A

MAJOR PROJECT REPORT

on

DETECTION OF MALARIA USING CONVOLUTIONAL NEURAL NETWORK

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Submitted by

BATCH - 22

V. ABHINAV SAI :207Y1A1201

Under the guidance

of

Mrs. A. LEELA SRAVANTHI

Assistant professor

DEPARTMENT OF INFORMATION TECHNOLOGY

MARRI LAXMAN REDDY

INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(AUTONOMOUS)

(Affiliated to JNTUH, Approved by AICTE New Delhi and Accredited by NBA &NAAC With A grade)

(MARCH 2024)

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

CERTIFICATE

This is to certify that the project titled **"DETECTION OF MALARIA USING CONVOLUTIONAL NEURAL NETWORK"** is being submitted by **V. Abhinav Sai (207Y1A1201)** in IV B. Tech II semester in **INFORMATION TECHNOLOGY** is a record bonafide work carried out by him. The results embodied in this report have not been submitted to any other university for the award of any degree.

Internal Guide **HOD**

Principal **External Examiner**

DECLARATION

I here by declare that the Major Project report entitled **"DETECTION OF MALARIA USING CONVOLUTIONAL NEURAL NETWORK"** submitted for the B.Tech degree is entirely my work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree.

Date:

V. Abhinav Sai
(207Y1A1201)

ACKNOWLEDGEMENT

I am happy to express my deep sense of gratitude to **Dr.K.Venkateswara Reddy,** Professor, Department of Computer Science and Engineering, Marri Laxman Reddy Institute of Technology & Management, for having provided with adequate facilities to pursue my project.

I would like to thank **Dr.M.Nagalakshmi,** Professor and Head, Department of Information Technology, Marri Laxman Reddy Institute of Technology & Management, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

I am very grateful to my project guide **Mrs.A.Leela Sravanthi,** Asst. Professor,Department of Information Technology, Marri Laxman Reddy Institute of Technology & Management, for her extensive patience and guidance throughout my project work.

I sincerely thank my seniors and all the teaching and non-teaching staff of the Department of Information Technology for their timely suggestions, healthy criticism and motivation during the course of this work.

I would also like to thank my classmates for always being there whenever I needed help or moral support. With great respect and obedience, I thank my parents and brother who were the backbone behind my deeds.

Finally, I express my immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to my need at the right time for the development and success of this work.

Abstract

Malaria is the deadliest disease in the earth and big hectic work for the health department. The traditional way of diagnosing malaria is by schematic examining blood smears of human beings for parasite-infected red blood cells under the microscope by lab or qualified technicians. This process is inefficient and the diagnosis depends on the experience and well knowledgeable person needed for the examination. Deep Learning algorithms have been applied to malaria blood smears for diagnosis before. However, practical performance has not been sufficient so far.

This paper proposes a new and highly robust machine learning model based on a Convolutional Neural Network (CNN) which automatically classifies and predicts infected cells in thin blood smears on standard microscope slides. A ten-fold cross-validation layer of the convolutional neural network on 27,558 single-cell images is used to understand the parameter of the cell. Three types of CNN models are compared based on their accuracy and select the precise accurate - Basic CNN, VGG-19 Frozen CNN, and VGG-19 Fine Tuned CNN. Then by comparing the accuracy of the three models, the model with a higher rate of accuracy is acquired.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

ABBREVATION	**DESCRIPTION**
CNN	Convolutional Neural Networks
VGG	Visual Geometry Graphics
OpenCV	Open-Source Computer Vision Library
API	Application Programming Interface
GUI	Graphical User Interface
CSV	Comma-Separated Values
XML	Extensible Markup Language
HTTP	Hypertext Transfer Protocol

CHAPTER 1

INTRODUCTION

The origin of malaria starts from the continent of Africa. The malaria was originated from the virus plasmodium falcipuram virus which is the cause for this disease. The disease has traveled through all around the world by the pathogen of mosquitoes. The virus can survive in hot and mild weather, but it cannot survive in very cold weather. The disease was 40 million years old from a very old age period. Malaria can infect all the animals to humans from infant to adult. Starting from fever to coma and death. The disease directly attacks the blood cell of the human body and breaks the white blood cell and stops the functionality of the organs of the human being. Malaria can be detected only by taking the blood samples from the human being and viewed in the microscope.

Malaria infection begins when an infected Anopheles mosquito bites a person, infecting Plasmodium parasites in the form of sporozoites. The sporozoites pass quickly into the liver of a human being and the sporozoites multiply asexually in the liver cells for 7 to 10 days. Then the parasites will form merozoites and will move through the bloodstream and settle at the lung capillaries. The merozoites are then moved through the red blood cells and multiply more, then the cell will burst. This will go through a process of a cycle of the carrier of a mosquito to another healthy person.

If a person is suffering from malaria, the person will get to know from the symptoms that his human body will emit as a warning symbol. The human body will start triggering the white blood cells to provide immune from the malarial cells. It causes high fever, headache, nausea, vomiting, abdominal pain or even coma. Even though there were many machine learning models to predict malaria. In the proposed work, a deep learning model is used to predict malaria with high accuracy.

1.1MALARIA

Malaria is a mosquito-borne infectious disease that affects humans and other animals. Malaria causes symptoms that typically include fever, tiredness, vomiting, and headaches. In severe cases, it can cause yellow skin, seizures, coma, or death. Symptoms usually begin ten to fifteen days after being bitten by an infected mosquito. If not properly treated, people may have recurrences of the disease months later. In those who have recently survived an infection, reinfection usually causes milder symptoms. This partial resistance disappears over months to years if the person has no continuing exposure to malaria.

Malaria is caused by single-celled microorganisms of the Plasmodium group. The disease is most commonly spread by an infected female Anopheles mosquito. The mosquito bite introduces the parasites from the mosquito's saliva into a person's blood. The parasites travel to the liver where they mature and reproduce. Five species of Plasmodium can infect and be spread by humans. Most deaths are caused by P. falciparum, whereas P. vivax, P. ovale, and P. malariae generally cause a milder form of malaria. The species P. knowlesi rarely causes disease in humans. Malaria is typically diagnosed by the microscopic examination of blood using blood films, or with antigen-based rapid diagnostic tests. Methods that use the polymerase chain reaction to detect the parasite's DNA have been developed, but are not widely used in areas where malaria is common due to their cost and complexity.

The risk of disease can be reduced by preventing mosquito bites through the use of mosquito nets and insect repellents or with mosquito-control measures such as spraying insecticides and draining standing water. Several medications are available to prevent malaria in travellers to areas where the disease is common. Occasional doses of the combination medication sulfadoxine/pyrimethamine are recommended in infants and after the first trimester of pregnancy in areas with high rates of malaria. As of 2020, there is one vaccine which has been shown to reduce the risk of malaria by about 40% in children in Africa. Efforts to develop more effective vaccines are ongoing. The recommended treatment for malaria is a combination of antimalarial medications that includes artemisinin. The second medication may be either mefloquine, lumefantrine, or sulfadoxine/pyrimethamine. Quinine, along with doxycycline, may be used if artemisinin is not available. It is recommended that in areas where the disease is common, malaria is confirmed if possible before treatment is started due to concerns of increasing drug resistance. Resistance among the parasites has developed to several antimalarial medications; for example, chloroquine-resistant P. falciparum has spread to most malarial areas, and resistance to artemisinin has become a problem in some parts of Southeast Asia.

The disease is widespread in the tropical and subtropical regions that exist in a broad band around the equator. This includes much of sub-Saharan Africa, Asia, and Latin America. In 2018 there were 228 million cases of malaria worldwide resulting in an estimated 405,000 deaths. Approximately 93% of the cases and 94% of deaths occurred in Africa. Rates of disease have decreased from 2010 to 2014 but increased from 2015 to 2017, during which there were

231 million cases. Malaria is commonly associated with poverty and has a significant negative effect on economic development. In Africa, it is estimated to result in losses of US$12 billion a year due to increased healthcare costs, lost ability to work, and adverse effects on tourism.

1.1.1 Signs and Symptoms

The signs and symptoms of malaria typically begin 8–25 days following infection, but may occur later in those who have taken antimalarial medications as prevention. Initial manifestations of the disease—common to all malaria species—are similar to flu-like symptoms, and can resemble other conditions such as sepsis, gastroenteritis, and viral diseases. The presentation may include headache, fever, shivering, joint pain, vomiting, hemolytic anemia, jaundice, hemoglobin in the urine, retinal damage, and convulsions.

The classic symptom of malaria is paroxysm—a cyclical occurrence of sudden coldness followed by shivering and then fever and sweating, occurring every two days (tertian fever) in P. vivax and P. ovale infections, and every three days (quartan fever) for P. malariae. P. falciparum infection can cause recurrent fever every 36–48 hours, or a less pronounced and almost continuous fever.

Severe malaria is usually caused by P. falciparum (often referred to as falciparum malaria). Symptoms of falciparum malaria arise 9–30 days after infection. Individuals with cerebral malaria frequently exhibit neurological symptoms, including abnormal posturing, nystagmus, conjugate gaze palsy (failure of the eyes to turn together in the same direction), opisthotonus, seizures, or coma.

1.1.2 Complications

Malaria has several serious complications. Among these is the development of respiratory distress, which occurs in up to 25% of adults and 40% of children with severe P. falciparum malaria. Possible causes include respiratory compensation of metabolic acidosis, noncardiogenic pulmonary oedema, concomitant pneumonia, and severe anaemia. Although rare in young children with severe malaria, acute respiratory distress syndrome occurs in 5–25% of adults and up to 29% of pregnant women. Coinfection of HIV with malaria increases mortality. Kidney failure is a feature of blackwater fever, where haemoglobin from lysed red blood cells leaks into the urine.

Infection with P. falciparum may result in cerebral malaria, a form of severe malaria that involves encephalopathy. It is associated with retinal whitening, which may be a useful clinical sign in distinguishing malaria from other causes of fever. An enlarged spleen, enlarged liver or both of these, severe headache, low blood sugar, and haemoglobin in the urine with kidney failure may occur. Complications may include spontaneous bleeding, coagulopathy, and shock.

Malaria in pregnant women is an important cause of stillbirths, infant mortality, miscarriage and low birth weight, particularly in P. falciparum infection, but also with P. vivax.

1.1.3 Cause

Malaria parasites belong to the genus Plasmodium (phylum Apicomplexa). In humans, malaria is caused by P. falciparum, P. malariae, P. ovale, P. vivax and P. knowlesi. Among those infected, P. falciparum is the most common species identified (~75%) followed by P. vivax (~20%). Although P. falciparum traditionally accounts for the majority of deaths, recent evidence suggests that P. vivax malaria is associated with potentially life-threatening conditions about as often as with a diagnosis of P. falciparum infection. P. vivax proportionally is more common outside Africa. There have been documented human infections with several species of Plasmodium from higher apes; however, except for P. knowlesi—a zoonotic species that causes malaria in macaques—these are mostly of limited public health importance.

1.1.4 Diagnosis

Owing to the non-specific nature of the presentation of symptoms, diagnosis of malaria in non-endemic areas requires a high degree of suspicion, which might be elicited by any of the following: recent travel history, enlarged spleen, fever, low number of platelets in the blood, and higher-than-normal levels of bilirubin in the blood combined with a normal level of white blood cells. Reports in 2016 and 2017 from countries where malaria is common suggest high levels of over diagnosis due to insufficient or inaccurate laboratory testing.

Malaria is usually confirmed by the microscopic examination of blood films or by antigen-based rapid diagnostic tests (RDT). In some areas, RDTs must be able to distinguish whether the malaria symptoms are caused by Plasmodium falciparum or by other species of parasites since treatment strategies could differ for non-P. falciparum infections. Microscopy is the most commonly used method to detect the malarial parasite—about 165 million blood films were examined for malaria in 2010. Despite its widespread usage, diagnosis by microscopy suffers from two main drawbacks: many settings (especially rural) are not equipped to perform the test, and the accuracy of the results depends on both the skill of the person examining the blood film and the levels of the parasite in the blood. The sensitivity of blood films ranges from 75 to 90% in optimum conditions, to as low as 50%. Commercially available RDTs are often more accurate than blood films at predicting the presence of malaria parasites, but they are widely variable in diagnostic sensitivity and specificity depending on manufacturer, and are unable to tell how many parasites are present. However, incorporating

RDTs into the diagnosis of malaria can reduce antimalarial prescription. Although RDT does not improve the health outcomes of those infected with malaria, it also does not lead to worse outcomes when compared to presumptive antimalarial treatment.

In regions where laboratory tests are readily available, malaria should be suspected, and tested for, in any unwell person who has been in an area where malaria is endemic. In areas that cannot afford laboratory diagnostic tests, it has become common to use only a history of fever as the indication to treat for malaria—thus the common teaching "fever equals malaria unless proven otherwise". Although polymerase chain reaction-based tests have been developed, they are not widely used in areas where malaria is common as of 2012, due to their complexity.

1.2 MACHINE LEARNING

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly. Machine learning algorithms are often categorized as supervised or unsupervised.

Supervised Machine Learning Algorithms:

Supervised machine learning algorithms can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

Unsupervised Machine Learning Algorithms:

In contrast, unsupervised machine learning algorithms are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn’t figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

Semi-Supervised Machine Learning Algorithms:

Semi-supervised machine learning algorithms fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn’t require additional resources.

Reinforcement Machine Learning Algorithms:

Reinforcement machine learning algorithms is a learning method that interacts with its environment by producing actions and discovers errors or rewards. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

1.2.1 Deep Learning

Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

How Deep Learning Works?

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support. Deep learning, a subset of machine learning, utilizes a hierarchical level of artificial neural networks to carry out the process of machine learning.

Deep Learning Versus Machine Learning:

One of the most common AI techniques used for processing big data is machine learning, a self-adaptive algorithm that gets increasingly better analysis and patterns with experience or with newly added data.

If a digital payments company wanted to detect the occurrence or potential for fraud in its system, it could employ machine learning tools for this purpose. The computational algorithm built into a computer model will process all transactions happening on the digital platform, find patterns in the data set and point out any anomaly detected by the pattern.

Deep learning, a subset of machine learning, utilizes a hierarchical level of artificial neural networks to carry out the process of machine learning. The artificial neural networks are built like the human brain, with neuron nodes connected together like a web. While traditional programs build analysis with data in a linear way, the hierarchical function of deep learning systems enables machines to process data with a nonlinear approach.

A traditional approach to detecting fraud or money laundering might rely on the amount of transaction that ensues, while a deep learning nonlinear technique would include time, geographic location, IP address, type of retailer and any other feature that is likely to point to fraudulent activity.

The first layer of the neural network processes a raw data input like the amount of the transaction and passes it on to the next layer as output.

The second layer processes the previous layer’s information by including additional information like the user's IP address and passes on its result.

The next layer takes the second layer’s information and includes raw data like geographic location and makes the machine’s pattern even better. This continues across all levels of the neuron network.

CHAPTER 2

LITERATURE SURVEY

2.1. Shuying Liu and Weihong Deng., “Very deep convolutional neural network based image classification using small training sample size” 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR).

Since Krizhevsky won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 competition with the brilliant deep convolutional neural networks (D-CNNs), researchers have designed lots of D-CNNs. However, almost all the existing very deep convolutional neural networks are trained on the giant ImageNet datasets. Small datasets like CIFAR-10 has rarely taken advantage of the power of depth since deep models are easy to overfit. In this paper, we proposed a modified VGG-16 network and used this model to fit CIFAR-10. By adding stronger regularizer and using Batch Normalization, we achieved 8.45% error rate on CIFAR-10 without severe overfitting. Our results show that the very deep CNN can be used to fit small datasets with simple and proper modifications and don't need to re-design specific small networks. We believe that if a model is strong enough to fit a large dataset, it can also fit a small one.

2.2. NimaHatami., Yann Gavet and Johan Debayale., “Classification of time-series images using deep convolutional neural network” Tenth International Conference on Machine Vision (ICMV 2017) Vol.10696.

Convolutional Neural Networks (CNN) has achieved a great success in image recognition task by automatically learning a hierarchical feature representation from raw data. While the majority of Time-Series Classification (TSC) literature is focused on 1D signals, this paper uses Recurrence Plots (RP) to transform time-series into 2D texture images and then take advantage of the deep CNN classifier. Image representation of time-series introduces different feature types that are not available for 1D signals, and therefore TSC can be treated as texture image recognition task. CNN model also allows learning different levels of representations together with a classifier, jointly and automatically. Therefore, using RP and CNN in a unified framework is expected to boost the recognition rate of TSC. Experimental results on the UCR time-series classification archive demonstrate competitive accuracy of the proposed approach, compared not only to the existing deep architectures, but also to the state-of-the art TSC algorithms.

2.3. Nicholas E. Ross., Charles J. Pitchard., David M. Rubin and Adriano G. Duse "Automated image processing method for the diagnosis and classification of malaria on thin blood smears" IEEE Vol.44.

Malaria is a serious global health problem, and rapid, accurate diagnosis is required to control the disease. An image processing algorithm to automate the diagnosis of malaria on thin blood smears is developed. The image classification system is designed to positively identify malaria parasites present in thin blood smears, and differentiate the species of malaria. Images are acquired using a charge-coupled device camera connected to a light microscope. Morphological and novel threshold selection techniques are used to identify erythrocytes (red blood cells) and possible parasites present on microscopic slides. Image features based on colour, texture and the geometry of the cells and parasites are generated, as well as features that make use of a priori knowledge of the classification problem and mimic features used by human technicians. A two-stage tree classifier using backpropogation feedforward neural networks distinguishes between true and false positives, and then diagnoses the species (Plasmodium falciparum, P. vivax, P. ovale or P. malariae) of the infection. Malaria samples obtained from the Department of Clinical Microbiology and Infectious Diseases at the University of the Witwatersrand Medical School are used for training and testing of the system. Infected erythrocytes are positively identified with a sensitivity of 85% and a positive predictive value (PPV) of 81%, which makes the method highly sensitive at diagnosing a complete sample provided many views are analyzed. Species were correctly determined for 11 out of 15 samples.

2.4. K.M. Khatri., V.R. Ratnaparkhe., S.S. Agarwal and A.S. Bhalchandra "Image processing approach for malarial parasite identification" International Journal of computer Application (IJCA).

Malaria is a very serious infectious disease caused by a peripheral blood parasite of the genus Plasmodium. Conventional microscopy, which is currently "the gold standard" for malaria diagnosis has occasionally proved inefficient since it is time consuming and results are difficult to reproduce. As it poses a serious global health problem, automation of the evaluation process is of high importance. In this work, an accurate, rapid and affordable model of malaria diagnosis using stained thin blood smear images was developed. The method made use of the intensity features of Plasmodium parasites and erythrocytes. Images of infected and non-infected erythrocytes were acquired, pre-processed, relevant features extracted from them and eventually diagnosis was made based on the features extracted from the images. A set of features based on intensity have been proposed, and the performance of these features on the red blood cell samples from the created database have been evaluated using an artificial neural network (ANN) classifier. The results have shown that these features could be successfully used for malaria detection.

2.5. Vishnu V. Makkapati and Raghuveer M. Rao., "Segmentation of malarial parasites in peripheral blood smear images" IEEE International Conference on Acoustics, Speech and Signal Processing.

Detection of malaria parasites in stained blood smears is critical for treatment of the disease. Automation of this process will help in reducing the time taken for diagnosis and the chance for human errors. However, the variability and artifacts in microscope images of blood samples pose significant challenges for accurate detection. A scheme based on HSV color space that segments Red Blood Cells and parasites by detecting dominant hue range and by calculating optimal saturation thresholds is presented in this paper. Methods that are less computation-intensive than existing approaches are proposed to remove artifacts. The scheme is evaluated using images taken from Leishman-stained blood smears. Sensitivity and specificity of the scheme are found to be 83% and 98% respectively.

2.6. CorentinDallet., Saumya Kareem and Izzet Kale., "Real time blood image processing application for malaria diagnosis using mobile phones" IEEE International Symposium on Circuits and Systems (ISCAS).

This paper describes a fast and reliable mobile phone Android application platform for blood image analysis and malaria diagnosis from Giemsa-stained thin blood film images. The application is based on novel Annular Ring Ratio Method which is already implemented, tested and validated in MATLAB. The method detects the blood components such as the Red Blood Cells (RBCs), White Blood Cells (WBCs), and identifies the parasites in the infected RBCs. The application also recognizes the different life stages of the parasites and calculates the parasitemia which is a measure of the extent of infection. In this paper, an attempt has been made to implement the malarial diagnosis algorithm, that has already been implemented, tested and evaluated on a MATLAB platform, into an Android mobile phone. The main objective of the research is to successfully implement the application on to the mobile platform without the loss of information integrity, with minimal memory footprint on the mobile phone. The results have to be the same as the Matlab output, as well as keeping to an acceptable processing speed and duration.

2.7. Zhaoui Liang., Andrew Powell., IlkerErsoy.,Mahdieh Poostchi., Kamolraut Silmaut and Kannapan Palani., "CNN-based image analysis for malaria diagnosis" IEEE International Conference on Bioinformatics and Biomedicine (BIBM).

Malaria is a major global health threat. The standard way of diagnosing malaria is by visually examining blood smears for parasite-infected red blood cells under the microscope by qualified technicians. This method is inefficient and the diagnosis depends on the experience and the knowledge of the person doing the examination. Automatic image recognition technologies based on machine learning have been applied to malaria blood smears for diagnosis before. However, the practical performance has not been sufficient so far. This study proposes a new and robust machine learning model based on a convolutional neural network (CNN) to automatically classify single cells in thin blood smears on standard microscope slides as either infected or uninfected. In a ten-fold cross-validation based on 27,578 single cell images, the average accuracy of our new 16-layer CNN model is 97.37%. A transfer learning model only achieves 91.99% on the same images. The CNN model shows superiority over the transfer learning model in all performance indicators such as sensitivity (96.99% vs 89.00%), specificity (97.75% vs 94.98%), precision (97.73% vs 95.12%), F1 score (97.36% vs 90.24%), and Matthews correlation coefficient (94.75% vs 85.25%).

2.8. Yuhang Dong., Zhuocheng Jiang., Hongda Shen., W. David Pan., Lance A. Williams., Vishnu V. B. Reddy., "Evaluations of deep convolutional neural network for automatic identification of malaria infected cells" IEEE EMBS International Conference on Biomedical & Health Informatics (BHI).

This paper studied automatic identification of malaria infected cells using deep learning methods. We used whole slide images of thin blood stains to compile an dataset of malaria-infected red blood cells and non-infected cells, as labeled by a group of four pathologists. We evaluated three types of well-known convolutional neural networks, including the LeNet, AlexNet and GoogLeNet. Simulation results showed that all these deep convolution neural networks achieved classification accuracies of over 95%, higher than the accuracy of about 92% attainable by using the support vector machine method. Moreover, the deep learning methods have the advantage of being able to automatically learn the features from the input data, thereby requiring minimal inputs from human experts for automated malaria diagnosis. Malaria is a life-threatening disease caused by parasites that are transmitted to people through the bites of infected female Anopheles mosquitoes. According to the report released by World Health Organization (WHO), there were 214 million cased of malaria in 2015 and 438,000 deaths. In most cases, malaria can only be diagnosed by manual examination of the microscopic slide. Whole slide imaging, which scans the conventional glass slides in order to produce digital slides, is the most recent imaging modality being employed by pathology departments worldwide. In order to provide a reliable diagnosis, necessary training and specialized human resource are required. Unfortunately, these resources are far from being adequate and frequently often unavailable in underdeveloped areas where malaria has a marked predominance. Therefore, an automated diagnostic system can provide an efficient solution to this problem.

2.9. Gopalakrishna Pillai Gopakumar., "Convolutional neural network-based malaria diagnosis from stack of blood smear images acquired using custom-built slide scanner" Journal of Biophontics, Vol. 11 No.3.

The present paper introduces a focus stacking-based approach for automated quantitative detection of Plasmodium falciparum malaria from blood smear. For the detection, a custom designed convolutional neural network (CNN) operating on focus stack of images is used. The cell counting problem is addressed as the segmentation problem and we propose a 2-level segmentation strategy. Use of CNN operating on focus stack for the detection of malaria is first of its kind, and it not only improved the detection accuracy (both in terms of sensitivity [97.06%] and specificity [98.50%]) but also favored the processing on cell patches and avoided the need for hand-engineered features. The slide images are acquired with a custom-built portable slide scanner made from low-cost, off-the-shelf components and is suitable for point-of-care diagnostics. The proposed approach of employing sophisticated algorithmic processing together with inexpensive instrumentation can potentially benefit clinicians to enable malaria diagnosis.

2.10. S.Revathy., B.Bharathi., P.Jeyanthi., M.Ramesh., "Chronic Kidney Disease Prediction using Machine Learning Models", International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249 – 8958, Vol.9, Issue-1, October 2019.

In today's era everyone is trying to be conscious about health although due to workload and busy schedule, one gives attention to the health when it shows any symptoms of some kind. But CKD is a disease which doesn't shows symptoms at all or in some cases it doesn't show any disease specific symptoms it is hard to predict, detect and prevent such a disease and this could be led to permanently health damage, but machine learning can be hope in this problem it is best in prediction and analysis. By using data of CKD patients with 14 attributes and 400 record we are going to use various machine learning techniques like Decision Tree, SVM, etc. To build a model with maximum accuracy of predicting whether CKD or not and if yes then its Severity. We all knows, that Kidney is essential organ in human body. Which has main functionalities like excretion and osmoregulation. In simple words we can say that all the toxic and unnecessary material from the body is collected and thrown out by kidney and excretion system. There are approximately 1 million cases of Chronic Kidney Disease (CKD) per year in India. Chronic kidney disease is also called renal failure. It is a dangerous disease of the kidney which produces gradual loss in kidney functionality. CKD is a slow and periodical loss of kidney function over a period of several years. A person will develop permanent kidney failure. If CKD is not detected and cured in early stage then patient can show following Symptoms: Blood Pressure, anaemia, weekboans, poor nutrition health and nerve damage, decreased immune response because at advanced stages dangerous levels of fluids, electrolytes, and wastes can build up in your blood and body. Hence it is essential to detect CKD at its early stage but it is unpredictable as its Symptoms develop slowly and aren't specific to the disease. Some people have no symptoms at all so machine learning can be helpful in this problem to predict that the patient has CKD or not. Machine learning does it by using old CKD patient data to train predicting model. Glomerular Filtration Rate (GFR) is the best test to measure your level of kidney function and determine your stage of chronic kidney disease. It can be calculated from the results of your blood creatinine, age, race, gender, and other factors. The earlier disease is detected the better chance of showing or stopping its progression.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

A model for the new genotypic signature for the detection of the malarial cell. So, using that concept, the bloodstain concept is selected that variability and artifacts are important for taking the microscopic images on malarial cells. The model shows that they have taken Leishman's blood smears for this project. So, understanding the concept of the Leishman blood smears and undergoing our project with the same concept. The concept of image processing using OpenCV and the contour detection concept which is sued in the proposed work to use contour detection on the blood cell to find the attributes of the blood cell. So, once the attributes are detected, the number of dots will be counted to conclude that the given cell is a malarial cell or not.

The introduction of the deep learning concept called the convolutional neural network and using a convolutional network finding the accuracy of the infected or not infected blood cell. So, the concept of building the scratch convolutional neural network is used in our proposed work and other convolutional neural networks. The advanced concept of a convolutional neural network called the VGG which is Visual Geometry Graphics. The VGG-16 is used in their model. The concept of the VGG-16 model is considered and developed the proposed research work in the VGG-19 model. The automatic classification of blood cells using the deep convolutional neural network. The concept uses the database of the malarial cell and used it with the LeNet and AlexNet and GoogleNet. So, understanding the concept of the three convolutional neural networks and using it on our project for building the three convolutional neural networks. The backpropagation feedforward neural network concept. The learning rate of this project was higher than the basic convolutional neural network. So, understanding the concept of the neural network used in that model. A model featuring the time series classification algorithm and recurrence plot. Both these concepts were used to build a unified network using a convolutional neural network. By using the method understanding the working of the Time Series Classification algorithm. A real-time identification of malarial cells in the human body using a mobile phone. By understanding the concept of building a tracking watch, which will monitor our pulse rate and calorie rate. The concept of forming the artificial microscopic slide of plasmodium falcipuram cell, which causes malaria. Using the concept of the cell structure and cell attributes are studied to understand the morphology of the malarial cell.

3.1.1 Disadvantages

- Well knowledgeable person needed for the examination
- Inefficient
- Low accuracy
- High time consumption

3.2 PROPOSED SYSTEM

The problem starts with the decision taking whether the particular cell is infected or healthy. Start training the machine by giving all the attributes of the images. So, by collecting as much as images through the internet, where totally 27,558 images were collected. Once all the images are acquired, they started the process of training, validation, and testing. Provided 17,361 to the training set, 1,929 to the validation set, and 8,268 to the testing set. Then imported the OpenCV library to the process.

By using OpenCV, the property will use contour detect on the particular cell. Contour can be described simply as a curve joining all the continuous points having the same color or same intensity. A contour is a useful tool for shape analysis and object detection. When seeing the images, it can be concluded that it has some black dark spots inside the cell. So, these contours will draw the curve near the dark spot forming a circle around.

3.2.1 Modules

Once you know exactly what you want and the equipment are in hand, it takes you to the first real step of machine learning.

Gathering Data:

- This step is very crucial as the quality and quantity of data gathered will directly determine how good the predictive model will turn out to be.
- The data collected is then tabulated and called as Training Data.

Preparing Dataset:

- The data is loaded into a suitable place and then prepared for use in machine learning training.

Choosing a Model:

- The next step that follows in the workflow is choosing a model among the many that researchers and data scientists have created over the years. Make the choice of the right one that should get the job done.

Training:

- The training process involves initializing some random values for say A and B of our model, predict the output with those values.
- Then compare it with the model's prediction and then adjust the values so that they match the predictions that were made previously.
- This process then repeats and each cycle of updating is called one training step.

Evaluation:

- Evaluation allows the testing of the model against data that has never been seen and used for training and is meant to be representative of how the model might perform when in the real world.

Hyperparameter Tuning:

- Hyperparameter is a parameter whose value is set before the learning process begins. These parameters have to be tuned so that the model can optimally solve the machine learning problem.

Prediction:

- This is the point where the value of machine learning is realized. Here you can Finally use your model to predict the outcome of what you want.

3.2.2 Advantages

- High accuracy
- Low time consumption
- Qualified technicians are not needed.

CHAPTER 4

SYSTEM DESIGN

4.1 BLOCK DIAGRAM

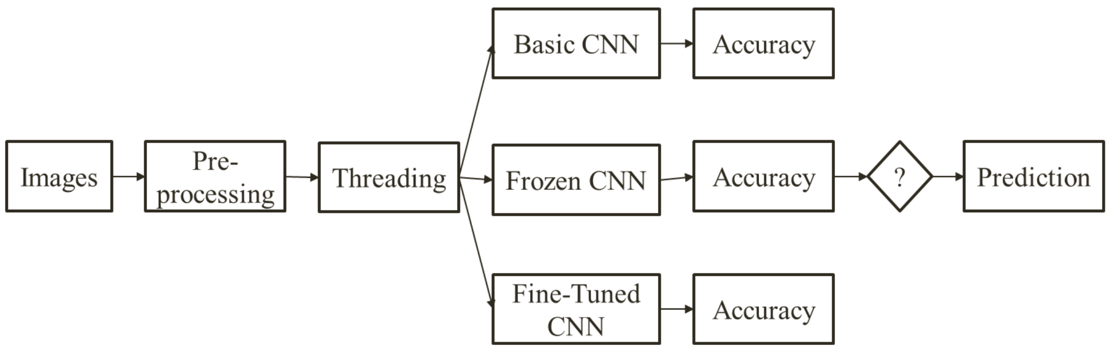

Fig 4.1: Block Diagram

4.2 ARCHITECTURE

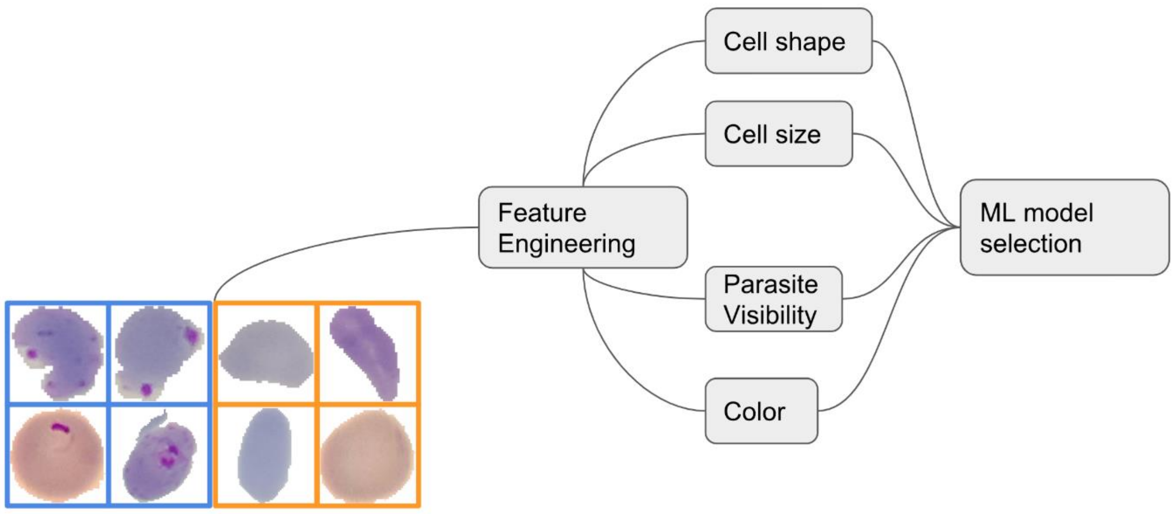

Fig 4.2: Proposed Architecture

4.3 EXPLANATION

When the process is finished with the contour detection, then it has been moved on to the most important process called threading. Threading is a separate flow of execution. That means the program will have two things happening at once. Threading can also be called as multiprocessing. So, in this project, the threading process happens with an attribute named Thread Pool Executor. Thread Pool Executor creates a context manager, telling it how many worker threads it wants in the pool. It uses .map() to step through an iterable of things. The library will produce the minimum dimension, average dimension, median dimension, and the maximum dimension in the format of an array. Then loading and resizing of images will take place through Thread Pool Executor for each training set, validation set, and testing set. Once running the images into the machine, the XYZ points will be acquired. Images will move to the setup configuration settings, scaling of images, and label encoding. In this stage, encoding and scaling will take place, where the images are converted into binary code of 1's or 0's and the new term will be used called epochs. Epoch is a point where the time starts.

Epoch = (number of iterations * batch size) / total number of images in training

4.3.1 Convolutional Neural Network

In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Object detections, recognition faces etc., are some of the areas where CNNs are widely used. CNN image classifications take an input image, process it and classify it under certain categories (Eg. Dog, Cat, Tiger, Lion). Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d (h = Height, w = Width, d = Dimension). Eg. An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to over fitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

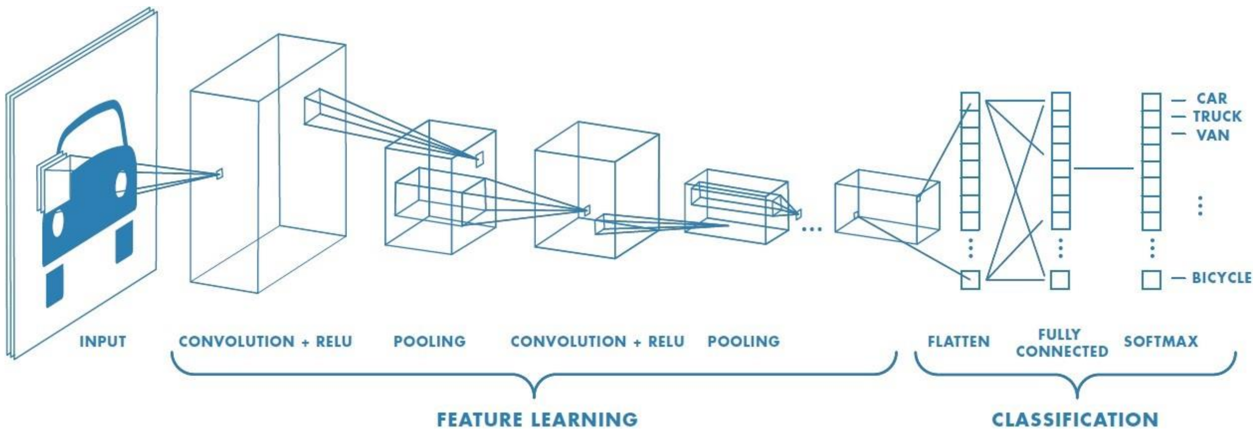

Fig 4.3: Neural network with many convolutional layers

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

4.3.2 Basic Convolutional Neural Network

The Basic Convolutional Neural Network is done from scratch. So, by using the Tensor Flow, Python library releases the Keras. Keras is the open-source neural network library in Python. By doing so, the Keras has an attribute

named Conv2D. It is also Conv1D but in this project, Conv2D is being used because the image being used here is a two-dimensional image. Then the max-pooling part comes there, to maximize the cluster of neurons at the prior layers. Max pooling is a down-sampling strategy on CNN. Normally max pooling is used to converge any type of matrix to the smallest possible value. For Example, by taking the 4X4 matrix in the matrix, there are four corner values. By using the max-pooling effect, the determinants of the four values are detected and the resultant will be a 2X2 matrix. In the Keras layers, there is another layer called Flatten which is used to prepare to become a vector for the fully connected layers by using the flatten command with respective to Keras. So once finished with the attributes of the Keras, then the activation of sigmoid will occur by calculating the accuracy. Once the attributes of the Keras are acquired, the model is built using Keras.model().

Then printing the model summary takes place to identify the given parameters full-fledged with both training and testing. There are a total of 15,102,529 trainable parameters that are acquired and zero Non-trainable parameters. To finish it with the model summary and the training of the model will take place by giving the details of the epochs, batch size, callbacks, verbose, and validation data. Then, fitting the model will start the process of 25 epoch processes. By doing so, the loss percentage will start decreasing slower and slower and comes to the halt after the 25th epoch. At 25th epoch, the readings are, loss: 0.0054, accuracy: 0.9985, validation accuracy: 0.9430 and validation loss: 0.4244.

By seeing these values, it is not convincing because the loss value is way higher of what the expectation was. To get lesser loss value, going to the next version of CNN may produce accurate value. In the graph depicting the Accuracy vs Epoch and Loss vs Epoch graph states that there is a huge gap between the dark line with a grey line that shows the validation percentage of both accuracy and loss graph. Whereas the loss graph as the most deviation between the dark line and the grey line this will give us the graphical representation of the CNN process of the project. So, once the model is trained and acquired the accuracy, it is saved in the extension of basic.h5. The h5 extension is a hierarchical data format used to store scientific data.

4.3.3 Frozen Convolutional Neural Network

The order of presentation is same for the both basic and frozen CNN but the given attributes will change as the frozen CNN takes place. In this CNN same, but using the Keras with VGG. VGG comes in two types VGG16 and VGG19. The main difference between the VGG16 and VGG19 is that VGG16 uses 16 layers and VGG19 uses 19 layers in the deep neural network. In this project, VGG19 is used to make it effective run on the images and model for training.

VGG is built under 16 layers the main layers are convolutions layer and max-pooling layer. They are fully connected at the end of the model. The model size is 528MB. The first layer is the convolution layer with 64 filters goes where the image goes through the first layer. Then the second layer is the convolution + max-pooling layer with 64 filters. These layers have a 3X3 matrix this example already given above. Next is the third layer that is convolution layer with 128 filters when by doing this the pool is divide by 2 when it goes from one filter to another filter. Then the next layer is the convolutional layer + maxpooling layer with 128 filters. From this layer, the images will move from images to the pixel unit through the filtering method and produce the matrix. Then comes the 256 filter, 512 filter, at last, the fully connected node layer where it has 4096 nodes where the images will be encoded and then at the output side the softmax activation will be activated with the 1000 nodes. The 224 x 224 x 3 image will further transform into 1 x 1000 when the image comes out from the softmax activation.

So once the model is fitted and the model will be using the history value so from that the model summary can be retrieved. From the model summary, the total parameter is 22,647,367 in that trainable parameter is 2,622,977 and the non-trainable parameter is 20,024,384. From here, it has been concluded that these non-trainable parameters are somehow escaped from the Basic CNN resulting in the zero nontrainable parameters. The model fitting process is over the training of the model that will be occurred by using the 25 epochs. It is the same as the basic CNN. 25 Epoch will start with 17361 samples then the loss and accuracy values will be generated. Once the process is generated and moves to the 25th Epoch the value will be also generated. Loss: 0.1017, accuracy: 0.9956, validation accuracy: 0.9430 and validation loss: 0.1751. Once the values are acquired by concluding that the value which was generated is not enough as the loss is less but not accurate. The Epoch vs Loss graph shows the loss percentage which is very low when compared with the basic CNN but not very accurate so it is good to go with the next high certain and pre-trained convolutional neural network.

4.3.4 Fine-Tuned Convolutional Neural Network

This is the last stage of the convolutional neural network. Fine-tuning means taking weights of a trained neural network and use it as initialization for a new model being trained on data from the same domain. It is often used to speed up the training and overcome small dataset sizes. The fine-tuned convolutional neural network has some techniques to accomplish the tasks. They remove the final layer of the past trained network and change it with our new softmax layer that is used in our problem. They use a low learning rate to train the network. The finely tuned practice will make an initial learning rate 10 times smaller than the one used for the basic convolutional neural network. The next practice of the fine-tuned convolutional neural network is locking up the weights of the first layers of the past trained network. This is because the first layers capture unique features like curves and edges that are also relevant to our new problem.

In this convolutional neural network, the imagenet is used as a weight and GG 19 is used with the Keras. So, the layers are getting ready to freeze by training the VGG layers. Then by using the for loop the layers are extracted and checked with the processed weights of imagenet. Then the output of Vgg is analyzed. In Keras, the flattening process will be executed. The flattening is converting the data into a 1-dimensional array for inputting it to the next layer. The output of the convolutional layers will create a single long feature vector and it is connected to the final classification model, which will become a fully connected layer. Then it moves to the next layer called the dense layer. A dense layer is a classical fully connected neural network layer where each input node is connected to each output node. Normally the dense layer will act as a matrix-vector multiplication. The next layer in the Keras is the dropout layer.

The dropout layer is a technique used to prevent a model from overfitting. Dropout works by randomly setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0 at each update of the training phase. Then the model is ready to be fitted with the activation of the sigmoid and then producing metrics with the model. Once the model is trained, the total layers are 28 and total trainable layers are 16. So, the numbers are achieved by running the network. The next step is to train the model. The callback response is used during training because they provide specific functionality with a set of methods called testing and prediction. After 25th Epoch the loss: 0.0792, accuracy: 0.9989 validation loss is 0.1127 and validation accuracy is 0.9641. So once the model is fully trained and tested, it is then represented by a graph and check the accuracy vs loss in the epoch. So once the model is trained and acquired the accuracy, it is saved in the extension of frozen.h5. The h5 extension is a hierarchical data format used to store scientific data.

4.3.5 Proposed Algorithm

Step 1: Collect the pre-processed images and merge them under one file for easy transportation

Step 2: Split the images according to train and test using sklearn

Step 3: Use OpenCV at the images and understand the parameters of the images and do the process of contour detection

Step 4: Process the image using thread pool executor for not to face time straining.

Step 5: Create the Basic CNN model from scratch and fit the model

Step 6: Now insert the images into the model and run the model by using the tensor flow and Keras package.

Step 7: Use Epoch = (number of iterations * batch size) / total number of images in training

Step 8: Check the accuracy if it is not sufficient then move to the next CNN model

Step 9: Create the Frozen CNN model and fit the model and repeat step 6-7

Step 10: Check the accuracy if it is not sufficient then move to the next CNN model

Step 11: Create the Fine-Tuned CNN model and fit the model and repeat step 6-7

Step 12: If the accuracy is sufficient to stop here and get the accuracy rate.

4.3.6 OpenCV

OpenCV (Open-Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU acceleration for real-time operations.

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. All of the new developments and algorithms appear in the C++ interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation. Wrappers in several programming languages have been developed to encourage adoption by a wider audience. In version 3.4, JavaScript bindings for a selected subset of OpenCV functions was released as OpenCV.js, to be used for web platforms.

OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

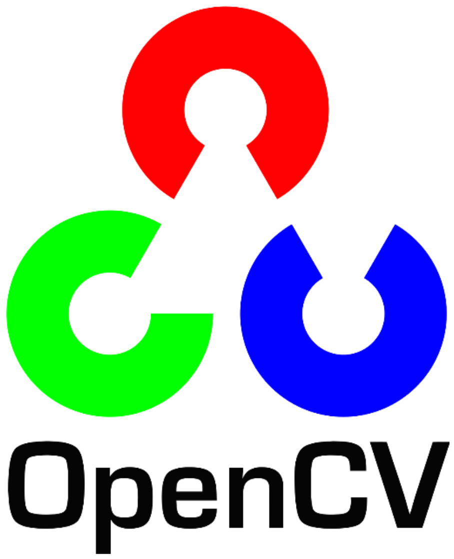

Fig 4.4: OpenCV

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching street-view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

CHAPTER 5

SYSTEM DESCRIPTION

5.1 HARDWARE DESCRIPTION

System	:	Pentium IV 3.5 GHz or Latest Version.
Hard Disk	:	40 GB.
Monitor	:	14' Color Monitor.
Mouse	:	Optical Mouse.
Ram	:	1 GB.

5.2 SOFTWARE DESCRIPTION

Operating system	:	Windows 10
Coding Language	:	Python
Tools used	:	Anaconda

5.2.1 Anaconda

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012.

Fig 5.1: Anaconda

As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, both of which are not free.

Package versions in Anaconda are managed by the package management system conda. This package manager was spun out as a separate open-source package as it ended up being useful on its own and for other things than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depend on, and a small number of other packages.

Overview:

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI). The big difference between conda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.

When pip installs a package, it automatically installs any dependent Python packages without checking if these conflict with previously installed packages. It will install a package and any of its dependencies regardless of the state of the existing installation. Because of this, a user with a working installation of, for example, Google Tensorflow, can find that it stops working having used pip to install a different package that requires a different version of the dependent numpy library than the one used by Tensorflow. In some cases, the package may appear to work but produce different results in detail. In contrast, conda analyses the current environment including everything currently installed, and, together with any version limitations specified (e.g., the user may wish to have Tensorflow version 2,0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done.

Open-source packages can be individually installed from the Anaconda repository, Anaconda Cloud (anaconda.org), or the user's own private repository or mirror, using the conda install command. Anaconda, Inc. compiles and builds the packages available in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit. Anything available on PyPI may be installed into a conda environment using pip, and conda will keep track of what it has installed itself and what pip has installed.

Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it is possible to create new environments that include any version of Python packaged with conda.

Anaconda Navigator:

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder
- Glue
- Orange
- RStudio
- Visual Studio Code

Conda:

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The conda package and environment manager is included in all versions of Anaconda, Miniconda and Anaconda Repository.

Anaconda Cloud:

Anaconda Cloud is a package management service by Anaconda where users can find, access, store and share public and private notebooks, environments, and conda and PyPI packages. Cloud hosts useful Python packages, notebooks and environments for a wide variety of applications. Users do not need to log in or to have a Cloud account, to search for public packages, download and install them.

Users can build new packages using the Anaconda Client command line interface (CLI), then manually or automatically upload the packages to Cloud.

5.2.2 Python Programming

Python is an interpreted, high-level and general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, a free and open-source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly.

- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" design philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the CPython reference implementation that would offer marginal increases in speed at the cost of clarity.[60] When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic. Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as Pythonistas.

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 SOFTWARE IMPLEMENTATION

Implementation includes all those activities that take place to convert from the old system to the new. The old system consists of manual operations, which is operated in a very different manner from the proposed new system. A proper implementation is essential to provide a reliable system to meet the requirements of the organizations. An improper installation may affect the success of the computerized system.

6.1.1 Implementation Methods:

There are several methods for handling the implementation and the consequent conversion from the old to the new computerized system.

The most secure method for conversion from the old system to the new system is to run the old and new system in parallel. In this approach, a person may operate in the manual older processing system as well as start operating the new computerized system. This method offers high security. However, the cost for maintaining two systems in parallel is very high. This outweighs its benefits.

A working version of the system can also be implemented in one part of the organization and the personnel will be piloting the system and changes can be made as and when required. But this method is less preferable due to the loss of entirety of the system.

The implementation plan includes a description of all the activities that must occur to implement the new system and to put it into operation. It identifies the personnel responsible for the activities and prepares a time chart for implementing the system.

The implementation plan consists of the following steps.

- List all files required for implementation.
- Identify all data required to build new files during the implementation.
- List all new documents and procedures that go into the new system.

The implementation plan should anticipate possible problems and must be able to deal with them. The usual problems may be missing documents; mixed data formats between current and files, errors in data translation, missing data etc.

CHAPTER 7

SYSTEM TESTING

System testing is a critical aspect of Software Quality Assurance and represents the ultimate review of specification, design and coding. Testing is a process of executing a program with the intent of finding an error. A good test is one that has a probability of finding an as yet undiscovered error. The purpose of testing is to identify and correct bugs in the developed system. Nothing is complete without testing. Testing is the vital to the success of the system.

In the code testing the logic of the developed system is tested. For this every module of the program is executed to find an error. To perform specification test, the examination of the specifications stating what the program should do and how it should perform under various conditions. Unit testing focuses first on the modules in the proposed system to locate errors. This enables to detect errors in the coding and logic that are contained within that module alone. Those resulting from the interaction between modules are initially avoided. In unit testing step each module has to be checked separately.

System testing does not test the software as a whole, but rather than integration of each module in the system. The primary concern is the compatibility of individual modules. One has to find areas where modules have been designed with different specifications of data lengths, type and data element name. Testing and validation are the most important steps after the implementation of the developed system. The system testing is performed to ensure that there are no errors in the implemented system. The software must be executed several times in order to find out the errors in the different modules of the system.

Validation refers to the process of using the new software for the developed system in a live environment i.e., new software inside the organization, in order to find out the errors. The validation phase reveals the failures and the bugs in the developed system. It will be come to know about the practical difficulties the system faces when operated in the true environment. By testing the code of the implemented software, the logic of the program can be examined. A specification test is conducted to check whether the specifications stating the program are performing under various conditions. Apart from these tests, there are some special tests conducted which are given below:

Peak Load Tests: This determines whether the new system will handle the volume of activities when the system is at the peak of its processing demand. The test has revealed that the new software for the agency is capable of handling the demands at the peak time.

Storage Testing: This determines the capacity of the new system to store transaction data on a disk or on other files. The proposed software has the required storage space available, because of the use of a number of hard disks.

Performance Time Testing: This test determines the length of the time used by the system to process transaction data.

In this phase the software developed Testing is exercising the software to uncover errors and ensure the system meets defined requirements. Testing may be done at 4 levels

- Unit Level
- Module Level
- Integration & System
- Regression

7.1 UNIT TESTING

A Unit corresponds to a screen /form in the package. Unit testing focuses on verification of the corresponding class or Screen. This testing includes testing of control paths, interfaces, local data structures, logical decisions, boundary conditions, and error handling. Unit testing may use Test Drivers, which are control programs to co-ordinate test case inputs and outputs, and Test stubs, which replace low-level modules. A stub is a dummy subprogram.

7.2 VALIDATION TESTING

In this requirement established as part of software requirements analysis are validated against the software that has been constructed. Validation testing provides final Assurance that software meets all functional, behavioral and performance requirements. Validation can be defined in many ways but a simple definition is that validation succeeds when software Function in a manner that can be reasonably by the customer.

1. Validation test criteria
2. Configuration review
3. Alpha and Beta testing (conducted by end user)

7.3 MODULE LEVEL TESTING

Module Testing is done using the test cases prepared earlier. Module is defined during the time of design.

7.4 INTEGRATION & SYSTEM TESTING

Integration testing is used to verify the combining of the software modules. Integration testing addresses the issues associated with the dual problems of verification and program construction. System testing is used to verify, whether the developed system meets the requirements. System testing is actually a series different test whose primary purpose is to full exercise the computer base system. Where the software and other system elements are tested as whole. To test computer software, we spiral out along streamlines that broadens the scope of testing with each turn.

The last higher-order testing step falls outside the boundary of software Engineering and in to the broader context of computer system engineering. Software, once validated, must be combining with order system Elements (e.g. hardware, people, databases). System testing verifies that all the elements Mesh properly and that overall system function/performance is achieved.

1.Recovery Testing

2.Security Testing

3.Stress Testing

7.5 REGRESSION TESTING

Each modification in software impacts unmodified areas, which results serious injuries to that software. So, the process of re-testing for rectification of errors due to modification is known as regression testing.

Installation and Delivery: Installation and Delivery is the process of delivering the developed and tested software to the customer. Refer the support procedures.

Acceptance and Project Closure: Acceptance is the part of the project by which the customer accepts the product. This will be done as per the Project Closure, once the customer accepts the product, closure of the project is started. This includes metrics collection, PCD, etc.

CHAPTER 8

SOURCE CODE

8.1 SOURCE CODE

```
from __future__ import absolute_import, division, print_function
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import tensorflow as tf
import cv2
import matplotlib.pyplot as plt
import seaborn as sns
from PIL import Image
import os
print(os.listdir("../input/cell_images/cell_images"))
infected = os.listdir('../input/cell_images/cell_images/Parasitized/')
uninfected = os.listdir('../input/cell_images/cell_images/Uninfected/')
data = []
labels = []

for i in infected:
    try:
        image = cv2.imread("../input/cell_images/cell_images/Parasitized/"+i)
        image_array = Image.fromarray(image , 'RGB')
        resize_img = image_array.resize((50 , 50))
        rotated45 = resize_img.rotate(45)
        rotated75 = resize_img.rotate(75)
        blur = cv2.blur(np.array(resize_img) ,(10,10))
        data.append(np.array(resize_img))
        data.append(np.array(rotated45))
        data.append(np.array(rotated75))
        data.append(np.array(blur))
        labels.append(1)
        labels.append(1)
        labels.append(1)
        labels.append(1)

    except AttributeError:
        print('')

for u in uninfected:
    try:
        image = cv2.imread("../input/cell_images/cell_images/Uninfected/"+u)
        image_array = Image.fromarray(image , 'RGB')
        resize_img = image_array.resize((50 , 50))
        rotated45 = resize_img.rotate(45)
        rotated75 = resize_img.rotate(75)
        data.append(np.array(resize_img))
        data.append(np.array(rotated45))
        data.append(np.array(rotated75))
        labels.append(0)
        labels.append(0)
        labels.append(0)

    except AttributeError:
        print('')
cells = np.array(data)
labels = np.array(labels)
```

```
np.save('Cells' , cells)
np.save('Labels' , labels)
print('Cells : {} | labels : {}'.format(cells.shape , labels.shape))
plt.figure(1 , figsize = (15 , 9))
n = 0
for i in range(49):
    n += 1
    r = np.random.randint(0 , cells.shape[0] , 1)
    plt.subplot(7 , 7 , n)
    plt.subplots_adjust(hspace = 0.5 , wspace = 0.5)
    plt.imshow(cells[r[0]])
    plt.title('{} : {}'.format('Infected' if labels[r[0]] == 1 else 'Unifected' ,
                            labels[r[0]]) )
    plt.xticks([]) , plt.yticks([])

plt.show()
plt.figure(1, figsize = (15 , 7))
plt.subplot(1 , 2 , 1)
plt.imshow(cells[0])
plt.title('Infected Cell')
plt.xticks([]) , plt.yticks([])

plt.subplot(1 , 2 , 2)
plt.imshow(cells[60000])
plt.title('Uninfected Cell')
plt.xticks([]) , plt.yticks([])

plt.show()
n = np.arange(cells.shape[0])
```

```
np.random.shuffle(n)
cells = cells[n]
labels = labels[n]
cells = cells.astype(np.float32)
labels = labels.astype(np.int32)
cells = cells/255
from sklearn.model_selection import train_test_split

train_x , x , train_y , y = train_test_split(cells , labels ,
                                    test_size = 0.2 ,
                                    random_state = 111)

eval_x , test_x , eval_y , test_y = train_test_split(x , y ,
                                    test_size = 0.5 ,
                                    random_state = 111)
plt.figure(1 , figsize = (15 ,5))
n = 0
for z , j in zip([train_y , eval_y , test_y] , ['train labels','eval labels','test labels']):
    n += 1
    plt.subplot(1 , 3  , n)
    sns.countplot(x = z )
    plt.title(j)
plt.show()
print('train data shape {} ,eval data shape {} , test data shape
{}'.format(train_x.shape, eval_x.shape, test_x.shape))

tf.reset_default_graph()
def cnn_model_fn(features , labels , mode):
    input_layers = tf.reshape(features['x'] , [-1 , 50 , 50 ,3])
    conv1 = tf.layers.conv2d(
    inputs = input_layers ,
    filters = 50 ,
    kernel_size = [7 , 7],
    padding = 'same',
    activation = tf.nn.relu
    )

conv2 = tf.layers.conv2d(
    inputs = conv1,
    filters = 90,
    kernel_size = [3 , 3],
    padding = 'valid',
    activation = tf.nn.relu
    )

conv3 = tf.layers.conv2d(
    inputs = conv2 ,
    filters = 10,
    kernel_size = [5 , 5],
    padding = 'same',
    activation = tf.nn.relu
    )

pool1 = tf.layers.max_pooling2d(inputs = conv3 , pool_size = [2 , 2] ,
                                strides = 2 )
conv4 = tf.layers.conv2d(
    inputs = pool1 ,
    filters = 5,
    kernel_size = [3 , 3],
    padding = 'same',
    activation = tf.nn.relu
    )

pool2 = tf.layers.max_pooling2d(inputs = conv4 , pool_size = [2 , 2] ,
                                strides = 2 , padding = 'same')

pool2_flatten = tf.layers.flatten(pool2)
fc1 = tf.layers.dense(
    inputs = pool2_flatten,
    units = 2000,
    activation = tf.nn.relu
    )
fc2 = tf.layers.dense(
    inputs = fc1,
    units = 1000,
    activation = tf.nn.relu
    )
fc3 = tf.layers.dense(
    inputs = fc2 ,
    units = 500 ,
    activation = tf.nn.relu
    )
logits = tf.layers.dense(
    inputs = fc3 ,
    units = 2
    )

predictions = {
        'classes': tf.argmax(input = logits , axis = 1),
        'probabilities': tf.nn.softmax(logits , name = 'softmax_tensor')
    }

    if mode == tf.estimator.ModeKeys.PREDICT:
        return tf.estimator.EstimatorSpec(mode = mode ,
                                          predictions = predictions)

    loss = tf.losses.sparse_softmax_cross_entropy(labels = labels ,
                                                  logits = logits)

    if mode == tf.estimator.ModeKeys.TRAIN:
        optimizer = tf.train.GradientDescentOptimizer(learning_rate = 0.001)
        train_op = optimizer.minimize(loss = loss ,
                                      global_step = tf.train.get_global_step())

        return tf.estimator.EstimatorSpec(mode = mode ,
                                          loss = loss ,
                                          train_op = train_op
                                         )
    eval_metric_op = {'accuracy' : tf.metrics.accuracy(labels = labels ,
                                    predictions =  predictions['classes'])}

    return tf.estimator.EstimatorSpec(mode = mode ,
                                      loss = loss ,
                                      eval_metric_ops = eval_metric_op)
malaria_detector = tf.estimator.Estimator(model_fn = cnn_model_fn, model_dir
= '/tmp/modelchkpt')
tensors_to_log = {'probabilities':'softmax_tensor'}
logging_hook = tf.train.LoggingTensorHook(
    tensors = tensors_to_log , every_n_iter = 50
    )
train_input_fn = tf.estimator.inputs.numpy_input_fn(
    x = {'x': train_x},
    y = train_y,
    batch_size = 100 ,
    num_epochs = None ,
    shuffle = True
    )
malaria_detector.train(input_fn = train_input_fn , steps = 1 , hooks =
[logging_hook])
malaria_detector.train(input_fn = train_input_fn , steps = 10000)
eval_input_fn = tf.estimator.inputs.numpy_input_fn(
    x = {'x': eval_x},
    y = eval_y ,
    num_epochs = 1 ,
    shuffle = False
    )
eval_results = malaria_detector.evaluate(input_fn = eval_input_fn)
print(eval_results)
pred_input_fn = tf.estimator.inputs.numpy_input_fn(
    x = {'x' : test_x},
    y = test_y,
    num_epochs = 1,
    shuffle = False
    )

y_pred = malaria_detector.predict(input_fn = pred_input_fn)
classes = [p['classes'] for p in y_pred]
```

```
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score
print('{} \n{} \n{}'.format(confusion_matrix(test_y, classes),
classification_report(test_y , classes) , accuracy_score(test_y , classes)))
plt.figure(1 , figsize = (15 , 9))
n = 0
for i in range(49):
    n += 1
    r = np.random.randint( 0  , test_x.shape[0] , 1)
    plt.subplot(7 , 7 , n)
    plt.subplots_adjust(hspace = 0.5 , wspace = 0.5)
    plt.imshow(test_x[r[0]])
    plt.title('true {} : pred {}'.format(test_y[r[0]] , classes[r[0]]) )
    plt.xticks([]) , plt.yticks([])

plt.show()
```

CHAPTER 9

RESULTS AND DISCUSSION

9.1 RESULTS AND DISCUSSION

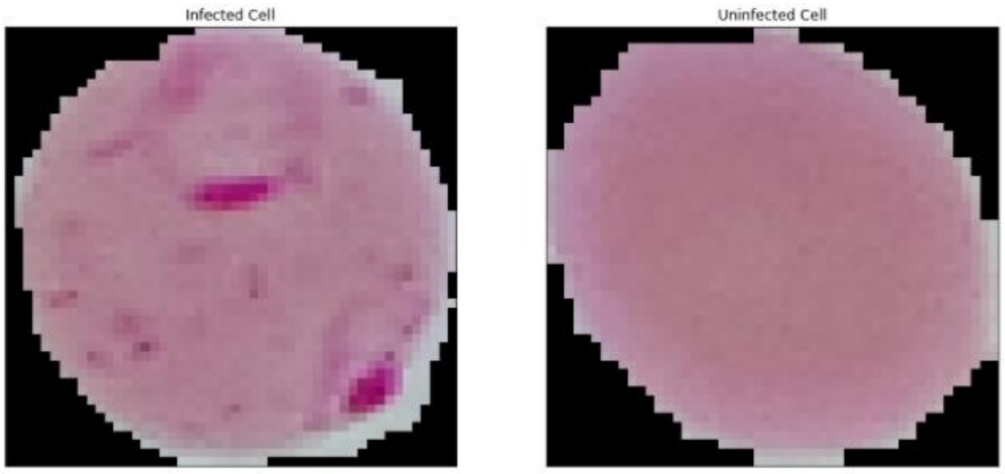

Fig 8.1: Sample of blood cell images of both healthy and infected

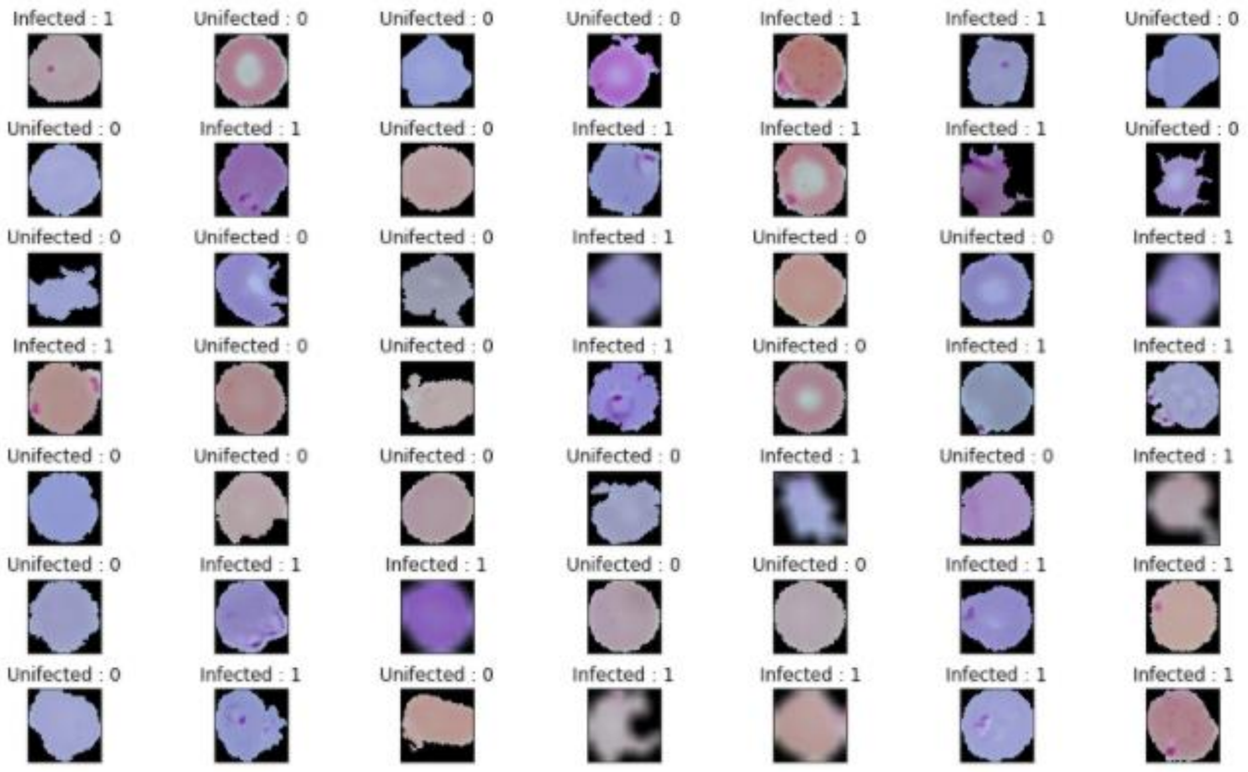

Fig 8.2: Infected and uninfected cell

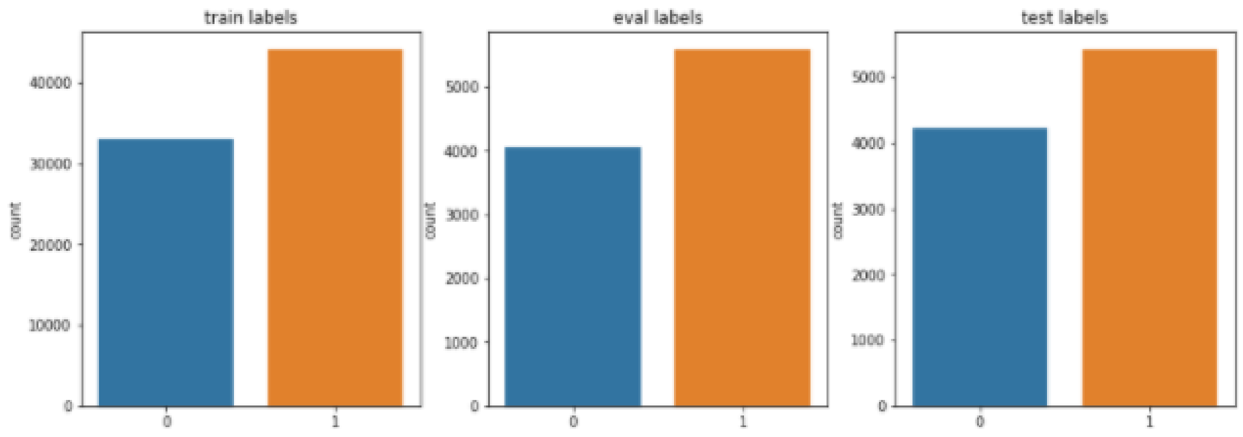

Fig 8.3: Dataset Split

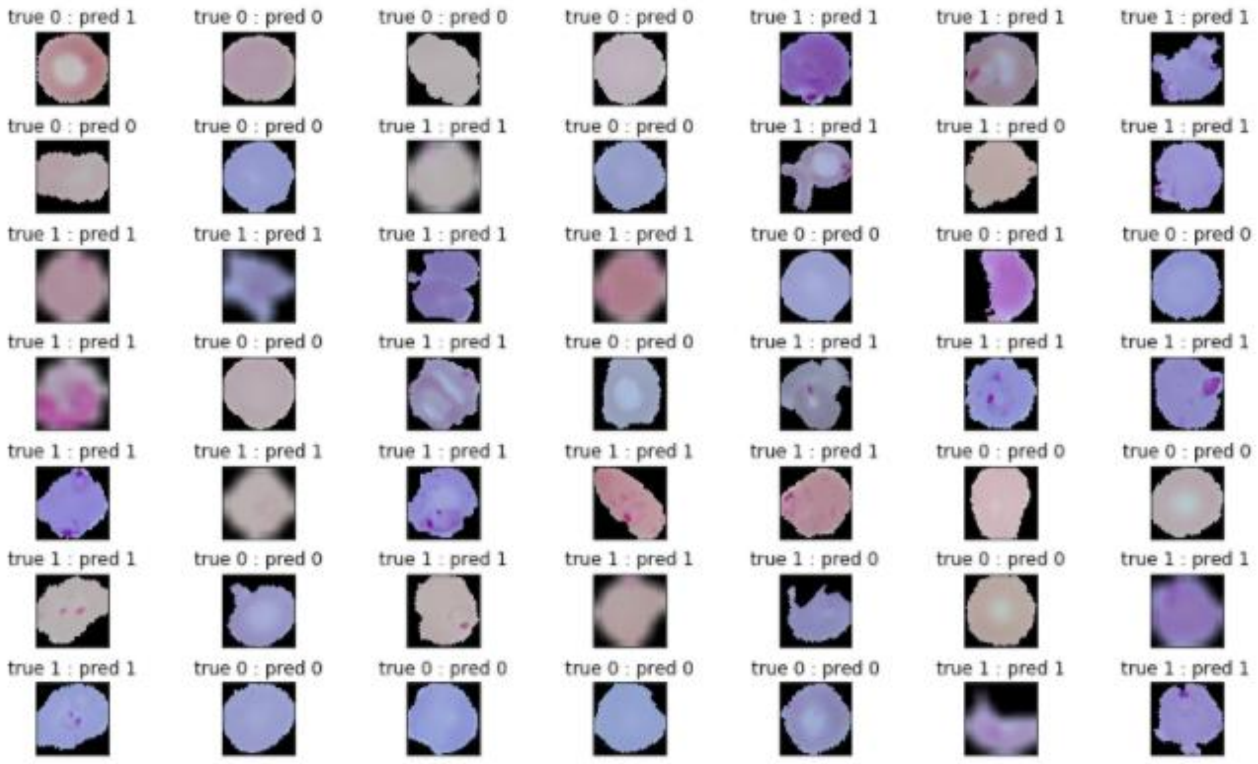

Fig 8.4: Prediction Outcome

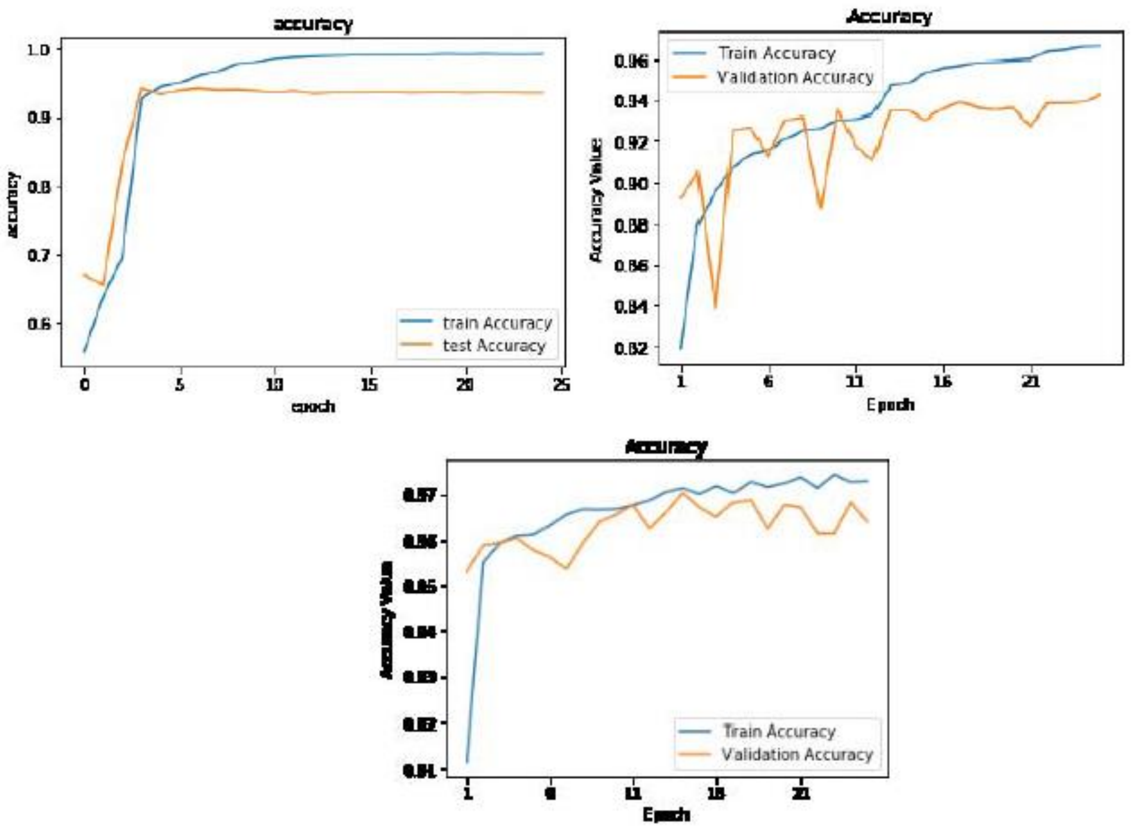

Fig 8.5: The Accuracy vs Epoch graph starts from the top left is the Basic CNN model, frozen CNN Model, and fine-tuned CNN model

The three convolutional neural networks. The fine-tuned convolutional network shows the highest accuracy among the three convolutional neural networks. The only disadvantage of the fine-tuned convolutional neural network is that it is a very cost-effective process and it takes more time. The confusion matrix is a performance measurement for machine learning classification. A confusion matrix is a two-dimensional matrix with four attributes true positive (TP), false positive (FP), false negative (FN), and true negative (TN). The confusion matrix is used to calculate the precision values, accuracy, recall, and F-measure.

T able 1: The confusion matrix of the basic CNN

Actual/Predicted	**Healthy**	**Malaria**
Healthy	3884	191
Malaria	225	3968

Table 1 shows that the basic convolutional neural network's confusion matrix states that there are 3884 cells are healthy and 3968 cells are unhealthy or infected but the accuracy is 94% which is not enough to predict the values.

T able 2: The confusion matrix of the Frozen CNN

Actual/Predicted	**Healthy**	**Malaria**
Healthy	3871	204
Malaria	212	3881

Table 2 shows that the frozen convolutional neural network's confusion matrix states that there are 3871 cells are healthy and 3881 cells are unhealthy or infected but the accuracy is 92% which is less than Basic CNN which cannot be used to predict the values.

T able 3: The confusion matrix of the Fine-Tuned CNN

Actual/Predicted	**Healthy**	**Malaria**
Healthy	4004	71
Malaria	260	3933

Table 3 shows that the fine-tuned convolutional network's confusion matrix states that there are 4004 cells are healthy and 3933 cells unhealthy or infected have a higher accuracy of 96% which will be used to predict accurate values.

Table 4: The calculation values of the three convolutional neural networks

CNN	**Acc**	**Class.**	**Sen**	**Pre**	**F1**	**Fb1**	**Spe**	**Fal**	**Mat**
Basic CNN	0.94	0.05	0.94	0.95	0.95	0.95	0.95	0.04	0.89
Frozen CNN	0.92	0.07	0.90	0.93	0.91	0.91	0.94	0.05	0.85
Fine-Tuned CNN	0.96	0.03	0.93	0.98	0.96	0.96	0.98	0.01	0.92

Table 4 shows that the calculation of different parameters for three different convolutional neural networks, where Acc means Accuracy, Class means Classification Error, Sen means Sensitivity, Pre means Precision, F1 means F1 Score, Fb means F_beta, Spe means Specificity, Fal means False positive rate and Mat means Mathew correlation.

CHAPTER 10

CONCLUSION & FUTURE ENHANCEMENTS

10.1 Conclusion

Malaria detected from the traditional method that is bringing the samples and analyzing cell growth requires more time. So, in the proposed work, a deep learning model has been constructed to predict Malaria with a high accuracy rate and low time duration. Three CNN models were constructed and identified as the highest accuracy model. The Fine-Tuned CNN provided a high accuracy rate compared to the other CNN models. The future work will be, working on disease detection like pneumonia, breast cancer using CNN, and planning for the detection of COVID-19 smears in the lungs of the human body.

10.2 Future Enhancements:

Data Augmentation Explore techniques like image rotation, flipping, and adding noise to artificially increase dataset size and improve model robustness. Multi-Species Detection Train the model to differentiate between various Plasmodium species causing malaria, enabling targeted treatment. Mobile Integration Develop a mobile app for on-site diagnosis in resource-limited regions, making malaria testing more accessible. Explainable AI Implement techniques to understand the model's decision-making process, fostering trust and transparency in diagnoses. Integration with Medical Devices to Explore integrating the model with automated blood analysis systems for seamless and efficient workflow**.**

REFERENCES

[1] Shuying Liu and Weihong Deng., “Very deep convolutional neural network based image classification using small training sample size” 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR).

[2] NimaHatami., Yann Gavet and Johan Debayale., “Classification of time-series images using deep convolutional neural network” Tenth International Conference on Machine Vision (ICMV 2017) Vol.10696.

[3] Nicholas E. Ross., Charles J. Pitchard., David M. Rubin and Adriano G. Duse “Automated image processing method for the diagnosis and classification of malaria on thin blood smears” IEEE Vol.44.

[4] K.M. Khatri., V.R. Ratnaparkhe., S.S. Agarwal and A.S. Bhalchandra “Image processing approach for malarial parasite identification” International Journal of computer Application (IJCA).

[5] Vishnu V. Makkapati and Raghuveer M. Rao., “Segmentation of malarial parasites in peripheral blood smear images” IEEE International Conference on Acoustics, Speech and Signal Processing.

[6] CorentinDallet., Saumya Kareem and Izzet Kale., “Real time blood image processing application for malaria diagnosis using mobile phones” IEEE International Symposium on Circuits and Systems (ISCAS).

[7] Zhaoui Liang., Andrew Powell., IlkerErsoy.,Mahdieh Poostchi., Kamolraut Silmaut and Kannapan Palani., “CNN-based image analysis for malaria diagnosis” IEEE International Conference on Bioinformatics and Biomedicine (BIBM).

[8] Yuhang Dong., Zhuocheng Jiang., Hongda Shen., W. David Pan., Lance A. Williams., Vishnu V. B. Reddy., “Evaluations of deep convolutional neural network for automatic identification of malaria infected cells” IEEE EMBS International Conference on Biomedical & Health Informatics (BHI).

[9] Gopalakrishna Pillai Gopakumar., “Convolutional neural network-based malaria diagnosis from stack of blood smear images acquired using custom-built slide scanner” Journal of Biophontics, Vol. 11 No.3.

[10] S.Revathy., B.Bharathi., P.Jeyanthi., M.Ramesh.,"Chronic Kidney Disease Prediction using Machine Learning Models", International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249 – 8958, Vol.9, Issue-1, October 2019.