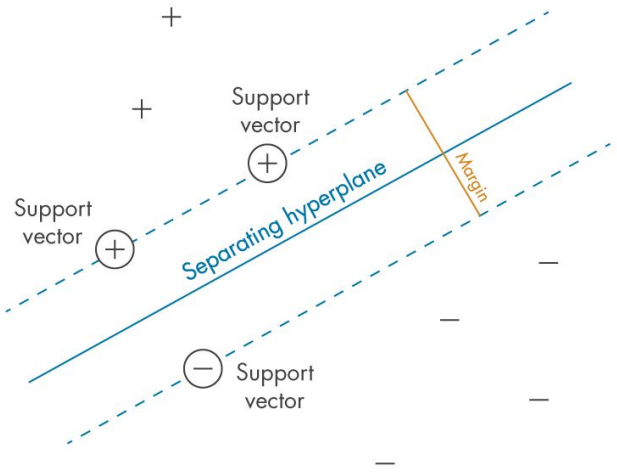

Figure 2.1: SVM for linearly separable datasets [1].

2.4.1.2 K-Nearest Neighbor

K-Nearest Neighbor (KNN) is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. It is a non-parametric classification technique that relies on readily available data to predict classes on new data. The KNN works on the principle that similar data will fall in the vicinity of its own clan, i.e., novel labels are predicted based on properties shared with other data points of its neighbors (K). It depends on the distance function used to measure similarity. With KNN, proximity is estimated by using the Euclidean distance function as shown in equation 2.1.

$$D(x, y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2} \tag{2.1}$$

where $x = x_1, ..., x_m, y = y_1, ..., y_m$ and m= the attribute values of two points x and y.

2.4.1.3 Decision Tree

Decision Trees (DTs) are one of the most powerful tools commonly used in supervised ML algorithms for image processing and pattern identification. It builds a flow-chart like tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label as depicted in Fig. 2.2. It is constructed by recursively splitting

the training data into subsets based on the values of the attributes until a stopping criterion is met, such as the maximum depth of the tree or the minimum number of samples required to split a node. It is usually utilized for classification models [31].

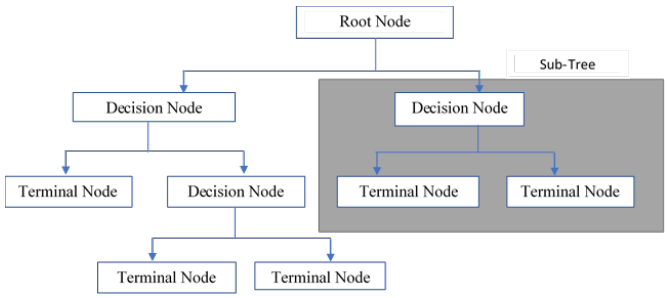

Figure 2.2: Decision Tree.

2.4.1.4 Random Forest

Random Forest (RF) is one of the supervised ML algorithms known for its simplicity and effectiveness. It can be defined as a *"Decision Tree-Based Classifier that chooses the best classification tree as the final classifier's classification of the algorithm via voting"* [32]. It follows specific rules for tree growing, tree combination, self-testing and post-processing, as it is robust to overfitting and it is considered more stable in the presence of outliers and in very high dimensional parameter spaces than other machine learning algorithms.

2.4.2 Deep Learning Based Malaria Diagnosis

Deep learning(DL) models revolutionized the process of traditional malaria detection steps by skipping the feature selection step that the experts in the field has to determine which feature to use for the classification. A deep learning CNN model recognizes patterns in microscopic images with much higher accuracy than ML algorithm approaches.

2.4.2.1 Convolutional Neural Network

Convolutional neural network(CNN) is one of the most popular forms of neural networks usually employed in image data classification. This is due to the fact that it is intrinsically enabled at automatically learning the spatial features of objects like edges, textures, and shapes which are imperative for the identification of objects in the image forms [33], similar to what human brains do in real world. It is widely used in automatic malaria detection researches [34].

In a convolutional neural network, input data are processed and channeled through layers of interconnected nodes, usually referred to as neurons, to produce an output. These interconnected layers are: **The input layer, the hidden layers and the output layer**.

The input layer is responsible for receiving an input data that is to be passed onto the next layer without performing any computational activity. The hidden layer is composed of several sets of the network such as Convolution layers, Pooling layers, and Fully connected layers. The final layer in the network is the output layer which is responsible for producing the output of the network. The input layer and layers from the hidden layers such as the convolution layers and the Pooling layers are used for extracting features from the input data whereas the fully connected layer and output layer are used for data classification.

The basic architecture of a CNN is depicted in Fig.2.3. As can be observed from the figure, one of the layers of the hidden layers is convolutional layer, which is the first layer in the sequence, where mathematical operation known as convolution is performed. Convolution is an element wise multiplication between a filter of specific kernel size and entries of the image data that fall under the size of the filter. The process continues by sliding the center element of the filter over each element of the image pixels. The convlutional layer is composed of many such filters. The concept of padding the original image comes into picture here inorder not to lose edge information. By convolving the input data, each filter generates a feature map. The combination of these feature maps is then passed through non-linear activation functions, such as the ReLU, Tanh and H-Switch functions, which are used to introduce non-linearities into the model to allow it learn several features of the input image such as corners and edges. The next layers of the CNN under consideration may include more convolutional layers, pooling layers and fully connected layers. Usually, a convolutional layer is succeeded by a Pooling layer. The aim of the Pooling layer is to decrease the size of the convolved feature maps in order to enhance the computational efficiency. Features generated by the convolutional layers are basically summarized by the Pooling layer, whose choice among the several types of pooling layer depend on the method being utilized. For example, a Max Pooling searches for the largest element in the feature map. Whereas Average Pooling computes the mean of the set of elements in the confined size of an image section. This CNN layer helps to generalize the extracted features and helps the network easily recognize the features independently. The fully-connected layers are typically situated after the Pooling layer and before the final output layer. It connects all the neurons in a layer to all the neurons in the subsequent layer, allowing the model to learn possible non-linear combinations of the features learned by the convolutional layers. The output of the preceding layers are flattened and inputted to the fully-connected layer. In this layer, some math-

ematical computations are performed on the flattened feature vectors and it is where the classification process commences.

The ouput layer of a CNN is the final layer that produces a probability distribution across the labels of the input data. The class with the largest probability distribution is taken as the output of the model.

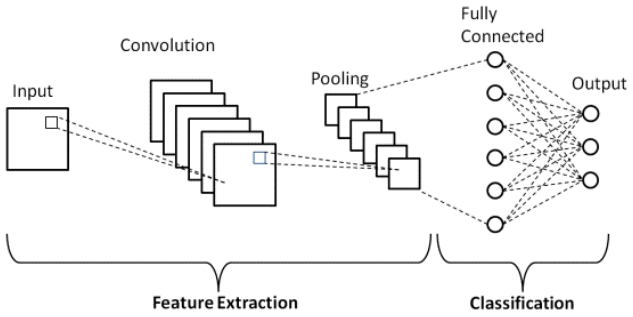

Figure 2.3: Basic Architecture of a CNN [2].

Clearly stating, CNN has the following structural components:

❶ **Convolution**: used to extract features from the input images. It preserves the spatial relationships between pixels by learning image features using small square of input data. Every image can be considered as a matrix of pixel values.

❷ **Max Pooling**: is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map. *Keras MaxPooling2D* is a pooling or max pooling operation which calculates the largest or maximum value in every patch and feature map. The results will be down sampled, or it will pool feature map which was highlighting the most present feature into the patch which contains the average feature presence from the average pooling. The max pooling is found to work well in average pooling for vision tasks.

Keras MaxPooling2D Arguments: Below are the arguments that we are used at the time of defining a keras maxpooling2d

- **Pool_size** this is a defined integer or tuple of two integers. This is specified as the window size which takes the maximum value into the pooling window which is 2*2. If we have specified only a single integer, then the same length is used for all dimensions.
- **Stride** is a number that indicates how many steps window of the filter or pool_size moves horizontally, vertically or both directions. Usually stride parameter is given to be 1 or 2.

Stride with parameter 1 is more common as it preserves high resolution features. When the dimensions of the input image are large, stride size of 2 and even more is employed to reduce the number of features and minimize computational cost.

- **Padding** is used to preserve the edge pixels of the image and keeps its dimension to be the same after convolution operations. In the Pooling operations, there are two parameters known as **Same and Valid.** When the **valid** parameter is used, there will be no image border padding with zeros, and the edge pixels will be lost. But when **Same** is employed, the height and width of the original image will be kept intact for convolutional operation and in the Pooling operation the edge pixels will also be considered.
- **Data_format** this defines the string which contains one of channel_last or channel_first. The dimensions ordering defines the input. Channels_last define the input shape as (batch, height, width, channels) while channel_first defines the input as (batch, channels, height, width). The default value (i.e. when unspecified) is the value of the image data format which was found in the keras configuration file.

❸ **Flattening**: is used to convert the multidimensional input to one-dimensional arrays, commonly used in the transition from the convolutional layer to the fully connected layer. The flattened matrix is fed as input to the fully connected layer to classify the image.

❹ **Full Connection**: A fully connected layer utilizes the output from the convolution process and predicts the class of the image based on the features extracted in the previous stages.

2.4.2.2 Mobile Apps for Malaria Detection

Various smartphone applications for different purposes are being developed using opens source software. For example, Android Studio is used for developing applications that run on Android operated devices. Recently released versions of Android, V4.2+, has an inbuilt TensorFlowLite library that facilitates the integration of deep learning models to smartphone applications. The availability of smartphones with sufficient speed and storage is allowing them to be one of the best candidates in the implementation of disease diagnosis automation [35]. The diseases range from those of plants to animals.

Different researches have considered the malaria disease diagnosis automation using models integrated on mobile phones. In a project that developed **Malaria Screener App** [36], 10 researchers collaborated with the National Institute of Health (NIH), National Library for Medicine (NLM) and other research organizations to design the Android Mobile Application which runs on affordable smartphones and yet

competing to be effective solution for automatic malaria detection through continuous upgrading. The application utilizes the embedded smartphone camera attached to the light-microscope via adaptor, to acquire slide image of RBCs. The app is capable of performing image acquisition, smear image analysis, and result visualization in its slide screening process, and is also equipped with a database to provide easy access to the acquired data. The open source codes of the application can be freely accessed and require advanced knowledge of Java programming language for customization.

In another study [34], a team of six engineers from the North South University of Bagladesh developed a program that can automatically diagnose malaria using a smartphone given a segmented and cropped microscopic images of RBCs as an input. This was intended to solve the issue of the requirement for the expensive equipment and highly trained personnel needed for malaria diagnosis in resource-limited countries. The model developed by this group of engineers was capable to be implemented offline on mobile phones as well as online in web based applications. The researchers used a public dataset containing 27,558 images of RBCs from 150 infected and 50 healthy patients to train the model.

CHAPTER 3

MATERIALS AND METHODS

This section explains the material and methods used for the Android Application development both for awareness creation and automatic malaria parasite detection on MGG stained RBC microscopic images. It deals with materials and methods used to develop the ML models, DL model and the android application.

3.1 Dataset and Data Preprocessing

The Android app developed in the current project uses deep learning (DL) CNN model to perform RBC image classification for automatic malaria detection. Neural Networks (NNs) have performed well on automatic feature extraction and acted as very good image classifiers [37]. In the dataset to be used in this project, the infected cells seem to contain some red globular structures whereas healthy cells do not seem to contain such structures in them. The proposed deep learning model will use these patterns in cell images to effectively detect malaria parasites in a patient. Though the Android app integrates only the deep learning scheme for parasite detection, different machine learning schemes have also been developed and tested in the current project for use in malaria parasite detection and each of the schemes are described in the subsequent sections in this chapter.

3.1.1 The Dataset

The malaria cell image dataset has been acquired from Kaggle, a freely available database which cab be accessed from the NIH official website. It has 27,558 images with an equal number of parasitized and uninfected cells. The images are variable in color intensity and size, ranging from 46x46 to 385x395 pixels with 3 color channels (RGB) [38]. Selection of some value around the median dimension of the dataset will trade-off between easing computational extensiveness of large dimensional images and accuracy of performance of developed automatic malaria detection model. Proper selection of the image dimension to make the images uniform has a great impact on the performance of ML and DL models.

3.1.2 Image Data Preprocessing

Data preprocessing is essential for the effectiveness and efficiency of the next processing steps. It enhances the quality of the blood smear images to improve the accuracy of later cell segmentation, feature extraction and classification steps.

In the course of capturing an image, some unintended phenomena may be encountered and that may distort the image information. Image preprocessing improves image quality by suppressing unwillingly added artefacts that distorts image features. It is not only limited to the removal of these artefacts but also reach far to include geometric transformations such as rotation, scaling, translation etc [39]. Image preprocessing in the current project is meant to improve the quality of the microscopic images allowing easier differentiation of normal and malaria infected RBCs.

Five preprocessing steps were employed for the ML classifiers, namely: color space conversion, image filtering, histogram normalization, image resizing and feature extraction. In the case of the DL algorithm, two preprocessing steps were considered: image resizing and image normalization. As mentioned in section 3.1.1, the image dataset varies in size and is exposed to different noises. First the original images found in Red-Green-Blue (RGB) color space are converted to Hue-Saturation-Value (HSV) color space where each channel of the HSV is separately filtered using median filter and then merged together to ease the process of re-conversion into its RGB format. ***cv2.cvtColor()*** is a very powerful python tool in such color space conversions. Median filter is selected over the other vastly found noise filtering techniques because it preserves edges during the noise removal [40]. The conversion between the color spaces is required because in the HSV color space, each channel can be separately manipulated without affecting the other channels, a problem which is noticed in RGB color space.

3.2 Input Data Features For Machine Learning

The prediction performance of ML models is influenced by the selected and extracted features from the input data. As it will be obvious, not all features of data will be important for the ML model performances. This means, some of the features could be irrelevant and replicas of the others. These kinds of features can even negatively affect the gross performance of the model. Therefore, it is imperative to pick out and take essential features from the data and exclude those that do nothing for the outcome of the model.

In this regard, there are two methods usually employed in ML data analytics. The first method deals with minimization of input variables by reducing repetitive and non-relevant input data features. This method is known as feature selection. The other method that plays important role for the enhancement of the ML performance is the input data feature extraction process. These two methods have the same goal but completely different from each other.

Feature selection deals with selecting those essential features necessary for the ML performance from the pool of input data features leaving those non-relevant ones. In doing so, it does not create new features, but those selected features become subsets of the original feature set. Feature selections maintains the features in their native format.

On the other hand, in the feature extraction process, new features are created from native input data features. Like feature selection technique, feature extraction is used to minimize the vast number of input data features, makes the model less complex, reduce model overfitting and increase model computational efficiency.

For the Parasitized and Uninfected RBC image datasets, the differentiating features could be the characteristic color variations observed between the two classes. Haralick textural features of the gray level images and the shape changes that could result from the infection can also be considered. Out of these features, **Color Histogram** was found to be more effective in its distinguishing ability between the two classes in the current study. It has also been used by other researchers as a very powerful feature for automatic malaria detection researches that involved analysis of microscopic RBC images [41], and hence adopted in the current study. A bar plot of a quantized pixels of color image over a certain intervals is known as color histogram representation. These intervals into which the image pixels were to be assigned are called bins. This project utilized 8 bins to collect the image pixels of the dataset. Bin size selection is a subjective judgmental determination and is considered to be enough if it falls between 5 to 20, in most cases [42].

3.3 Data Preparation for Machine Learning

From the total dataset, discussed in section 3.1.1, 70% of the data were used for training, 10% were used for vlidation while the remaining 20% were used for final testing. These correspond to 19,290 microscopic images for training, 2,756 for validation and 5,512 for testing. The 5,512 test images were randomly selected using a random sampling technique and stored separately in a folder named *test-data*. The remaining 22,046 images were stored in a folder named *dataset*. Using the Python sklearn's *train_test_split* library functionality, the data in the *dataset* were split into training and validation sets. Python randomly selected 2,756 images for validation and the rest 19,290 were used for training. The final partitioned data is indicated in Table 3.1.

Category	Training	Validation	Testing
Parasitized	9633	1390	2756
Uninfected	9657	1366	2756

Table 3.1: Data For Machine Learning Classifiers.

3.4 Data Preparation for Deep Learning

In the implementation of the CNN based deep learning model, the publicly available free blood smear image dataset acquired from the kaggle website were first imported to the Colab platform. The dataset is grouped in two separate folders each of them consisting of 13,779 images representing the infected and healthy RBC microscopic samples. Note that the dataset is composed of a total of 27,558 images with equal number of healthy and infected microscopic samples.

Three folders were created with each of the folders composing two sub folders. The three folders were labelled as *training, validation* and *testing*. The two subfolders in each category are named as *infected* and *uninfected*. Among the 13,779 infected samples, 9,645 were randomly picked to the training folder's infected sub-folder, 1,378 to the validation folder's infected sub-folder, and 2,756 to the testing folder's infected sub-folder. Same step was applied on the uninfected samples. In total, 19,290 images were set for training, 2,756 for validation and 5,512 images for testing, i.e. 70% of the dataset were used for training, 10% for validation and 20% for finally testing the performance of the trained model. The proportion of the data after this splitting operation is indicated in Fig. 3.1. After the data partitioning, all images were resized to the median size of all the 27,558 images considered in this study. The median size was calculated to be 130x130 pixels and all assumed a uniform image size. All images were further

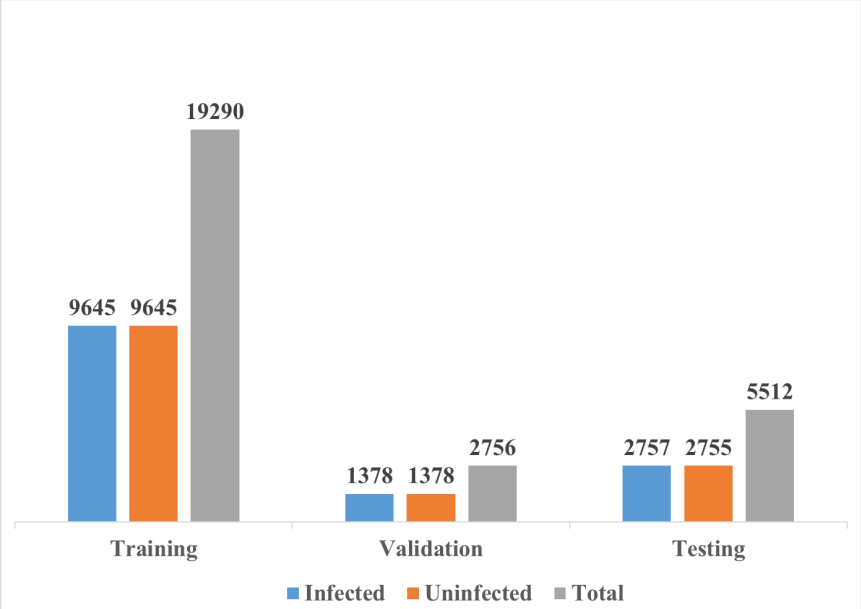

Figure 3.1: CNN Data proportions.

normalized between 0 and 1 before they are inputted to the DL model. Selected sample images from both infected and normal classes are presented in Fig. 3.2.

3.5 Training Classifiers

Once the data preparation is complete, it was subjected to training using selected selected ML and DL classifiers. In the case of ML classifiers, Decision tree, KNN, SVM, Random Forest, Gaussian naive Bayes, and Multi-layer Perceptron were used and in the case of DL, the classical CNN algorithm was utilized.

3.5.1 Training the Machine Learning

The ML algorithms were trained on a Jupyter Integrated Development Environment (IDE), an open source software, found on Anaconda Navigator, which helps to write modular python codes. The complete python code for all the ML algorithms considered in the current study are presented in Appendix 5.2C . As discussed previously, the features extracted to be inputted to the ML algorithms was Color histogram. A bin size of 8 was assumed when computing the Color histogram. Note that the feature extraction was executed once the input images passed through the four preprocessing stages (RGB to HSV color space conversion, median filtering, hitogram normalization and image resizing into size 130x130).

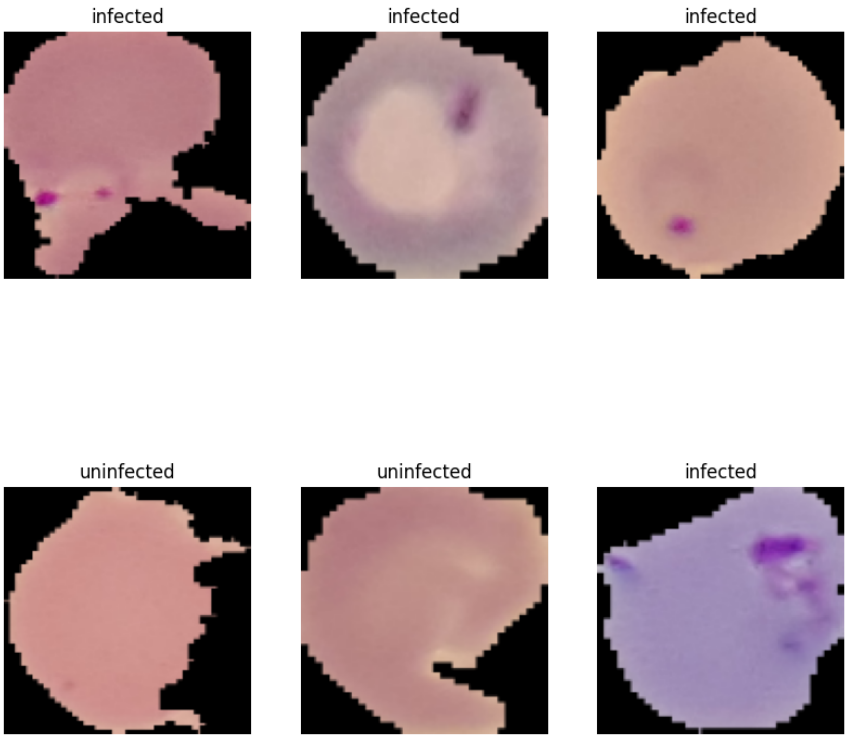

Figure 3.2: Sample Datasets.

3.5.2 Training the Convolutional Neural Network

Google Colab, open-source software, was used to train the CNN model. The platform supports libraries including TensorFlow (used for image classification), matplotlib (for plotting graphs), pandas (for data manipulation and analysis), OpenCv (for computer vision tasks), sklearn (for data mining and analysis) and Numpy (for scientific operations), all discussed in section 3.6.1.

The model was trained with three convolutions having 32 filters, 3x3 kernel size, a ReLu activation function, same padding and a 2x2 maxpooling in the first convolution; 64 filters, 3x3 kernel size, ReLu activation, same padding and a 2x2 maxpooling in the second convolution and finally with 128 filters, 3x3 kernel size, a ReLu activation, same padding and a 2x2 maxpooling in the third convolution. The data was then Flattened to be passed to the Dense layer (256 Dense layers with ReLu activation). As an optimizer function, Adam was utilized since its learning rate is optimal in such image datasets. Finally, 2 Dense layers were used since we are dealing with a two class of data, *Parasitized* and *Uninfected*. The complete code used for the DL model training is given in the Appendix 5.2 B.

3.6 Materials

In the implementation of the current project, several software, supporting libraries, an android operated smartphone, and a personal computer were used and each are briefly explained below.

3.6.1 Major Software and Libraries

☞ **Android Studio:** is an official IDE for Android App development. This project used Flamingo 2022.2.2 Android Studio version.

☞ **Jupyter Notebook:** is an open-source web application that allows us to write and execute python codes in a modular form. In this project, it was integrated with 3.8.8 python version. It has many inbuilt libraries and it is easy to add and upgrade other python libraries. It is considered an excellent IDE for most ML code development.

☞ **Google Colab:** Colab, a short for Colaboratory, is a product from Google Research that allows to write and execute python code through a browser, similar to Jupyter Notebook. It is also a well suited environment for ML, DL and data analysis. More technically speaking, Colab is a hosted notebook service that requires no setup to use, while providing access free of charge to computing resources including Graphics Processing Unit (GPU) that are specially required for image processing.

☞ **Latex Overleaf:** is a free mathematical typesetting software available through different means. It is a cloud-based system, which enables users do their word processing online and download the output files in a PDF format.

✍ **Scikit-learn:** is an open-source python library integrating a wide range of state-of-the-art ML algorithms for medium scale supervised and unsupervised problems. It is used for data preprocessing, cross-validation and building ML models.

✍ **Matplotlib:** is a comprehensive python library for creating static, animated, and interactive visualizations. It enables us to create various graphs.

✍ **TensorFlow:** is an open-source python library which makes things easy for beginners and experts to create ML and DL models for desktop, mobile, web and cloud applications. In the current project, a TensorFlow Lite has been used, the version which is used on mobile and embedded devices like Android, iOS,and Raspberry Pi.

✍ **Keras:** is a simple, flexible, and powerful DL library written in python empowering engineers and researchers to take full advantage of the scalability and cross-platform capabilities of the TensorFlow platform.

✍ **OpenCV:** is a cross-platform library used for developing real-time computer vision applications. It mainly focuses on image processing, video capturing and analysis including face and object detection.

3.6.2 Hardware

The project work was implemented on a PC with the following specifications:

✍ **Processor:** Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz 3.19 GHz

✍ **RAM:**8.00 GB (7.84 GB usable)

✍ **OS:** Windows 10 Pro

✍ **System type:** 64-bit operating system, x64-based processor

✍ **Hard Disk:** 1TB

The smart phone used for the App development has the following specifications:

✆ **Android Version:** 10

✆ **Device storage:** 16GB

✆ **RAM:** 2GB

3.7 Performance Metrics

The performance of the proposed ML and DL models built for automatic malaria diagnosis are evaluated using appropriate metrics which are most commonly used in the assessment of the effectiveness of the models. **Confusion Matrix, Precision, Recall, F-Score, Accuracy, and AUC (Area Under the Curve)-ROC** are some of the most commonly used performance metrics in the machine and deep learning models.

- **A confusion matrix**, also known as error matrix, is a kind of truth table that provides summarised classification results, showing a predicted and actual categories. For a binary classification, such as *Parasitized* or *Uninfected*, the confusion matrix is 2x2 table where the cells of the table are represented by: True Positive, True Negative, False Positive and False Negative:

- **True Positive (TP)**: refers to a sample belonging to the positive class being classified correctly.
- **True Negative (TN)**: refers to a sample belonging to the negative class being classified correctly.
- **False Positive (FP)**: refers to a sample belonging to the negative class but wrongly classified as it belongs to the positive class.
- **False Negative (FN)**: refers to a sample belonging to the positive class but wrongly classified as it belongs to the negative class.

- **Accuracy** is a very popular metric that tells us out of all the predictions, what percentage are correct. In other terms, it is a fraction of correctly classified samples from the total samples, and mathematically expressed using elements of confusion matrix as follows:

$$Accuracy = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{3.1}$$

- **Precision** is the fraction of true positives(TPs) out of the sum of true positives (TPs) and false positives(FPs). It can be computed using the following formula:

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.2}$$

- **Specificity** is the fraction of true negatives(TNs) out of the sum of true negatives(TNs) and false positives(FPs). It can be computed using the following equation:

$$Specificity = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{3.3}$$

- **Recall**: usually known as **sensitivity**, is the fraction of true positives(TPs) out of the sum of true positives(TPs) and false negatives(FNs). It can be computed using the following formula:

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.4}$$

- **F1-score** is the harmonic mean of the precision and recall. It can be calculated using the following formula:

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3.5}$$

- **Matthews Correlation Coefficient (MCC)** is one of the ML performance matrics that considers all the four elements of the confusion matrix. It is a kind of correlation coefficient that ranges in value from -1 to 1. It is helpful for identifying if the classifier is a simple random guesser or effective by classifying especially the negative class samples. MCC can be computed using the following formula:

$$\text{MCC} = \frac{\text{TN} * \text{TP} - \text{FP} * \text{FN}}{\sqrt{(\text{TN} + \text{FN})(\text{FP} + \text{TP})(\text{TN} + \text{FP})(\text{FN} + \text{TP})}} \tag{3.6}$$

- **Receiver Operating Characteristics (ROC)** curve is a plot of True Positive Rate (TPR) against False Positive Rate (FPR) at different classification threshold settings. It shows the model's distinguishing ability between infected and uninfected cases.

  To understand the definition of ROC, it is imperative knowing what we mean by TPR and FPR.

  - **True Positive Rate(TPR)** coincides with sensitivity or recall in definition. Therefore, it is computed using the formula given in equation 3.4.
  - **False Positive Rate(FPR)** defined as the number of negative class samples predicted wrongly to be in the positive class, out of all the samples in the dataset that actually belong to the negative class. Mathematically it is represented as the following:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{3.7}$$

    which is same as $\text{FPR} = 1 - \frac{\text{TN}}{\text{FP} + \text{TN}}$ or 1-**Specificity**.

- **Area Under ROC (Area Under the ROC Curve (AUC))** is another important metric that measures the overall performance of a classifier. As the name suggests, it is simply the area measured under the ROC curve. A higher value of AUC represents a better classifier performance.

3.8 System Development

3.8.1 The Agile Environment

Among the two most commonly employed system development processes, viz., Plan-driven and Agile processes, we are going to follow an agile development method which is a more recent and came to fill the gap observed in the traditional plan-driven methods which is rigid and hard to entertain dynamically

changing user requirements in software development activities. As indirectly stated, agile development method is flexible to changes with user requirement which is an inevitable nature of human beings. In agile software development, the system development processes (life-cycles) such as specification, design, implementation and testing are interleaved and the outputs from the development processes are decided through a process of negotiation among the stakeholders during the software development.

3.8.2 Requirements Engineering

Requirements engineering is the process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed. The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process. Roughly speaking, requirements can be categorized into two: Functional Requirements and Non-Functional Requirements.

3.8.2.1 The Android Application Functional Requirements

Functional requirements are statements of services that the system should provide, how the system should react to a particular inputs and how the system should behave in particular situations. It may also state what the system should not do. The following list shows some of the functional requirements of the application:

- **Display information about malaria:** The app displays information about malaria quickly with a simple click.
- **Screen Size:** Operates on different smartphone screen sizes.
- **Switch language option:** The user can choose any of the three languages (Amharic, Afaan Oromoo or English) to read about malaria at the starting of the application or any time in between.
- **Switch between malaria information content:** A user can select any of the topics he/she wanted to read about from the page that appears after selecting language option in the user interface that appears first or can choose from a menu by simply swiping the screen any time he/she has an interest of changing title.
- **Connect to the provided YOUTUBE link:** opens a video about malaria with useful lessons about ways of transmission, prevention and the like by known specialists in the area.

- ☞ **Dial instantly:** The App also provides a link to dial to the ministry of health professionals to consult about malaria.

- ☞ **Detect infected RBCs on microscopic images:** check if an RBC is malaria infected or not based on processing a microscopic image of the RBC.

3.8.2.2 The Non-Functional Requirements

Non-functional requirements may include constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc. They often apply to the system as a whole rather than individual features or services. The following non-functional requirements have been identified for the Application developed in the current project:

- ☞ The App requires on android operated smartphones with an application programming interface (API) level of equal to or greater than 21 (Android version 5.0 Lollipop).

- ☞ The App has a fast responsive time, responds to the delay a user may not notice or complains about.

- ☞ The App executes without consuming much battery and doesn't require much memory for its operation.

- ☞ The App doesn't require an internet connection for its normal operation except to open the **YOUTUBE** link.

- ☞ The App requires a microscopic image of a single RBC properly cropped (manually) by a professional before it detects it as either normal or infected. This might also mean focusing on a single RBC when viewing the blood sample on the microscope with out the need for cropping.

- ☞ The App can predict malaria infection from microscopic images saved as any image format and with variable image dimension.

CHAPTER 4

RESULTS AND DISCUSSION

These days, mobile phones are being possessed almost by everyone and are used in day to day activities. Most people use these powerful devices not only for communication but also for many other purposes due to their being available at an affordable cost [34]. In the current project, an android app that can give tips about malaria and automatically detect if a RBC was infected with malaria parasite or not is developed. In the course of the automatic malaria detection model development, the proposed supervised ML classifiers and the CNN based DL model were tested for their efficacy in correctly differentiating between normal and infected RBCs using useful quantitative metrics discussed in the previous chapter. A prototype of android operated smartphone application has been developed using the DL model. The development of such system has been undertaken assuming an alleviation of the challenges faced in the proper and timely diagnosis of the parasite in malaria endemic regions.

4.1 Machine Learning Results

As it can be recalled from sections 3.1.2 and 3.2 of this project document, targeting better performance of the selected ML classifiers, prior to classification, image preprocessing and feature selection steps were performed. Then the generated features were inputted to the different classifiers and the outcomes are indicated below.

4.1.1 Results of Random Forest Classifier

Table 4.1 and Fig. 4.1 show the prediction report and confusion matrix of the random forest classifier during model testing after successful training on the training image dataset.

Table 4.1: Prediction Report of Random Forest

	Precision	**Recall**	**F1-score**	**Support**
Uninfected	0.96	0.97	0.96	2756
Parasitized	0.97	0.96	0.96	2756
Accuracy			0.96	5512
Macro Average	0.96	0.96	0.96	5512
Weighted Average	0.96	0.96	0.96	5512

From Fig.4.1, we can see that TP is **2632**, TN is **2672**, FN is **124** and FP is **84**. Accordingly, the *precision, recall, f1-score* can be computed for the infected class and the overall accuracy.

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{2632}{2632 + 84} = \frac{2632}{2716} = 0.97$$

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{2632}{2632 + 124} = \frac{2632}{2756} = 0.96$$

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * 0.97 * 0.96}{0.97 + 0.96} = \frac{1.8624}{1.93} = 0.96$$

$$Accuracy = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}} = \frac{2632 + 2672}{2632 + 2672 + 124 + 84} = \frac{5304}{5512} = 0.96$$

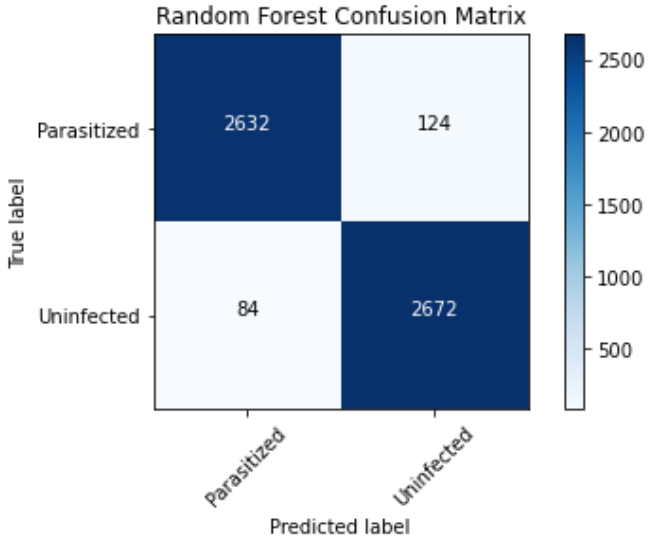

Figure 4.1: Confusion Matrix for Random Forest Model.

Table 4.1 also presents the **Macro Average** and **Weighted Average** of the classifier's **Precision, Recall and F1-score** parameters. As we have a perfectly balanced infected and normal samples, the two are

computed to be equal in our case. The RF classifier resulted in 96% precision, 96% recall and 96% F1-score. The MCC of this classifier was also computed using equation 3.6 and that resulted to be 0.92.

4.1.2 Results of K-Nearest Neighbor Classifier

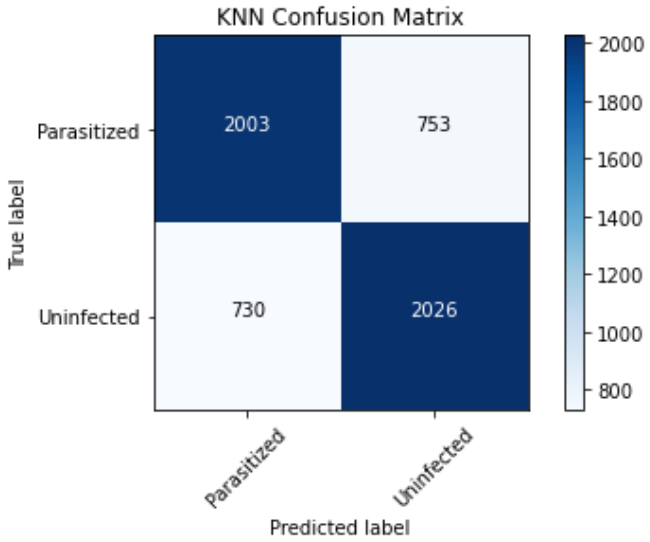

Figure 4.2: Confusion Matrix for KNN Model.

The quantitative metrics computed to show the perfiormance of the KNN are presented in Table 4.2. Accordingly, the classifier resulted in 73% precision, 73% recall and 73% F1-score.

Table 4.2: Prediction Report of KNN.

	Precision	**Recall**	**F1-score**	**Support**
Uninfected	0.73	0.74	0.73	2756
Parasitized	0.73	0.73	0.73	2756
Accuracy			0.73	5512
Macro Average	0.73	0.73	0.73	5512
Weighted Average	0.73	0.73	0.73	5512

4.1.3 Results of Support Vector Machine Classifier

As indicated in Table 4.3, the SVM resulted in 72% average precision, 71% Recall, and 71% F1-score.

Table 4.3: Prediction Report of SVM.

	Precision	**Recall**	**F1-score**	**Support**
Uninfected	0.69	0.78	0.73	2756
Parasitized	0.75	0.64	0.69	2756
Accuracy			0.71	5512
Macro Average	0.72	0.71	0.71	5512
Weighted Average	0.72	0.71	0.71	5512

The confusion matrix presented in Fig. 4.3 for the SVM classifier indicates that it has correctly classified 3,929 samples which constitutes 1,777 parasitized and 2,152 uninfected samples. The model misclassified 604 uninfected samples and 979 parasitized samples resulting in 21.92% FPR and 64% TPR.

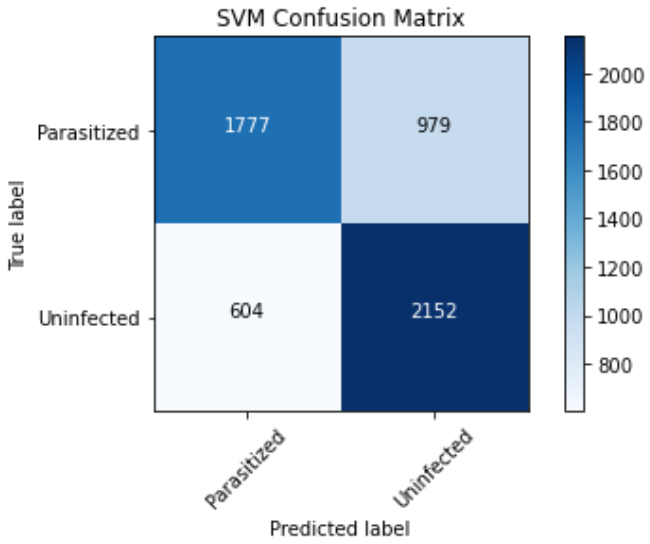

Figure 4.3: Confusion Matrix for SVM Model.

4.1.4 Results of Decision Tree Classifier

Table 4.4: Prediction Report of Decision Tree.

	Precision	**Recall**	**F1-score**	**Support**
Uninfected	0.94	0.93	0.94	2756
Parasitized	0.93	0.94	0.94	2756
Accuracy			0.94	5512
Macro Average	0.94	0.94	0.94	5512
Weighted Average	0.94	0.94	0.94	5512

As indicated in Table 4.4, the performance metrics computed for the DT show that it has gained weighted average of 94% Precision, 94% Recall, and 94% F1-score. The corresponding MCC was calculated to be 0.87.

4.1.5 Comparison of Machine Learning Classifiers

Fig.4.5 summarizes the comparison between the four ML classifiers considred in the current project. As can be seen from the figure, RF has shown best performance among its sibling classifiers for the dataset considered for training and testing. DT comes second in performance while SVM and KNN have shown relatively inferior performance given the dataset considered in the current project. Since RF is an ensemble of several DTs, it has shown a better performance than its igredient DT.

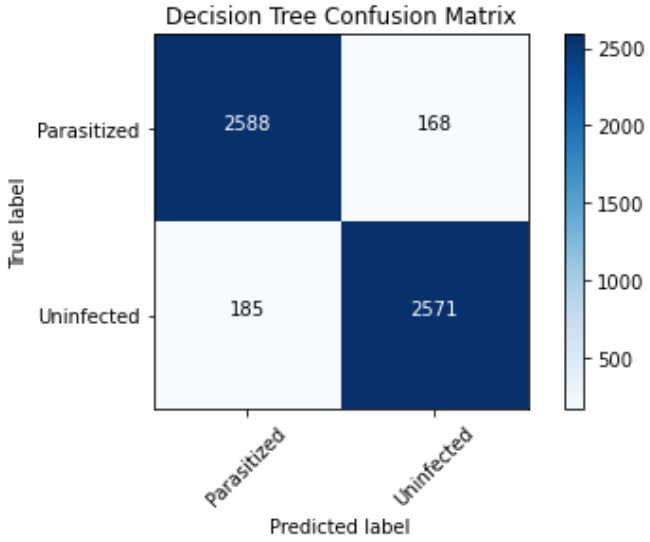

Figure 4.4: Confusion Matrix for Decision Tree Model.

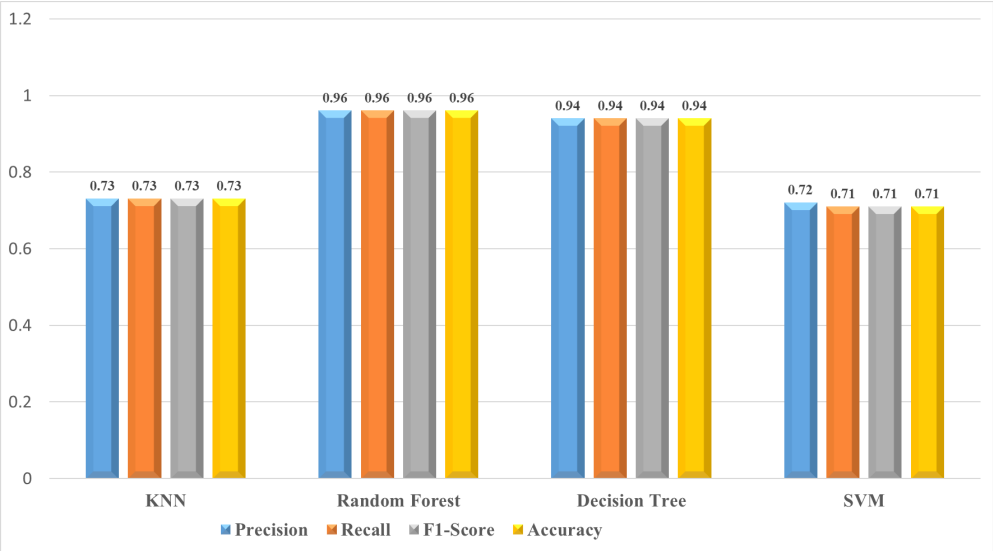

Figure 4.5: Comparison of ML Classifier.

4.2 Deep Learning Results

This section discusses the training and testing performances of the proposed CNN based DL classifier. CNN is a widely used tool in image processing due to its excellent ability to perform image classification through edge detection and pattern recognition [43].

There are several CNN variants of DL models as can be observed in different studies with applications in various datasets. The current project considered the custom CNN) model. The reason behind is that these variants of CNN have downside effects to be applicable in android app development projects due to their large size and complexity in addition to the requirements of extra feature extraction tasks and longer training time, which hinders them from being applicable in battery operated mobile devices [38]. The custom CNN utilized in the current project work and integrated to the mobile app has got a performance

accuracy which is comparable to the acclaimed variants of CNN for the datasets under consideration. The experiment was conducted with free T4 GPU Google Colab processor. The data partitioning (into training, validation an testing) method used in the case of the DL scheme was exactly the same as the scheme used in the case of ML.

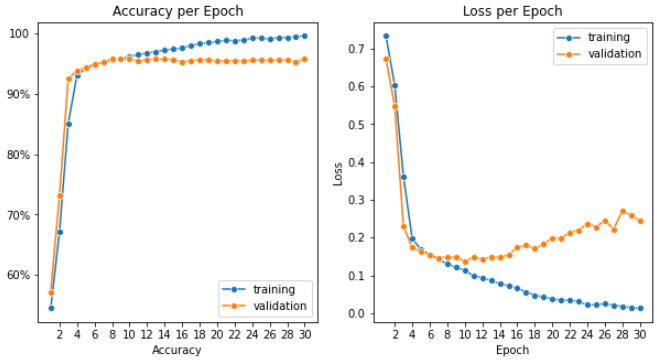

Figure 4.6: Accuracy and Loss graphs for the CNN model during Training and Validation.

Fig.4.6 presents the accuracy and loss graphs of the CNN scheme during both training as well as validation. The accuracy level is determined for different epochs. From the figure, we could see that as the number of epochs increases, training and validation accuracies improve, until they finally saturate after a certain number of epochs. Optimum number of epochs will be determined by trial and error, until the training and validation accuracies show no further improvements. The accuracies were also checked with various batch sizes. The determination of batch-size has to take into account the processing power (RAM) of the computer being used for training. In the current project, 30 epochs with 64 batch-size were finally used.

Fig.4.7 presents a bar graph showing the training accuracy, validation accuracy, training loss, and validation loss together for different epochs. Again, the bar plot also shows that the training accuracy increases as the number of epochs increase while the training loss decreases. Table 4.5 presents the prediction report computed (duirng model final testing) for the CNN model in terms of Precision, Recall, F1-score

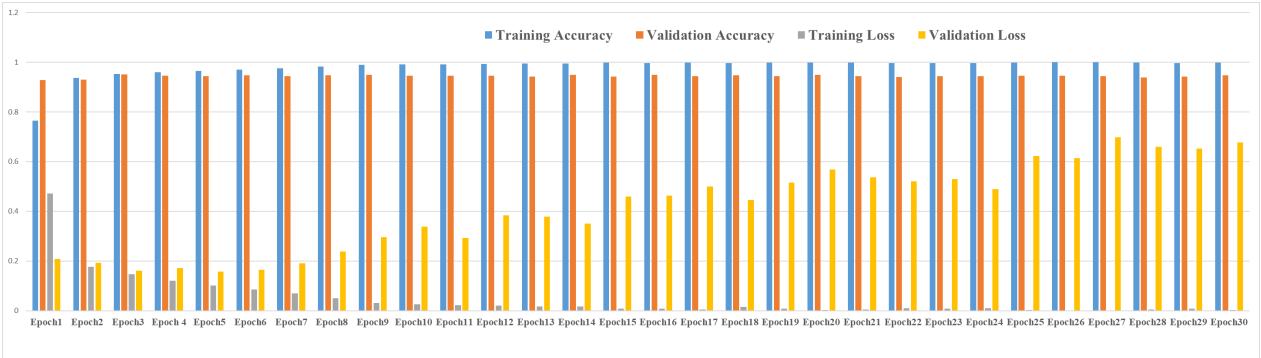

Figure 4.7: Training Performance for Different Epochs.

Figure 4.8: User Interface.

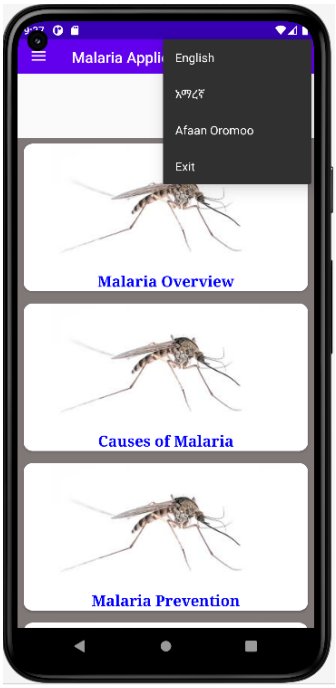

Figure 4.9: Language Switching Options.

and overall accuracy. Accordingly, the model was able to achieve 96% Accuracy, 96% Precision, 96% Recall and 96% F1-Score at epoch 30 with batch size 64.

Table 4.5: Classification Report of the CNN Performance.

	Precision	**Recall**	**F1-Score**	**Support**
Uninfected	0.96	0.95	0.96	2759
Parasitized	0.95	0.97	0.96	2753
Accuracy			0.96	5512
Macro Average	0.96	0.96	0.96	5512
Weighted Average	0.96	0.96	0.96	5512

4.3 The Application Contents and User Interfaces

The proposed malaria App was developed using the successfully trained and tested DL model. Though the observed accuracies of the ML and DL algorithms developed in this project both offered comparable results, however, the malaraia App was developed utilizaing only the DL scheme. The very reason was its convenience during implementation on an Android studio environment. The App was developed to be interactive, easy to use and vivid User Interface (UI). A snapshot of the front page seen on the App is depicted on Fig.4.8.

A user can select one of the language options displayed on the UI that appears when he/she opens the application from any android operated smart phone. The user also has the option to switch between

different languages (Amharic, Afaan Oromoo, English). For instance, let's say that some one has clicked **ENGLISH** from the language list. Then that click action redirects the user to different malaria related information option written in English. From there, the user can also use the left side menu, an option that is hidden until one can swipe it to the right or click on a menu icon which is found on the upper left corner. The user can also click on an option icon, a vertical ellipses dots, to switch from one language to another. Figure 4.9 shows a UI with a side menu icon and language selection options. The image was taken after the vertical ellipses icon is clicked. In the selected language options, we see an *exit* option that helps to exit from the current page. From the right swiped option, a user can select any of the listed topics to read about malaria. On that page, there are also **Home** and **Exit** options. The **Home** option takes us back to the plain view of the UI and the **Exit** option closes the application altogether.

Such options and interaction are also available in all the other language selections. In order to bring back the side options display page to its home position, the user has to either click a *Lock* icon found on the upper center of the display or swipe the page to the left. To bring back the language options, displayed as a result of clicking the vertical ellipses, to the home position, the user has to click outside of the drop down page if he/she has no intention of changing the current language. Clicking on the same language the user currently using is an other way of exiting from the drop down page.

From the UI image, shown in Fig.4.8, we have an option to make a malaria diagnosis using the developed DL model, displayed as **MALARIA DETECTION USING DEEP LEARNING** button. RBC images to be diagnosed will be either captured using the mobile phone camera or browsed from storage gallery. The cell image to be diagnosed will be displayed in the image view of the android app and the result is displayed instantly.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The current project has dealt with building an Android App after developing a malaria classifier model based on a CNN scheme. In the process, we have explored different machine learning models which are popular in applications including object detection and classification purposes. The DL model has shown commendable performance in differentiating malaria infected and normal RBCs. Useful quantitative measures have been used to test the performance. Overall, the model achieved 100% accuracy during training and 96% during testing stage when applied the image datasets considered in the current project. The latest labeled dataset available on Kaggle has been used for the model development and testing [34]. The developed Android App is not only capable of automatically detecting malaria infected RBCs from microscopic images but it also gives information about malaria in three popularly spoken and well understood languages in Ethiopia, in order to create awareness so that potential vulnerable people can take certain measures against the deadly malaria disease. The app is hoped to be a user friendly and easy to switch from one user interface to another, both in content and language options with a simple click.

5.2 Recommendations

This project will be more applicable if the limitations stated in section 1.5 and the following recommendations are considered in the future work.

- ☞ This project considered detecting a single RBC found on a microscopic slide. That requires a third person either to focus on a single RBC when imaging or do a segmentation of a single RBC identified on the whole slide. In that sense the proposed detection scheme in the current project could be assumed semi-automatic. For the model (hence the App) be more applicable in a clinical setting, considering the whole slide may be essential. That may require the model to be modified in such a way that it could automatically segment all RBCs identified on a given slide and number them. The model can then identify infected and normal RBCs automatically. Most previous literature used single RBC microscopic images and the current project followed a similar route.

- ☞ Comparing the proposed CNN based DL scheme with other deep learning models available in the literature might be important to demonstrate more its efficiency. In the current project, a single DL model and several ML models have been considered.

- ☞ The information about malaria that the App provides is to be updated regularly.

REFERENCES

[1] https://www.mathworks.com/discovery/support-vector-machine.html, 2023. Accessed 02 Oct,2023.

[2] https://www.upgrad.com/blog/basic-cnn-architecture/, 2023. Accessed 09 Oct,2023.

[3] W. H. Organization, *World malaria report 2022*. World Health Organization, 2022.

[4] W. H. Organization, "WHO guideline for malaria, 13 July 2021," tech. rep., World Health Organization, 2021.

[5] W. H. Organization, "The potential impact of health service disruptions on the burden of malaria: a modelling analysis for countries in sub-saharan africa," 2020.

[6] G. Bugssa and K. Tedla, "Feasibility of malaria elimination in ethiopia," *Ethiopian Journal of Health Sciences*, vol. 30, no. 4, 2020.

[7] F. A. Kendie, T. Hailegebriel W/kiros, E. Nibret Semegn, and M. W. Ferede, "Prevalence of malaria among adults in ethiopia: a systematic review and meta-analysis," *Journal of tropical medicine*, vol. 2021, pp. 1–9, 2021.

[8] Z. Babure, Y. Ahmed, S. Likasa, F. Jiru, T. Weldemarium, and M. Fite, "Trend analysis of malaria prevalence in east wollega zone, oromia regional state, western ethiopia, 2020: A retrospective study," *J Women's Health Care*, vol. 10, no. 515, pp. 2167–0420, 2021.

[9] T. Adhanom, W. Deressa, K. Witten, A. Getachew, and T. Seboxa, "Malaria. in the epidemiology and ecology of health and disease in ethiopia," *Addis Ababa: Shama Books*, 2006.

[10] F. Chen, B. R. Flaherty, C. E. Cohen, D. S. Peterson, and Y. Zhao, "Direct detection of malaria infected red blood cells by surface enhanced raman spectroscopy," *Nanomedicine: Nanotechnology, Biology and Medicine*, vol. 12, no. 6, pp. 1445–1451, 2016.

[11] W. Erhun, E. Agbani, and S. Adesanya, "Malaria prevention: knowledge, attitude and practice in a southwestern nigerian community," *African Journal of Biomedical Research*, vol. 8, no. 1, pp. 25–29, 2005.

[12] A. Ababa, "National malaria elimination roadmap," 2017.

[13] https://www.bing.com/search?q=Health+information+systemsgs. Accessed 05 Nov,2023.

[14] J. C. Moses, S. Adibi, S. M. Shariful Islam, N. Wickramasinghe, and L. Nguyen, "Application of smartphone technologies in disease monitoring: a systematic review," in *Healthcare*, vol. 9, p. 889, MDPI, 2021.

[15] Y. H. Kwan, W. J. Ong, M. Xiong, Y. Y. Leung, J. K. Phang, C. T. M. Wang, and W. Fong, "Evaluation of mobile apps targeted at patients with spondyloarthritis for disease monitoring: systematic app search," *JMIR mHealth and uHealth*, vol. 7, no. 10, p. e14753, 2019.

[16] https://www.measureevaluation.org/our-work/malaria.html. Accessed 05 Nov,2023.

[17] H. K. Heggenhougen, V. Hackethal, P. Vivek, *et al.*, *The behavioural and social aspects of malaria and its control: an introduction and annotated biliography*. 2003.

[18] I. of Medicine (US). Committee on the Economics of Antimalarial Drugs, *Saving Lives, Buying Time: Economics of Malaria Drugs in an Age of Drug Resistance*. National Academies Press, 2004.

[19] R. Neghina, I. Iacobiciu, A. M. Neghina, and I. Marincu, "Malaria, a journey in time: in search of the lost myths and forgotten stories," *The American journal of the medical sciences*, vol. 340, no. 6, pp. 492–498, 2010.

[20] L. Feezer, "Theories concerning the causation of disease," *American journal of public health*, vol. 11, no. 10, pp. 908–912, 1921.

[21] S. Tang, L. Ji, T. Hu, R. Wang, H. Fu, T. Shao, C. Liu, P. Shao, Z. He, G. Li, *et al.*, "Public awareness of malaria in the middle stage of national malaria elimination programme. a cross-sectional survey in rural areas of malaria-endemic counties, china," *Malaria Journal*, vol. 15, pp. 1–8, 2016.

[22] K. R. Awasthi, J. Jancey, A. C. A. Clements, and J. E. Leavy, "Community engagement approaches for malaria prevention, control and elimination: a scoping review protocol," *BMJ Open*, vol. 11, no. 10, 2021.

[23] https://www.ibm.com/topics/mobile-application-development. Accessed 03 Nov,2023.

[24] S. C. Redd, S. Luby, A. Hightower, P. Kazembe, O. Nwanyanwu, C. Ziba, L. Chitsulo, C. Franco, M. Olivar, and J. Wirima, "Clinical algorithm for treatment of plasmodium falciparum malaria in children," *The Lancet*, vol. 347, no. 8996, pp. 223–227, 1996.

[25] N. Tangpukdee, C. Duangdee, P. Wilairatana, and S. Krudsood, "Malaria diagnosis: a brief review," *The Korean journal of parasitology*, vol. 47, no. 2, p. 93, 2009.

[26] A. Molina, J. Rodellar, L. Boldú, A. Acevedo, S. Alférez, and A. Merino, "Automatic identification of malaria and other red blood cell inclusions using convolutional neural networks," *Computers in Biology and Medicine*, vol. 136, p. 104680, 2021.

[27] J. BEKTAŞ, "Comparison of cnns and svm for detection of activation in malaria cell images," *Natural and Applied Sciences Journal*, vol. 2, no. 2, pp. 38–50, 2019.

[28] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, no. 1, pp. 381–386, 2020.

[29] A. Maqsood, M. S. Farid, M. H. Khan, and M. Grzegorzek, "Deep malaria parasite detection in thin blood smear microscopic images," *Applied Sciences*, vol. 11, no. 5, p. 2284, 2021.

[30] M. Habibzadeh, *Automatic Segmentation and Classification of Red and White Blood cells in Thin Blood Smear Slides*. PhD thesis, Concordia University, 2015.

[31] B. Charbuty and A. Abdulazeez, "Classification based on decision tree algorithm for machine learning," *Journal of Applied Science and Technology Trends*, vol. 2, no. 01, pp. 20–28, 2021.

[32] N. M. Abdulkareem and A. M. Abdulazeez, "Machine learning classification based on radom forest algorithm: A review," *International journal of science and business*, vol. 5, no. 2, pp. 128–142, 2021.

[33] https://datagen.tech/guides/image-classification/image-classification-using-cnn. Accessed 17 Nov,2023.

[34] K. Fuhad, J. F. Tuba, M. R. A. Sarker, S. Momen, N. Mohammed, and T. Rahman, "Deep learning based automatic malaria parasite detection from blood smear and its smartphone based application," *Diagnostics*, vol. 10, no. 5, p. 329, 2020.

[35] C. R. Maturana, A. D. de Oliveira, S. Nadal, B. Bilalli, F. Z. Serrat, M. E. Soley, E. S. Igual, M. Bosch, A. V. Lluch, A. Abelló, *et al.*, "Advances and challenges in automated malaria diagnosis using digital microscopy imaging with artificial intelligence tools: A review," *Frontiers in microbiology*, vol. 13, p. 1006659, 2022.

[36] H. Yu, F. Yang, S. Rajaraman, I. Ersoy, G. Moallem, M. Poostchi, K. Palaniappan, S. Antani, R. J. Maude, and S. Jaeger, "Malaria screener: a smartphone application for automated malaria screening," *BMC Infectious Diseases*, vol. 20, no. 1, pp. 1–8, 2020.

[37] C. Affonso, A. L. D. Rossi, F. H. A. Vieira, A. C. P. de Leon Ferreira, *et al.*, "Deep learning for biological image classification," *Expert systems with applications*, vol. 85, pp. 114–122, 2017.

[38] M. Masud, H. Alhumyani, S. S. Alshamrani, O. Cheikhrouhou, S. Ibrahim, G. Muhammad, M. S. Hossain, and M. Shorfuzzaman, "Leveraging deep learning techniques for malaria parasite detection using mobile application," *Wireless Communications and Mobile Computing*, vol. 2020, pp. 1–15, 2020.

[39] M. Sonka, V. Hlavac, R. Boyle, M. Sonka, V. Hlavac, and R. Boyle, "Image pre-processing," *Image processing, analysis and machine vision*, pp. 56–111, 1993.

[40] C. T. Leondes, *Multidimensional Systems: Signal Processing and Modeling Techniques: Advances in Theory and Applications*. Elsevier, 1995.

[41] B. G. Gezehegn, "Automatic malaria detection using machine learning approaches," Master's thesis, Addis Ababa University, 2020.

[42] https://medium.com/@maxmarkovvision/optimal-number-of-bins-for-histograms-3d7c48086fde. Accessed 19 Nov, 2023.

[43] H. S. Abdullahi, R. Sheriff, and F. Mahieddine, "Convolution neural network in precision agriculture for plant image recognition and classification," in *2017 Seventh International Conference on Innovative Computing Technology (INTECH)*, vol. 10, pp. 256–272, Ieee New York, 2017.

APPENDIX A

XML code for the first UI

```
    <?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:id="@+id/drawerlayout"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">
    <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <androidx.appcompat.widget.Toolbar
        android:layout_width="match_parent"
        android:layout_height="45dp"
        android:id="@+id/toolbar"
```

```
app:titleTextColor="#FFFFFF"
app:title="Malaria Application"
android:textAlignment="center"
android:gravity="center"
android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
app:titleMargin="10dp"
android:layout_marginBottom="5dp"
tools:ignore="MissingConstraints"
android:background="@color/colorPrimary"/>
<ScrollView
android:layout_width="match_parent"
android:layout_height="match_parent">
<androidx.constraintlayout.widget.ConstraintLayout
android:layout_width="match_parent"
android:layout_height="match_parent">
<LinearLayout
 android:layout_width="match_parent"
android:layout_height="match_parent">
<GridLayout
 android:layout_marginTop="80sp"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:background="#807777"
 android:rowCount="4"
 android:columnCount="1">
 <androidx.cardview.widget.CardView
 android:id="@+id/card1e"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_row="0"
 android:layout_column="0"
 android:layout_gravity="fill"
```

```
 android:layout_rowWeight="1"
 android:layout_columnWeight="1"
 android:layout_margin="8dp"
app:cardCornerRadius="12dp"
android:elevation="8dp">
<RelativeLayout
android:layout_width="match_parent"
 android:layout_height="match_parent"
android:orientation="vertical"
android:layout_gravity="center_vertical|center_horizontal">
<ImageView
android:id="@+id/image11"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:src="@drawable/mosquito"/>
<TextView
android:layout_below="@id/image11"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Malaria Overview"
android:textAlignment="center"
android:textColor="#0000FF"
android:textSize="20dp"
android:fontFamily="serif"
android:textStyle="bold"
android:id="@+id/textView_1"/>
</RelativeLayout>
</androidx.cardview.widget.CardView>
<androidx.cardview.widget.CardView
android:id="@+id/card2e"
 android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

```
 android:layout_row="1"
 android:layout_column="0"
android:layout_gravity="fill"
android:layout_rowWeight="1"
android:layout_columnWeight="1"
android:layout_margin="8dp"
app:cardCornerRadius="12dp"
android:elevation="8dp">
<RelativeLayout
android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 android:layout_gravity="center_vertical|center_horizontal">
 <ImageView
 android:id="@+id/image12"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:src="@drawable/mosquito"/>
 <TextView
  android:id="@+id/textView_2"
android:layout_width="match_parent"
android:layout_height="wrap_content"
 android:layout_below="@id/image12"
android:text="Causes of Malaria"
android:textAlignment="center"
android:textSize="20dp"
 android:fontFamily="serif"
android:textColor="#0000FF"
android:textStyle="bold"/>
 </RelativeLayout>
 </androidx.cardview.widget.CardView>
 <androidx.cardview.widget.CardView
```

```
android:id="@+id/card3e"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_row="2"
android:layout_column="0"
android:layout_gravity="fill"
android:layout_rowWeight="1"
android:layout_columnWeight="1"
 android:layout_margin="8dp"
 app:cardCornerRadius="12dp"
 android:elevation="8dp">
<RelativeLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:layout_gravity="center_vertical|center_horizontal">
<ImageView
 android:id="@+id/image13"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:src="@drawable/mosquito"/>
<TextView
android:layout_below="@id/image13"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Malaria Prevention"
android:textAlignment="center"
android:textColor="#0000FF"
android:textSize="20dp"
android:fontFamily="serif"
android:textStyle="bold"
android:id="@+id/textView_3"/>
```

```
</RelativeLayout>
</androidx.cardview.widget.CardView>
 <androidx.cardview.widget.CardView
 android:id="@+id/card4e"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_row="3"
 android:layout_column="0"
 android:layout_gravity="fill"
 android:layout_rowWeight="1"
 android:layout_columnWeight="1"
 android:layout_margin="8dp"
 app:cardCornerRadius="12dp"
 android:elevation="8dp">
 <RelativeLayout
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
android:layout_gravity="center_vertical|center_horizontal">
<ImageView
 android:id="@+id/image14"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:src="@drawable/mosquito"/>
 <TextView
 android:id="@+id/textView_4"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_below="@id/image14"
android:text="Malaria Diagnosis"
android:textAlignment="center"
android:textSize="20dp"
```

```
        android:fontFamily="serif"
         android:textColor="#0000FF"
        android:textStyle="bold"/>
        </RelativeLayout>
         </androidx.cardview.widget.CardView>
        </GridLayout>
       </LinearLayout>
            </androidx.constraintlayout.widget.ConstraintLayout>
        </ScrollView>
    </LinearLayout>
    <RelativeLayout
        android:layout_width="300dp"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:background="@color/white"
        tools:ignore="MissingConstraints">
        <include
            layout="@layout/main_nav_drawer"/>
    </RelativeLayout>
  </androidx.drawerlayout.widget.DrawerLayout>
```

Java code for the first UI

```
    package com.example.malariaapp;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.cardview.widget.CardView;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;

import android.app.Activity;
```

```
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;

import com.google.android.material.navigation.NavigationView;
public class MainActivity extends AppCompatActivity {
    DrawerLayout drawerLayout;
    NavigationView navigationView;
    Toolbar toolbar;
    CardView cardView1e,cardView2e,cardView3e,cardView4e;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        toolbar=findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        cardView1e=findViewById(R.id.card1e);
        cardView1e.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
        Intent intent=new Intent(MainActivity.this,MalariaOverView.class);
                startActivity(intent);
            }
        });
        cardView2e=findViewById(R.id.card2e);
        cardView2e.setOnClickListener(new View.OnClickListener() {
        @Override
```

```
        public void onClick(View view) {
         Intent intent=new Intent(MainActivity.this,Causes.class);
                startActivity(intent);
            }
        });
        cardView3e=findViewById(R.id.card3e);
        cardView3e.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent=new Intent(MainActivity.this,Prevention.class);
                startActivity(intent);
            }
        });
        cardView4e=findViewById(R.id.card4e);
        cardView4e.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
             Intent intent=new Intent(MainActivity.this,Diagnosis.class);
                startActivity(intent);
            }
        });
    drawerLayout=findViewById(R.id.drawerlayout);
    setSupportActionBar(toolbar);
ActionBarDrawerToggle toggle=new ActionBarDrawerToggle(this, ...
drawerLayout,toolbar,R.string.navigation_open,
R.string.navigation_close);
        drawerLayout.addDrawerListener(toggle);
        toggle.syncState();
    }
    public void ClickMenu(View view){openDrawer(drawerLayout);}
    public static void openDrawer(DrawerLayout drawerLayout) {
        drawerLayout.openDrawer(GravityCompat.START);
```

```
    }
    public void ClickLogo(View view){closeDrawer(drawerLayout);}
    public static void closeDrawer(DrawerLayout drawerLayout) {
        if (drawerLayout.isDrawerOpen(GravityCompat.START)){
            drawerLayout.closeDrawer(GravityCompat.START);
        }
    }
public void ClickHome(View view){recreate();}
public void ClickMalariaOverview(View view){redirectActivity(this, ...
MalariaOverView.class);}
public void ClickCausesOfMalaria(View view){redirectActivity(this, ...
Causes.class);}
public void ClickMalariaPrevention(View view){redirectActivity(this, ...
Prevention.class);}
public void ClickMalariaDiagnosis(View view){redirectActivity(this, ...
Diagnosis.class);}
public void ClickExit(View view){exit(this);}
 public static void exit(final Activity activity){
AlertDialog.Builder builder=new AlertDialog.Builder(activity);
 builder.setTitle("Exit");
 builder.setMessage("Are you sure you want to exit?");
 builder.setPositiveButton("YES", new DialogInterface.OnClickListener() {
 @Override
public void onClick(DialogInterface dialogInterface, int i) {
 activity.finishAffinity();
    System.exit(0);
            }
        });
    builder.setNegativeButton("NO", new DialogInterface.OnClickListener() {
  @Override
     public void onClick(DialogInterface dialogInterface, int i) {
   dialogInterface.dismiss();
```

```
            }
        });
        builder.show();
    }
    public static void redirectActivity(Activity activity,Class aClass){
        Intent intent=new Intent(activity,aClass);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        activity.startActivity(intent);
    }
    protected void onPause(){
        super.onPause();
        closeDrawer(drawerLayout);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu){
    MenuInflater inflater=getMenuInflater();
    inflater.inflate(R.menu.language_menu,menu);
return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item){
        int id1=item.getItemId();
        if(id1==R.id.exit){
            AlertDialog.Builder builder=new AlertDialog. ...
            Builder(MainActivity.this);
            builder.setMessage("Do you really want to Exit?");
            builder.setCancelable(true);

            builder.setNegativeButton("YES", new DialogInterface. ...
            OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, ...
```

```
            int which) {
                finish();
            }
        });
        builder.setPositiveButton("NO", new DialogInterface. ...
        OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, ...
            int which) {
                dialogInterface.cancel();
            }
        });
        AlertDialog alertDialog=builder.create();
        alertDialog.show();
    }
    int id= item.getItemId();
    Intent intent1=new Intent(Intent.ACTION_SEND);
    intent1.setType("text/plain");
    intent1.putExtra(Intent.EXTRA_SUBJECT,"checkout this cool ...
    Application");
    intent1.putExtra(Intent.EXTRA_TEXT,"");
    startActivity(Intent.createChooser(intent1,"Share Via"));
    if (id==R.id.language1){
        Intent intent=new Intent(MainActivity.this,MainActivity.class);
        startActivity(intent);
        return true;
    }
    else if(id==R.id.language2){
        Intent intent=new Intent(MainActivity.this,Amharic.class);
        startActivity(intent);
        return true;
    }
```

```
        else if (id==R.id.language3){
            Intent intent=new Intent(MainActivity.this, AfaanOromoo.class);
            startActivity(intent);
            return true;
        }
        return true;
    }
}
```

Similar code goes with the others with little modification

Appendix B

Deep Learning Code Used for the Project

```
    !pip install kaggle

    !mkdir .kaggle
!touch ~/.kaggle/kaggle.json

import json
token={"username":"berihunn","key":"6155f5579077954d5184b0cc74d789d0"}
with open('/root/.kaggle/kaggle.json','w') as file:
  json.dump(token,file)

  !cp '/content/.kaggle/kaggle.json' ~'/.kaggle/kaggle.json'
#Configure the file
!kaggle config set -n path -v{/content}

!chmod 600 ~/.kaggle/kaggle.json

!kaggle datasets list
```

```
!kaggle datasets download -d iarunava/cell-images-for-detecting-malaria...
-p /content

import os
import zipfile
local_zip='/content/cell-images-for-detecting-malaria.zip'
zip_ref=zipfile.ZipFile(local_zip,'r')
zip_ref.extractall('/content/cell-images-for-detecting-malaria')
zip_ref.close()

#directory with our training uninfected pictures
uninfected_dir=os.path.join('/content/cell-images-for- ...
detecting-malaria/cell_images/Uninfected')

#directory with our training infected or parasitized pictures
infected_dir=os.path.join('/content/cell-images-for- ...
detecting-malaria/cell_images/Parasitized')

print(len(os.listdir(infected_dir)))
print(len(os.listdir(uninfected_dir)))

try:
  os.mkdir('/content/cell-images-for-detecting-malaria/cell_images/training
  os.mkdir('/content/cell-images-for-detecting-malaria/cell_images/testing'
  os.mkdir('/content/cell-images-for-detecting-malaria/cell_images/validati

  os.mkdir('/content/cell-images-for-detecting-malaria/cell_images/...
  training/uninfected')
  os.mkdir('/content/cell-images-for-detecting-malaria/cell_images/...
  training/infected')
  os.mkdir('/content/cell-images-for-detecting-malaria/cell_images/...
```

```
  testing/uninfected')
  os.mkdir('/content/cell-images-for-detecting-malaria/cell_images/...
  testing/infected')
  os.mkdir('/content/cell-images-for-detecting-malaria/cell_images/...
  validation/uninfected')
  os.mkdir('/content/cell-images-for-detecting-malaria/cell_images/...
  validation/infected')
except OSError as e:
  print('error:',e)

  import random
from shutil import copyfile

def split_data(SOURCE,TRAINING,TESTING,VALIDATION,SPLIT_SIZE):
  files=[]
  for filename in os.listdir(SOURCE):
    file=SOURCE+filename
    if os.path.getsize(file)>0:
      files.append(filename)
    else:
      print(filename+" is zero length, so ignoring.")

  training_length=int(len(files)*SPLIT_SIZE)
  testing_length=int(len(files)-(training_length+training_length/7))
  validation_length=int(len(files)-(training_length+testing_length))

  shuffled_set=random.sample(files,len(files))
  training_set=shuffled_set[0:training_length]
  testing_set=shuffled_set[training_length:(training_length+testing_length)
  validation_set=shuffled_set[-validation_length:]

  for filename in training_set:
```

```
    this_file=SOURCE+filename
    destination=TRAINING+filename
    copyfile(this_file,destination)

  for filename in testing_set:
    this_file=SOURCE+filename
    destination=TESTING+filename
    copyfile(this_file,destination)

  for filename in validation_set:
    this_file=SOURCE+filename
    destination=VALIDATION+filename
    copyfile(this_file,destination)

INFECTED_SOURCE_DIR="/content/cell-images-for-detecting-malaria/...
cell_images/Parasitized/"
TRAINING_INFECTED_DIR="/content/cell-images-for-detecting-malaria/...
cell_images/training/infected/"
TESTING_INFECTED_DIR="/content/cell-images-for-detecting-malaria/...
cell_images/testing/infected/"
VALIDATION_INFECTED_DIR="/content/cell-images-for-detecting-malaria/...
cell_images/validation/infected/"

UNINFECTED_SOURCE_DIR="/content/cell-images-for-detecting-malaria/...
cell_images/Uninfected/"
TRAINING_UNINFECTED_DIR="/content/cell-images-for-detecting-malaria/...
cell_images/training/uninfected/"
TESTING_UNINFECTED_DIR="/content/cell-images-for-detecting-malaria/...
cell_images/testing/uninfected/"
VALIDATION_UNINFECTED_DIR="/content/cell-images-for-detecting-malaria/...
cell_images/validation/uninfected/"
```

```
split_size=0.7
split_data(INFECTED_SOURCE_DIR,TRAINING_INFECTED_DIR,...
TESTING_INFECTED_DIR,...
VALIDATION_INFECTED_DIR,split_size)
split_data(UNINFECTED_SOURCE_DIR,TRAINING_UNINFECTED_DIR,...
TESTING_UNINFECTED_DIR,VALIDATION_UNINFECTED_DIR,split_size)

print(len(os.listdir(TRAINING_INFECTED_DIR))+ ...
len(os.listdir(TRAINING_UNINFECTED_DIR)))

print(len(os.listdir(TESTING_INFECTED_DIR))+ ...
len(os.listdir(TESTING_UNINFECTED_DIR)))

print(len(os.listdir(VALIDATION_INFECTED_DIR))+ ...
len(os.listdir(VALIDATION_UNINFECTED_DIR)))

import tensorflow as tf
import matplotlib.pyplot as plt

img_height,img_width=112,112
batch_size=64

train_ds=tf.keras.utils.image_dataset_from_directory(
  "cell-images-for-detecting-malaria/cell_images/training",
  image_size=(img_height,img_width),
  batch_size=batch_size
)

val_ds=tf.keras.utils.image_dataset_from_directory(
  "cell-images-for-detecting-malaria/cell_images/validation",
  image_size=(img_height,img_width),
  batch_size=batch_size
```

```
)

test_ds=tf.keras.utils.image_dataset_from_directory(
  "cell-images-for-detecting-malaria/cell_images/testing",
  image_size=(img_height,img_width),
  batch_size=batch_size
)

class_names=["infected","uninfected"]
plt.figure(figsize=(10,10))
for images,labels in train_ds.take(1):
  for i in range(6):
    ax=plt.subplot(2,3,i+1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")

   import tensorflow as tf
tf.keras.layers.Activation
#from keras.layers import activation
model=tf.keras.Sequential([
    tf.keras.layers.Rescaling(1./255),
    tf.keras.layers.Conv2D(32,3,activation="relu"),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.Conv2D(64,3,activation="relu"),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.Conv2D(128,3,activation="relu"),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256,activation="relu"),
    tf.keras.layers.Dense(2)
])
```

```
model.compile(
    optimizer="adam",
    loss=tf.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy']
)
model.fit(train_ds,
          validation_data=val_ds,
          epochs=10
          )
          model.evaluate(test_ds)

          import numpy
plt.figure(figsize=(10,10))
for images, labels in test_ds.take(1):
  classifications=model(images)
  #print(classifications)

  for i in range(6):
      ax=plt.subplot(2,3,i+1)
      plt.imshow(images[i].numpy().astype("uint8"))
      index=numpy.argmax(classifications[i])
      plt.title("pred: "+class_names[index]+ ...
      " | Real:"+class_names[labels[i]])

converter=tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model=converter.convert()
with open("model.tflite",'wb') as f:
  f.write(tflite_model)
```

Appendix C

Machine Learning Code

```
    #import the necessary packages
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from PIL import Image
from imutils import paths
import numpy as np
import os
import random
import cv2
import mahotas as mt
import matplotlib.pyplot as plt
import itertools
from sklearn.tree import export_graphviz
```

In the above all the necessary packages and models were imported into the Jupyter IDE.

```
    def fd_histogram(image,mask=None):
    #convert the image to HSV color-space
    image=cv2.cvtColor(image,cv2.COLOR_BGR2HSV)
    h,s,v=cv2.split(image)
    h_fil=cv2.medianBlur(h,3)
```

```
    s_fil=cv2.medianBlur(s,3)
    v_fil=cv2.medianBlur(v,3)
    hsv_merge=cv2.merge([h_fil,s_fil,v_fil])
    #compute the color histogram
    hist=cv2.calcHist([hsv_merge],[0,1,2],None,[bins, bins, bins],...
    [0,256,0,256,0,256])
    #normalize the histogram
    cv2.normalize(hist,hist)
    #return the histogram
    return hist.flatten()

    #grab all image paths in the input dataset directory, then initialize
#our list of images and corresponding class labels
print('[INFO] loading images...')
imagePaths=sorted(list(paths.list_images("dataset")))
#dataset is the folder name that contains images
random.seed(42)
random.shuffle(imagePaths)
global_feature=[]
data=[]
labels=[]
IMAGE_DIMS=(130,130,1)
bins=8

    #loop over the input images
for imagePath in imagePaths:
    image=cv2.imread(imagePath)
    image=cv2.resize(image,(IMAGE_DIMS[1],IMAGE_DIMS[0]))
    fv_histogram=fd_histogram(image)

    data.append(fv_histogram)
    l=label=imagePath.split(os.path.sep)[-2]
    labels.append(l)
```

```
    #encode the labels,converting them from strings to integers
le=LabelEncoder()
labelsl=le.fit_transform(labels)

    models={
    "knn":KNeighborsClassifier(n_neighbors=1),
    "naive_bayes":GaussianNB(),
    "logit":LogisticRegression(solver='lbfgs',multi_class="auto"),
    "svm":SVC(kernel="poly",degree=2),
    "decision_tree":DecisionTreeClassifier(),
    "random_forest":RandomForestClassifier(n_estimators=100),
    "mlp":MLPClassifier()
}

    #partition the data into training and testing splits, using 87.5% of...
    the data for training and the remaining 12.5% for testing
print("[INFO] constructing training/testing split...")
(trainData,testData,trainLabels,testLabels)=train_test_split(data,...
labels,test_size=0.125,random_state=0)

    #train the model
print("[INFO] using '{}' model".format("decision_tree"))
model=models["decision_tree"]
model.fit(trainData,trainLabels)

    #make predictions on our data and show a classification report
print("[INFO] evaluating..." )
predictions=model.predict(testData)
print(classification_report(testLabels,predictions))

    #grap all image paths in the input dataset directory,...
    #then initialize our list of images and corresponding class labels
print('[INFO] loading images...')
imagePaths=sorted(list(paths.list_images("testdata")))
```

```
random.seed(0)
random.shuffle(imagePaths)
datatest=[]
labelstest=[]
IMAGE_DIMS=(130,130,1)

    #loop over the input images
for imagePath in imagePaths:
    #load the image, pre_process it, and store it in the data list
    image=cv2.imread(imagePath)
    image=cv2.resize(image,(IMAGE_DIMS[1],IMAGE_DIMS[0]))
    #image=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    features=fd_histogram(image)
    datatest.append(features)
    labell=imagePath.split(os.path.sep)[-2]
    labelstest.append(labell)

    #encode the labels,converting them from strings to integers
lb=LabelEncoder()
labelsll=lb.fit_transform(labelstest)

    #make predictions on our data and show a classification report
print("[INFO] evaluating...")
predictionstest=model.predict(datatest)
print(classification_report(labelstest,predictionstest))

    def plot_confusion_matrix(cm,classes,normalize=False,...
    title='confusion matrix',cmap=plt.cm.Blues):
    plt.imshow(cm,interpolation='nearest',cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks=np.arange(len(classes))
    plt.xticks(tick_marks,classes,rotation=45)
    plt.yticks(tick_marks,classes)
```

```
    if normalize:
        cm=cm.astype('float')/cm.sum(axis=1)[:,np.newaxis]
        print('Normalized confusion matrix')
    else:
        print('confusion matrix, without normalization')
    print(cm)

    thresh=cm.max()/2.
    for i,j in itertools.product(range(cm.shape[0]),range(cm.shape[1])):
        plt.text(j,i,cm[i,j],
                 horizontalalignment="center",
                 color="white" if cm[i,j]>thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

    cnf_matrix=confusion_matrix(labelstest,predictionstest)
np.set_printoptions(precision=2)
plt.figure()
plot_confusion_matrix(cnf_matrix,classes=lb.classes_,...
title='Decision Tree Confusion Matrix')
plt.show()
```

This code snippet illustrate how ML algorithm was trained for a Decision Tree. All other algorithms go similarly with a few modifications in the program code.