CLASSIFICATION OF BLOOD CELL USING CNN

A Project report submitted in partial fulfillment of the requirement for the award of the degreeof

MASTER OF TECHNOLOGY IN
COMPUTER SCIENCE AND TECHNOLOGY
WITH SPECIALIZATION IN COMPUTER NETWORKS

Submitted by
BANDARU NIKITH KUMAR
(Regd.No.320206418001)

Under the Supervision of
PROF. M. SHASHI

Department of Computer Science and Systems EngineeringAndhra University College of Engineering (A)

DEPARTMENT OF COMPUTER SCIENCE AND SYSTEMS ENGINEERING,ANDHRA UNIVERSITY COLLEGE OF ENGINEERING (A),
ANDHRA UNIVERSITY, VISAKHAPATNAM -530003
2020-2022

ANDHRA UNIVERSITY COLLEGE OF ENGINEERING (A)

DEPARTMENT OF COMPUTER SCIENCE AND SYSTEMS ENGINEERING

CERTIFICATE

This is to certify that the main project entitled "CLASSIFICATION OF BLOOD CELL USING CNN " is the bonafide project work carried out by BANDARU NIKITH KUMAR, bearing Roll No: 320206418001 in partial fulfillment of the requirement for the Award of Degree of Master of Technology with specialization in computer networks in Computer Science and Technology from the department of Computer Science and Systems Engineering, Andhra University College of Engineering (A),Andhra University, Visakhapatnam during the year 2020 – 2022.

Signature of Project Guide	**Signature of Head of the Department**
Prof. M. Shashi	**Prof. Kujam Nageswara Rao**

DECLARATION

I BANDARU NIKITH KUMAR, hereby declare that the project entitled "CLASSIFICATION OF BLOOD CELL USING CNN", submitted to the Department of Computer Science and Systems Engineering, Andhra University College of Engineering (A), Andhra University, was done by me in partial fulfillment of the requirement for the award of Degree of Master of Technology in Computer Science and Technology with specialization in computer networks by ANDHRA UNIVERSITY, Visakhapatnam.

BANDARU NIKITH KUMAR
Regd. No 320206418001

Place: Visakhapatnam
Date :

ACKNOWLEDGEMENT

Firstly, I would like to convey my sincere thanks to my project guide Prof. M. Shashi for her valuable guidance, supervision and official support throughout the project, which was greatly helpful in progression and smoothness of the entire project work.

I think Prof. Kunjam Nageswara Rao, Head of the Department of Computer Science and Systems Engineering, who has encouraged us a lot and also his extremely valuable guidance and advice.

I would also like to convey my gratitude to our beloved principal Prof. P. Srinivas Rao for his encouragement in the aspect of our course.

I also extend my gratitude to all the teaching and non-teaching staff of the Department of Computer Science and Systems Engineering for their support in completing the project. I also thank my parents and friends, who have willingly helped me out with their abilities, in completing theproject work.

BANDARU NIKITH KUMAR

ABSTRACT

Deep Learning has already shown power in many application fields, and is accepted by more and more people as a better approach than the traditional machine learning models. In particular, the implementation of deep learning algorithms, especially Convolution Neural Networks (CNN), brings huge benefits to the medical field, where a huge number of images are to be processed and analysed. This paper aims to develop a deep learning model to address the blood cell classification problem, which is one of the most challenging problems in blood diagnosis. A CNN-based framework is built to automatically classify the blood cell images into subtypes of the cells. Experiments are conducted on a dataset of 13k images of blood cells with their subtypes, and the results show that our proposed model provide better results in terms of evaluation parameters.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 CLASSIFICATION OF BLOOD CELL USING CNN

The blood test plays imperative part in recognizing infections. It gives the points of interest approximately the common state of a person’s prosperity conditions. Based on information from blood test pros, they select the treatment for the quiet required. White blood cells (WBC) are an imperative component of blood framework. It is fundamental for great wellbeing and security against malady. WBC contains in a general sense five parts and depends on its assortment of measure, number and shape. Based on variety in these highlights there can happen numerous illnesses.

Blood passes on basic substance to all standard of the body, it could be a transporting fluid. The white blood cell accept basic portion in immunity of the body. The white blood cell is something else called leukocytes which are nucleated cell conveyed from bone marrow. The major obligation of WBC is to battle disease and cancer. Monocytes, neutrophils, basophiles, eosinophil’s, lymphocytes are basically sorts of leukocytes. Each kind of leukocyte has there on properties for ID like tally, shape, morphological changes. When a person’s advise with master for check-up or sickness concern you've got, your master may approach criticalinformation from your blood test.

About 90 rate of the data within the normal therapeutic chart comes from research facility information. Within the occasion of developments and distinctive perilous maladies, blood tests deliver strong signs around how the body is working. To recognizing the contaminations take more noteworthiness of blood testing is legitimately basic. Pros see the total blood tally for tall or moo tallies of diverse blood cells additionally for irregular blood cells.

For recognizable confirmation of sicknesses most commonly utilized two frameworks, they are include up to WBC check and check of each kind of WBC, which is known as differential number or diff test. WBC number can be discovered by utilizing manual strategy and laser based cytometer.

1.2 PROBLEM DEFINITION

Computerized strategies to recognize and arrange platelet subtypes have significant therapeutic applications. Intertwined CNN model which combines the element maps of Convolutional layers to bolster into completely associated Neural Network (NN) model to arrange the WBC picture. WBC contains on a very basic level five sections and depends on its assortment of size, check, shape. In light of variety in these highlights there can happen numerous illnesses. By and by, we have stood up to numerous issues in blood testing. One normal issue is that a one of a kind blood was getting a substitute and unprecedented estimation of cell check from the assorted lab professional. The vast majority of the research center pursues the manual checking method, which is extremely monotonous and less exact. The proposed structures help to characterization of every sort of white platelets using multi class bolster vector machine arrangement and convolutional neural systems (CNN). Recognizing the Neutrophils, Lymphocytes, Monocytes, Eosinophil and Basophils variety can be distinguished utilizing profound learning for finding infections. In like manner, find the degree of threatening cell in blood and leukocytes for various age area.

1.3 PROJECT PURPOSE

The purpose of this project is to extract knowledge about classification of blood cell. The proposed blood cell classification method has four major contributions compared to the existing methods as follows.

- This application of the model is CNN in the classification of blood cells. This model can effectively use the temporal and spatial characteristics of information to achieve better classification results.
- Using the pre-trained method of the CNN model, its chosen weights are close to a good local optimum, so that they are kept within a high gradient range and they can be effectively fine-tuned.
- CNN is a neural network that captures dynamic information in serialized data by hiding the periodic connections of nodes in layers, and can classify serialized data. Different from other forward neural networks, CNN can save the context state, and even can store in any arbitrarily long context window, learn and express relevant information, and is no longer limited to the spatial boundaries of traditional neural networks. Based on this theory, we introduce a CNN with memory function to generate a continuous time outputstate for features extracted from CNN.
- The proposed method adjusts the loss function and activation function of CNN, and Adam optimizers to train the network. At the same time, we have also introduced a fine-tuning strategy to train the CNN framework. The experimental results show that this method has achieved good results.

1.4 PROJECT FEATURES

The dataset is increased by rotating the images randomly. This is one kind of data argumentation; the total number of blood cell images is 12444 where 9957 images will be training data and 2487 will be testing data. Before and after argumentation we will show the difference in the images, this type of data set is called imbalance data distribution which can force any trained model to high distribution of data. To balance the dataset, the random rotation will be performed and after this, each type of WBC image is almost equally distributed.

We will design a fused CNN model to inspired by a concept paper for WBC Classification. We will use overlapping max-pooling layer, constant number of filters in each convolutional layer and avoiding normalization layer where we are not planning to use.

We will try to build up such a model that can reduce the FP and FN for each type of WBC image on the test data. For evaluating the quality of model, we will measure accuracy, precision, and the recall of the predictions on the test data set.

CHAPTER 2

LITERATURE SURVEY

A number of experimental investigations have been carried out by using partial replacement of basalt fiber in concrete. The study of various literatures on use of basalt fiber in concrete as below.

Image to Speech Conversion for Visually Impaired (International Journal of Latest Research in Engineering and Technology-2017) [3]

Asha G. Hagargund carried out a work and they concluded that the basic framework is an embedded system that captures an image, extracts only the region of interest(i.e. region of the image that contains text) and converts that text to speech. It is implemented using a Raspberry Pi and a Raspberry Pi camera. The captured image undergoes a series of image pre-processing steps to locate only that part of the image that contains the text and removes the background. Two tools are usedconvert the new image (which contains only the text) to speech. They are OCR (Optical Character Recognition) software and TTS (Text-to-Speech) engines. The audio output is heard through the raspberry pi ˃ s audio jack using speakers or earphones.

OCR based automatic book reader for the visually impaired using RaspberryPI (International Journal of Innovative Research in Computer and Communication Engineering - 2016) [8]

Aaron James S carried out a work and they concluded that Optical character recognition (OCR) is the identification of printed characters using photoelectric devices and computer software. It coverts images of typed, handwritten or printed text into machine encoded text from scanned document or from subtitle text superimposed on an image. In this research these images are converted into audio output. OCR is used in machine process such as cognitive computing, machine translation, text to speech, key data and text mining. It is mainly used in the field of research in Character recognition, Artificial intelligence and computer vision. In this research, as the recognition process is image processing based Multilingual Translator for visually impaired And Travelers Using OCR on Raspberry PI done using OCR the character code in text files are processed using Raspberry Pi device on which it recognizes character using tesseract algorithm and python programming and audio output is listened. To use OCR for pattern recognition to perform Document image analysis (DIA) we use information in grid format in virtual digital libraries design and construction. This research mainly focuses on the OCR based automatic book reader for the visually impaired using Raspberry PI. Raspberry PI features a Broad com system on a chip (SOC) which includes ARM compatible CPU and an on chip graphics-processing unit GPU. It promotes Python programming as main programming language with support for BBC BASIC.

A Smart Reader for Visually Impaired People Using Raspberry PI (International Journal of Engineering Science and Computing – 2016) [11]

D. Velmurugan carried a work and they concluded that this work proposes a smart reader for visually challenged people using Raspberry Pi. This report addresses the integration of a complete Text Read-out system designed for the visually challenged. The system consists of a webcam interfaced with raspberry pi which accepts a page of printed text. The OCR (Optical Character Recognition) package installed in raspberry pi scans it into a digital document which is then subjected to skew correction, segmentation, before feature extraction to perform classification. Once classified, the text is readout by a text to speech conversion unit (TTS engine) installed in Raspberry Pi. The output is fed to an audio amplifier before it is read out. The simulation for the proposed project can be done in MATLAB. The simulation is just an initiation of image processing, the image to text conversion and text to speech conversion done by the OCR software installed in raspberry pi. The system finds interesting applications in libraries, auditoriums, offices where instructions and notices are to be read and also in assisted filling of application forms. Though there are many existing solutions to the problem of assisting individuals who are blind to read, however none of them provides a reading experience that in any way parallels that of the sighted population. In particular, there is a need for a portable text reader that is affordable and readily available to the blind community.

Camera based Text to Speech Conversion, Obstacle and Currency Detectionfor Blind Persons (Indian Journal of Science and Technology – 2016) [13]

J. K. R. Sastry carried out a work the main object of this report is to present an innovated system that can help the blind for handling currency. Methods/Statistical Analysis: Many image processing techniques have been used to scan the currency, remove the noise, mark the region of interest and convert the image into text and then to sound which can be heard by the blind. The entire system is implemented by using Raspberry Pi Micro controller based system. In the prototype model an IPR sensor is used instead of camera for sensing the object. Findings: In this report a novel method has been presented using which one can recognize the object, mark the interesting region within the object, scan the text and convert the scanned text into binary characters through optical recognition. A second method has been presented using which the noise present in the scanned image is eliminated before characters are recognized. A third method that can be used to convert the recognized characters into e-speech through pattern matching has also be presented. Applications: An embedded system has been developed based on ARM technology which helps the blind persons to read the currency notes. All the methods presented in this project have been implemented within an embedded application. The embedded board has been tested with different currency notes and the speech in English has been generated that identify the value of the currency.

2.2 EXISTING SYSTEM

Some of the existing systems are:-

Literature review of the problem shows that there have been several approaches to address the issue of blood cells recognition in image using several different methods. A traditional cell classification process that is characterized by manually segmented cytoplasm/nuclei. In contrast, our method can adaptively learn the feature of the input image and is therefore not limited by the cell segmentation or feature design. The traditional method of cell classification is greatly affected by the balance of the sample, which causes the classifier to predict more cell abnormalities.

Disadvantages:

This approach has the drawback of having intermediary blood cells which may lead to ambiguity in training.

Performance is not worthy.

It is time consumption.

2.3 PROPOSED SYSTEM

The proposed model can adaptively learn the characteristics of the blood cell images; thereby avoiding manual feature extraction (no cytoplasmic/nuclear segmentation is needed). Moreover, we used of CNN models to classify blood cells. In particular, we use the transfer learning method to migrate the weight parameters pre-trained on the ImageNet dataset to the CNN branch, thereby enhancing the robustness of the combined model and accelerating the convergence of the model.

Combination model uses the characteristics of the nural neural network to use CNN classify cells. This method enables our model to take into account the spatiotemporal characteristics of the information contained in the image, and can effectively learn the structural characteristics of blood cell images, which will result in higher classification accuracy.

Our method with Xception and achieved a good classification effect in the experiment. As is well- known, the Xception structure is a linear stack of depthwise separable convolution layers with residual connections. Unlike Inception, Xception does not divide the input data into several compressed data blocks, but instead maps the spatial correlation for each output channel and then performs a 1×1 depth convolution to obtain a cross-channel correlation. Xception introduces a deeply separable convolution, which basically does not increase the complexity of the network under the premise of improving the model. Therefore, our combined model also has the same features as Xception. In addition, the neural network attention mechanism we introduce incorporates features from the CNN branch, allowing the model to be more focused on finding useful information in the input data that is relevant to the current output, and thereby improving the quality of the output. The experiment confirmed the validity of our proposal.

In addition, compared with some classical neural network models (ResNet, GoogleNet, etc.), our combined model can make full use of the spatiotemporal information of image features. Therefore, our new method has great application potential in the field of blood cell image classification.

Advantages:

The main advantage we can take of previous training and use a relatively small trainingset.

The proposed method performed well compared to all the existing methods.

Execution time is less.

2.4 SOFTWARE DESCRIPTION

2.4.1 Python

Python is an object-oriented programming language created by Guido Rossum in 1989. It is ideally designed for rapid prototyping of complex applications. It has interfaces to many OS system calls and libraries and is extensible to C or C++. Many large companies use the Python programming language include NASA, Google, YouTube, Bit Torrent, etc.

Python is widely used in Artificial Intelligence, Natural Language Generation, Neural Networks and other advanced fields of Computer Science. Python had deep focus on code readability & this class will teach you python from basics.

Characteristics of Python

- It provides rich data types and easier to read syntax than any other programminglanguages
- It is a platform independent scripted language with full access to operating system API's
- Compared to other programming languages, it allows more run-time flexibility
- It includes the basic text manipulation facilities of Perl and Awk
- A module in Python may have one or more classes and free functions
- Libraries in Pythons are cross-platform compatible with Linux, MacIntosh, and Windows

2.4.2 Jupyter Notebook

The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting data science projects. This article will walk you through how to set up Jupyter Notebooks on your local machine and how to start using it to do data science projects.

First, though: what is a “notebook”? A notebook integrates code and its output into a single

document that combines visualizations, narrative text, mathematical equations, and other rich media. This intuitive workflow promotes iterative and rapid development, making notebooks an increasingly popular choice at the heart of contemporary data science, analysis, and increasingly science at large.Best of all, as part of the open source Project Jupyter, they are completely free.

The Jupyter project is the successor to the earlier IPython Notebook, which was first published as a prototype in 2010. Although it is possible to use many different programming languages within Jupyter Notebooks, this article will focus on Python as it is the most common use case. (Among R users, R Studio tends to be a more popular choice).

To get the most out of this tutorial you should be familiar with programming, specifically Python and pandas specifically. That said, if you have experience with another language, the Python in this article shouldn’t be too cryptic, and will still help you get Jupyter Notebooks set up locally. Jupyter Notebooks can also act as a flexible platform for getting to grips with pandas and even Python, as will become apparent in this article.

We will:

- ➢ Cover the basics of installing Jupyter and creating your first notebook
- ➢ Delve deeper and learn all the important terminology

- Explore how easily notebooks can be shared and published online. Indeed, this article *is* a Jupyter Notebook! Everything here was written in the Jupyter Notebook environment, though you are viewing it in a read-only form.

2.4.3 The Library & Packages

- **Open CV:**

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. 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 invideos, 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 14 million. The library is used extensively in companies, research groups and by governmental bodies. 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 CUDAand 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. OpenCV's application areas include :

- 2D and 3D feature toolkits
- Egomotion estimation
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics
- Motion understanding
- Object identification
- Segmentation and recognition
- Stereopsis stereo vision: depth perception from 2 cameras
- Structure from motion (SFM)
- Motion tracking
- Augmented reality

To support some of the above areas, Open CV includes a statistical machine learning library thatcontains :

- Boosting
- Decision tree learning
- Gradient boosting trees
- Expectation-maximization algorithm
- k-nearest neighbor algorithm
- Naive Bayes classifier
- Artificial neural networks
- Random forest
- Random forest
- Support vector machine (SVM)
- Deep neural networks (DNN)

❖ **Numpy:**

Numpy is an acronym for "Numeric Python" or "Numerical Python". It is an open source extension module for Python, which provides fast precompiled functions for mathematical and numerical routines. Furthermore, NumPy enriches the programming language Python with powerful data structures for efficient computation of multi-dimensional arrays and matrices. The implementation is even aiming at huge matrices and arrays. Besides that the module supplies a large library of high-level mathematical functions to operate on these matrices and arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions

- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier Transform, and random number capabilities.

❖ **Numpy Array:**

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

❖ **SciPy:**

SciPy (Scientific Python) is often mentioned in the same breath with NumPy. SciPy extends the capabilities of NumPy with further useful functions for minimization, regression, Fourier- transformation and many others. NumPy is based on two earlier Python modules dealing with arrays. One of these is Numeric. Numeric is like NumPy a Python module for high-performance, numeric computing, but it is obsolete nowadays. Another predecessor of NumPy is Numarray, which is a complete rewrite of Numeric but is deprecated as well. NumPy is a merger of those two, i.e. it is build on the code of Numeric and the features of Numarray.

❖ **Keras:**

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Keras contains numerous implementations of commonly used neural network building blocks suchas layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier. Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep learning models on clusters of Graphics Processing Units (GPU).

❖ TensorFlow:

Tensor Flow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of Tensor Flow.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FUNCTIONAL REQUIREMENTS

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behaviour, and outputs. Functional requirements may be data manipulation, processing, technical details and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Here, the system has to perform the following tasks:

GUI can select the blood cells using the web application.

The blood cells image is processed and features are extracted using deep learning.

- ➢ Can detect the blood cells.
- ➢ Application can send the instruction to the deep learning model.
- ➢ Can process the instruction model.
- ➢ The web app to current blood cells image is select with GUI.
- ➢ The blood cells status is displayed the classification and detection.

3.2 NON-FUNCTIONAL REQUIREMENTS

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional

requirements is detailed in the system design. The plan for implementing non-functionalrequirements is detailed in the system architecture.

- **Usability**

The client acknowledge be typical nearly the buyer interfaces and committed to ask forambush pressure in relocating to a unique framework with another condition.

- **Reliability**

The progressions made by the Programmer ought to be obvious both to the Project pioneerand in addition the Test design.

- **Security**

Counting bug following the framework must give important security and must secure theentire procedure from slamming.

- **Performance**

The framework will be facilitated on a solitary web server with a solitary database server outof sight, consequently execution turns into a noteworthy concern.

- **Portability**

This is required when the web server, which is facilitating the framework stalls out because ofa few issues, which requires their framework to be taken to another framework.

- **Reusability**

The framework ought to be separated into such modules that it could be utilized as a piece ofanother framework without requiring a lot of work.3.6 Technologies used.

3.2.1 ACCESSIBILITY

Accessibility is a general term used to describe the degree to which a product, device, service, or environment is accessible by as many people as possible.

3.2.2 MAINTAINABILITY

In software engineering, maintainability is the ease with which a software product can bemodified. In order to:-

- Correct defects
- Meet new requirements

3.2.3 SCALABILITY

System is capable of handling increase total throughput under an increased load when resources (typically hardware) are added.

System can work normally under situations such as low bandwidth and large number of users.

3.2.4 PORTABILITY

Portability is one of the key concepts of high-level programming. Portability is the software code base feature to be able to reuse the existing code instead of creating new code whenmoving software from an environment to another.

3.3 HARDWARE REQUIREMENTS

Processor	: Any Processor
above 500 MHzSystem type	: 64-bit
Operating system	
RAM	: 2 GB
Hard Disk	: 80 GB

3.4 SOFTWARE REQUIREMENTS

1.	Operating system	: Windows 7/8/10
2.	Programming Language	: Python
3.	Library	: Numpy, Matplotlib, Sckit-learn, Tensorflow, Keras, Joblib.
4.	Simulation tool / IDE	: Anaconda Navigator, Python IDE 3.7.4, (Jupyter Notebook).

CHAPTER 4

4.1 DESIGN GOALS

DESIGN

The main goal is to provide an easy platform for the blood cells classification and recognize the infected or normal. The project work focuses on the blood cell using image processing. Previously, the classification is done manually. The manual process takes lot of time and it does not give accurate results. We are designing a system to overcome them. To classify the blood cells and it provides results with more accuracy.

- It gives the classification of blood cells. The count of the blood cell is provided long withthat.
- This can be affordable for everyone. .

4.2 SYSTEM ARCHITECTURE

The architectural configuration procedure is concerned with building up a fundamental basic system for a framework. It includes recognizing the real parts of the framework and interchanges between these segments. The beginning configuration procedure of recognizing these subsystems and building up a structure for subsystem control and correspondence is called construction modeling outline and the yield of this outline procedure is a portrayal of the product structural planning. The proposed architecture for this system is given below. It shows the way this system is designed and brief working of the system.

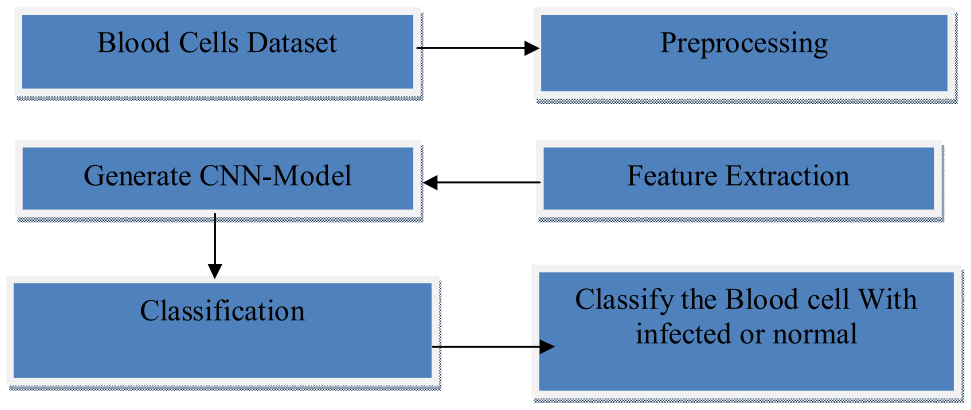


Figure 4.1 Architecture

4.3 DATA FLOW DIAGRAM

The DFD is straightforward graphical formalism that can be utilized to speak to a framework as far as the info information to the framework, different preparing did on this information and the yield information created by the framework. A DFD model uses an exceptionally predetermined number of primitive images to speak to the capacities performed by a framework and the information stream among the capacities.

The principle motivation behind why the DFD method is so famous is most likely in light of the way that DFD is an exceptionally basic formalism- It is easy to comprehend and utilization. Beginning with the arrangement of abnormal state works that a framework performs, a DFD display progressively speaks to different sub capacities. Actually, any various leveled model is easy to get it.

The human personality is such that it can without much of a stretch see any progressive model of a framework in light of the fact that in a various leveled model, beginning with an extremely straightforward and unique model of framework, distinctive points of interest of a framework are gradually presented through the diverse orders. A data-flow diagram (DFD) is a graphical representation of the "stream" of information through a data framework. DFDs can likewise be utilized for the perception of information handling.

Fig 4.2: Data Flow diagram(level 0)

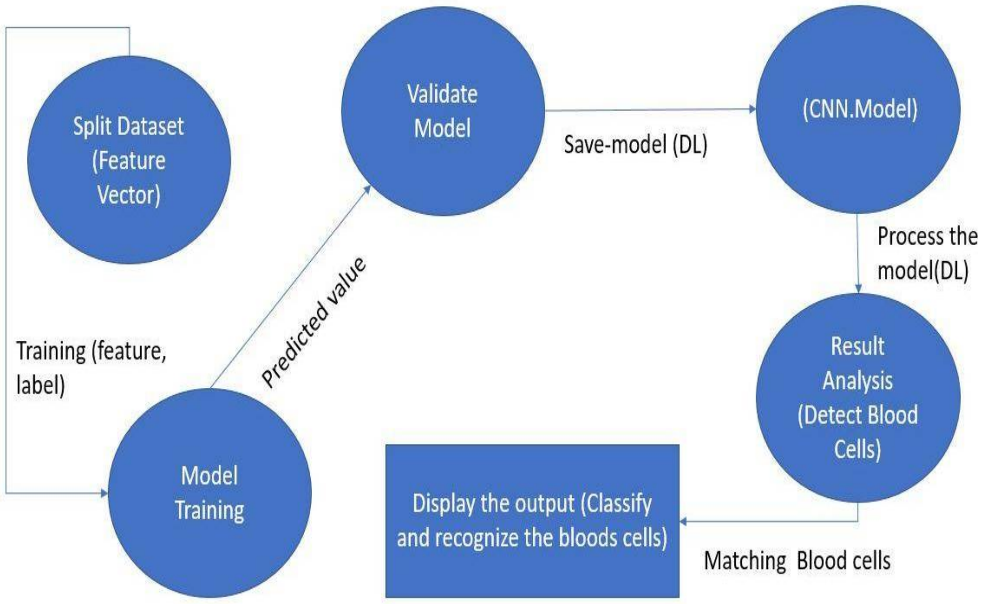

Split Dataset (Feature Vector)
Validate Model
Save-model (DL)
(CNN.Model)
Process the model(DL)
Training (feature, label)
Predicted value
Result Analysis (Detect Blood Cells)
Model Training
Display the output (Classify and recognize the bloods cells)
Matching Blood cells

Fig 4.3: Data Flow diagram(level 1)

4.4 SEQUENCE DIAGRAM

A sequence diagram is an integrated Modelling Language is a sort of communication diagram that shows procedures work with each other and in what request. Sequence diagrams are some of the time called occasion follow diagrams, occasion situations, and timing diagram. Sequence diagrams are utilized to formalize the conduct of the framework and to picture the correspondence among articles. They are valuable for recognizing extra questions that takes part in the utilization cases. A sequence diagram speaks to the associations that happen among these articles.

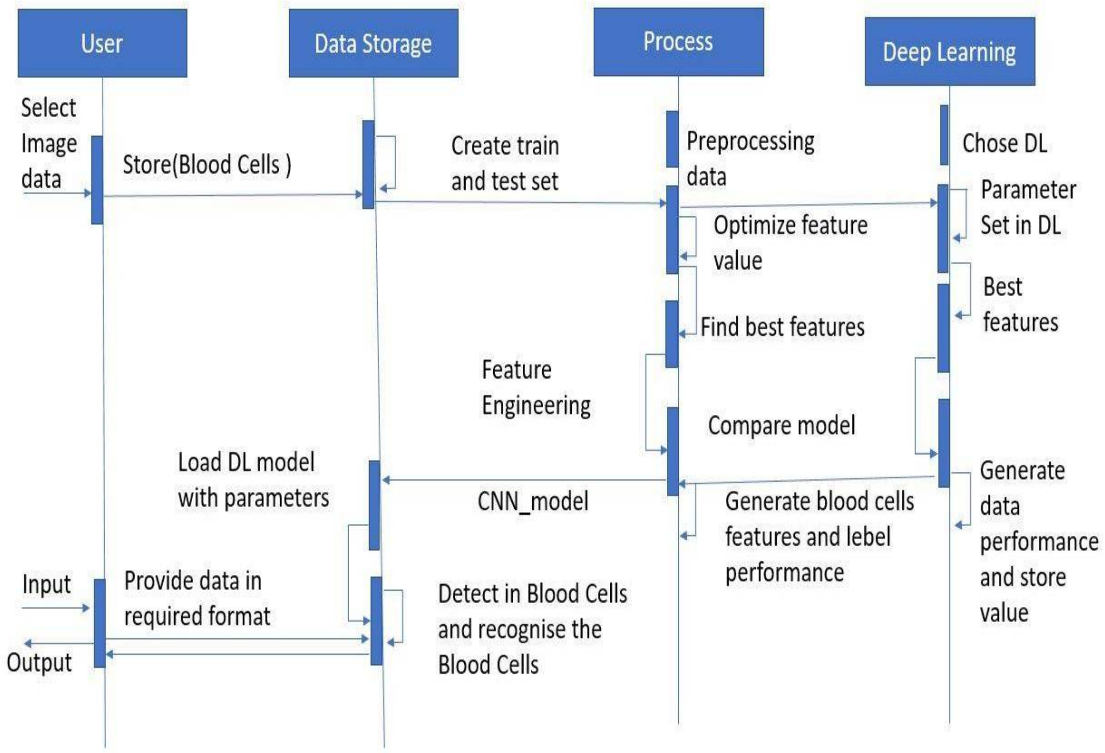

User
Data Storage
Process
Deep Learning
Select Image data
Store(Blood Cells)
Create train and test set
Preprocessing data
Chose DL
Parameter Set in DL
Optimize feature value
Best features
Find best features
Feature Engineering
Compare model
Load DL model with parameters
CNN_model
Generate blood cells features and lebel performance
Generate data performance and store value
Input
Provide data in required format
Output
Detect in Blood Cells and recognise the Blood Cells

Fig 4.4: Sequence diagram

4.5 USE CASE DIAGRAM

A use case chart is a kind of behavioral graph made from a Use-case examination. Its object is to present a graphical diagram of the usefulness gave by a framework regarding performers, their objectives (spoke to as utilization cases), and any conditions between those utilization cases. Use case chart gives us the data about how that clients and utilization cases are connected with the framework. Use cases are used amid prerequisites elicitation and examination to speak to the usefulness of the framework. Use cases concentrate on the conduct of the framework from an outside perspective.

A use case depicts a capacity gave by framework that yields an obvious result f or a performer. A performing artist portrays any element that collaborates with the system. The performers are outside the limit of the framework, while the use cases are inside the limit of the framework. On-screen characters are spoken to with stick figures, use cases with ovals, and the limit of the framework with a container encasing the use cases.

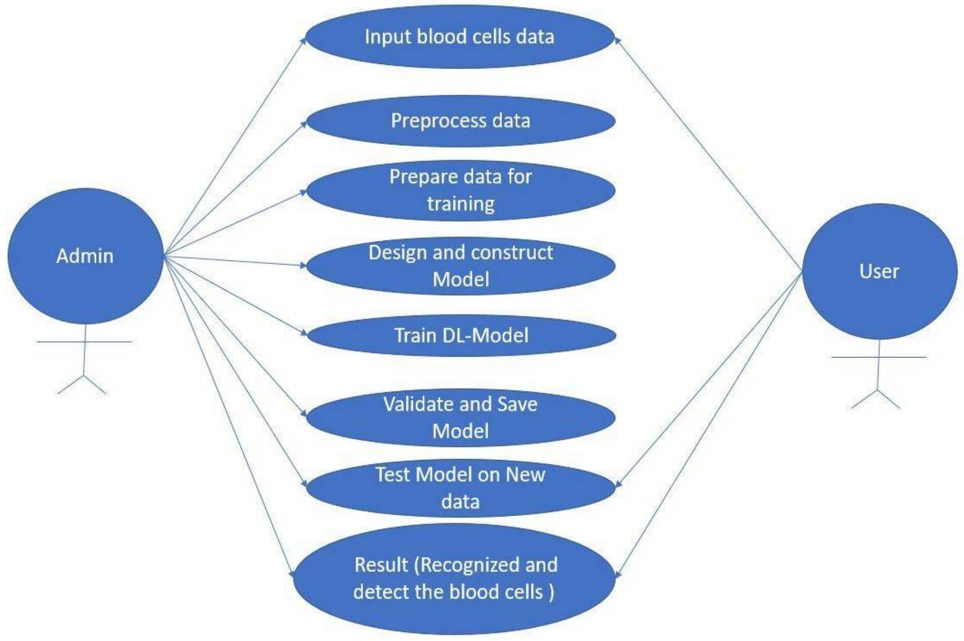


Fig 4.5: Usecase diagram

4.6 FLOW CHART

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

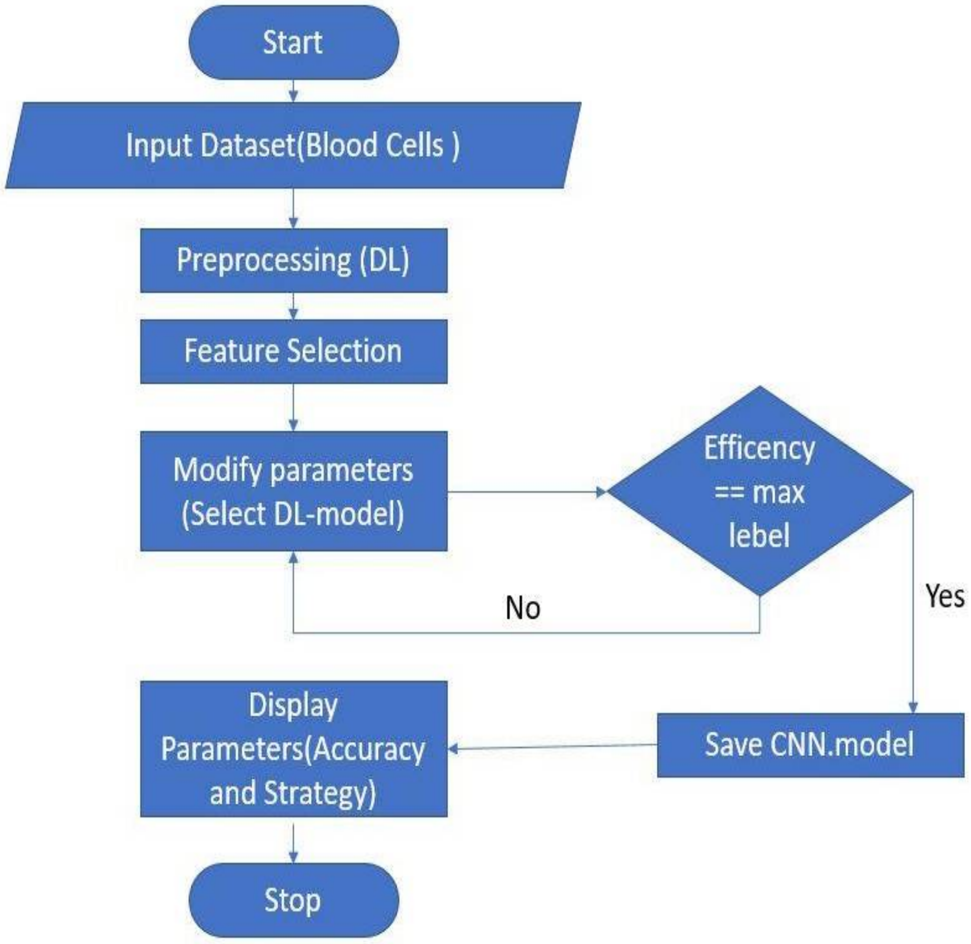
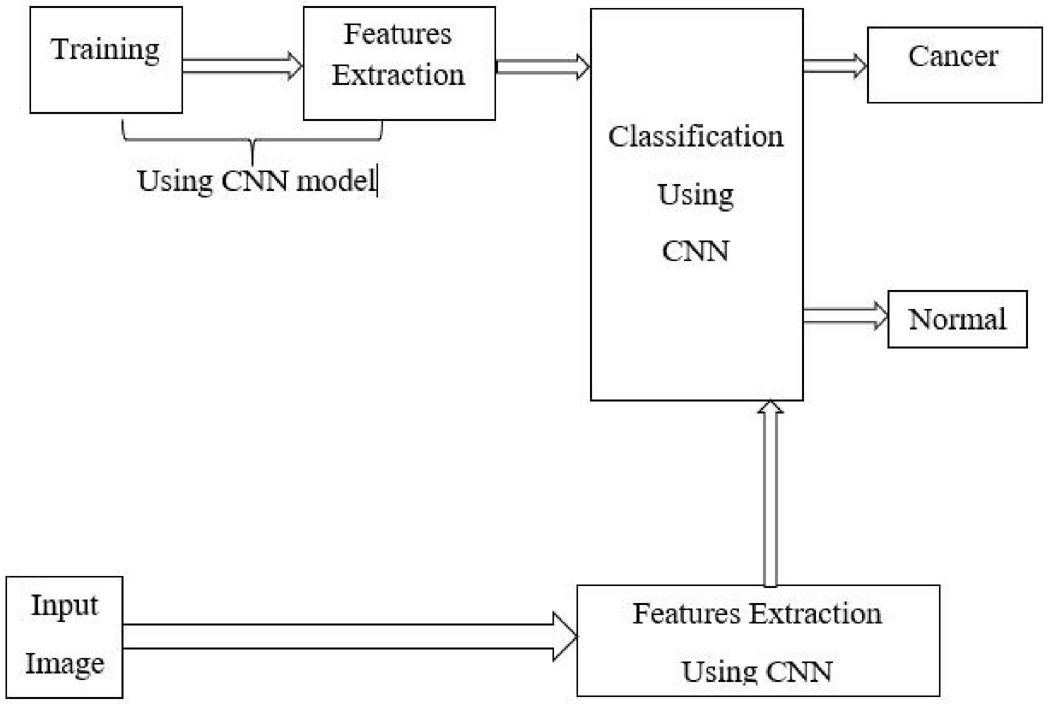

Start
Input Dataset(Blood Cells)
Preprocessing (DL)
Feature Selection
Modify parameters (Select DL-model)
Efficency == max lebel
No
Yes
Save CNN.model
Display Parameters(Accuracy and Strategy)
Stop

Fig 4.6: Flow chart

CHAPTER 5
IMPLEMENTATION

CNN TECHNIQUES

Have proposed the segmentation method using color-based clustering to obtain nucleus region and cytoplasm area from stained blood smear images. SVM classifiers are applied with relevant features and gain satisfactory results.
have proposed an automatic detection of white blood cells (WBCs) from peripheral blood images and classification of five types of WBCs: eosinophil, basophil, neutrophil, monocyte, and lymphocyte. Eosinophil and basophil from other WBCs are first classified by SVM with a granularity feature. Other three types are then recognized using convolutional neural network to extract features, and random forest uses these features to classify those WBCs.



Block Diagram DATASET

Image used in this project were obtained from Kaggle dataset which is a public dataset available online [9]. This dataset was divided into 2 classes. There was total 4961 training images where 2483 images were from healthy patients and 2478 images were from patients affected with blood cancer. We tested the model with total 1240 images 620 from each class. These images had resolution of 320*240.

CNN OVER OTHER ALGORITHMS

There are a lot of algorithms that people used for image classification before CNN became popular. People used to create features from images and then feed those features into some classification algorithm like SVM. Some algorithm also used the pixel level values of images as a feature vector too. To give an example, you could train a SVM with 784 features where each feature is the pixel value for a 28×28 image.

CNNs can be thought of automatic feature extractors from the image. While if we use a algorithm with pixel vector we lose a lot of spatial interaction between pixels, a CNN effectively uses adjacent pixel information to effectively down sample the image first by convolution and then uses a prediction layer at the end.

This concept was first presented by Yann le cun in 1998 for digit classification where he used a single convolution layer. It was later popularized by Alex net in 2012 which used multiple convolution layers to achieve state of the art on image net. Thus, making them an algorithm of choice for image classification challenges henceforth.

WORKING OF CNN

We have implemented CNN for the feature extraction and classification of the blood samples.

A CNN is a multilayered neural network with a special architecture to detect complex features in data. CNNs have been used in image recognition, powering vision in robots, text in images and for self-driving vehicles.

The CNN consist layer of neurons and it is optimized for two-dimensional pattern recognition. CNN has three types of layer namely convolutional layer, pooling layer and fully connected layer. Our network consists of 11 layers excluding the input layer. The input layer takes in a RGB color image where each color channel is processed separately.

The first 6 layers of convolution network are convolution layer. First 2 convolution layer applies 16 of 3*3 filters to an image in the layer. The other two layer applies 32 of 3*3 filters to an image. And the last 2 layers of convolution applies 64 of 3*3 filters to an image. The nonlinear transformation sublayer employs the ReLU activation function. The max pooling sublayer applies a 2*2 filter to the image which results in reducing the image size to its half. At this point, convolution network extracts 64 features, each represented by a 32*32 array for each color channel.

The eighth layer is the flatten layer. The flatten layer transforms a multidimensional array into one-dimensional array by simply concatenating the entries of the multidimensional array together. The output of this flatten layer is a one-dimensional array of size 4800. The ninth layer is the fully connected ANN with the ReLU activation function that maps 4800 input values to the 64 output values. The tenth layer is the dropout layer. 50% of the input values coming to the layer are dropped to zero to reduce the problem of overfitting. The eleventh and the final layer is a fully connected ANN with the sigmoid activation function that maps 64 input values to 2 class labels.

First, we train convolution network using the data in training set to find appropriated filters' weights in the three convolutional sublayers and the weights that yield minimum error in the two fully connected layers. Next, we evaluate convolution network using the data in the validation set to obtain validation error and cross-entropy loss. We repeat the training of convolution network in this same procedure until we complete 10 epochs. Last, we evaluate the performance of convolution network using data in the test set.

CLASSIFICATION

Neural networks are used in the automatic detection of cancer in blood samples. Neural network is chosen as a classification tool due to its well-known technique as a successful classifier for many real applications. The training and validation processes are among the important steps in developing an accurate process model using CNNs. The dataset for training and validation processes consists of two parts; the training features set which are used to train the CNN model; whilst a testing features sets are used to verify the accuracy of the trained using the feed- forward back propagation network. In the training part, connection weights were always updated until they reached the defined iteration Number or suitable error. Neural networks are used in the automatic detection of cancer in blood samples. Neural network is chosen as a classification tool due to its well-known technique as a successful classifier for many real applications. The training and validation processes are among the important steps in developing an accurate process model using CNNs.

5.1 CLASSIFICATION

```
#Blood cell subtype classification
import pandas as pd
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import os
import seaborn as sns
#TRAIN AND TEST DATASET ADDRESS
DATASET="./dataset2-master/images/TRAIN"
TEST_DATASET="./dataset2-master/images/TEST"
```

```
#Categroized images
#4 types of subcells
CATEGORIES=["EOSINOPHIL","LYMPHOCYTE","MONOCYT
E","NEUTROPHIL"]
#reading original image from directory
for category in CATEGORIES:
label=CATEGORIES.index(category)
 path=os.path.join(DATASET,category)
 for img_file in os.listdir(path):
# 1 indicates read image in RGB scale
# 0 indicates read image in grey scale
img=cv.imread(os.path.join(path,img_file),1)
#open cv read image in BGR format
#below we convert it to RGB format
img=cv.cvtColor(img,cv.COLOR_BGR2RGB)
#print(img.shape)
 plt.imshow(img)
 plt.show()
 break
 #plotting single image from each folderIn [ ]:
#reading image from directory
for category in CATEGORIES:
label=CATEGORIES.index(category)
path=os.path.join(DATASET,category)
for img_file in os.listdir(path):
# 1 indicates read image in RGB scale
# 0 indicates read image in grey scale
img=cv.imread(os.path.join(path,img_file),1)
img=cv.cvtColor(img,cv.COLOR_BGR2RGB)
dst = cv.fastNlMeansDenoisingColored(img,None,5,10,7,21)
#image convert to smaller pixels 60*60
#print(img.shape)
plt.figure(figsize=(10,8))
plt.subplot(121)
plt.imshow(dst)
plt.subplot(122)
plt.imshow(img)
plt.show()
break
#plotting single image from each folder
#make train data
train_data=[]
for category in CATEGORIES:
#each cateogry into unique integer
label=CATEGORIES.index(category)
path=os.path.join(DATASET,category)
for img_file in os.listdir(path):
img=cv.imread(os.path.join(path,img_file),1)
img=cv.cvtColor(img,cv.COLOR_BGR2RGB)
#dst = cv.fastNlMeansDenoisingColored(img,None,5,10,7,21)
img=cv.resize(img,(60,60))
train_data.append([img,label])
#make test data
test_data=[]
for category in CATEGORIES:
#each cateogry into unique integer
label=CATEGORIES.index(category)
path=os.path.join(TEST_DATASET,category)
for img_file in os.listdir(path):
img=cv.imread(os.path.join(path,img_file),1)
img=cv.cvtColor(img,cv.COLOR_BGR2RGB)
#dst = cv.fastNlMeansDenoisingColored(img,None,5,10,7,21)
img=cv.resize(img,(60,60))
test_data.append([img,label])
In [ ]:
#print total data in train and test
print(len(train_data))
print(len(test_data))
#shuffle the dataset fo good result
import random
random.shuffle(train_data)
random.shuffle(test_data)
In [ ]:
```

```
#check the data
for lbl in train_data[:10]:
print(lbl[1])
In [ ]:
#lets seprate the feature and target variable
train_X=[]
train_y=[]
for features,label in train_data:
train_X.append(features)
train_y.append(label)
len(train_X),len(train_y)
In [ ]:
#lets seprate the feature and target variable
test_X=[]
test_y=[]
for features,label in test_data:
test_X.append(features)
test_y.append(label)
len(test_X),len(test_y)
In [ ]:
#convert image array to numpy array
#-1 means same size
# 40*40 means height and width
# 3 for R+G+B
train_X=np.array(train_X).reshape(-1,60,60,3)
train_X=train_X/255.0
train_X.shape
#we divide the np array by 255 to close all values to 0
In [ ]:
#convert image array to numpy array
#-1 means same size
# 40*40 means height and width
# 3 for R+G+B
```

```
test_X=np.array(test_X).reshape(-1,60,60,3)
test_X=test_X/255.0
test_X.shape
#we divide the np array by 255 to close all values to 0
In [ ]:
#count labels
sns.countplot(train_y,palette='Set3')
#we can see each categroy has equal data
In [ ]:
#convert label into the one hot encode
from keras.utils import to_categorical
#train y
one_hot_train=to_categorical(train_y)
one_hot_train
In [ ]:
#test_y
one_hot_test=to_categorical(test_y)
one_hot_test
In [ ]:
#build the models
#import Keras libraries
In [ ]:
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,Dense,Flatten,MaxPooling2D,Dropout,Activation
In [ ]:
model=Sequential()
model.add(Conv2D(32,(3,3),activation='relu',input_shape=(60,60,3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.20))
model.add(Conv2D(64,(3,3,activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.20))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.40))
 model.add(Flatten())
 model.add(Dense(64,activation='relu'))
model.add(Dense(128,activation='relu'))
model.add(Dense(64,activation='relu'))
 model.add(Dense(4,activation='softmax'))
model.summary()
In [ ]:
#we will choose adam optimizer
#we have 4 categories so loss function is categorical_crossentropy
#metrics accuracy
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
In [ ]:
#lets split the 20% train dataset for validation
hist=model.fit(train_X,one_hot_train,epochs=50,batch_size=128,validation_split=0.2
In [ ]:
#model evaluation
test_loss,test_acc=model.evaluate(test_X,one_hot_test)
test_loss,test_acc
In [ ]:
#train and validation loss
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train','Validation'],loc='upper left')
plt.show()
In [ ]:
#train and validation accuracy
```

```
plt.plot(hist.history['acc'])
plt.plot(hist.history['val_acc'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train','Validation'],loc='upper left')
plt.show()
In [ ]:
#model prediction

y_pred=model.predict_classes(test_X)
y_pred
In [ ]:
 for i in range(10):
print("Actual=%s, Predicted=%s" % (test_y[i], y_pred[i]))
In [ ]:
#accuracy_score
from sklearn.metrics import accuracy_score,confusion_matrix
accuracy_score(test_y,y_pred)
In [ ]:
sns.heatmap(confusion_matrix(test_y,y_pred))
```

PREDICTION

```
app = dash.Dash()
im1='https://blog.placeit.net/wp-content/uploads/2018/09/Medical-Icons-for-
Healthcare- Logos2.png'
im2='https://www.nicepng.com/png/full/224-2248376_image-download-
healthy-clipart- health-subject-blood-health.png'
im3='https://www.doulos.com/images/logos/DeepLearning.jpg'
im4='https://upload.wikimedia.org/wikipedia/en/c/cd/Anaconda_Logo.png'
im5='https://seeklogo.com/images/S/spyder-logo-68D7CF8B2C-
seeklogo.com.png' im6='https://microbiologyinfo.com/wp-
content/uploads/2017/04/White-Blood-Cells- Leukocytes.jpg'
im7='https://upload.wikimedia.org/wikipedia/en/c/cd/Anaconda_Logo.png'
im8='https://upload.wikimedia.org/wikipedia/commons/thumb/3/3c/Flask_logo.s
vg/1200 px-Flask_logo.svg.png'

app.layout = html.Div([
html.H1(children=' BlOOD CELLS RECOGNITION(CLASSIFICATION)
USING DEEP
LEARNING ', style={'marginBottom': '12px'}),
   html.Div(children     =     [html.Img(src=im1,style={'width':     '20%'
        'marginLeft':'auto','marginRight': 'auto'}),
html.Img(src=im2,style={'width': '20%', 'marginLeft': 'auto', 'marginRight':
'auto'}),
             html.Img(src=im3,style={'width': '20%', 'marginLeft': 'auto',
             'marginRight':
'auto'})]),
html.H2(" Please Select Blood Cell Image file to Test ",style={'marginBottom':
'4px'}),html.Button(id='submit-button', n_clicks=0, children='Upload',
style={'marginTop': 15, 'marginBottom': 25}),html.Div(id='my-div'),
    html.Div(children=   [html.Img(src=im7,style={'width':
    '25%', 'marginLeft':   'auto','marginRight': 'auto'}),
html.Img(src=im6,style={'width': '25%', 'marginLeft': 'auto', 'marginRight':
  'aut    html.Img(src=im8,style={'width': '25%', 'marginLeft': 'auto',
  o'}),   'marginRight':

  'aut
  o'})
  ])
  ], style={'textAlign':
  'center',"backgroundColor":"#ff7700"})
  #"#FFE4FE"
  #"#ff7700"


  app.css.append_css({"external_url":
  "https://codepen.io/chriddyp/pen/bWLwgP.css"})

  @app.callback(
    Output(component_id='my-div',
    component_property='children'),
    [Input(component_id='submit-button',
    component_property='n_clicks')]

  def
      update_outpu
      t_div(n_click
      s):if n_clicks
      > 0:
          root = Tk()
          path =
          'C:/Users/Lenovo/Desktop/B_Classification/TEST_SIMPE

          root.filename  =  filedialog.askopenfilename(initialdir =
          path,title = "Select
  file",filetypes = (("Blood cell
          Image","*.jpg"),("all files","*.*")))
          print(root.filename)
          out =
          Predict_Classification.Blood_Cells_Classification(root.filen
          ame)root.destroy()
          return    html.H1(children='Selected Blood    Cell    is
                    '                                  +          str(out),
  style={'marginBottom': '14px'})

  f __name
     ________=
    = '
    ________
    main____':
    app.run_ser
    ver()
```

CHAPTER 6

TESTING

Testing is an important phase in the development life cycle of the product. This is the phase, where the remaining errors, if any, from all the phases are detected. Hence testing performs a very critical role for quality assurance and ensuring the reliability of the software.

During the testing, the program to be tested was executed with a set of test cases and the output of the program for the test cases was evaluated to determine whether the program was performing as expected. Errors were found and corrected by using the below stated testing steps and correction was recorded for future references. Thus, a series of testing was performed on the system, before it was ready for implementation.

It is the process used to help identify the correctness, completeness, security, and quality of developed computer software. Testing is a process of technical investigation, performed on behalf of stake holders, i.e. intended to reveal the quality-related information about the product with respect to context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding errors.

The quality is not an absolute; it is value to some person. With that in mind, testing can never completely establish the correctness of arbitrary computer software; Testing furnishes a 'criticism' or comparison that compares the state and behavior of the product against specification. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing.

There are many approaches to software testing, but effective testing of complex products is essentially a process of investigation not merely a matter of creating and following routine procedure.

Although most of the intellectual processes of testing are nearly identical to that of review or inspection, the word testing is connoted to mean the dynamic analysis of the product- putting the product through its paces. Some of the common quality attributes include capability, reliability, efficiency, portability, maintainability, compatibility and usability. A good test is sometimes described as one, which reveals an error; however, more recentthinking suggest that a good test is one which reveals information of interest to someone who matters within the project community.

TYPES OF TESTS

6.1 UNIT TESTING

Individual component are tested to ensure that they operate correctly. Each component is tested independently, without other system component. This system was tested with the set of proper test data for each module and the results were checked with the expected output. Unit testing focuses on verification effort on the smallest unit of the software design module. This is also known as MODULE TESTING. This testing is carried out during phases, each module is found to be working satisfactory as regards to the expected output from the module.

6.2 INTEGRATION TESTING

Integration testing is another aspect of testing that is generally done in order to uncover errors associated with flow of data across interfaces. The unit-tested modules aregrouped together and tested in small segment, which make it easier to isolate and correct errors. This approach is continued unit I have integrated all modules to form the systemas a whole.

6.3 VALIDATION TESTING

The validation testing can be defined in many ways, but a simple definition is that. Validation succeeds when the software functions in a manner that can be reasonably expected by the enduser.

Black Box testing

Black box testing is done to find the following

- ✓ Incorrect or missing functions
- ✓ Interface errors
- ✓ Errors on external database access
- ✓ Performance error
- ✓ Initialization and termination error

White Box Testing

- ➢ This allows the tests to
- ➢ Check whether all independent paths within a module have been exercised atleast once
- ➢ Exercise all logical decisions on their false sides
- ➢ Execute all loops and their boundaries and within their boundaries
- ➢ Exercise the internal data structure to ensure their validity
- ➢ Ensure whether all possible validity checks and validity lookups have beenprovided to validate data entry.
- ➢ The testing process overview is as follows:

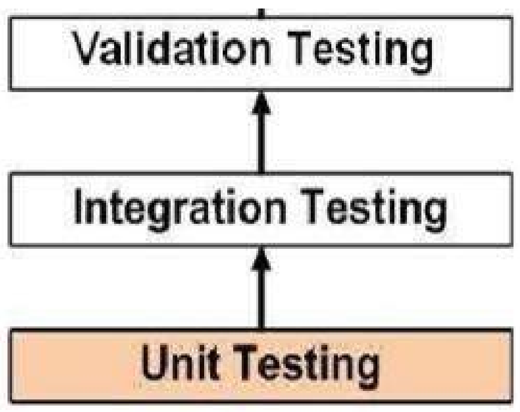


Figure 6.1: The testing process

6.4 SYSTEM TESTING

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration testing. System testing is based on process description and flows, emphasizing pre-driver process and integration points.

6.5 Acceptance Testing

This is the final stage of testing process before the system is accepted for operationaluse.

The system is tested within the data supplied from the system procurer rather thansimulated data

TESTING OF INITIALIZATION AND UI COMPONENTS

Sl # Test Case	UTC- 1
Name of Test	Pre-processing the blood cells images.
Expected Result	Pre-processing is done by image to a sequence of RGB color. Each image having the same dimensions. Blood cells color segmentation used to extract blood cells region, with the help of GRAY.
Actual output	Same as expected.
Remarks	Successful

Table 6.1 Unit Test Case 1

Sl # Test Case	UTC- 2
Name of Test	Segmentation.
Expected Result	Filtered image as input. Grey to binary image. Image morphological operation.
Actual output	Same as expected.
Remarks	Successful

Table 6.2 Unit Test Case 2

Sl # Test Case	UTC- 3
Name of Test	Feature extraction.
Expected Result	We proposed the CNN model extracted temporal features from the frames which was used further to predict blood cells based on sequence of images.
Actual output	Same as expected.
Remarks	Successful

Table 6.3 Unit Test Case 3

Sl # Test Case	UTC- 4
Name of Test	Training.
Expected Result	Train deep learning model with feature matrix. Saved trained model with **‘CNN_train.hdf5’**.
Actual output	Same as expected.
Remarks	Successful

Table 6.4 Unit Test Case 4

Sl # Test Case	UTC- 5
Name of Test	Classification using CNN model
Expected Result	Input RGB images Pass features to trained deep learning model
Actual output	Same as expected.
Remarks	Successful

Table 6.5 Unit Test Case 5

Sl # Test Case	UTC- 5
Name of Test	Blood Cells Classification and detection.
Expected Result	Our process was also capable of real-time prediction using image from a web application and the predictions from the Softmax layer. In **displaying result** are real time detect the blood cells and classification, when web application is start then select the blood cells after the process in display the result.
Actual output	Same as expected.
Remarks	Successful

Table 6.6 Unit Test Case 6

CHAPTER 7

F

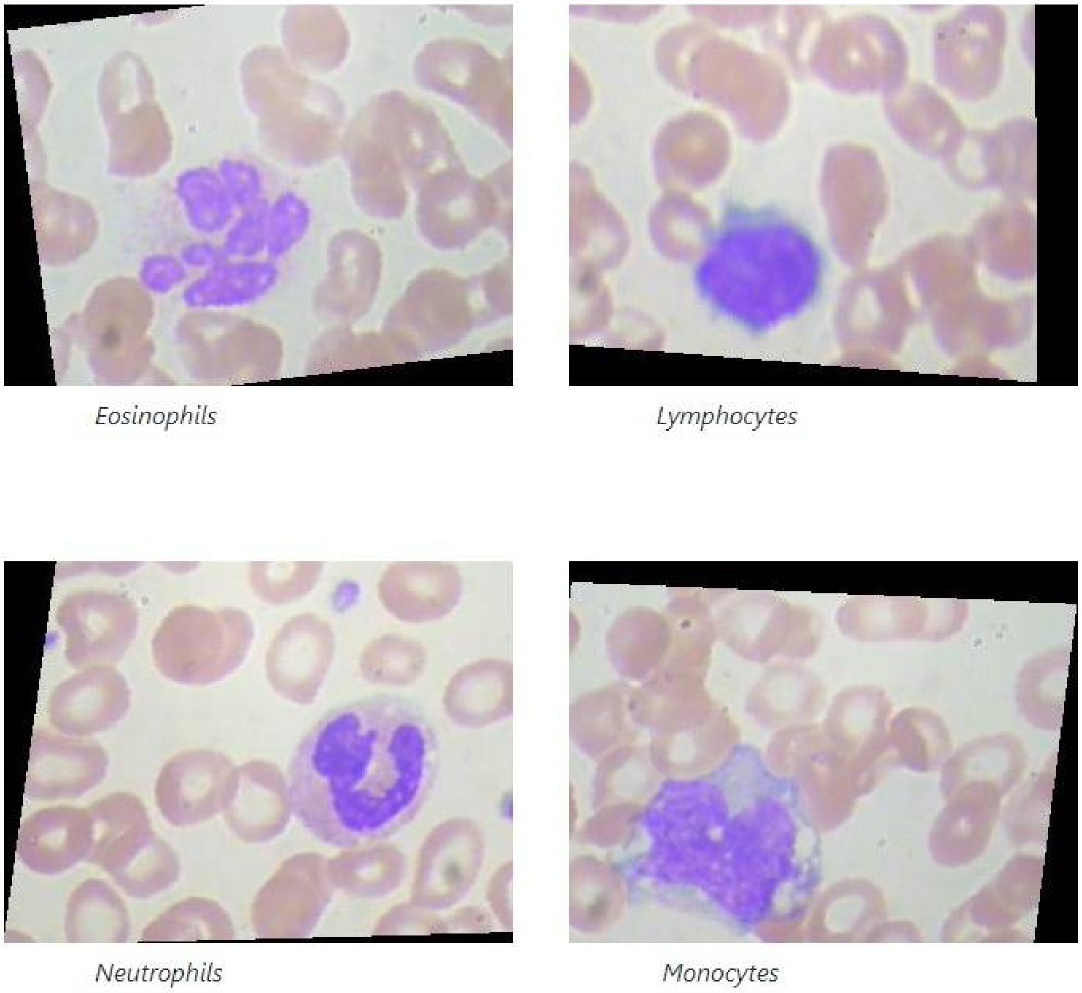


FIG :- TYPES OF BLOOD CELL TYPES

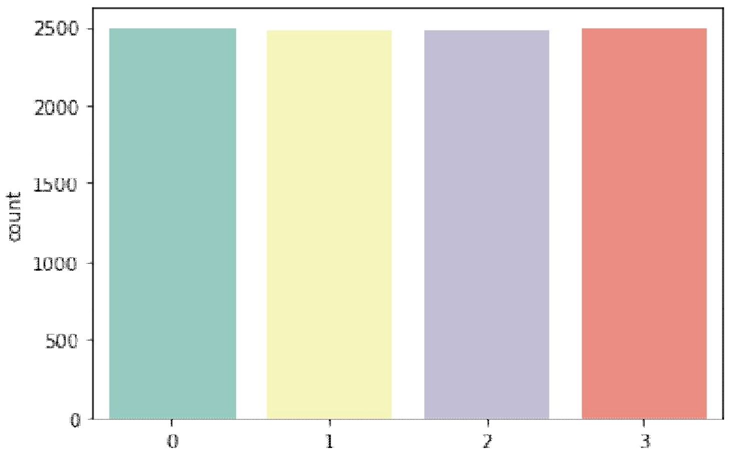


FIG :- COUNT OD TYPE OF BLOOD CELLS

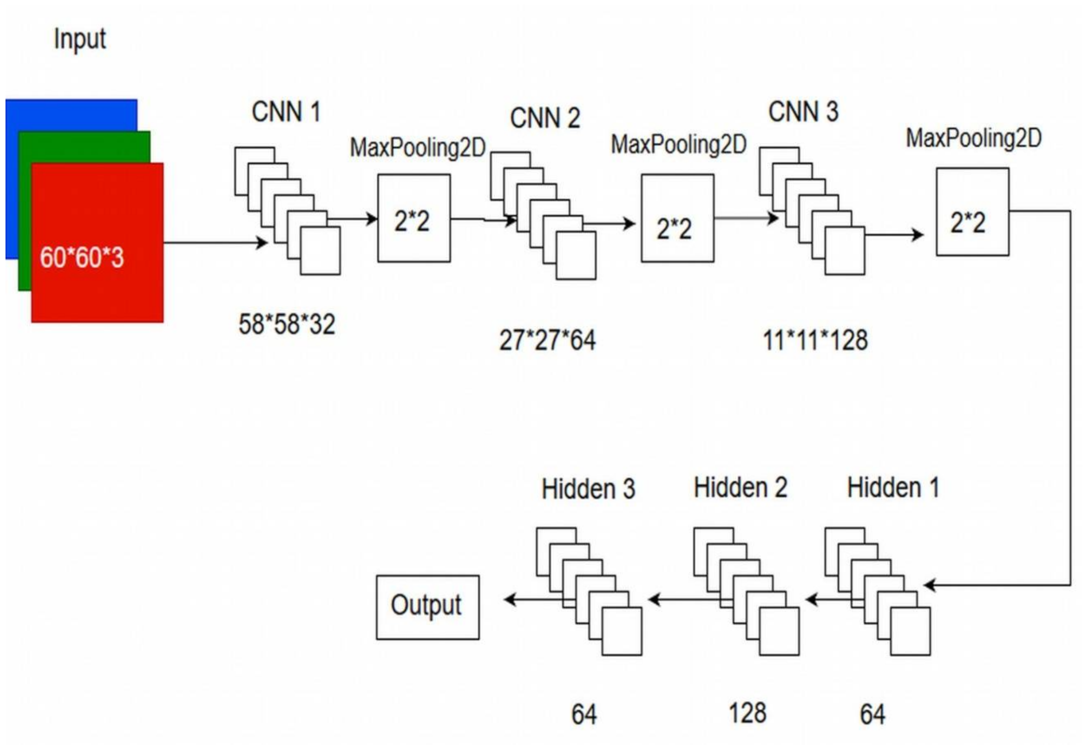
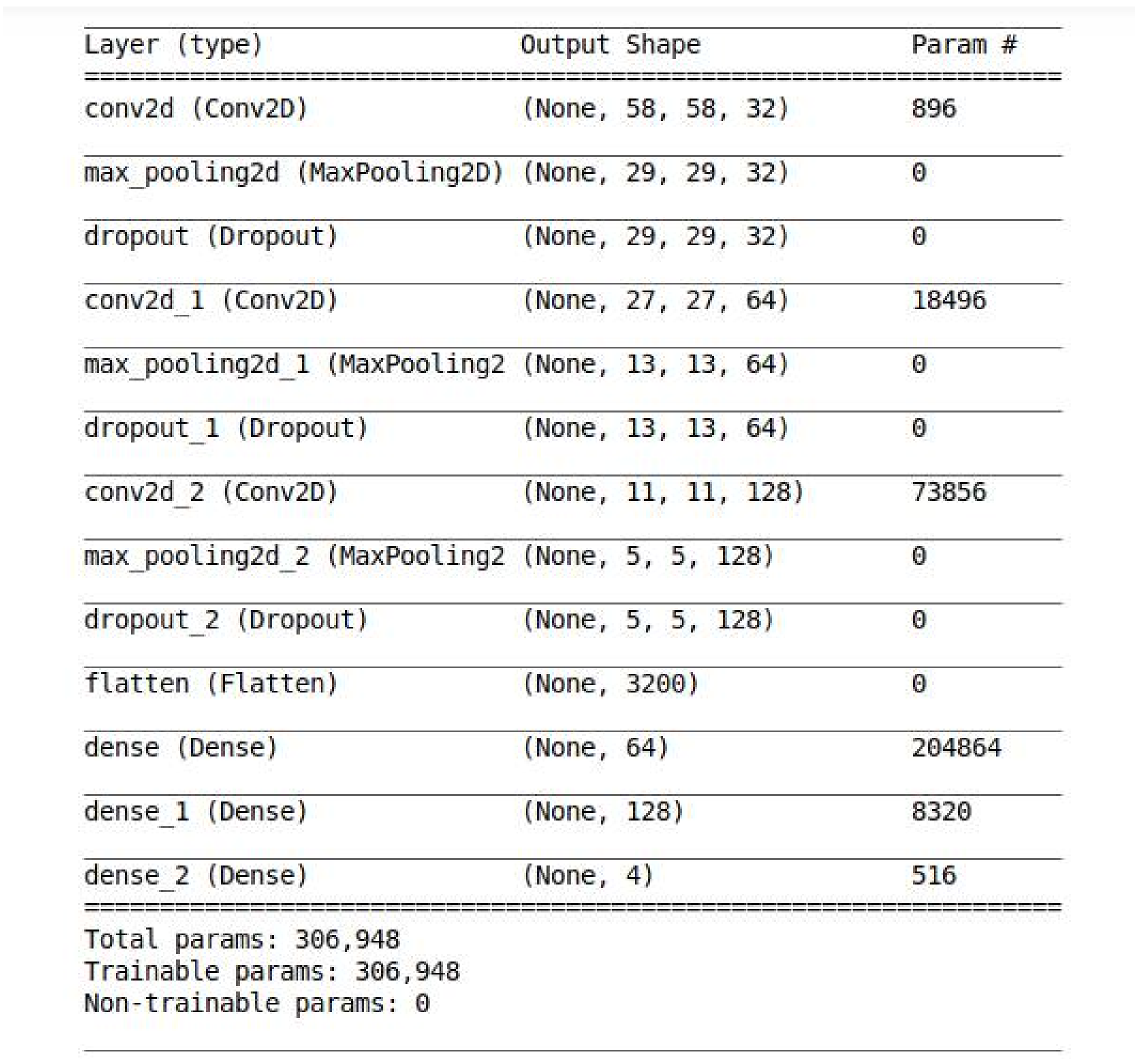


FIG : CNN MODEL USED IN PROPOSED SYSTEM



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 58, 58, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 29, 29, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 29, 29, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 27, 27, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 13, 13, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 5, 5, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 3200)              0
_________________________________________________________________
dense (Dense)                (None, 64)                204864
_________________________________________________________________
dense_1 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 516
=================================================================
Total params: 306,948
Trainable params: 306,948
Non-trainable params: 0
_________________________________________________________________
```

FIG MODEL SUMMARY

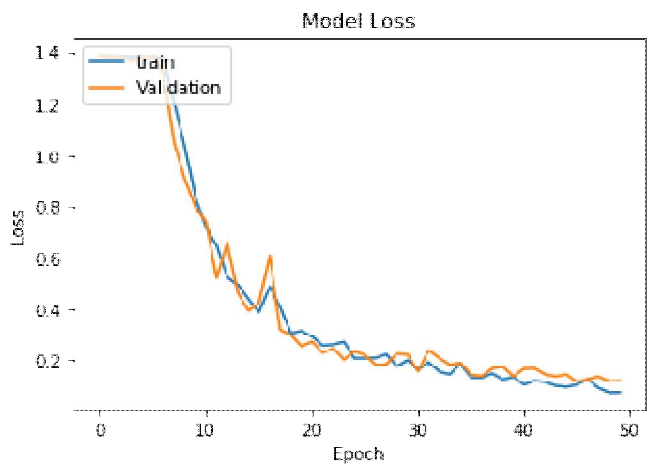


FIG :- MODEL LOSS GRAPH

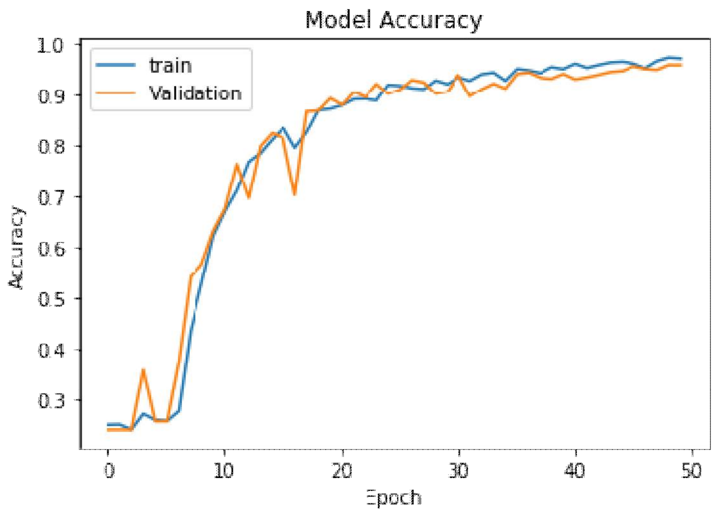


FIG :- MODEL ACCURACY GRAPH

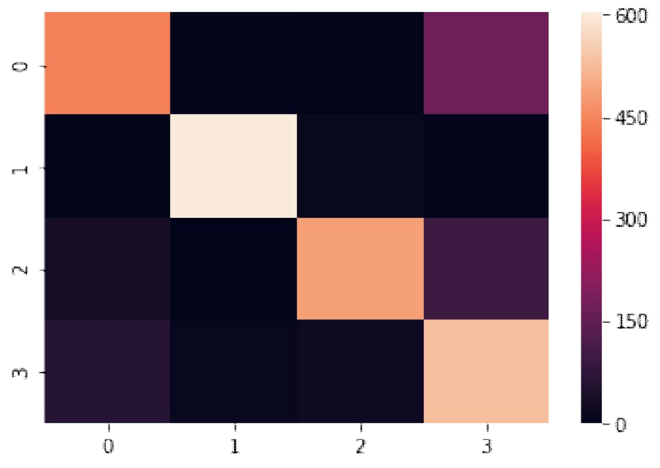


FIG :- CONFUSION MATRIX

SNAPSHOT

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

We presented an approach to real-time blood cells image classification using deep learning. Our proposed deep learning model based on CNN was able to perform with 90
% accuracy on our test dataset. We are developing a system that classifies the blood cells. We have achieved main advantages we propose a depth neural network architecture that combines the features of convolutional neural networks (Xception). We then implement the CNN framework for blood cell image classification. Our model preserves the temporal and spatial information of image features and can learn structured information of imagefeatures. Unlike previous manual feature extraction methods, which rely on cytoplasmic/nuclear segmentation, our method can automatically extract and classify the deep features embedded in cell image patches. Compared with the previous existing methods, our proposed technique achieved the highest performance in terms of classification based on the blood cell dataset.

8.2 FUTURE ENHANCEMENT

Future study may extend our work to accept training using deep learning with Tensorflow GPU (Graphics: Nvidia-Geforce-GTX) is high configuration to well prediction in classify the blood cell. We hope that this segmentation-free, highly accurate blood cell classificationmethod can be used to develop medical-aided diagnostic systems for blood-related diseases in the future.

REFERENCES

[1] N. Sinha and A. G. Ramakrishnan, "Automation of differential blood count," Conference on Convergent Technologies for Asia-Pacific Region (TENCON), pp. 547-551, 2003.

[2] M. Ghosh et al., "Statistical pattern analysis of white blood cell nuclei morphometry", IEEE Students Technology Symposium (TechSym), pp. 59-66, 2010.

[3] S. H. Rezatofighi and H. Soltanian-Zadeh, "Automatic recognition of five types of white blood cells in peripheral blood", in Comput. Med. Imag. Graph., vol. 35, no. 4, pp. 333-343,2011.

[4] S. F. Bikhet, A. M. Darwish, H. A. Tolba, and S. I. Shaheen, "Segmentation and classification of white blood cells," IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 2259-2261, June 2000.

[5] P. Szolovits, Artificial Intelligence In Medicine. Taylor & Francis, 2019.

[6] A. P. Sahastrabuddhe, "Counting of RBC and WBC using image processing: A review," International Journal of Research in Engineering and Technology, vol. 05, no. 05, pp. 356– 360, 2016.

[7] D. Cruz, C. Jennifer, Valiente, L. C. Castor, C. M. T. Mendoza, B. A. Jay, L. S. C. Jane, and
P. T. B. Brian, "Determination of blood components (WBCs, RBCs, and Platelets) count in microscopic images using image processing and analysis," in IEEEInternational Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM'17), Manila, Philippines, December 2017.

[8] V. D. Dvanesh, P. S. Lakshmi, K. Reddy, and A. S. Vasavi, “Blood cell count using digital image processing,” in International Conference on Current Trends towards Converging Technologies (ICCTCT’18), Coimbatore, India, March 2018.

[9] V. Aparna, T. V. Sarath, and K. I. Ramachandran, “Simulation model for anemia detection using RBC counting algorithms and watershed transform,” in International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT’17), Kannur, India, July 2017.

S. S. Savkare, A. S. Narote, and S. P. Narote, “Automatic blood cell segmentation using K-mean clustering from microscopic thin blood images,” in ACM Proceedings of the Third International Symposium on Computer Vision and the Internet (VisionNet’16), Jaipur, India, July