DESIGN AND IMPLEMENTATION OF AN AUTOMATIC ACUTE LYMPHOBLASTIC LEUKEMIA DIAGNOSIS SYSTEM

BY

ENOCH ADURAGBEMI OLUTUNMIDA

MATRI C NO : - 1 6 0 1 0 3 0 1 0 1 7

SUBMITTED TO

THE DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS, COLLEGE OF BASIC AND APPLIED SCIENCES, MOUNTAIN TOP UNIVERSITY, IBAFO, NIGERIA

IN FULFILMENT OF THE REQUIREMENTS FOR THE AWARD DEGREE OF BACHELOR OF SCIENCE (B.SC) IN COMPUTER SCIENCE

NOVEMBER 2020

Certification

This is to certify that this project, Design and Implementation of an Automatic Acute Lymphoblastic Leukemia Diagnosis System was carried out by me, Enoch Aduragbemi Olutunmida (Matriculation Number: 16010301017) and duly supervised by Dr. F.A Kasali.

Dr. Kasali F.A. (Supervisor)	Date
Dr. Akinyemi I.O. (Head of Department)	Date
Professor Olalusi A.P. (Dean of College)	Date

Dedication

This project work is dedicated to the giver of life and wisdom: The Almighty God.

Acknowledgement

Firstly, I would like to give all glory to God Almighty for helping me through the process of this project work.

I specially appreciate my Supervisor Dr. Kasali F.A. who took keen interest in my project work and guided me all along.

I cannot express enough thanks to the Head of Department Computer Science and Mathematics Dr. I.O. Akinyemi for his fatherly role, constant encouragement and guidance. A big thank you to all academic and non-academic staff of the Department of Computer Science and Mathematics for their continued support and impartation of knowledge and positive values. I extend my sincere appreciation to Mountain Top University for the adequate learning facilities and opportunities provided. I say God bless you richly.

My special gratitude goes to Dr. D.K. Olukoya, my parents and my siblings for their moral and financial support. The countless times you gave encouraging words and prayer support to help me get through difficult times in this project were not taken for granted.

Table of Contents

Title Page

List of Figures

Abstract

Automatic Leukemia Diagnosis is a computer-aided approach to diagnosing leukemia. This work focuses on the diagnosis of Acute Lymphoblastic Leukemia (ALL) which accounts for 12% of all childhood and adult leukemias diagnosed in developed countries and for nearly 60% of those diagnosed in persons under 20 years of age (Pui, 2011). The relevance of this work is to find a way to reduce the over-reliance on medical specialist for the diagnosis of ALL. Machine Learning and Deep Learning algorithms are the current trends adopted for the purpose of medical diagnosis involving image analysis. This approach has been adopted by many other researchers for the purpose of diagnosing breast cancer (Poorti & Neetu, 2019), and prostate cancer (Janney, Christilda, Mary & Haritha, 2017), amongst others.

This project work would be achieved using Python 3.7. A number of Machine Learning models will be compared to find the best performing algorithm. The best performing algorithm will be implemented as an API in Python (Flask) and then hosted using Google Cloud Platform (GCP). The

hosted API will then be consumed in an Android App for easy usage and diagnosis of ALL in medical facilities.

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

Decades of research have focused on the question of if computers can aid human in carrying out tasks that would rather have taken ages and achieve them within a shorter time, while also prioritizing efficiency. It has been widely assumed that concepts such as artificial intelligence (AI) and machine learning (ML) would serve as a foundation for creating a world we all deserve (Zuckerberg, n.d.). A world where we can avoid fatal accidents through the introduction of autonomous driving vehicles, handle the risk of heart-attack using smartwatches and the use of computer vision for early diagnosis of terminal illness such as leukemia. Leukemia could be explained to be a malignant disease involving the bone marrows, which in this case, produces an excessive number of leucocytes (Burgun et al., 2005).

The French-American-British (FAB) classification model categories acute leukaemia into two distinct types: acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML). In developed countries, ALL is said to have accounted for 12% of leukemia in children and adults, it was also reported to have higher prevalence of up to 60% in persons under 20 years. It is a very common type of cancer in children (25% of all cases) and elderly patients (Pui, 2011).

Of all the numerous indicators used in evaluating whether a patient is positive for leukemia or not, the presence of lymphoblasts in the blood sample is a sure and confident indicator for diagnosis. In this manner, tallying of lymphoblasts is the most trusted method for diagnosing leukemia (Gayathri & Jyothi, 2018). While the present approaches to diagnosis leukemia still appear to be effective and accurate, some of the challenges faced include time-consuming procedures and error-prone techniques due to human interference.

The relevance of machine learning algorithms for making the process of leukemia diagnosis more efficient and accurate has been proven by its application in the process of diagnosing other cancer ailments such as breast cancer (Poorti & Neetu, 2019), and prostate cancer (Janney, Christilda, Mary & Haritha, 2017), amongst others. Machine Learning uses mathematical equations to study the tiniest insights or patterns in the image data (which the human senses may ordinarily not be able to detect). These patterns are obtained by training models on several datasets in other to help them make an intelligent diagnosis (Bishop, 2006). Hence, the main focus of this work is to build a system for diagnosing the presence the of lymphoblast in the blood cell, while also carrying out a comparative analysis of various algorithms for diagnosing acute lymphoblastic leukemia (ALL) using blood sample images.

1.2 Statement of Problem

In recent times, there's been a lot of application of deep learning techniques and algorithms in problem solving across various domains, especially the medical field. This is because deep learning algorithms have the ability to draw patterns and insight from data (Najafabadi et al., 2015). This has had a high positive impact in medical domain especially in disease diagnosis and prediction. Cancer is a prevalent cankerworm that continues to eat deep into the health and wealth of many nations as ascertained by the World health Organization (WHO, 2018). It was reported that the disease alone accounted for an estimated 9.6 million deaths in 2018 globally which means that 1 in 6 deaths was due to cancer that year alone.

Leukemia is a type of cancer that affects the white blood cells of its patients, and from 2011 to 2015 – the most recent 5 years for which data are available, leukemia alone represented 39.6 percent of all malignant growth types in children, teenagers and young adults younger than 20 years (Facts, 2018). In the diagnosis of the disease, complete blood count (CBC) is normally used, however some of the problems associated with this procedure includes high cost of diagnosis and machineries, time consuming procedures and over-reliance on experts which are rarely available in developing

countries. The end goal of this research work is to help reduce the time, cost and over reliance on experts for the diagnosis of leukemia and other related ailments.

1.3 Aim and Objectives

The aim of this work is to develop a system to diagnose acute lymphoblastic leukemia from blood sample images. The specific objectives are to:

1. Collect and prepare dataset for image processing and classification.
2. Train and design a model for leukemia cell segmentation.
3. Compare result of the diagnosis when directly classifying images against how its performance when using image processing techniques to extract features and classify them.
1. Implement the proposed design as an android-based system for diagnosing leukemia by consuming the machine learning model as a RESTful API.

1.4 Methodology

1. Performing image segmentation, enhancement and feature extraction on ALL-IDB1 cell images.

2. Analyzing the performance of some image classification algorithms on ALL-IDB2 dataset in comparison to the use of image processing to extract features form ALLIDB1 dataset which are eventually classified.

3.i. Systematically highlighting how the leukemia diagnosis system would work.

ii. Highlighting all resources and tools adopted to ensure the best performance of the system (like the cloud hosting platform used).

iii. Highlighting the model used in the framework based on evaluation of the relevant classification algorithms implemented using Python 3.7.

iv. Discuss the chosen approach to make the framework easily usable and adaptable to the present healthcare working environment; such as making it operate as an android-based leukemia diagnosis framework.

1.5 Significance of the Study

This study will redound to the benefit of society considering that providing a quicker and less expensive means of diagnosing diseases will play an important role by giving as many people as possible access to quality diagnosis of leukemia. The estimated 9.6 million deaths leukemia accounted for in 2018 globally (WHO, 2018) justifies the need for a more effective, reliable and cheaper approach to aid early discovery and diagnosis of the ailment. Thus, medical facilities that apply our technology derived from the result of this study will be able to diagnose leukemia faster, cheaper and without much reliance on experts in the field.

1.6 Scope of the Study

This study is for the creation of a device to aid leukemia detection. There will be an app for Android based systems through which the medical attendant will take an image of the blood smear (using microscopic lenses) to be used for the leukemia diagnosis. In other to ensure the quality of service, various image processing & classification algorithms will be compared using certain metrics like accuracy level, confusion matrix, etc.

1.7 Definition of Terms

- ***ALL: -*** **Acute Lymphoblastic Leukemia** is a malignant disease of the bone marrow.
- ***WBC: -*** **White Blood Cells** are components of the human blood that help the human body fight and defend against infection and other diseases.

- ***RBC:* - Red Blood Cells** are the component of the human blood that help to transport oxygen and carbon dioxide to and from the tissues. They are also responsible for the color of the blood.
- ***Confusion matrix:* -** A confusion matrix is an indicator (table) representing the performance of a model on a dataset.
- ***Algorithms:* -** A group of processes, rules or logical decisions upon which a computer system performs calculations or other problem-solving operations.
- ***Diagnosis:* -** the process of examining symptoms with the goal of identifying the nature of an illness or medical challenge.
- ***Machine Learning:* -** the study of computer algorithms and mathematical functions that learn patterns in data and improve automatically through experience gained.
- ***Artificial Intelligence:* -** using machines to simulate human intelligence by designing and programming them to mimic humans.
- ***Deep Learning:* -** a subset of machine learning that imitates the behaviour of humans in processing and patterning data for decision making – unsupervised learning. It imitates the way humans learn certain types of knowledge.
- ***Primary data:* -** a type of data that is collected by researchers directly from the main source through interviews, surveys and experiments.
- ***Secondary data:* -** a type of data that has already been collected through primary sources and made readily available for researchers to use.
- ***Python programming language:* -** is a general purpose, high-level programming language created by *Guido van Rossum*.

1.8 Organization of Work

This work is structured in such a way that, Chapter 1 introduces the concept and overview of the topic, Chapter 2 discusses recent related works as well as giving sufficient understanding on the

various terminologies and concepts that surround the work, Chapter 3 discusses the methods used in achieving the said objectives laid down in chapter 1, the various image classification techniques compared and the precautions taken during the cause of implementing those techniques. Chapter 4 talks about the implementation of the proposed system, while Chapter 5 discusses the summary, limitations experienced during the project, recommendation and conclusion.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

The diagnosis of leukemia much of the time follows a routine blood test that results in abnormal or irregular platelet count. In a case where the doctor suspects leukemia, samples of the bone marrow and blood are taken to examine the cell shape. This collected sample may likewise be sent to the pathology laboratory to distinguish proteins situated on a superficial level and chromosomal changes. This group of information serve important purposes in the process of diagnosing patients.

This work makes reference to other literatures around the idea of diagnosing leukemia and related ailments with the help of machine learning algorithms. While some studies suggested using image processing techniques to extract shape based features from the cell image such as roundness, standard deviation, and so on to diagnose the presence of malignant cells in a blood sample (Modi et al., 2016). Other studies took a slightly different approach by just getting the shape and texture of the blood sample images and then passing that into a classifier which would determine whether it is malignant or not (Gayathri & Jyothi, 2018).

Several techniques have been suggested for detecting blast cells and for diagnosing leukemia. (Ahmed et al., 2019) proposed an approach to diagnosis leukemia (and its subtypes) from microscopic blood cell images using convolutional neural networks (CNN). They made use of two publicly available leukemia data sources: ALL-IDB and American Society of Hematology (ASH) Image Bank in carrying out their research. CNN however requires large dataset to avoid memorization and perform optimally, so they performed data augmentation on the training dataset to synthetically increase the size of the dataset.

CNN performed well with 88% accuracy for the binary classification of one leukemia type and 81% accuracy for classifying all leukemia subtypes (Ahmed et al., 2019). Other works propose utilizing image processing techniques like automatic Otsu's threshold segmentation method, image enhancement and arithmetic for WBC segmentation, and KNN classifier to characterize blast cells from typical lymphocyte cells (Chatarwad et al., 2018). (Chatarwad et al., 2018) used Otsu's thresholding method for conversion of grayscale image into binary image. To remove noise, they applied image filtering using median filter and then used *sobel* operator for edge detection.

(Modi et al., 2016) also proposed methods for performing different image processing operations on leukemia detected images such as reducing image quality from RGB to gray level, thresholding methods for converting images into binary forms, area opening to remove connected component, dilation to add pixels to boundary of objects and erosion to remove the pixel on object boundaries. After detecting the boundary of object, they performed hole filling operations to detect perfect cells. By the end of this processes, they were able to detect boundaries around the cells using operators like *sobel*, *prewitt*, *canny*, etc. They computed shape-based features like *major axis, minor axis, area, perimeter, standard deviation, radius* and *roundness* using certain formulas:

Radius = (Major axis + Minor axis) / 4

Roundness = $(4*PI*area) / perimeter^2$

Standard deviation = $((Major\ axis - X)^2 + (Minor\ axis - X)^2)^{1/2}$

Where X = (Major axis + Minor axis) / 2

Using the value of major axis and minor axis, the number of overlapping cells and non-overlapping cells were detected and in-turn the number of overlapping cells and non-overlapping cells were used to compute the total number of malignant cells present.

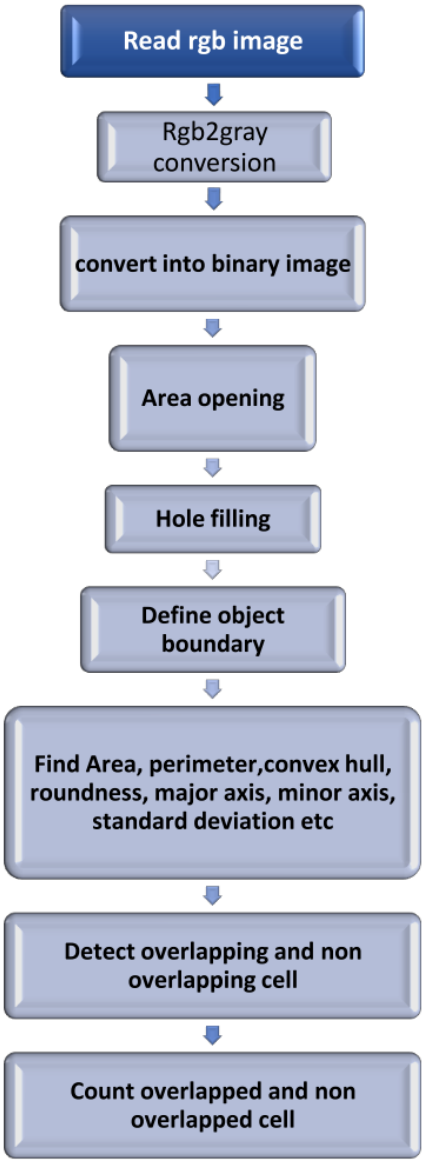


Figure 2.1 Algorithm to count the cells (Source: Modi et al., 2016)

2.2 Dimensionality Reduction

Dimensionality reduction involves the application of data encoding or transformation to obtain a reduced or compressed representation of the original data (Data Mining: Concepts and Techniques Second Edition, 2013 pg77). A major reason for dimensionality reduction is "degree of freedom" – the creation of simpler structures in a machine learning model, and according to (Machine Learning : A Probabilistic perspective, 2012 pg11) "when dealing with high dimensional data, it is often

useful to reduce the dimensionality by projecting the data to a lower dimensional subspace which captures the *essence* of the data.”.

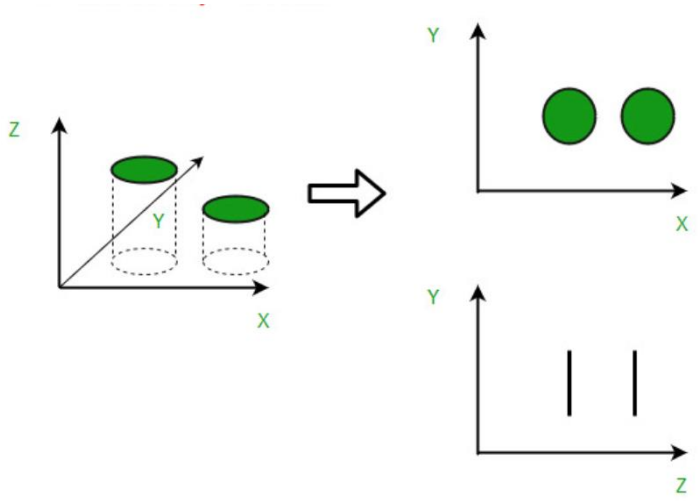


Figure 2.2 Diagrammatic representation of dimensionality reduction of image data 3D to 2D

2.3 Otsu’s Thresholding Method

Many works have referenced Otsu’s thresholding method to be the best method for *thresholding*. In some works like (Modi et al., 2016) – “Leukemia Detection using Digital Image Processing Techniques Leukemia Detection using Digital Image Processing Techniques” and (Joshi, Karode & Suralkar, 2013) – “White blood cells segmentation and classification to detect acute leukemia” , automatic Otsu’s Thresholding was proposed for blood cell segmentation before carrying out image enhancement . In the later work (Joshi, Karode & Suralkar, 2013), Otsu’s thresholding method was used to successfully segment the blood cell images, after which a K-Nearest Neighbor classifier was utilized to classify the blast cells – hence providing a way to distinguish them from normal lymphocyte cells. Otsu’s thresholding method is a data-driven way for adaptively finding the optimal threshold to distinguish two-class data.

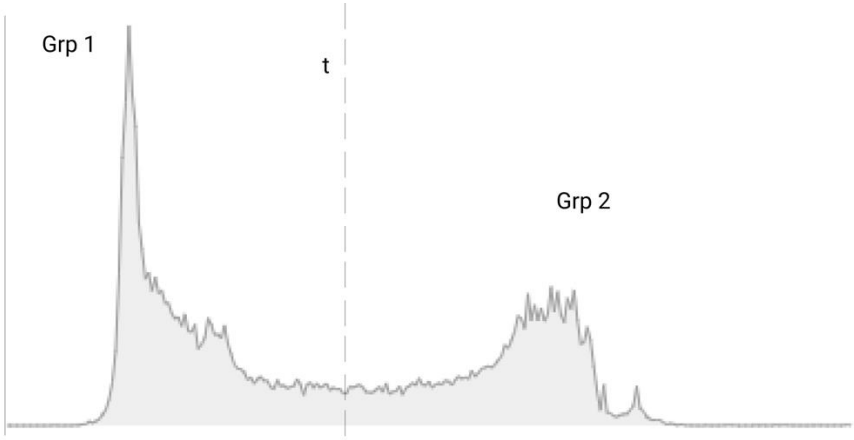


Figure 2.3 Bimodal histogram representation of a grayscale image

With reference to the above image, Otsu's thresholding method (also called automatic thresholding) finds the threshold *t* that minimizes the weighted sum of *within-group variances* for the two groups that result from separating the gray tones at value *t*.

The formula $\sigma_\omega{}^2(t) = q_{1(t)}\sigma_1{}^2(t) + q_{2(t)}\sigma_2{}^2(t)$ could be used to represent the Otsu's thresholding method, where *t* represents the threshold.

2.4 Identification and Classification of Leucocytes

(Gayathri & Jyothi, 2018) proposed an approach for identifying and classifying leucocytes. Their work revolved around developing an automated and accurate method to find whether a blood sample image is leukemia affected or not. To achieve their objective, they carried out certain procedures such as performing *segmentation on the input image, image cleaning, extracting features from the image, and finally using a classifier to classify the extracted features.* Beyond just building a model to classify leucocytes, they went further ahead to compare several classification models such as SVM, ANN, and CNN in other to find the best method possible for the classification of leucocytes. They built their system such that in addition to diagnosing leukemia, it is also able to detect which type of leukemia in particular i.e. ALL, AML, etc.

2.5 Segmentation and Morphological Image Processing

Segmentation is a very potent image processing technique in medical image analysis and in the diagnosis of various ailments like cancer. Segmentation has been used in detecting lumps and diagnosing breast cancer (Guzmán-Cabrera et al., 2013). (Guzmán-Cabrera et al., 2013) suggested an approach to distinguish the presence of breast cancer mass in mammograms. They utilized morphological operators for segmentation and clustering for clear identification of abnormalities such as microcalcifications. Their proposed algorithm was tried over more than a few images taken from the digital database for screening mammography for cancer examination and diagnosis. One of the current methods for leukemia diagnosis, Complete Blood Count (CBC) would usually require a medical specialist who counts the malignant cells and sort of "take stock" of the various components present in the blood sample (Rathee, 2013). To computationally simulate the Complete Blood Count (CBC) method of diagnosing leukemia, (Scotti, 2007) suggested an approach to automatically detect and count the blast cells present in the blood image (using image segmentation and some other morphological operations).

2.6 Classification Models and Classifiers

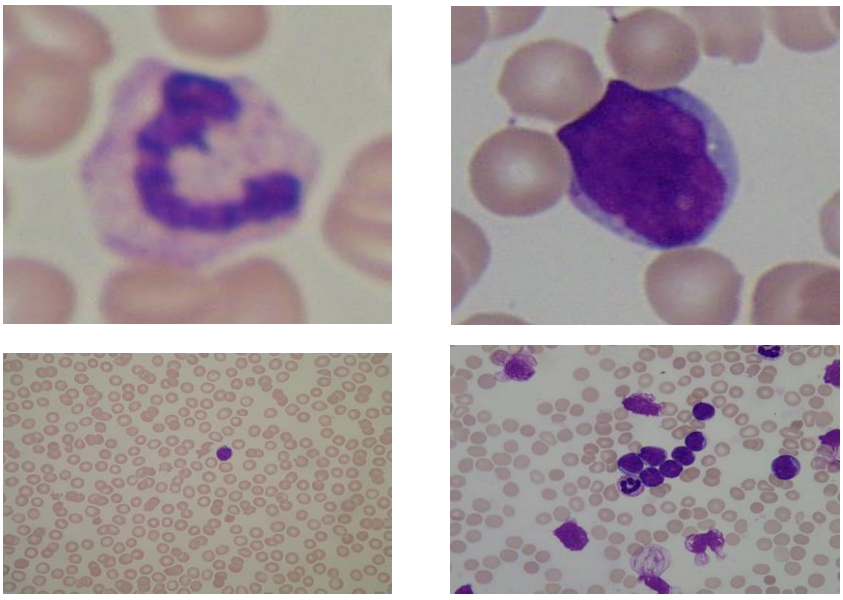
Classification is simply a process of categorizing a given set of data into classes (Data Mining: Concepts and Techniques Second Edition, 2013 pg24). This cycle of order can be performed on both structured and unstructured data. Other terminologies such as *targets, labels, or categories* can be used interchangeably with *classes* to communicate the same meaning. Classification can be either *binary* – involving just two possible outcomes, or *multiclass* – having more than two possible outcomes / labels. Since the classifier helps to determine whether a cell is malignant or not, the classifiers used in (Gayathri & Jyothi, 2018) are binary.

A ***Classifier*** is an algorithm used to map input data specific to a *category or label*.

2.7 Existing Methods for Leukemia Diagnosis

According to the Canadian Cancer Association, diagnosing leukemia usually begins with a medical history and physical examination. Some other commonly used techniques for ruling out or diagnosing leukemia are complete blood count (CBC), bone marrow aspiration, cytogenetic analysis, lymph node biopsy, lumbar puncture and Immunohistochemistry.

Complete blood count (CBC) involves taking the blood and checking under the microscope for the quantity of RBCs, WBCs and platelets. Leukemia can cause anomalous blood cell counts. Immature blood cells (called leukemia cells, or blasts) are not regularly found in the blood, so specialists will speculate leukemia if there are blasts or if blood cells do not look normal. Bone marrow aspiration on the other hand requires expulsion of the bone marrow with the assistance of a needle from the breastbone and observing the removed sample under a microscope to search for unusual cells.



2.8 Digital Image Processing Techniques

Image processing is a technique adopted for improving raw image data. Image processing holds a ton of significance in different fields of study, for example, remote sensing, textiles, material science, military, entertainment world, document processing, graphic arts, medical imaging, etc.

Image processing can be analog or digital, while analog image processing rely on using electrical means for altering images, digital image processing requires digital computers to process the images. Digital image processing can be defined as subjecting numerical representations of objects to a series of operations in order to obtain a desired result (Ravi & Ashokkumar, 2017).

(Ravi & Ashokkumar, 2017) in their work on Analysis of various image processing techniques, gave a classification of image processing techniques – Image representation, Image preprocessing, Image enhancement, Image analysis, and Image compression.

2.8.1 Image Representation

While creating a digital image, continuous sensed data is converted into digital forms, and this process references the relevance of concepts like Sampling – digitizing the co-ordinate values, and Quantization – digitizing the amplitude value. Sampling & Quantization are of relevance given that the sampling rates is a determinant factor of the spatial resolution of the digitized image and the quantization level determines the color depth.

Among a rundown of potential factors, the elements that essentially decide image quality are:

- Spatial resolution: constrained by spatial testing, and;
- Color depth: controlled by number of colors or grey levels designated for every pixel.

Manipulating any of these factors would in some way have an effect on the size of the image file. On this note, an image can be rightly defined as a 2D rectilinear array of pixels, having a fixed number of samples.

The figure below serves as a diagrammatic representation of how digital images appear – as a 2D array. Each of the elements in the matrix represents a pixel in the image. 0 represents black and 255 represents white. Other values within the 0 – 255 range represent the various combination of colors within the image.

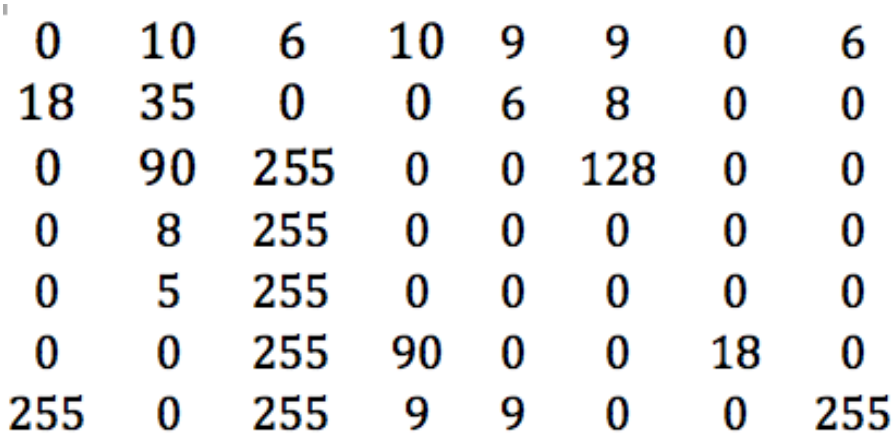


Figure 2.5 Diagrammatic representation of how digital images appear

2.8.2 Image Preprocessing

Image preprocessing (also known as data cleaning) describes a set of operations (feature engineering) carried out on images at the lowest level of abstraction to improve the image quality.

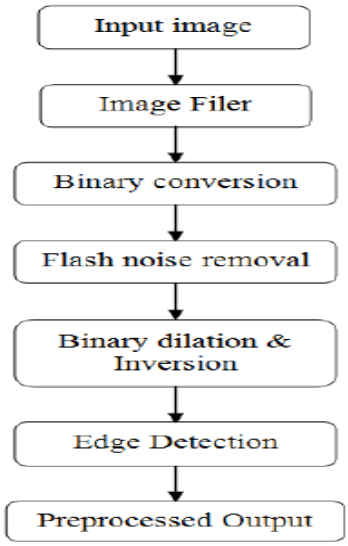


Figure 2.6 Block diagram of image processing stages

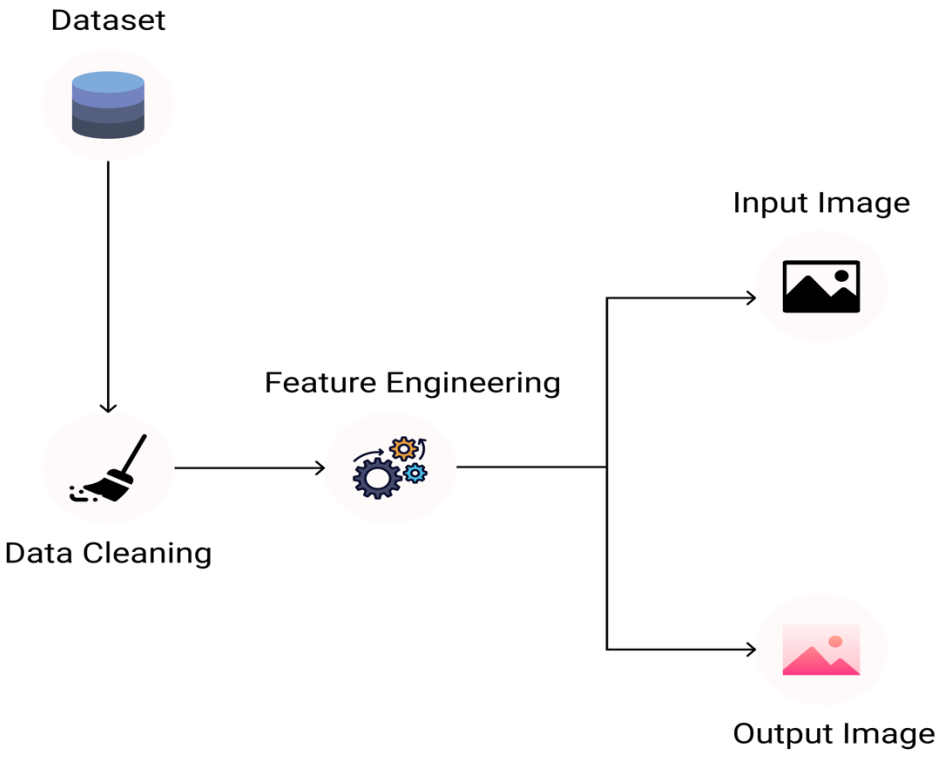


Figure 2.7 Diagrammatic representation of image preprocessing

2.8.3 Image Enhancement

Image Enhancement is the way toward adjusting digital images in other to make them more reasonable for a necessary assignment. Image enhancement can be either qualitative or quantitative. Qualitative image enhancement is used to make the image quality more appealing (look better) while Quantitative image enhancement is used to modify certain information within an image.

There are two types of Image Enhancement – Spatial domain and Frequency domain. Spatial domain has to do with the pixels of the images, and the Frequency domain pays more attention to the manipulation of the orthogonal transform in the image rather than the image itself.

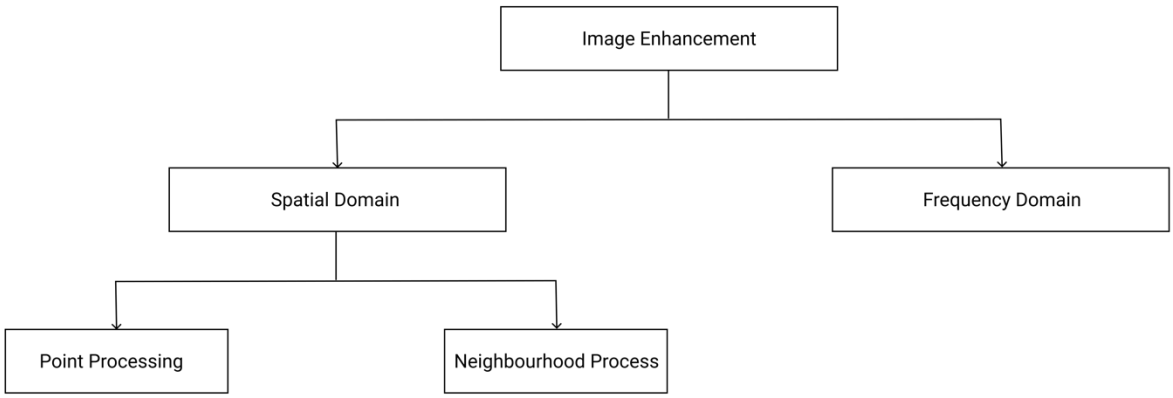


Figure 2.8 Types of image enhancement

2.8.4 Image Analysis

Image Analysis is the process of using a digital medium to recognize and study the attributes within an image. Over the years image analysis has been used in various fields like age prediction, extracting emotions from images, facial recognition, auto-caption from images and barcode reading. Image analysis adopts image processing techniques to extract meaningful information from images.

2.8.5 Image Compression

Image compression is the process of encoding or converting an image file in such a way that it requires less space than the original file (Techopedia, n.d.). Simply put, image compression is a compression technique applied to digital images in other to reduce their memory consumption and computational complexity. Compression can either be lossy or lossless. In lossy compression redundant data is removed in compression and added during decompression – i.e. some data is lost, while in lossless compression the data after compression and decompression are the same – i.e. no data is lost.

The model for image compression follows the process;

a. Input image (as a 2D array).

b. Encoding Image: this process encompasses the activities of the mapper, quantizer and symbol coder. The *mapper* reduces the spatial or temporal redundancy, the *quantizer* reduces the accuracy of the mapper's output (by rounding-down the value from the mapper), and the *symbol coder* generates fixed or variable length codes for the quantizer's output.

c. Decoding Image: this process involves the *symbol decoder* and *inverse mapper*, and;

d. Getting the compressed image as output.

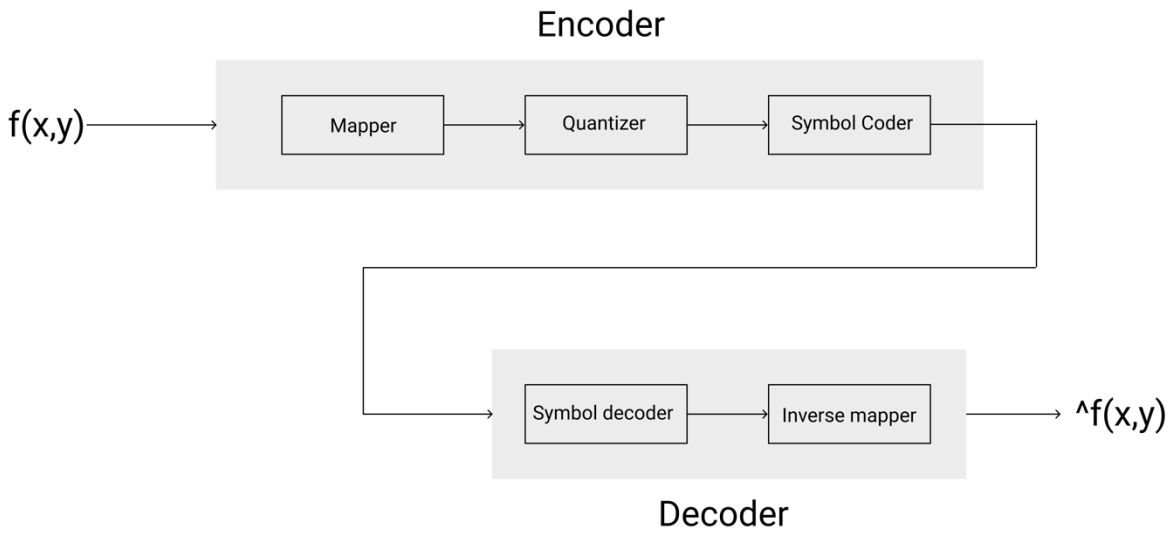


Figure 2.9 Image compression model

2.8.6 Image Processing and Medical Imaging

Image Processing has been the foundational text for the study of digital image processing. (Gonzalez & Woods, 2018) emphasizes that digital image processing is the use of a digital computer to process digital images through an algorithm. Generally, image processing consist of several stages: image acquisition, image preprocessing and analysis, manipulation and image output. (Gonzalez et al., 2009, p. 2) in their book titled *Digital Image Processing Using MATLAB*, gave an interesting and rather unconventional definition for *digital image processing*. They defined an image as a two-dimensional function, f(x, y), where x and y are spatial coordinates, and the amplitude f at any pair of coordinates (x, y) is called the intensity or gray-level of the image at that point. They eventually went ahead to defined *digital image processing* as a when x, y, and the amplitude values f are all finite discrete quantities.

Some other researching who have made certain contributions in the field of image processing for medical diagnosis include (Zadeh, Haddadnia & Janianpour, 2013) & (Supardi, Masher, Harun,

Bakri & Hassan, 2012). While the former (Zadeh, Haddadnia & Janianpour, 2013) did some work on the recognition and clarification of cancer cells with image processing in Lab VIEW. Lab VIEW used an image analysis approach for automated detection, preprocessing (smoothing, enhancement, segmentation, feature extraction morphological and calorimetric) and then detection and classification of particular cells, the later (Supardi, Masher, Harun, Bakri & Hassan, 2012) presented a study on classifying acute leukemia into two major forms which are ALL and AML by using K-NN. 12 primary highlights that speak to size, color and shape were extricated from the blood images. In other to find suitable parameters for classifying the blasts, a range of k values and distance metrics were tested.

A few other cases in which Image processing techniques were adopted for medical diagnosis include (Arena, Basile, Bucolo & Fortuna, 2018) in their work on Image Processing for Medical Diagnosis using Cellular Neural Networks. Though carried out almost two decades ago – when Deep Learning Algorithms were yet to hold firmer grounds, they were able to infer correctly how much potency Image Processing would hold in respect to medical diagnosis. They highlighted Image Processing as a very potent phase for improving the accuracy of the diagnosis procedure. They adopted the use of Cellular Neural Networks to process diagnostic images, like: Magnetic Resonance Imaging, Computed Tomography, and fluorescent cDNA microarray images. (Jindal, Gupta & Goyal, 2019) also emphasized the use of digital image processing techniques in the detection of cataract – one of the leading causes of blindness, especially among old people.

Several studies taken together have suggested various image processing procedures to achieve the common goal – leukemia diagnosis, it however remains an open question whether there could be an easier, more efficient and straightforward approach to diagnosing the ailment. Therefore, in this work we will be suggesting other techniques and also building upon some already laid down techniques by comparing more algorithms.

CHAPTER THREE

METHODOLOGY

3.0 Introduction

Research is a deliberate, formal, thorough and exact cycle employed to acquire answers for an issue or to find and decipher new facts and relationships (Waltz and Bansell, 1981). It is a quest for trust

with the assistance of study, perception, correlation and examination, the quest for knowledge through an unbiased and deliberate strategy for discovering solutions to a problem (Kothari, 2006).

The research methodology is a section containing explanations and clarifications of what was done and how it was done. It is a systematic description of how the initially stated objectives of the projects were achieved by adopting various problem-solving techniques. It gives an understanding of how the data used were collected, how they were analyzed, tools and material used in the research process, and rationales for adopting such approaches.

3.1 The Dataset

In this work secondary data from ALL-DB1 and ALL-IDB2 will be used. The datasets consist of images captured with a Canon Powershot G5 camera. The images have 24bit color depth and resolution of 2592 x 1944. The images contain about 39,000 blood elements with lymphocytes labelled by expert oncologists.

3.2 Proposed Work

In the proposed methodology the blood smear would undergo a number of steps from the point of inputting the image into the system all the way to point where the system returns an output (stating whether the blood sample is leukemia affected or not). In line with the work proposed by (Gayathri & Jyothi, 2018), this work would adopt a similar process for the diagnosis but would be comparing more algorithms to achieve the objective of comparatively reviewing algorithms to see which of them would perform best to classify the extracted features from the input image. After reviewing the performance of CNN, SVM, PNN, random forest, and naïve bayes algorithm on the extracted features, the best performing algorithm would be deployed as an API and consumed within an android app that would collect the microscopic images through the camera and pass it to our model for diagnosis, hereby achieving the fourth objective in this work.

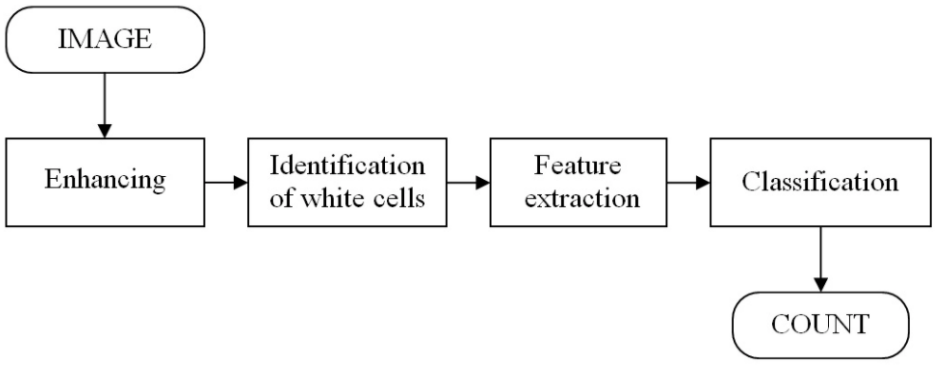


Figure 3.1 Proposed leukemia diagnosis system

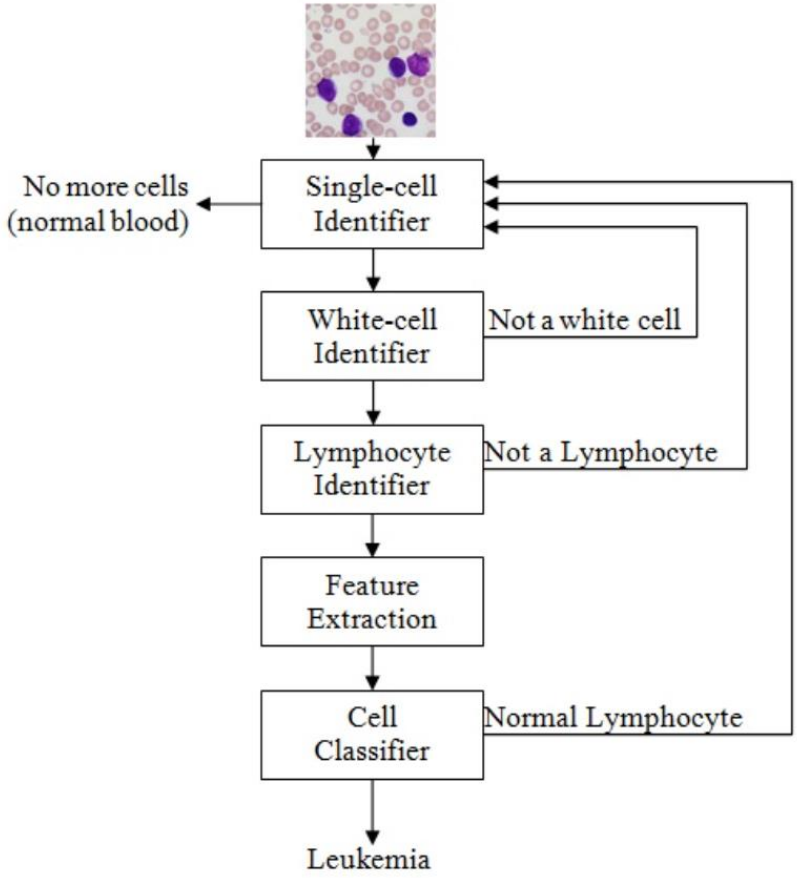


Figure 3.2 Process Model (Source: ALL-IDB initiative)

3.2.1 Image Preprocessing

Data acquired could sometimes be messy and uniformed. To make it easier for the leukemia diagnosis ML model to analyze and process data computationally, image preprocessing is used to standardize the image

data before feeding them to the model or neural network. In this work, the following methods of image preprocessing were adopted;

- Data Augmentation: provided with just 260 cell images from ALL-IDB initiative, this method is used to increase the dataset by computationally scaling, rotating, and performing other types of transformation on the image. It consists of de-texturization, decolorization and edge enhancement. Some algorithms and image classification techniques that adopt deep learning perform less effective and efficient with little image dataset. Data augmentation is used in this work to increase the image dataset before passing it to a CNN model.
- Image standardization: to achieve uniformity on the image dataset, it is necessary to sometimes resize images and align their width and height. Some algorithms like CNN require all training image dataset to be of uniform to perform optimally.
- Color images to grayscale: the adopted process of diagnosis leukemia in this work do not require colors, it was hereby important to reduce the input images to grayscale in other to avoid unnecessary memory consumption and computation complexity.

3.2.2 Image Segmentation

Image segmentation is a technique used to partition digital images into multiple parts or regions. Medical diagnosis that require studying some properties and elements of the cell image would mostly require image segmentation automate that process. To achieve instance segmentation in this work, a pretrained model called Mask R-CNN was used, the final layers were dropped and retrained on ALL-IDB1 dataset to uniquely classify only blast cells. Mask R-CNN uses RestNet101 architecture to extract features from images. It makes use of region proposal network (RPN) to generate a region of interest (ROI).

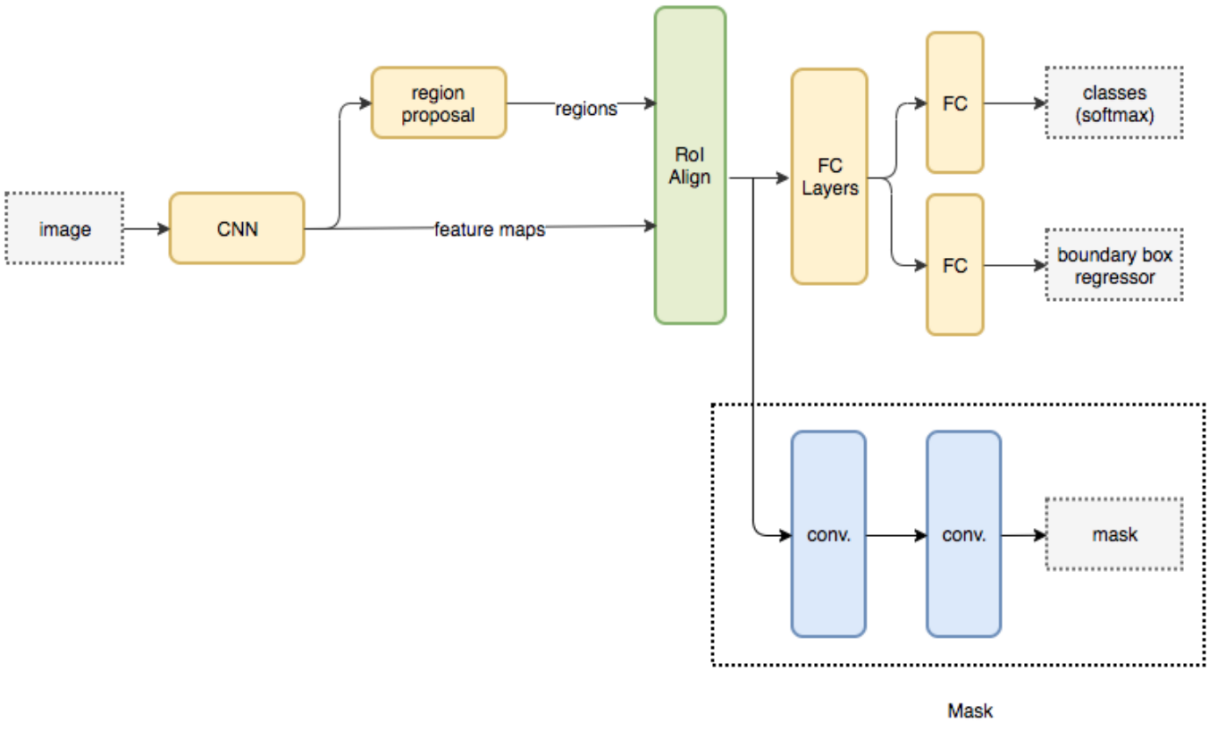


Figure 3.3 Mask R-CNN (Source: Analytics Vidhya, 2019)

3.2.3 Feature Extraction

Features play a very important role in image processing and medical image analysis. Haven carried out several other important steps of preprocessing, thresholding, and image segmentation, some filters where added to the image to achieve feature extraction. The features used in this work include, gabor, canny edge, robert edge, sobel, scharr, prewitt, gaussian, median, and variance. These extracted features are then used to train some ML classifiers and compare their performance.

3.3 Application Design

To achieve a robust, testable and maintainable design, this work adopts the use of android architecture component, MVVM (Model View View-Model). This proposed system will rely on android framework to collect new images and pass them to ML model hosted in the cloud. The model processes the input image and sends a response back to the android interface where the status of the blood sample is displayed.

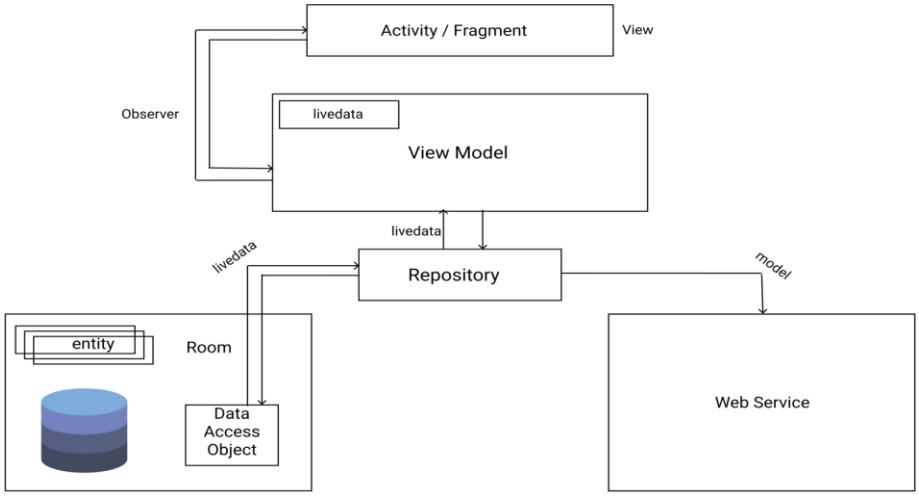


Figure 3.4 MVVM Architecture Component

3.3.1 Use Case Diagram

A Use Case Diagram is the simplest representation of a *user's* interaction with a *system* that shows the *relationship* between the user and the different *user cases* in which the user is involved.

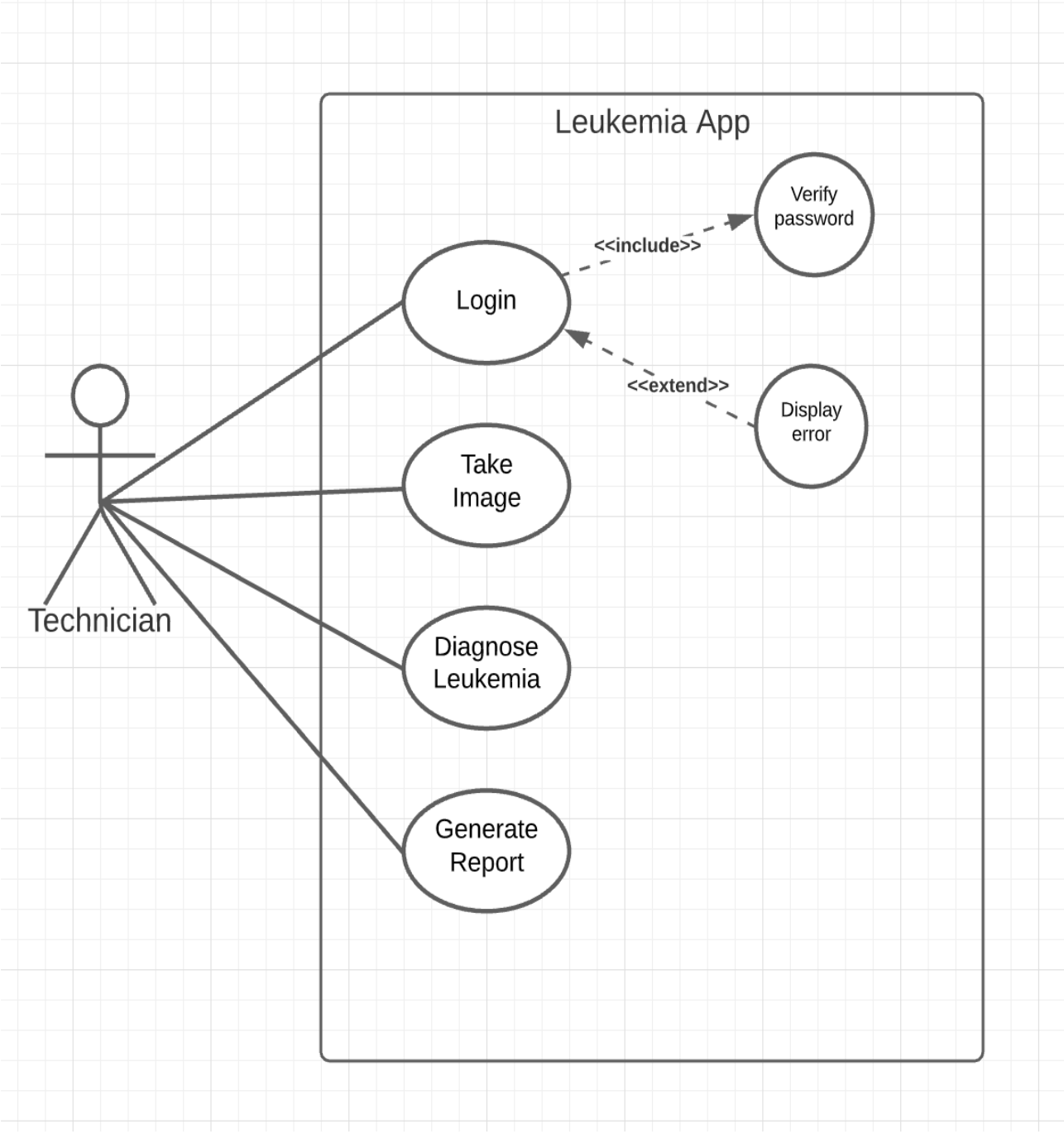


Figure 3.5 Use Case Diagram

3.3.2 UML Sequence Diagram

UML sequence diagram show the sequence of object interaction that take place. They are connection charts that detail how operations are carried out.

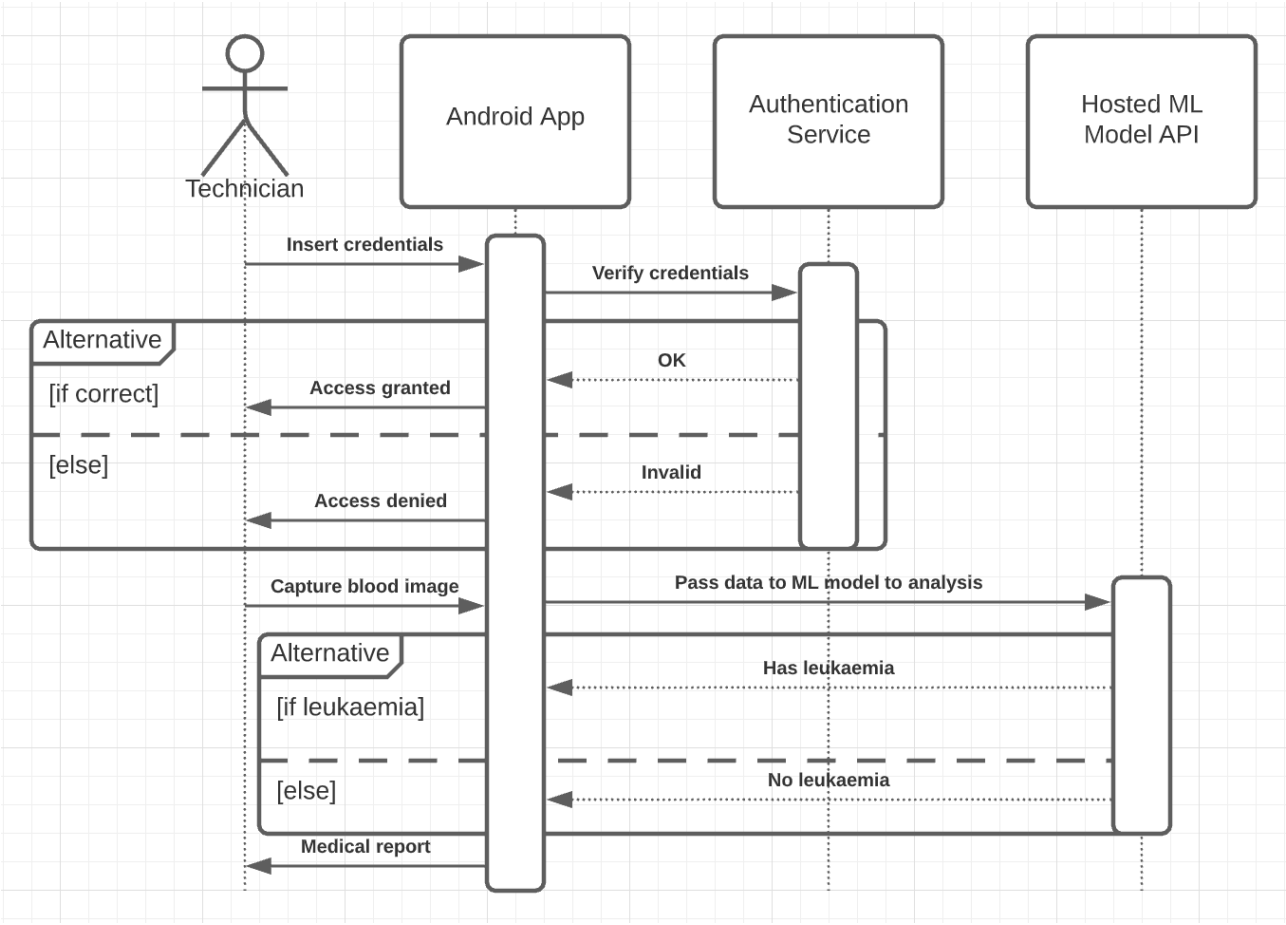


Figure 3.6 UML Sequence Diagram

3.4 Machine Learning Life Cycle (MLLC)

MLLC is a cyclical process for building and managing good quality models. It is an approach adopted by data science specialists to design, develop and test high quality models. The leukemia diagnosis ML model was developed through the processes of data gathering, exploratory analysis, training, model selection and verification, deployment and monitoring. A continuous deployment and automation pipeline was developed to ensure that the model can be easily improved and deployed for use in the long run.

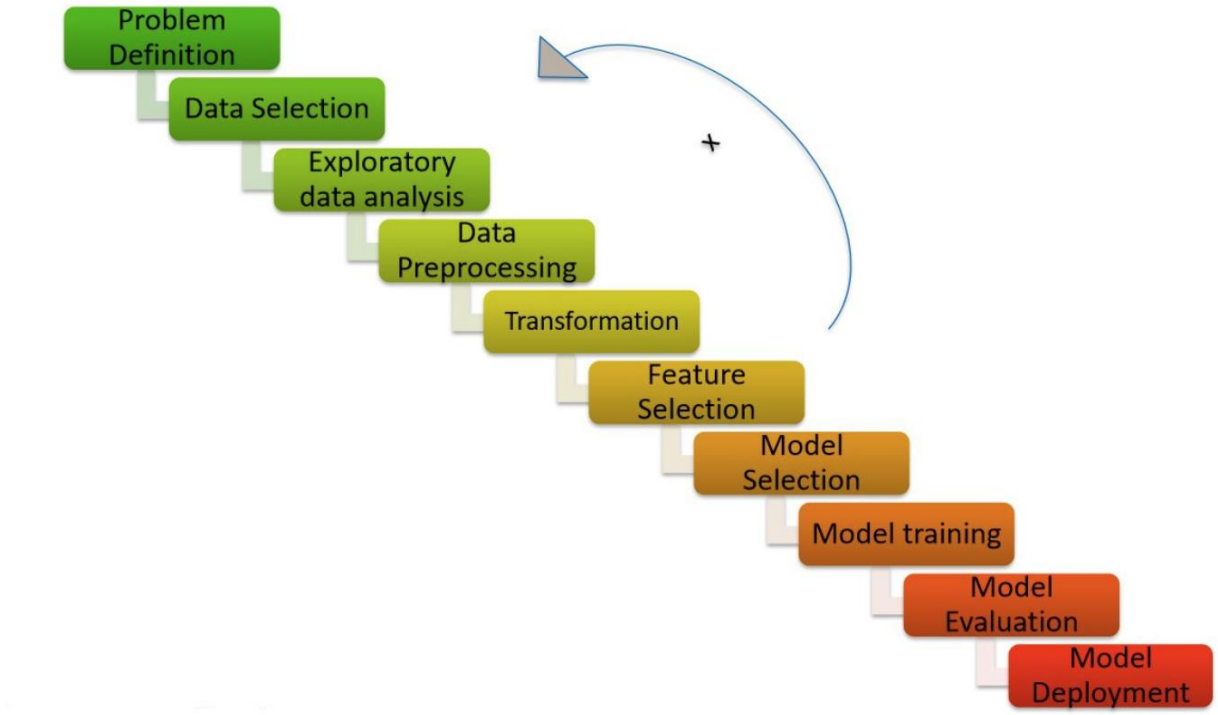


Figure 3.7 Machine Learning Model Development Lifecycle (Source: Analytics Vidhya, 2020)

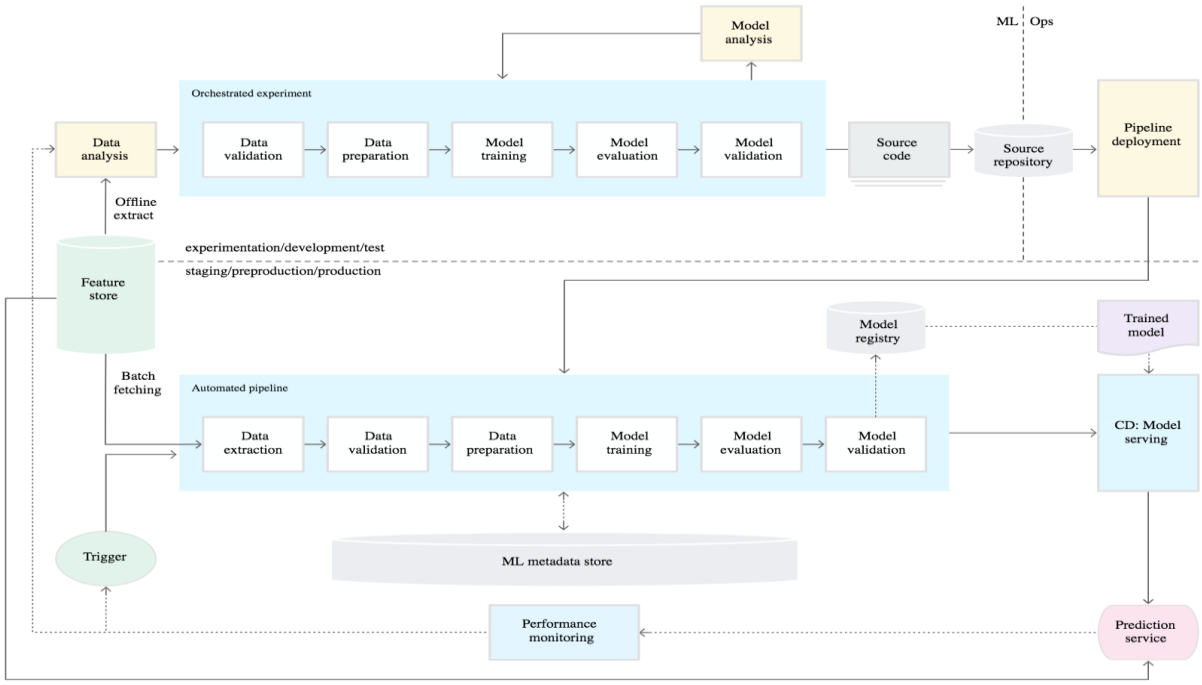


Figure 3.8 CI/CD and automated ML pipeline (Source: Google Cloud, n.d.)

CHAPTER FOUR

IMPLEMENTATION AND RESULT

4.0 Implementation and Documentation

This chapter demonstrates the experimental results of implementing an android-based automated leukemia diagnosis system. The output include an optimal method for diagnosis acute lymphoblastic leukemia (ALL) using image processing and image classification techniques.

It describes the tools used in developing and implementing the system. It describes the various steps and processes carried out on the image samples and their output at every step.

4.1 Programming Language, IDEs, Tools AND Technologies

Python 3.7 was used in building and deploying the ML model. The model was hosted as a RESTful API on Google Cloud Platform (GCP) with Python (Flask). Some of the libraries used in this work include Keras, Tensorflow, OpenCV, and Mask R-CNN pretrained RestNet101 instance segmentation framework. The frontend of the system was built in Java (Android) while adopting a MVVM architecture component. MVVM (Model View View-Model) is a software architectural pattern that facilitates the separation of the development of the graphical user interface (GUI code) from the development of the back-end logic so that the view is not dependent on any specific model platform. Spyder IDE and Jupyter Notebook were used to run all python programs, and Android Studio IDE (4.0.1) was used to in developing the android app.

4.2 How Automated Leukemia Diagnosis System Works

The diagnosis system works based on some functional and non-functional requirements. These requirements portray the features and behaviour of the system or software application. It conveys the expectation of the users from the software product.

4.2.1 Function Requirement

Functional requirements specify what the system should do. The functional requirements of the proposed automated leukemia diagnosis system include;

1. Registration and Login
2. Capturing blood smear images through the device camera
3. Performing diagnosis on the blood image gotten from the device camera
4. Provide a medical report containing the result of the diagnosis

4.2.2 Non-Functional Requirement

Non-functional requirements represent constraints under which a system should perform all it's functional requirements. The Non-functional requirements of the proposed automated leukemia diagnosis system include;

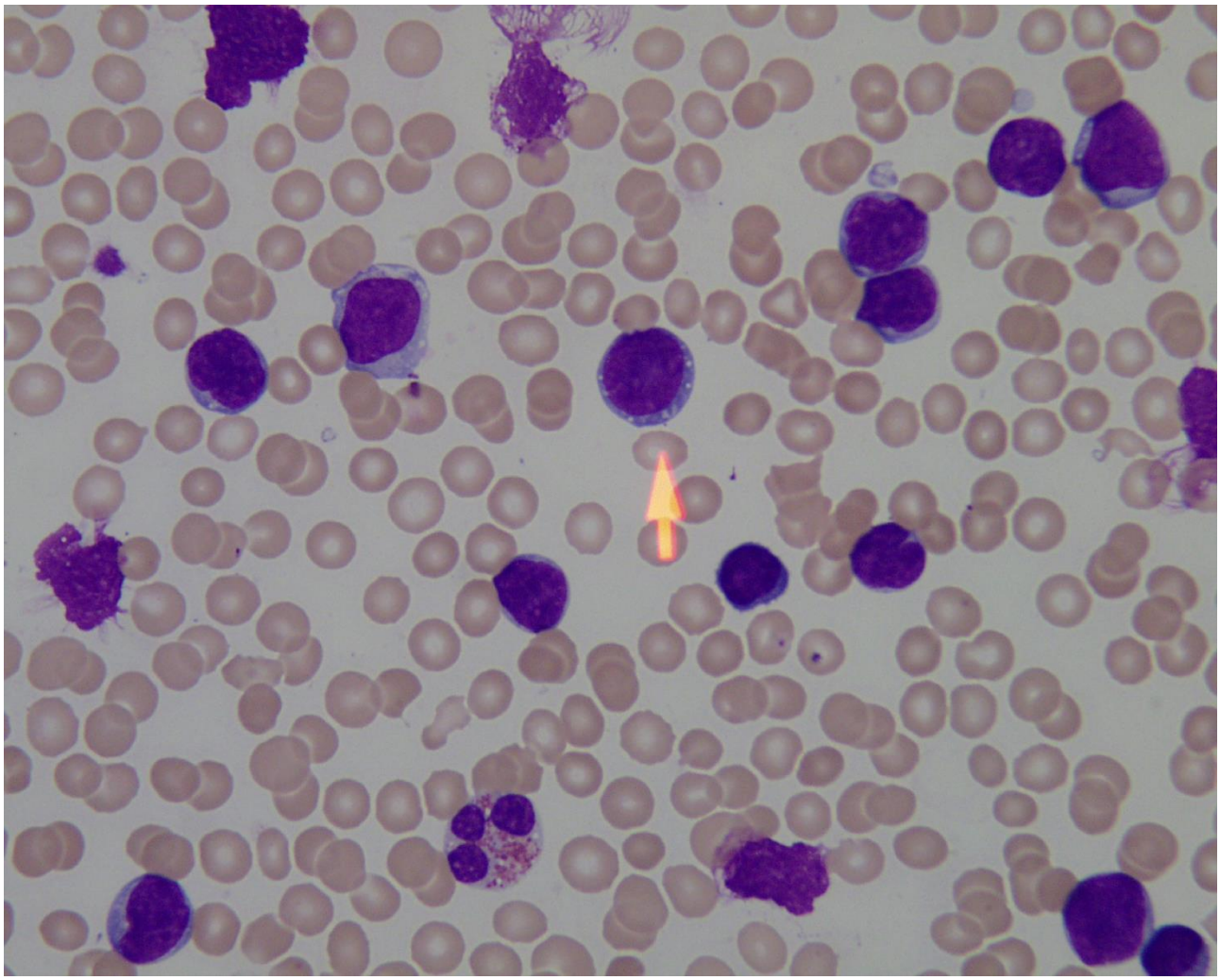
1. Security:
    a. The system should have a constraint on patient identity and information.
    b. The system should have multiple layers for authenticating users and deciding what level of data and information they have access to. These layers would include, user, admin, and super-admin.
2. Usability: the diagnosis system should be easily understandable and user friendly.
3. Scalability: the system should be robust enough to perform efficiently and accommodate changes across geographical boundaries.

4.3 Screenshots of Implementation Stages

The screenshots of the implementation stages show the interface of the android software to be used by the medical laboratory technician and the output image of the various processes happening in backend.

4.3.1 Leukemia Diagnosis Model (Using Image Processing)

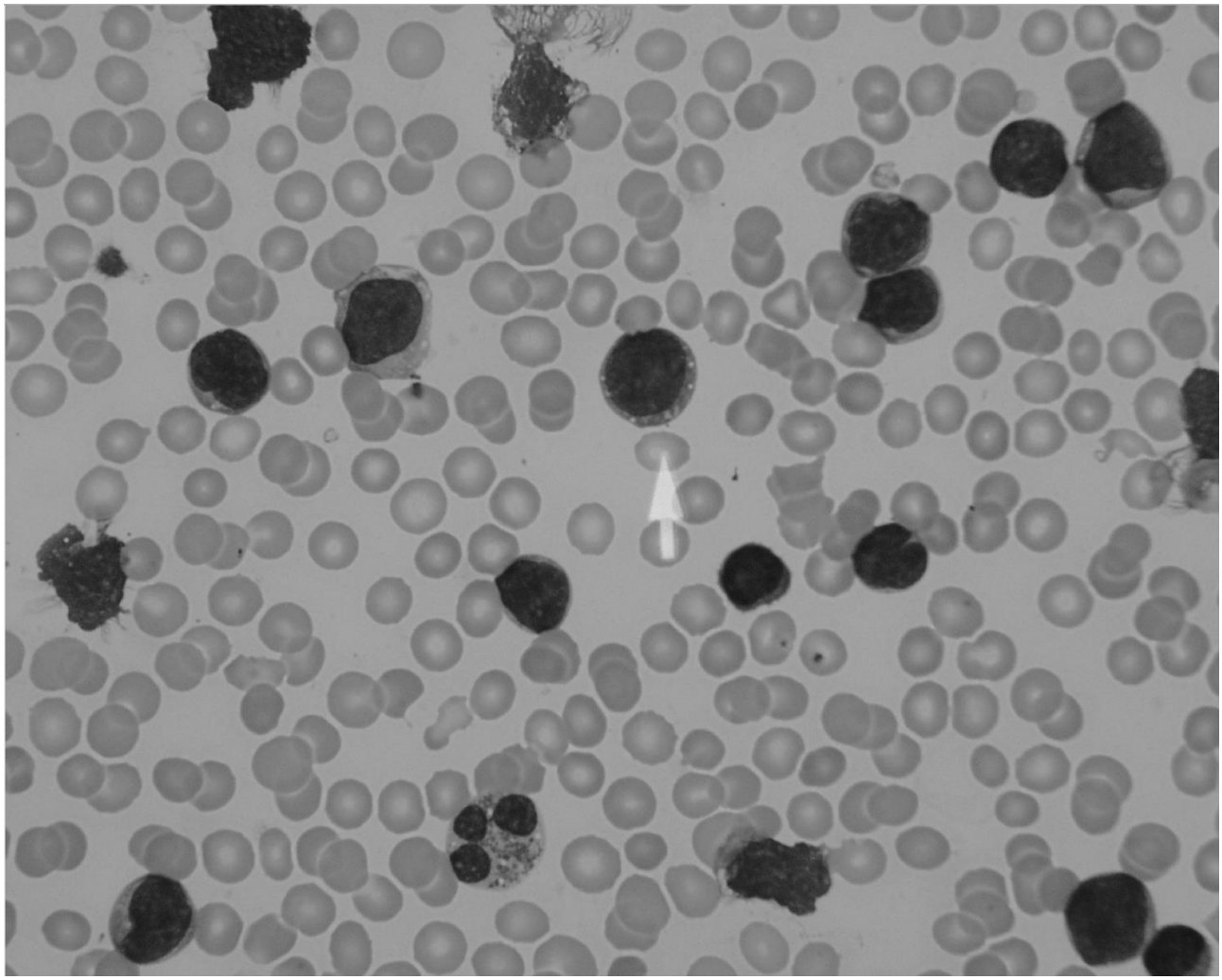
a. Original Image



This is the cell image before undergoing any form of preprocessing or cleaning.

Figure 4.1 Original image

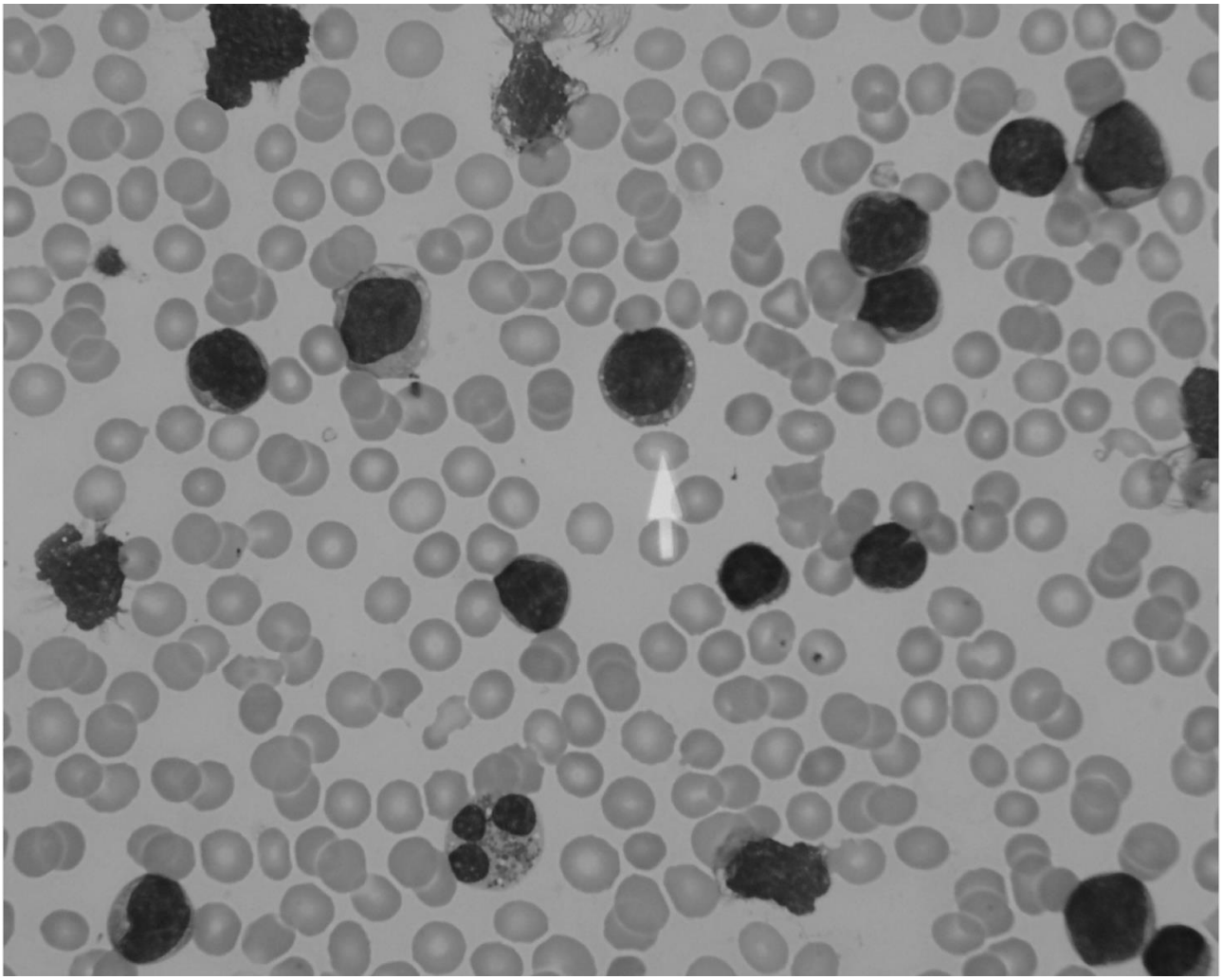
b. Grayscale Image



This is the cell image after programmatically converting it to grayscale

Figure 4.2 Grayscale Image

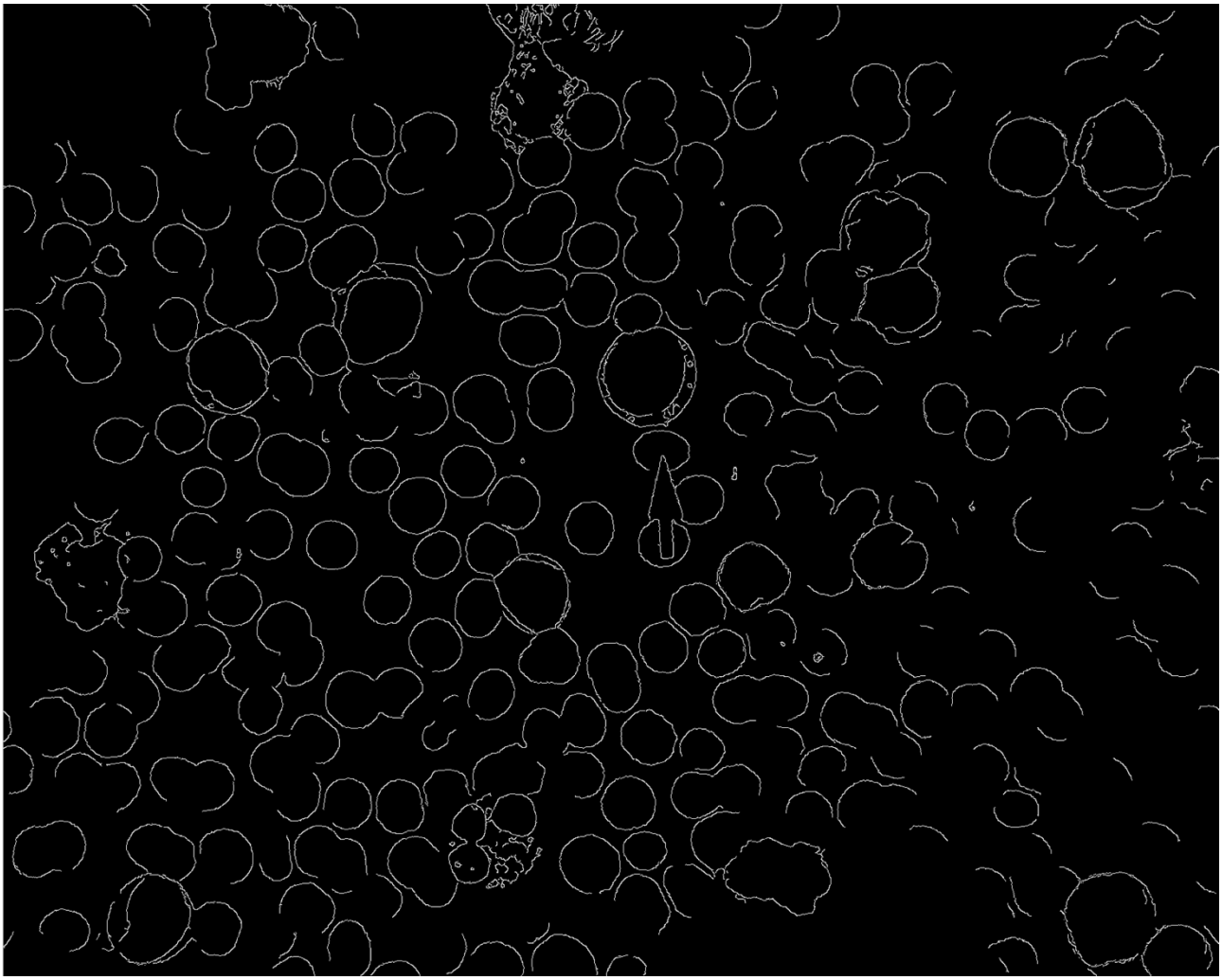
c. Enhanced Image



This is the cell image after applying median blur filter to enhance and smoothing the grayscale image.

Figure 4.3 Enhanced Image

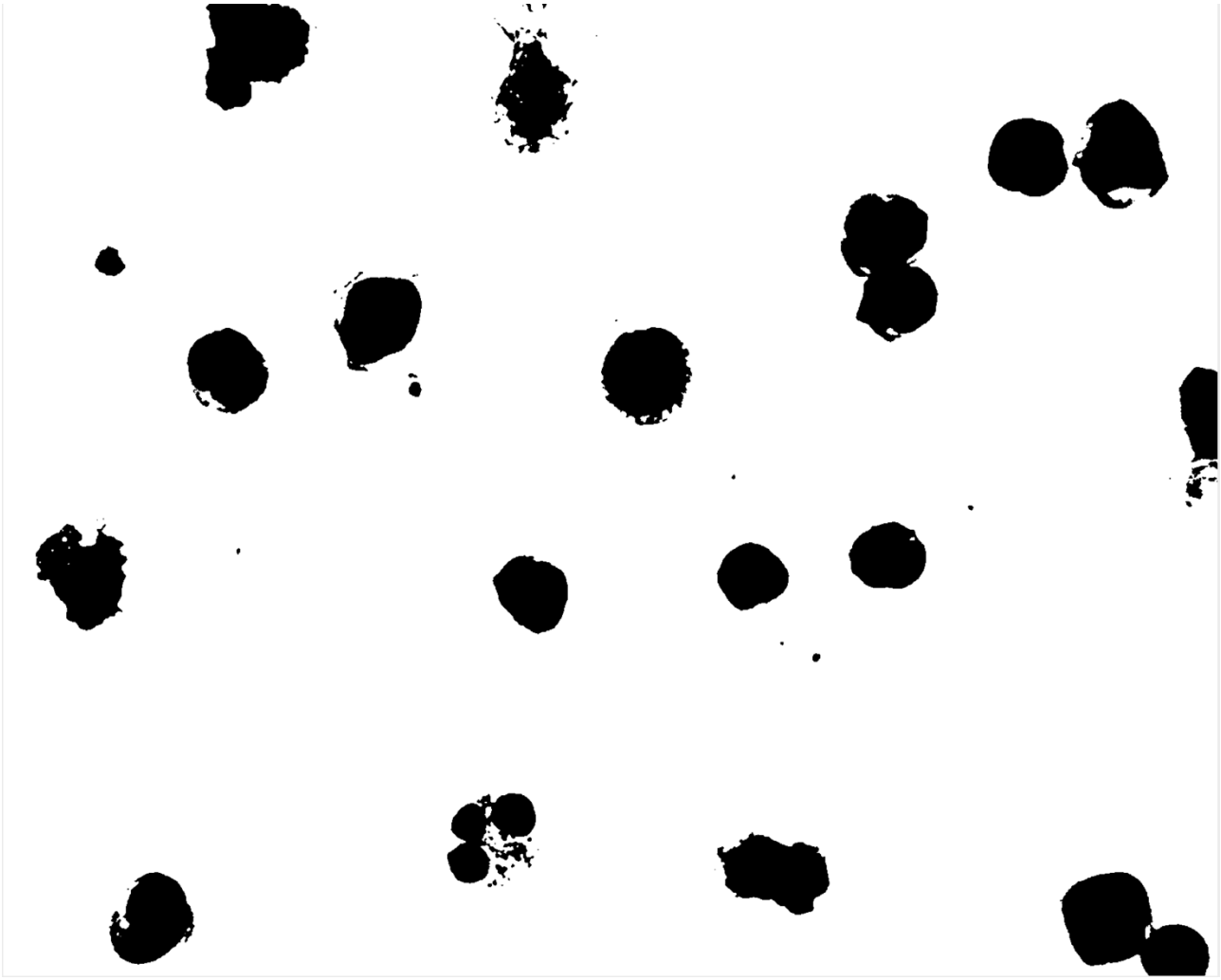
d. Edge Detection



This is the cell image after applying canny edge filter to get the edges of all cells present.

Figure 4.4 Edge Detection

e. Thresholding and Image Segmentation



This is the cell image after applying Otsu's thresholding to the enhanced image and performing segmentation with the Mask R-CNN pretrained model.

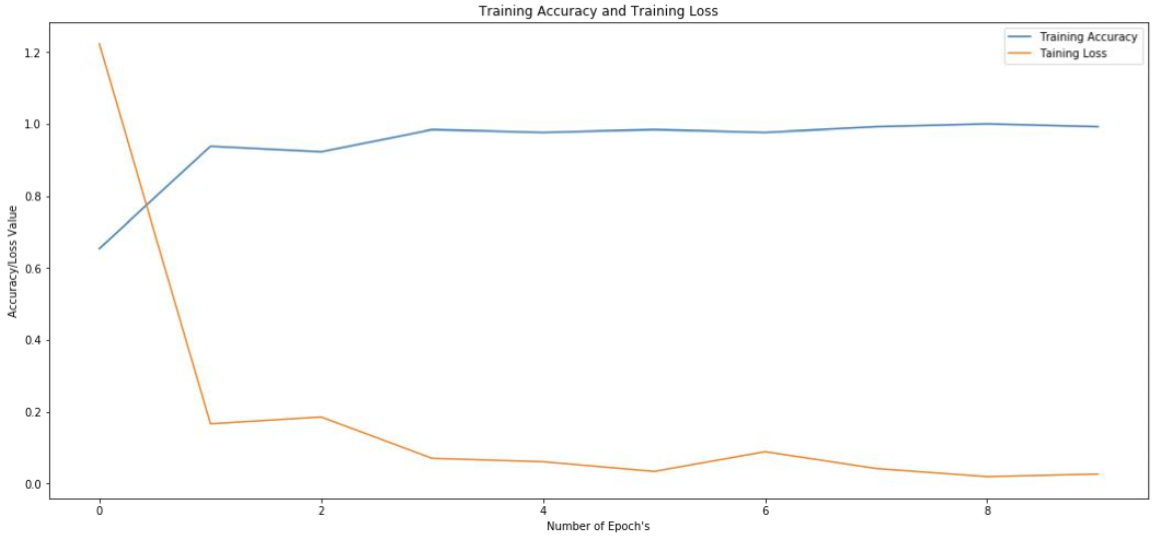
Figure 4.5 Thresholding and Image segmentation

4.3.2 Leukemia Diagnosis Model (Using Image Classification)

a. CNN Classifier

Out[17]:

```
<matplotlib.legend.Legend at 0x1a568e99e8>
```

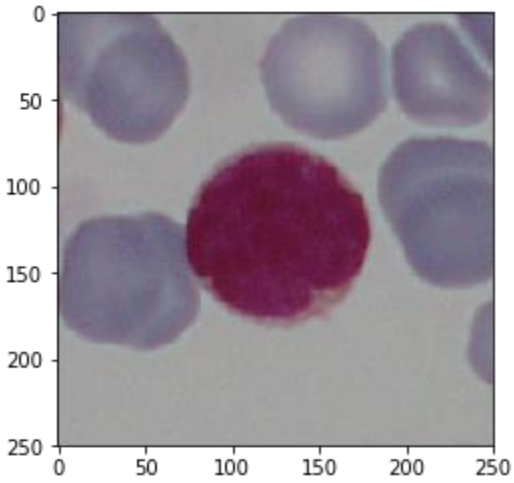


In [18]:

```
130/130 [==============================] - 3s 27ms/step
```

In [19]:

```
LOSS : 0.527237777870435
ACCURACY : 0.8461538461538461
```




```
[[0.03920074 0.9607993 ]]
Infected
```

Figure 4.6 Training Accuracy and Training Loss of CNN Classifier

b. SVM Classifier

```
Accuracy: 0.7076923076923077
Prediction is: infected
```

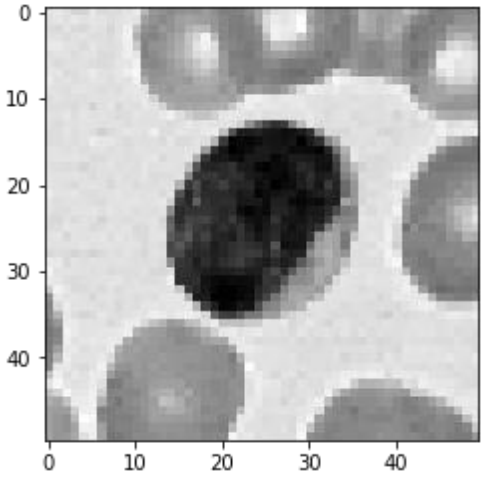


Figure 4.7 Training Accuracy and Training Loss of SVM Classifier

4.3.3 Android Interfaces

a. Take Image Screen

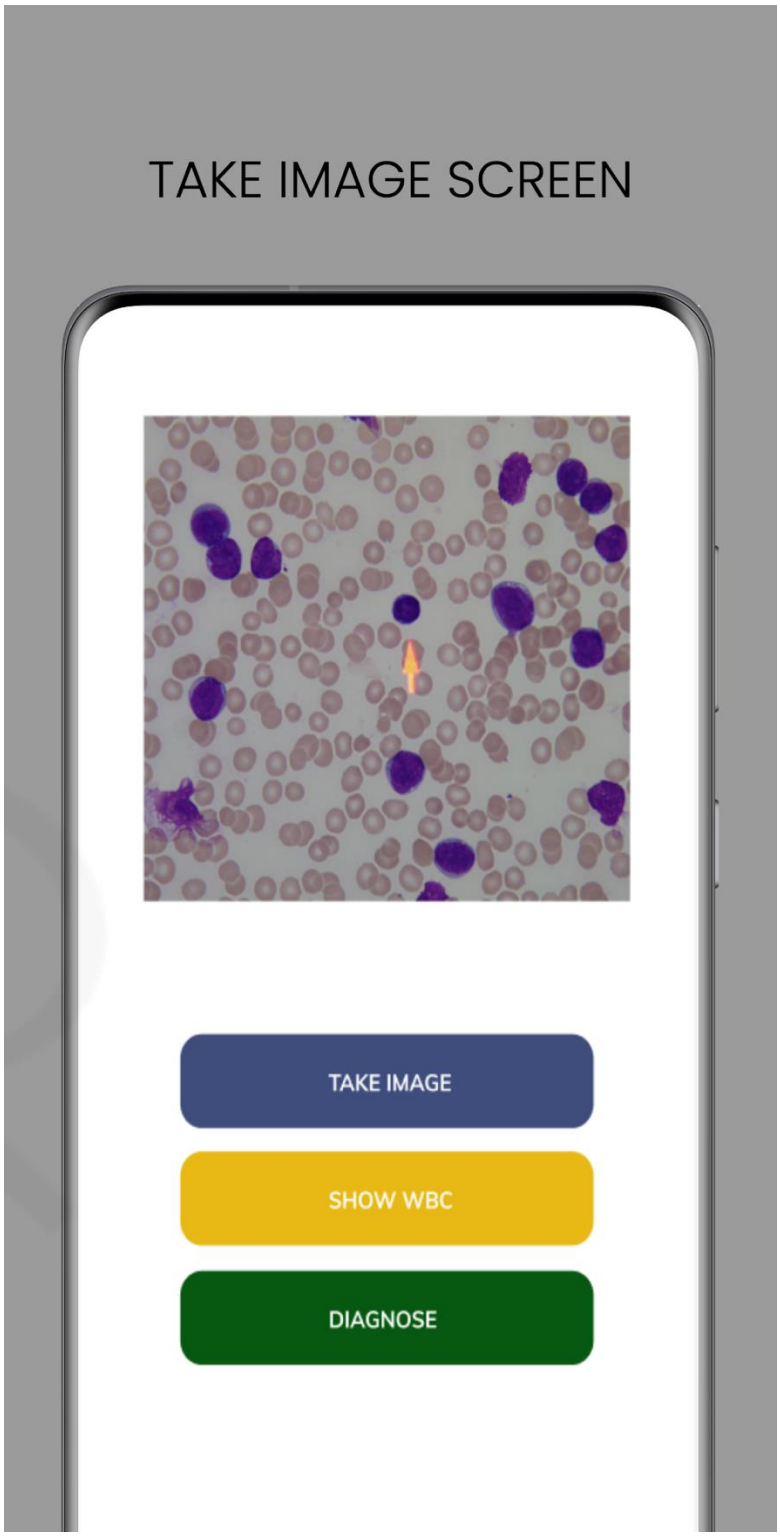


Figure 4.8 Capturing cell image

b. Show WBC Screen

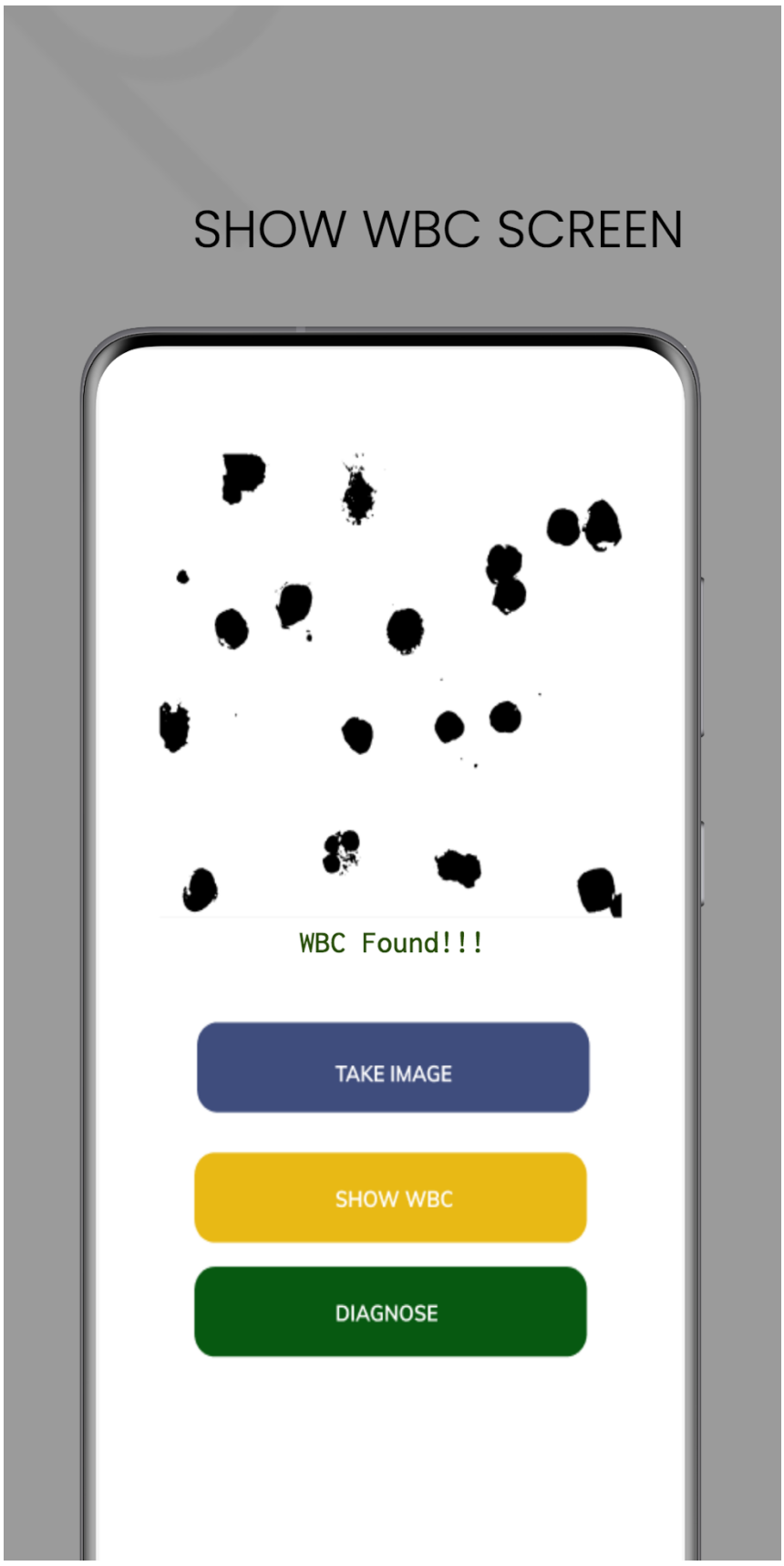


Figure 4.9 WBCs Found Screen

c. Diagnose Screen

Figure 4.10 Leukemia Found Screen

CHAPTER FIVE

SUMMARY AND CONCLUSION

5.0 Summary

The use of machine learning techniques like image processing to diagnose some disease has proved effective and valuable. The process of diagnosing leukemia in this work adopted two approaches. First was using image classification algorithms on the 260 ALL-IDB2 image dataset, while the second approach was to use image processing techniques to enhance the image, segment it, extract some features like shape from the segmented image and then passing those features into a random forest classifier. This work compared the performance of CNN, SVM, PNN, random forest, and naïve bayes classifiers. The image segmentation also compared performance of using Otsu's thresholding (as a basic form of segmentation) against the Mask R-CNN (which is a RestNet101 instance segmentation architecture).

5.1 Contribution to Knowledge

This research contributes to knowledge by utilizing machine learning techniques to develop a system that diagnoses acute lymphoblastic leukemia (ALL). The developed system aims to assist medical institutions and the society as large in timely and cheap diagnosis of ALL. This work provides some insight into the process of using image processing and classification for identifying blast cells in a blood sample. This provides opportunities for other researchers to expand the scope of this work and build models that would diagnose other related ailments using similar approaches of image processing techniques.

5.2 Limitations

i. Scarcity of previous works

ii. Scarcity of sufficient datasets

iii. Time constraints

5.3 Recommendation for Further Study

After designing and implementing a system for automatically diagnosing acute lymphoblastic leukemia, and also evaluating the performance and accuracy of five classification algorithms to ascertain the most effective and efficient one of them, the following are recommended;

i. Other leukemia subtypes like AML and CML can be diagnosed using a similar approach.
ii. Other image processing and classification algorithms can be compared outside the scope of this work to adequately extract features from images and also classify them.
iii. Other android architecture components like Model View Presenter (MVP) could be used in designing the android system for consuming the hosted ML model while also prioritizing security of patient data.

5.4 Conclusion

In conclusion, it was observed that a traditional machine learning random forest classifier performed better than CNN in classifying features extracted from the 260 ALL-IDB1 images dataset. CNN however performed best when image augmentation was adopted to increase the ALL-IDB2 image dataset used in training the deep learning classifier.

REFERENCES

Harun, N. H., Bakar, N. A. A., Mohan, U. A. P., Nadzir, M. M., Hassan, M. G., & Adollah, R. (2019). Automated Cell Counting System for Chronic Leukemia. *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology, JEEIT 2019 - Proceedings*, 502–506. https://doi.org/10.1109/JEEIT.2019.8717500

Kumar, S., Mishra, S., Asthana, P., & Pragya. (2018). Automated detection of acute leukemia using K-mean clustering algorithm. *Advances in Intelligent Systems and Computing*, *554*, 655–670. https://doi.org/10.1007/978-981-10-3773-3_64

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2015). Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, *18*, 1–43. http://arxiv.org/abs/1502.05767

Scotti, F. (2007). *Robust Segmentation and Measurements Techniques of White Cells in Blood Microscope Images*.

Suganda, R., Sutrisno, E., & Wardana, I. W. (2013). Data Mining: Concepts and Techniques Second Edition. In *Journal of Chemical Information and Modeling* (Vol. 53, Issue 9).

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, *2*(1). https://doi.org/10.1186/s40537-014-0007-7

Rathee, R. (2013). *Acute Lymphoblastic Leukemia : Characterization and its Prognostic Values*. *2*(December 2008), 27–36.

Guzmán-Cabrera, R., Guzmán-Sepúlveda, J. R., Torres-Cisneros, M., May-Arrioja, D. A., Ruiz-Pinales, J., Ibarra-Manzano, O. G., Aviña-Cervantes, G., & Parada, A. G. (2013). Digital image processing technique for breast cancer detection. *International Journal of Thermophysics*, *34*(8–9), 1519–1531. https://doi.org/10.1007/s10765-012-1328-4

Gayathri, S., & Jyothi, R. L. (2018). An Automated Leucocyte Classification For Leukemia Detection. *International Research Journal of Engineering and Technology (IRJET)*.

Pui, C.-H. (2011). Acute Lymphoblastic Leukemia BT - Encyclopedia of Cancer. *Encyclopedia of Cancer*, 23–26. https://doi.org/10.1007/978-3-642-16483-5_57

Shafique, S., & Tehsin, S. (2018). Computer-Aided Diagnosis of Acute Lymphoblastic Leukaemia. *Computational and Mathematical Methods in Medicine, 2018*. https://doi.org/10.1155/2018/6125289

Ahmed, N., Yigit, A., Isik, Z., & Alpkocak, A. (2019). *Identification of Leukemia Subtypes from Microscopic Images Using Convolutional Neural Network.*

Salah, H. T., Muhsen, I. N., Salama, M. E., Owaidah, T., & Hashmi, S. K. (2019). Machine learning applications in the diagnosis of leukemia: Current trends and future directions. *International Journal of Laboratory Hematology*, *41*(6), 717–725. https://doi.org/10.1111/ijlh.13089

Ravi, P., & Ashokkumar, A. (2017). Analysis of Various Image Processing Techniques. *International Journal of Advanced Networking & Applications*, *8*(5), 86–89.

Hu, R., & Li, C. (2015). Computational and Mathematical Methods in Medicine Indexed in Science Citation Index Expanded Indexed in Science Citation Index Expanded. *Computational and Mathematical Methods in Medicine*, *2015*(Figure 1), 1–8.

Chatarwad, S., Bansode, P., Burade, A., & Chaware, P. T. S. (2018). Automatic Blood Cancer Detection Using Image Processing. *International Journal of Recent Trends in Engineering and Research*, *4*(3), 204–210. https://doi.org/10.23883/ijrter.2018.4117.o3kbv

Modi, H., Pandya, M., Vaghela, H., & Potdar, M. (2016). Leukemia Detection using Digital Image Processing Techniques Leukemia Detection using Digital Image Processing Techniques. *International Journal of Applied Information Systems (IJAIS), November 2015*. https://doi.org/10.5120/ijais2015451461

Gonzalez, R., Woods, R., & Eddins, S. (2009). *Digital Image Processing Using Matlab* (Second Edition). Gatesmark, LLC.

Sharma, P. (2019, April 1). *Computer Vision Tutorial: A Step-by-Step Introduction to Image* Segmentation Techniques (Part 1). Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/

WHO, (2018). Cancer. Retrieved from https://www.who.int/news-room/fact-sheets/detail/cancer

Facts 2018-2019 provides updates from the American Cancer Society's *Cancer Facts & Figures 2019* (published online 2019, https://www.cancer.org/research/cancer-facts-statistics.html)

APPENDIX

Source code for Automated Leukemia Diagnosis System

IMAGE CLASSIFICATION

Convolutional Neural Network (CNN)

```
In [1]:
# This Python 3 environment comes with many helpful analytics libraries installe
d
# It is defined by the kaggle/python docker image: https://github.com/kaggle/doc
ker-python
# For example, here's several helpful packages to load in import numpy as np #
linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list
 the files in the input directory
import os
# print(os.listdir('../enocholutunmida/Downloads/cell_images/Parasitized'))
# print(os.listdir('../enocholutunmida/Downloads/cell_images/Uninfected'))
parasitized_data = os.listdir('../enocholutunmida/Downloads/ALL_IDB2/infected')
uninfected_data = os.listdir('../enocholutunmida/Downloads/ALL_IDB2/uninfected')
print(parasitized_data[1:]) #the output we get are the .png files print('\n')
print(uninfected_data[1:])
import cv2
import matplotlib.pyplot as plt
import seaborn as sns
import os
from PIL import Image
from keras.preprocessing.image import img_to_array from
keras.preprocessing.image import load_img from keras.utils import np_utils

['Im090_1.tif', 'Im050_1.tif', 'Im015_1.tif', 'Im108_1.tif', 'Im074_
1.tif', 'Im031_1.tif', 'Im089_1.tif', 'Im111_1.tif', 'Im049_1.tif',
'Im113_1.tif', 'Im076_1.tif', 'Im033_1.tif', 'Im052_1.tif', 'Im017_
1.tif', 'Im092_1.tif', 'Im056_1.tif', 'Im013_1.tif', 'Im096_1.tif',
'Im117_1.tif', 'Im072_1.tif', 'Im037_1.tif', 'Im128_1.tif', 'Im070_
1.tif', 'Im035_1.tif', 'Im008_1.tif', 'Im115_1.tif', 'Im094_1.tif',
'Im069_1.tif', 'Im054_1.tif', 'Im011_1.tif', 'Im032_1.tif', 'Im077_
1.tif', 'Im112_1.tif', 'Im093_1.tif', 'Im016_1.tif', 'Im053_1.tif',
'Im109_1.tif', 'Im014_1.tif', 'Im051_1.tif', 'Im029_1.tif', 'Im091_
1.tif', 'Im048_1.tif', 'Im110_1.tif', 'Im030_1.tif', 'Im075_1.tif',
'Im088_1.tif', 'Im114_1.tif', 'Im009_1.tif', 'Im034_1.tif', 'Im071_
1.tif', 'Im129_1.tif', 'Im010_1.tif', 'Im055_1.tif', 'Im130_1.tif',
'Im095_1.tif', 'Im068_1.tif', 'Im097_1.tif', 'Im012_1.tif', 'Im057_
1.tif', 'Im036_1.tif', 'Im073_1.tif', 'Im116_1.tif', 'Im042_1.tif',
'Im007_1.tif', 'Im082_1.tif', 'Im127_1.tif', 'Im103_1.tif', 'Im066_
1.tif', 'Im023_1.tif', 'Im099_1.tif', 'Im064_1.tif', 'Im021_1.tif',
'Im059_1.tif', 'Im101_1.tif', 'Im080_1.tif', 'Im038_1.tif', 'Im125_
1.tif', 'Im118_1.tif', 'Im040_1.tif', 'Im005_1.tif', 'Im121_1.tif',
'Im079_1.tif', 'Im084_1.tif', 'Im044_1.tif', 'Im001_1.tif', 'Im060_
1.tif', 'Im025_1.tif', 'Im105_1.tif', 'Im018_1.tif', 'Im107_1.tif',
```

```
'Im062_1.tif', 'Im027_1.tif', 'Im046_1.tif', 'Im003_1.tif', 'Im123_
1.tif', 'Im086_1.tif', 'Im100_1.tif', 'Im058_1.tif', 'Im098_1.tif',
'Im020_1.tif', 'Im065_1.tif', 'Im004_1.tif', 'Im041_1.tif', 'Im119_
1.tif', 'Im124_1.tif', 'Im081_1.tif', 'Im039_1.tif', 'Im126_1.tif',
'Im083_1.tif', 'Im006_1.tif', 'Im043_1.tif', 'Im022_1.tif', 'Im067_
1.tif', 'Im102_1.tif', 'Im026_1.tif', 'Im063_1.tif', 'Im106_1.tif',
'Im087_1.tif', 'Im122_1.tif', 'Im002_1.tif', 'Im047_1.tif', 'Im045_
1.tif', 'Im078_1.tif', 'Im085_1.tif', 'Im120_1.tif', 'Im019_1.tif',
'Im104_1.tif', 'Im024_1.tif', 'Im061_1.tif']
['Im255_0.tif', 'Im154_0.tif', 'Im169_0.tif', 'Im194_0.tif', 'Im209_
0.tif', 'Im135_0.tif', 'Im170_0.tif', 'Im234_0.tif', 'Im236_0.tif',
'Im137_0.tif', 'Im172_0.tif', 'Im196_0.tif', 'Im156_0.tif', 'Im212_
0.tif', 'Im257_0.tif', 'Im192_0.tif', 'Im216_0.tif', 'Im253_0.tif',
'Im152_0.tif', 'Im133_0.tif', 'Im176_0.tif', 'Im232_0.tif', 'Im149_
0.tif', 'Im248_0.tif', 'Im189_0.tif', 'Im230_0.tif', 'Im131_0.tif',
'Im174_0.tif', 'Im150_0.tif', 'Im214_0.tif', 'Im251_0.tif', 'Im229_
0.tif', 'Im190_0.tif', 'Im237_0.tif', 'Im173_0.tif', 'Im136_0.tif',
'Im157_0.tif', 'Im256_0.tif', 'Im213_0.tif', 'Im197_0.tif', 'Im168_
0.tif', 'Im195_0.tif', 'Im254_0.tif', 'Im211_0.tif', 'Im155_0.tif',
'Im171_0.tif', 'Im134_0.tif', 'Im235_0.tif', 'Im208_0.tif', 'Im188_
0.tif', 'Im231 0.tif', 'Im175 0.tif', 'Im148 0.tif', 'Im249 0.tif',
'Im228_0.tif', 'Im191_0.tif', 'Im151_0.tif', 'Im250_0.tif', 'Im215_
0.tif', 'Im252_0.tif', 'Im217_0.tif', 'Im153_0.tif', 'Im193_0.tif',
'Im177_0.tif', 'Im132_0.tif', 'Im233_0.tif', 'Im186_0.tif', 'Im202_
0.tif', 'Im247_0.tif', 'Im146_0.tif', 'Im162_0.tif', 'Im226_0.tif',
'Im219_0.tif', 'Im224_0.tif', 'Im160_0.tif', 'Im144_0.tif', 'Im200_
0.tif', 'Im245_0.tif', 'Im184_0.tif', 'Im179_0.tif', 'Im204_0.tif',
'Im241_0.tif', 'Im140_0.tif', 'Im138_0.tif', 'Im239_0.tif', 'Im180_
0.tif', 'Im258_0.tif', 'Im159_0.tif', 'Im164_0.tif', 'Im199_0.tif',
'Im220_0.tif', 'Im222_0.tif', 'Im166_0.tif', 'Im182_0.tif', 'Im142_
0.tif', 'Im206_0.tif', 'Im243_0.tif', 'Im260_0.tif', 'Im225_0.tif',
'Im161_0.tif', 'Im218_0.tif', 'Im185_0.tif', 'Im178_0.tif', 'Im145_
0.tif', 'Im244_0.tif', 'Im201_0.tif', 'Im246_0.tif', 'Im203_0.tif',
'Im147_0.tif', 'Im187_0.tif', 'Im163_0.tif', 'Im227_0.tif', 'Im223_
0.tif', 'Im167_0.tif', 'Im143_0.tif', 'Im242_0.tif', 'Im207_0.tif',
'Im183_0.tif', 'Im139_0.tif', 'Im238_0.tif', 'Im181_0.tif', 'Im240_
0.tif', 'Im205_0.tif', 'Im141_0.tif', 'Im165_0.tif', 'Im198_0.tif',
'Im221_0.tif', 'Im259_0.tif', 'Im158_0.tif']


Using TensorFlow backend.
```

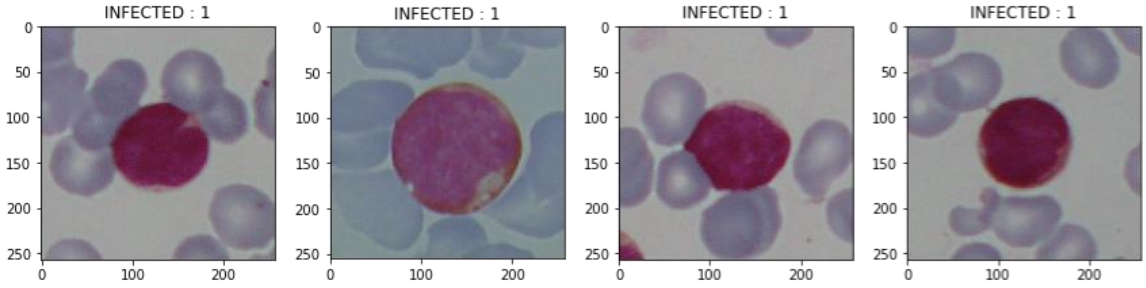
In [2]:

```
['Im028_1.tif', 'Im090_1.tif', 'Im050_1.tif', 'Im015_1.tif', 'Im108_
1.tif', 'Im074_1.tif', 'Im031_1.tif', 'Im089_1.tif', 'Im111_1.tif',
'Im049_1.tif']
['Im210_0.tif', 'Im255_0.tif', 'Im154_0.tif', 'Im169_0.tif', 'Im194_
0.tif', 'Im209 0.tif', 'Im135 0.tif', 'Im170 0.tif', 'Im234 0.tif',
'Im236_0.tif']
```

In [3]:

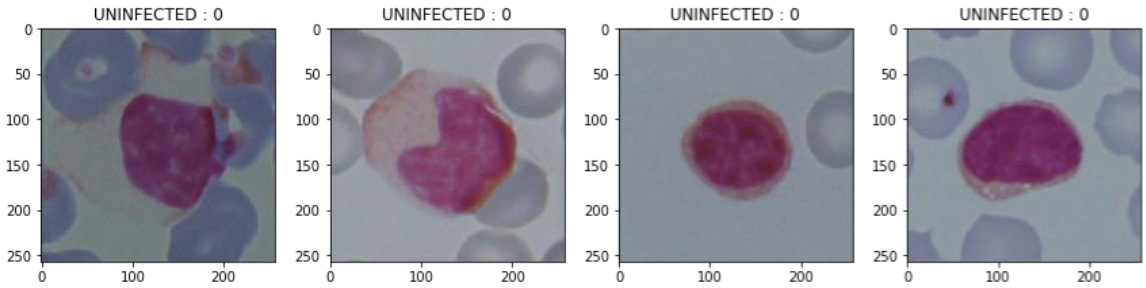
```
parasitized_data = os.listdir('../enocholutunmida/Downloads/ALL_IDB2/infected')
print(parasitized_data[:10]) #the output we get are the .png files

uninfected_data = os.listdir('../enocholutunmida/Downloads/ALL_IDB2/uninfected')
print('\n')
print(uninfected_data[:10])

plt.figure(figsize = (12,12)) for i in range(4):

    plt.subplot(1, 4, i+1)
    img = cv2.imread('../enocholutunmida/Downloads/ALL_IDB2/infected' + "/" + pa
rasitized_data[i])
    plt.imshow(img)
```

```
    plt.title('INFECTED : 1')
    plt.tight_layout()
plt.show()
```



In [4]:

```
plt.figure(figsize = (12,12)) for i in range(4):
    plt.subplot(1, 4, i+1)
    img = cv2.imread('../enocholutunmida/Downloads/ALL_IDB2/uninfected' + "/" +
uninfected_data[i+1])
    plt.imshow(img)
    plt.title('UNINFECTED : 0')
    plt.tight_layout()
plt.show()
```



In [5]:

```
data = []
labels = []
for img in uninfected_data: try:
        img_read = plt.imread('../enocholutunmida/Downloads/ALL_IDB2/uninfecte
d/' + "/" + img)
        img_resize = cv2.resize(img_read, (250, 250))
        img_array = img_to_array(img_resize)
        data.append(img_array)
        labels.append(0)
except: None
for img in parasitized_data: try:
        img_read = plt.imread('../enocholutunmida/Downloads/ALL_IDB2/infected/'
+ "/" + img)
        img_resize = cv2.resize(img_read, (250, 250))
        img_array = img_to_array(img_resize)
        data.append(img_array)
        labels.append(1)
except: None
```

In [6]:

```
Clipping input data to the valid range for imshow with RGB data
([0..1] for floats or [0..255] for integers).
```

In [7]:

In [8]:

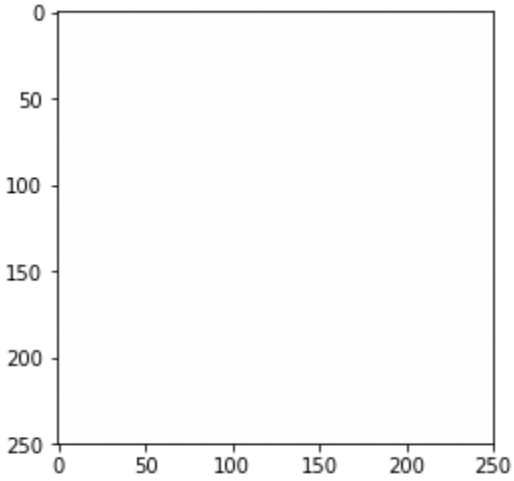
In [9]:

In [10]:

In [11]:

```
SHAPE OF TRAINING IMAGE DATA : (130, 250, 250, 3)
SHAPE OF TESTING IMAGE DATA : (130, 250, 250, 3)
SHAPE OF TRAINING LABELS : (130, 2)
SHAPE OF TESTING LABELS : (130, 2)
```

```
plt.imshow(data[50])
plt.show()
```




```
image_data = np.array(data)
labels = np.array(labels)
idx = np.arange(image_data.shape[0])
np.random.shuffle(idx)
image_data = image_data[idx]
labels = labels[idx]
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(image_data, labels, test_siz
e = 0.5, random_state = 101)
```

```
y_train = np_utils.to_categorical(y_train, num_classes = 2)
y_test = np_utils.to_categorical(y_test, num_classes = 2)
```

```
print(f'SHAPE OF TRAINING IMAGE DATA : {x_train.shape}') print(f'SHAPE OF
TESTING IMAGE DATA : {x_test.shape}') print(f'SHAPE OF TRAINING LABELS :
{y_train.shape}') print(f'SHAPE OF TESTING LABELS : {y_test.shape}')
```

In [12]:

```
import keras
from keras.layers import Dense, Conv2D
from keras.layers import Flatten
from keras.layers import MaxPooling2D, GlobalAveragePooling2D from keras.layers
import Activation
from keras.layers import BatchNormalization
from keras.layers import Dropout
```

```
from keras.models import Sequential
from keras import backend as K
from keras import optimizers
from keras.optimizers import Adam
```

In [13]:

```
def CNNbuild(height, width, classes, channels): model = Sequential()
    inputShape = (height, width, channels)
    chanDim = -1
if K.image_data_format() == 'channels_first': inputShape = (channels, height,
width)
    model.add(Conv2D(32, (3,3), activation = 'relu', input_shape = inputShape))
    model.add(MaxPooling2D(2,2))
    model.add(BatchNormalization(axis = chanDim))
    model.add(Dropout(0.2))
    model.add(Conv2D(32, (3,3), activation = 'relu'))
    model.add(MaxPooling2D(2,2))
    model.add(BatchNormalization(axis = chanDim))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(512, activation = 'relu'))
    model.add(BatchNormalization(axis = chanDim))
    model.add(Dropout(0.5))
    model.add(Dense(classes, activation = 'softmax'))
return model
```

In [14]:

```
#instantiate the model
height = 250
width = 250
classes = 2
channels = 3
model = CNNbuild(height = height, width = width, classes = classes, channels = c
hannels)
model.summary()
```

```
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rat
e = 1 - keep_prob`.
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 248, 248, 32)      896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 124, 124, 32)      0
_________________________________________________________________
batch_normalization_1 (Batch (None, 124, 124, 32)      128
_________________________________________________________________
dropout_1 (Dropout)          (None, 124, 124, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 122, 122, 32)      9248
_________________________________________________________________
max pooling2d 2 (MaxPooling2 (None, 61, 61, 32)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 61, 61, 32)        128
_________________________________________________________________
dropout_2 (Dropout)          (None, 61, 61, 32)        0
```

```
_________________________________________________________________
flatten_1 (Flatten)          (None, 119072)            0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               60965376
_________________________________________________________________
batch_normalization_3 (Batch (None, 512)               2048
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 1026
=================================================================
Total params: 60,978,850
Trainable params: 60,977,698
Non-trainable params: 1,152
_________________________________________________________________
```

In [15]:

```
#compile the model
opt = Adam(lr=0.0001)
model.compile(loss = 'categorical_crossentropy', optimizer = opt, metrics = ['ac
curacy'])
```

In [16]:

```
#fit the model onto the dataset
h = model.fit(x_train, y_train, epochs = 10, batch_size = 32)
model.save('my_model.h5')
```

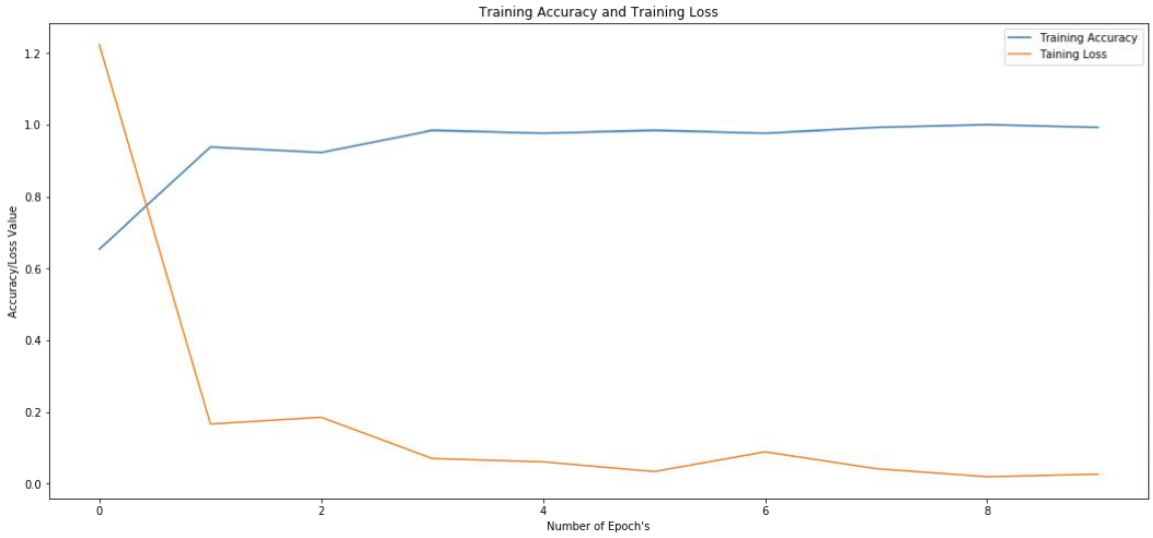
```
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Epoch 1/10
130/130 [==============================] - 22s 169ms/step - loss: 1.
2231 - acc: 0.6538
Epoch 2/10
130/130 [==============================] - 10s 77ms/step - loss: 0.1
661 - acc: 0.9385
Epoch 3/10
130/130 [==============================] - 19s 146ms/step - loss: 0.
1851 - acc: 0.9231
Epoch 4/10
130/130 [==============================] - 21s 159ms/step - loss: 0.
0699 - acc: 0.9846
Epoch 5/10
130/130 [==============================] - 19s 149ms/step - loss: 0.
0604 - acc: 0.9769
Epoch 6/10
130/130 [==============================] - 28s 214ms/step - loss: 0.
0334 - acc: 0.9846
Epoch 7/10
130/130 [==============================] - 29s 220ms/step - loss: 0.
0884 - acc: 0.9769
Epoch 8/10
130/130 [==============================] - 19s 147ms/step - loss: 0.
0414 - acc: 0.9923
Epoch 9/10
130/130 [==============================] - 18s 136ms/step - loss: 0.
0189 - acc: 1.0000
Epoch 10/10
130/130 [==============================] - 16s 122ms/step - loss: 0.
0261 - acc: 0.9923
```

In [17]:

```
plt.figure(figsize = (18,8))
plt.plot(range(10), h.history['acc'], label = 'Training Accuracy')
plt.plot(range(10), h.history['loss'], label = 'Taining Loss')
#ax1.set_xticks(np.arange(0, 31, 5))
plt.xlabel("Number of Epoch's")
plt.ylabel('Accuracy/Loss Value')
plt.title('Training Accuracy and Training Loss')
plt.legend(loc = "best")
```

Out[17]:

```
<matplotlib.legend.Legend at 0x1a568e99e8>
```



In [18]:

```
130/130 [==============================] - 3s 27ms/step
```

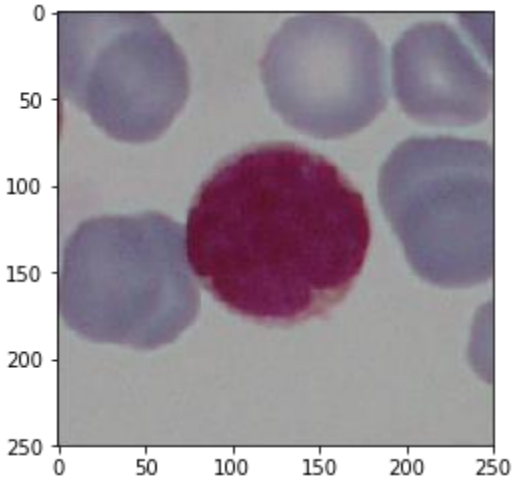
In [19]:

```
LOSS : 0.527237777870435
ACCURACY : 0.8461538461538461

#evaluate the model on test data
predictions = model.evaluate(x_test, y_test)


print(f'LOSS : {predictions[0]}') print(f'ACCURACY : {predictions[1]}')
```

In [25]:

```
import cv2
import tensorflow as tf from PIL import Image

CATEGORIES = ["Infected", "Uninfected"]

def prepare(filepath): IMG_SIZE = 250;

#

img_array = cv2.imread(filepath)
img_resize = cv2.resize(img_array,(250,250),interpolation=cv2.INTER_AREA)
img_array = img_to_array(img_resize)
plt.imshow(img_resize)
plt.show()
  return img_resize

return img_resize.reshape(-3, IMG_SIZE, IMG_SIZE, 3)

IMG_SIZE = 50;
# model = tf.keras.models.load_model("my_model.h5")
# prediction = model.predict([data[22000].reshape(-1, IMG_SIZE, IMG_SIZE, 3)])
```

```
prediction = model.predict(prepare('Im003_1.tif'))
print(prediction) # will be a list in a list.

for pred in prediction: print(CATEGORIES[int(round(pred[0]))]) #
print(int(pred[0]))
```




```
[[0.03920074 0.9607993 ]]
Infected
```

Support Vector Machine (SVM)

In [106]:

```
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import pickle
import random
from sklearn.model_selection import train_test_split from sklearn.svm import SVC
import seaborn as sns
from PIL import Image
from keras.preprocessing.image import img_to_array from
keras.preprocessing.image import load_img from keras.utils import np_utils
```

```
Using TensorFlow backend.
```

In [110]:

```
dir = '../enocholutunmida/Downloads/ALL_IDB2'
categories = ['infected', 'uninfected']
data = []
for category in categories:
path = os.path.join(dir, category) label = categories.index(category)
for img in os.listdir(path):
imgpath = os.path.join(path, img) cell_img = cv2.imread(imgpath, 0) cell_img =
cv2.resize(cell_img, (50,50)) image = np.array(cell_img).flatten()
        data.append([image, label])
```

In [111]:

```
print(len(data))
```

In [142]:

```
random.shuffle(data)
features = []
labels = []
for feature , label in data: features.append(feature) labels.append(label)
```

In [143]:

```
xtrain, xtest, ytrain, ytest = train_test_split(features, labels, test_size= 0.2
5)
```

In [150]:

```
model = SVC(C=1, kernel = 'poly', gamma = 'auto')
model.fit(xtrain, ytrain)
prediction = model.predict(xtest)
accuracy = model.score(xtest, ytest)
categories = ['infected', 'uninfected']
print('Accuracy:', accuracy)
print('Prediction is:', categories[prediction[6]])
img_cell = xtest[14].reshape(50,50)
plt.imshow(img_cell, cmap='gray')
plt.show()
```

```
Accuracy: 0.7076923076923077
Prediction is: infected
```

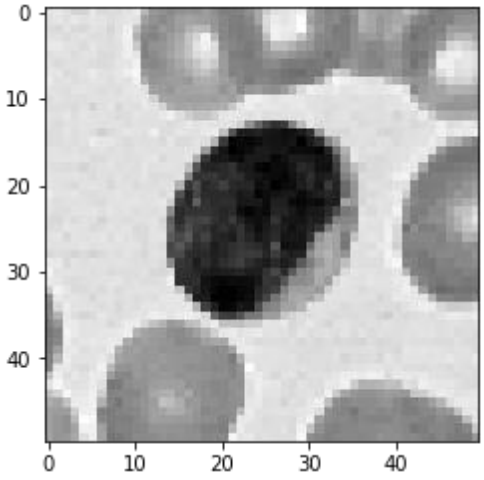


IMAGE PROCESSING

RGB TO GRAYSCALE

```
import cv2

import numpy as np

from matplotlib import pyplot as plt

import seaborn as sns

import os

from PIL import Image

img_data = os.listdir('../Downloads/ALL_IDB1/im')

img = cv2.imread('../Downloads/ALL_IDB1/im' + "/" + img_data[1], 1)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

cv2.imshow("Original Image", gray)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

IMAGE ENHANCING

```
import cv2

import numpy as np

from matplotlib import pyplot as plt

import seaborn as sns

import os

from PIL import Image

img_data = os.listdir('../Downloads/ALL_IDB1/im')
```

```
img = cv2.imread('../Downloads/ALL_IDB1/im' + "/" + img_data[1], 1)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

median = cv2.medianBlur(gray, 3)

cv2.imshow("Median Image", median)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

EDGE DETECTION

```
import cv2

import numpy as np

from matplotlib import pyplot as plt

import seaborn as sns

import os

from PIL import Image

img_data = os.listdir('../Downloads/ALL_IDB1/im')

img = cv2.imread('../Downloads/ALL_IDB1/im' + "/" + img_data[1], 1)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

median = cv2.medianBlur(gray, 3)

edges = cv2.Canny(median,60,120)

cv2.imshow("Edge Detection", edges)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

THESHOLDING AND SEGMENTATION

```
import cv2

import numpy as np

from matplotlib import pyplot as plt

import seaborn as sns

import os

from PIL import Image

img_data = os.listdir('../Downloads/ALL_IDB1/im')

img = cv2.imread('../Downloads/ALL_IDB1/im' + "/" + img_data[1], 1)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

median = cv2.medianBlur(gray, 3)

edges = cv2.Canny(median,60,120)

ret,th = cv2.threshold(median,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

cv2.imshow("Thresholded Image", th)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

FEATURE EXTRACTION

```
import numpy as np

import cv2

import pandas as pd
```

```
import os


#img = cv2.imread('BSE_Image.jpg')


img_data = os.listdir('../Downloads/ALL_IDB1/im')


img = cv2.imread('../Downloads/ALL_IDB1/im' + "/" + img_data[1])

img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

#Here, if you have multichannel image then extract the right channel instead of converting the image to grey.

#For example, if DAPI contains nuclei information, extract the DAPI channel image first.


#Multiple images can be used for training. For that, you need to concatenate the data


#Save original image pixels into a data frame. This is our Feature #1.

img2 = img.reshape(-1)

df = pd.DataFrame()

df['Original Image'] = img2


#Generate Gabor features

num = 1

kernels = []

for theta in range(2):

    theta = theta / 4. * np.pi
```

```
    for sigma in (1, 3):  #Sigma with 1 and 3
        for lamda in np.arange(0, np.pi, np.pi / 4):   #Range of wavelengths
            for gamma in (0.05, 0.5):   #Gamma values of 0.05 and 0.5



                gabor_label = 'Gabor' + str(num)
                print(gabor_label)
                ksize=9
                kernel = cv2.getGaborKernel((ksize, ksize), sigma, theta, lamda, gamma, 0, ktype=cv2.CV_32F)
                kernels.append(kernel)
                #Now filter the image and add values to a new column
                fimg = cv2.filter2D(img2, cv2.CV_8UC3, kernel)
                filtered_img = fimg.reshape(-1)
                df[gabor_label] = filtered_img
                print(gabor_label, ': theta=', theta, ': sigma=', sigma, ': lamda=', lamda, ': gamma=', gamma)
                num += 1  #Increment for gabor column label


#CANNY EDGE
edges = cv2.Canny(img, 100,200)   #Image, min and max values
edges1 = edges.reshape(-1)
df['Canny Edge'] = edges1 #Add column to original dataframe


from skimage.filters import roberts, sobel, scharr, prewitt
```

```
#ROBERTS EDGE

edge_roberts = roberts(img)

edge_roberts1 = edge_roberts.reshape(-1)

df['Roberts'] = edge_roberts1


#SOBEL

edge_sobel = sobel(img)

edge_sobel1 = edge_sobel.reshape(-1)

df['Sobel'] = edge_sobel1


#SCHARR

edge_scharr = scharr(img)

edge_scharr1 = edge_scharr.reshape(-1)

df['Scharr'] = edge_scharr1


#PREWITT

edge_prewitt = prewitt(img)

edge_prewitt1 = edge_prewitt.reshape(-1)

df['Prewitt'] = edge_prewitt1


#GAUSSIAN with sigma=3

from scipy import ndimage as nd
```

```
gaussian_img = nd.gaussian_filter(img, sigma=3)

gaussian_img1 = gaussian_img.reshape(-1)

df['Gaussian s3'] = gaussian_img1


#GAUSSIAN with sigma=7

gaussian_img2 = nd.gaussian_filter(img, sigma=7)

gaussian_img3 = gaussian_img2.reshape(-1)

df['Gaussian s7'] = gaussian_img3


#MEDIAN with sigma=3

median_img = nd.median_filter(img, size=3)

median_img1 = median_img.reshape(-1)

df['Median s3'] = median_img1


#VARIANCE with size=3

variance_img = nd.generic_filter(img, np.var, size=3)

variance_img1 = variance_img.reshape(-1)

df['Variance s3'] = variance_img1  #Add column to original dataframe


#Import the labeled/masked image

labeled_img = cv2.imread('../Downloads/ALL_IDB1/im' + "/" + img_data[1])

#Remember that you can load an image with partial labels

# drop the rows with unlabeled data
```

```
labeled_img = cv2.cvtColor(labeled_img, cv2.COLOR_BGR2GRAY)

labeled_img1 = labeled_img.reshape(-1)

df['Labels'] = labeled_img1


print(df.head())


#df.to_csv("Gabor.csv")


#Define the dependent variable that needs to be predicted (labels)

Y = df["Labels"].values


#Define the independent variables

X = df.drop(labels = ["Labels"], axis=1)


#Split data into train and test to verify accuracy after fitting the model.

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.4, random_state=20)



# Import the model we are using

#RandomForestRegressor is for regression type of problems.

#For classification we use RandomForestClassifier.

#Both yield similar results except for regressor the result is float
```

#and for classifier it is an integer.

RANDOM FOREST CLASSIFIER

```
from sklearn.ensemble import RandomForestClassifier

# Instantiate model with n number of decision trees

model = RandomForestClassifier(n_estimators = 100, random_state = 42)
```

SUPPORT VECTOR MACHINE (SVM)

```
# Training the SVM to compare against Random Forest

from sklearn.svm import LinearSVC

model = LinearSVC(max_iter=100)  #Default of 100 is not converging


# Train the model on training data

model.fit(X_train, y_train)


# verify number of trees used. If not defined above.

#print('Number of Trees used : ', model.n_estimators)


#First test prediction on the training data itself.

prediction_test_train = model.predict(X_train)


#Test prediction on testing data.
```

```
prediction_test = model.predict(X_test)


#Let us check the accuracy on test data

from sklearn import metrics


#Print the prediction accuracy

print ("Accuracy on training data = ", metrics.accuracy_score(y_train, prediction_test_train))


#Check accuracy on test dataset.

print ("Accuracy = ", metrics.accuracy_score(y_test, prediction_test))


feature_list = list(X.columns)

feature_imp = pd.Series(model.feature_importances_,index=feature_list).sort_values(ascending=False)

print(feature_imp)


# store the model for future use.

#Train on training images, validate on test images and deploy the model on unknown images.

import pickle


#Save the trained model as pickle string to disk for future use

filename = "sandstone_model"

pickle.dump(model, open(filename, 'wb'))
```

```
#To test the model on future datasets

loaded_model = pickle.load(open(filename, 'rb'))

result = loaded_model.predict(X)

segmented = result.reshape((img.shape))

from matplotlib import pyplot as plt

plt.imshow(segmented, cmap ='jet')
```

API ENDPOINT

DIAGNOSE API

```
@app.route('/diagnose', methods=['POST'])

def main():

    if flask.request.method == 'POST':

        image_data = flask.request.form['image_data']

        prediction = model.predict(image_data)

        return flask.render_template('main.html',

                                original_input={'image_data':image_data},

                                result=prediction,

                                )
```

AUTHENTICATION API

```
@app.route('/api/users', methods = ['POST'])
def new_user():
    username = request.json.get(‘user_id’)
    password = request.json.get(‘pass_id’)
    user = User(user_id = username)
    user.hash_password(password)
    db.session.add(user)
    db.session.commit()
    return jsonify({ 'username': user.user_id }), 201, {'message': ‘successful’})
```

USER MODEL

```
class User(db.Model):
    __tablename__ = 'users'
    id = db.Column(db.Integer, primary_key = True)
    user_id = db.Column(db.String(32), index = True)
    password_hash = db.Column(db.String(128))
```