DETECTION OF LEUKEMIA AND MULTI MYELOMA USING DENSE CONVOLUTIONAL NEURAL NETWORK

A project report submitted to the

BHARATHIDASAN UNIVERSITY,

TAMILNADU

in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

Submitted by

GOKUL A

(Register Number 19MTCS07)

Under the Supervision and Guidance of

Dr. P. SUMATHY,
Assistant Professor

SCHOOL OF COMPUTER SCIENCE, ENGINEERING & APPLICATIONS
BHARATHIDASAN UNIVERSITY
TAMILNADU
MAY-2023

SCHOOL OF COMPUTER SCIENCE, ENGINEERING & APPLICATIONS
BHARATHIDASAN UNIVERSITY
TAMILNADU

CERTIFICATE

This is to certify that the major project work entitled **"DETECTION OF LEUKEMIA AND MULTI MYELOMA USING DENSE CONVOLUTIONAL NEURAL NETWORK"** is a bonafide work doneby **Mr.GOKUL A** , Reg. No**19MTCS07** in partial fulfilment of the requirement for the award of the degree of **"BACHELOR OF TECHNOLOGY"** under the guidance of **Dr. P. SUMATHY**, Assistant Professor, School of Computer Science, Engineering & Applications, Bharathidasan University, Tamil Nadu during the academic year 2019 – 2023.

Signature of the Guide **Signature of the HOD**

The Project evaluation and viva-voice examination held on

at the School of Computer Science, Engineering and Applications, Bharathidasan University, Tiruchirappalli – 23.

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

First and foremost, I would like to thank **God Almighty** for the abundant shower of grace blessings on me throughout my entire project.

I wish to express my deep sense of gratitude to **Dr. G. Gopinath, MCA., Ph.D., Professor and Head, School of Computer Science, Engineering and Applications, Bharathidasan University, Tiruchirappalli-23** for giving me this opportunity to do this project.

I would like to express my genuine regard and respect to my project guide **Dr. P. Sumathy, Assistant Professor, School of Computer Science, Engineering and Applications, Bharathidasan University, TamilNadu** for the encouragement, guidance, inspiration, and constructive suggestions that help me in carrying out this project successfully.

It is my duty to express my thanks to all **Teaching** and **Non-Teaching Staff members** of Department of Computer Science & Applications, Bharathidasan University, Tiruchirappalli - 23 those who offered me help directly and indirectly by their suggestions.

Finally, I would like to thank my beloved family members, friends those who extended support towards the successful completion of my project.

(GOKUL A)

19MTCS07

ABSTRACT

Microscopic image analysis plays a significant role in initial leukemia screening and its efficient diagnostics. Since the present conventional methodologies partly rely on manual examination, which is time consuming and depends greatly on the experience of domain experts, automated leukemia detection opens up new possibilities to minimize human intervention and provide more accurate clinical information. This project proposes a novel approach based on conventional digital image processing techniques and machine learning algorithms and deep learning algorithms to automatically identify acute lymphoblastic leukemia from peripheral blood smear images. The proposed model eradicates the probability of errors in the manual process by employing deep learning techniques, namely convolutional neural networks. The model, trained on cells' images, first pre-processes the images and extracts the best features. This is followed by training the model with the optimized Dense Convolutional neural network framework (termed DCNN here) and finally predicting the type of cancer present in the cells. The model was able to reproduce all the measurements correctly while it recollected the samples exactly 94 times out of 100. The overall accuracy was recorded to be 97.2%, which is better than the conventional machine learning methods like Support Vector Machine (SVMs), Decision Trees, Random Forests, Naive Bayes, etc. This study indicates that the DCNN model's performance is close to that of the established CNN architectures with far fewer parameters and computation time tested on the retrieved dataset. Thus, the model can be used effectively as a tool for determining the type of cancer in the bone marrow. To overcome the greatest challenges in the segmentation phase, This work implemented extensive pre-processing and applied a three phase filtration algorithm to achieve the best segmentation results.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Leukemia is a hematological disorder and type of cancer that weakens the human immune system by generating malignant White Blood Cells (WBC). Leukemia is considered as one of the fatal cancers with a high death rate. Leukemia is usually classified based on myelogenous or lymphoblastic disorders of the WBCs. If the affected cells are lymphoblastic, then the leukemia is called Acute Lymphoblastic Leukemia (ALL). If the affected WBCs are monocytes and granulocytes, then the leukemia will be called Acute Myeloid Leukemia (AML).Leukemia is a blood cancer resulting from an abundance of abnormal white blood cells in humans. Usually, a hematology analyzer is used to diagnose leukemia through manual counting. Cell classification usually depends on the morphological characteristics of the cells and requires a skilled medical operator. This procedure can be time-consuming, tedious, and costly.Moreover, the manual analyzer may sometimes lead to the incorrect counting and classification of leukocytes.Undoubtedly, this manual examination mechanism can be replaced by machine-learning-based automated techniques that can save precious time and significantly reduce human effort and error.Deep learning algorithms are powerful and versatile algorithms used efficiently in significant research areas such as medical image processing, supercomputing, investment modeling, and fraud detections . Convolutional Neural Network (CNN) is a popular subcategory of deep learning algorithms, specially designed for visual pattern recognition Among all types of blood cancers, leukemia is the most common form of malignancy in different age groups, especially in children. abnormal phenomenon is caused by excessive proliferation and immature growth of blood kidneys, and then metastasize to important tissues of the body . there are different types of leukemia that haematologists in cell transplant aboratories can differentiate diagnose based on microscopic images. If the slide is correctly stained, some types of leukemia can be more easily identified and distinguished than others, but more equipment is needed to determine underlying leukemia. shows the stained slides of the most common different types of leukemia. An early diagnosis of leukemia has always been a challenge to researchers, doctors, and hematologists. Enlargement of lymph nodes, pallor, fever, and weight loss are the symptoms of leukemia, but they can also be associated with other diseases. Leukemia diagnosis is difficult in its early stages due to the mild nature of the symptoms. most common leukemia diagnosis method is the microscopic evaluation of PBS, but the golden standard for leukemia diagnosis only involves taking and analyzing bone marrow samples. In the last two decades, various studies have adopted machine learning (ML) and computer-aided diagnostic methods for laboratory image analysis, hoping to overcome the limitations of a late leukemia diagnosis and determine its sub groups. these studies have analyzed blood smears images for diagnosing, differentiating, and counting the cells in various types of leukemia . ML is a well-known branch of artificial intelligence, comprising algorithms and mathematical relations, which was quickly introduced to the domain of clinical research. ML enables computers to be programmed without explicit experience and learns from that experience. the outcome of using these methods in medical data processing has been extraordinary, and they have made remarkable success in disease diagnosis. Research indicates that, in medical image processing, ML methods greatly aid complex medical decision-making processes by extracting and then analysing the features of these images. As the number of medical diagnosis tools increased and a large volume of high-quality data was produced, there was an urgent need for more advanced data analysis methods. Traditional methods could not analyse such a large volume of data or find data patterns.

OVERVIEW

Leukemia is cancer of the body's blood-forming tissues, including the bone marrow and the lymphatic system. Many types of Leukemia exist. Some forms of leukemia are more common in children. Other forms of leukemia occur mostly in adults.Leukemia usually involves the white blood cells. Your white blood cells are potent infection fighters — they normally grow and divide in an orderly way, as your body needs them. But in people with leukemia, the bone marrow produces an excessive amount of abnormal white blood cells, which don't function properly.Treatment for leukemia can be complex — depending on the type of leukemia and other factors. But there are strategies and resources that can help make your treatment successful.

SYMPTOMS

Leukemia symptoms vary, depending on the type of leukemia. Common leukemia signs and symptoms include

1. Fever or chills
2. Persistent fatigue, weakness
3. Frequent or severe infections
4. Losing weight without trying
5. Swollen lymph nodes, enlarged liver or spleen
6. Easy bleeding or bruising
7. Recurrent nosebleeds
8. Tiny red spots in your skin (petechiae)
9. Excessive sweating, especially at night
10. Bone pain or tenderness

CAUSES

Scientists don't understand the exact causes of leukemia. It seems to develop from a combination of genetic and environmental factors.

HOW LEUKEMIA FORMS?

In general, leukemia is thought to occur when some blood cells acquire changes (mutations) in their genetic material or DNA. A cell's DNA contains the instructions that tell a cell what to do. Normally, the DNA tells the cell to grow at a set rate and to die at a set time. In leukemia, the mutations tell the blood cells to continue growing and dividing. When this happens, blood cell production becomes out of control. Over time, these abnormal cells can crowd out healthy blood cells in the bone marrow, leading to fewer healthy white blood cells, red blood cells and platelets, causing the signs and symptoms of leukemia.

HOW LEUKEMIA IS CLASSIFIED?

Doctors classify leukemia based on its speed of progression and the type of cells involved.

The first type of classification is by how fast the leukemia progresses

- **Acute leukemia.** In acute leukemia, the abnormal blood cells are immature blood cells (blasts). They can't carry out their normal functions, and they multiply rapidly, so the disease worsens quickly. Acute leukemia requires aggressive, timely treatment.
- **Chronic leukemia.** There are many types of chronic leukemias. Some produce too many cells and some cause too few cells to be produced. Chronic leukemia involves more-mature blood cells. These blood cells replicate or accumulate more slowly and can function normally for a period of time.

Some forms of chronic leukemia initially produce no early symptoms and can go unnoticed or undiagnosed for years.

The second type of classification is by type of white blood cell affected

- **Lymphocytic leukemia.** This type of leukemia affects the lymphoid cells (lymphocytes), which form lymphoid or lymphatic tissue. Lymphatic tissue makes up your immune system.
- **Myelogenous (my-uh-LOHJ-uh-nus) leukemia.** This type of leukemia affects the myeloid cells. Myeloid cells give rise to red blood cells, white blood cells and platelet-producing cells.

TYPES OF LEUKEMIA

The major types of leukemia are

- **Acute Lymphocytic Leukemia (ALL).** This is the most common type of leukemia in young children. ALL can also occur in adults.
- **Acute Myelogenous Leukemia (AML).** AML is a common type of leukemia. It occurs in children and adults. AML is the most common type of acute leukemia in adults.
- **Chronic Lymphocytic Leukemia (CLL).** With CLL, the most common chronic adult leukemia, you may feel well for years without needing treatment.
- **Chronic Myelogenous Leukemia (CML).** This type of leukemia mainly affects adults. A person with CML may have few or no symptoms for months or years before entering a phase in which the leukemia cells grow more quickly.

CHAPTER 2
LITREATURE SURVEY

1.Automatic Detection of White Blood Cancer from Bone Marrow Microscopic Images Using Convolutional Neural Networks DEEPIKA KUMARI , NIKITA JAIN1 , AAYUSH KHURANA1 , SWETA MITTAL1 , SURESH CHANDRA SATAPATHY.

2. Leukemia Diagnosis Based on Machine Learning Algorithms. Patil Babaso S , S.K. Mishra , Aparna Junnarkar.

3.Machine Learning based System for Automatic Detection of Leukemia Cancer Cell . Supriya Mandal , Vani Daivajna , Rajagopalan V.

CHAPTER 3
EXISTING SYSTEM

This existing study provides a robust mechanism for the classification of Acute Lymphoblastic Leukemia (ALL) and Multiple Myeloma (MM) using the SN-AM dataset. Acute lymphoblastic leukemia (ALL) is a type of cancer where the bone marrow forms too many lymphocytes. On the other hand, Multiple Myeloma (MM), a different kind of cancer, causes cancer cells to accumulate in the bone marrow rather than releasing them into the bloodstream. Therefore, they crowd out and prevent the production of healthy blood cells. The model eradicates the probability of errors in the manual process by employing machine learning techniques, namely k-beast algorithm. The model, trained on cells' images, first pre-processes the images and extracts the best features. This existing study indicates that the model's performance is close to that of the established K-beast algorithms architectures with far fewer parameters and computation time tested on the retrieved dataset.

PROPOSED SYSTEM

In the proposed methodology, the image of blood smear goes through different stages. First the input image undergo segmentation. Then the image cleaning operation is performed. After that features are extracted from the image. Finally these features are classified by a classifier. The objectives of proposed method are developing automated and accurate method to find whether the blood image is leukaemia affected or not. if leukaemia is affected then it is classified the leukaemia is Acute Lymphoblastic Luekmia (ALL) or Multiple Myloma (MM). It includes identification and classification of leukocytes. Also find the best method which classifies the leucocytes among three classification methods.

The proposed method also aims in analysing performance of Decision Tree and CNN. Convolutional neural network (CNN) is a tpe of artificial neural network that is most commonly used in all application of image processing widely. It is a multilayer neural network, and it is based on supervised learning method. It is a complex feed forward neural network. It is used for image classification and recognition because of its high accuracy and small error rates. The following is the steps involved in the CNN method to produce a classified output. The model, containing three types of layers, namely convolution layer, max pool, and fully connected layer, is trained on the training set, and then it is used for prediction on the testing set.

SOFTWARE DESCRIPTION

- **Python 3.7.3**
- **Tensrflow**
- **Keros**
- **Numpy**
- **Pillow**

PYTHON

- Python is a wonderful and powerful programming language that's easy to use (easy to read **and** write) and with Raspberry Pi lets you connect your project to the real world.
- Python syntax is very clean, with an emphasis on readability and uses standard English keywords. Start by opening IDLE from the desktop.

IDLE

- The easiest introduction to Python is through IDLE, a Python development environment. Open IDLE from the Desktop or applications menu
- IDLE gives you a REPL (Read-Evaluate-Print-Loop) which is a prompt you can enter Python commands in to. As it's a REPL you even get the output of commands printed to the screen without using print.

TENSORFLOW- INTRODUCTION

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. The belove figure 3.1 describe the TensorFlow software It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions. The official website of TensorFlow is mentioned below: https//www.tensorflow.org/

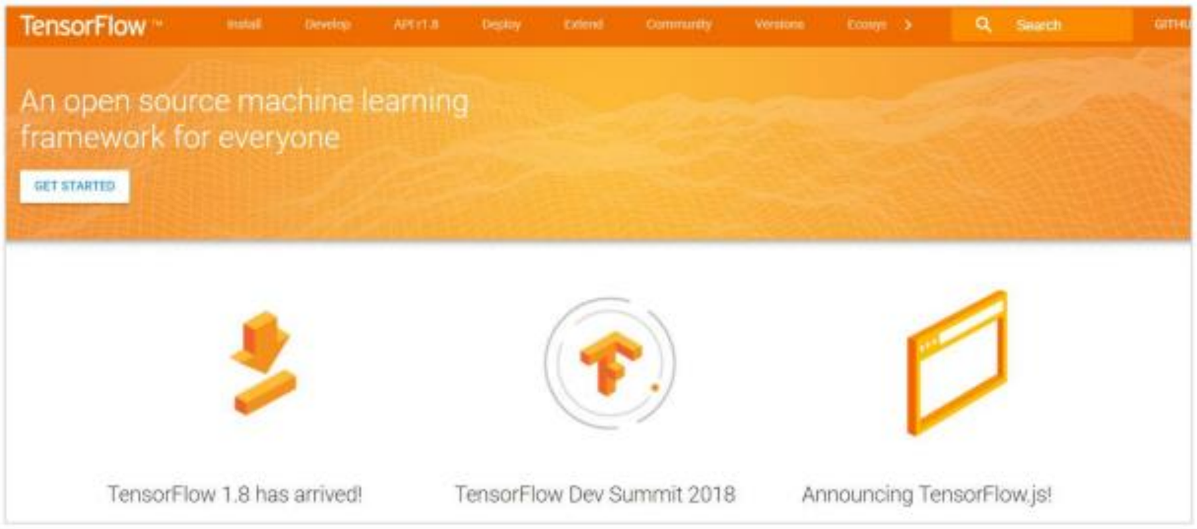

TensorFlow
Install
Develop
API r1.8
Deploy
Extend
Community
Versions
Ecosys
Search
An open source machine learning framework for everyone
GET STARTED
TensorFlow 1.8 has arrived!
TensorFlow Dev Summit 2018
Announcing TensorFlow.js!

Fig3.1 TensorFlow

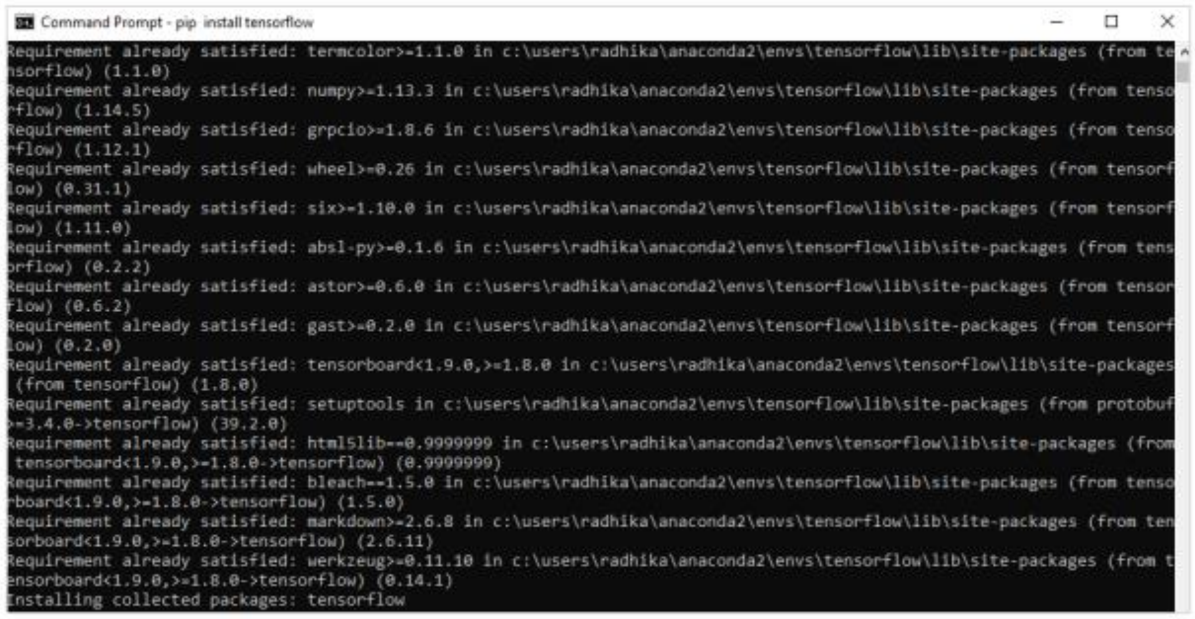
Let us now consider the following important features of TensorFlow

- It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes a programming support of deep neural networks and machine learning techniques.
- It includes a high scalable feature of computation with various data sets.
- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

TensorFlow Installation

To install TensorFlow, it is important to have "Python" installed in your system. Python version 3.4+ is considered the best to start with TensorFlow installation. Consider the following steps to install TensorFlow in Windows operating system.

pip install Tensorflow


Command Prompt - pip install tensorflow

```
Requirement already satisfied: termcolor>=1.1.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from te
nsorflow) (1.1.0)
Requirement already satisfied: numpy>=1.13.3 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tenso
rflow) (1.14.5)
Requirement already satisfied: grpcio>=1.8.6 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tenso
rflow) (1.12.1)
Requirement already satisfied: wheel>=0.26 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorf
low) (0.31.1)
Requirement already satisfied: six>=1.10.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorf
low) (1.11.0)
Requirement already satisfied: absl-py>=0.1.6 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tens
orflow) (0.2.2)
Requirement already satisfied: astor>=0.6.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensor
flow) (0.6.2)
Requirement already satisfied: gast>=0.2.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorf
low) (0.2.0)
Requirement already satisfied: tensorboard<1.9.0,>=1.8.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages
 (from tensorflow) (1.8.0)
Requirement already satisfied: setuptools in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from protobuf
>=3.4.0->tensorflow) (39.2.0)
Requirement already satisfied: html5lib==0.9999999 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from
 tensorboard<1.9.0,>=1.8.0->tensorflow) (0.9999999)
Requirement already satisfied: bleach==1.5.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tenso
rboard<1.9.0,>=1.8.0->tensorflow) (1.5.0)
Requirement already satisfied: markdown>=2.6.8 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from ten
sorboard<1.9.0,>=1.8.0->tensorflow) (2.6.11)
Requirement already satisfied: werkzeug>=0.11.10 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from t
ensorboard<1.9.0,>=1.8.0->tensorflow) (0.14.1)
Installing collected packages: tensorflow
```

Fig 3.2 TensorFlow installation

The above figure shows installing TensorFlow using pip

TensorFlow — Convolutional Neural Networks

After understanding machine-learning concepts, we can now shift our focus to deep learning concepts. Deep learning is a division of machine learning and is considered as a crucial step taken by researchers in recent decades. The examples of deep learning implementation include applications like image recognition and speech recognition.

Following are the two important types of deep neural networks

- Convolutional Neural Networks
- Recurrent Neural Networks In this chapter, we will focus on the CNN, Convolutional Neural Networks

Convolutional Neural Networks

Convolutional Neural networks are designed to process data through multiple layers of arrays. This type of neural networks is used in applications like image recognition or face recognition. The primary difference between CNN and any other ordinary neural network is that CNN takes input as a two-dimensional array and operates directly on the images rather than focusing on feature extraction which other neural networks focus on. The dominant approach of CNN includes solutions for problems of recognition. Top companies like Google and Facebook have invested in research and development towards recognition projects to get activities done with greater speed.

A convolutional neural network uses three basic ideas

- Local respective fields
- Convolution
- Pooling

CNN or convolutional neural networks use pooling layers, which are the layers, positioned immediately after CNN declaration. It takes the input from the user as a feature map that comes out of convolutional networks and prepares a condensed feature map. Pooling layers helps in creating layers with neurons of previous layers.

KERAS

Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Keras is designed to quickly define deep learning models. Keras is an optimal choice for deep learning applications.

Artificial Neural Networks

The most popular and primary approach of deep learning is using "Artificial neural network" (ANN). ANNs are made up of multiple nodes which is similar to neurons. Nodes are tightly interconnected and organized into different hidden layers. The input layer receives the input data and the data goes through one or more hidden layers sequentially and finally the output layer predict something useful about the input data. For example, the input may be an image and the output may be the thing identified in the image, say a "Cat".

Multi-Layer Perceptron

Multi-Layer perceptron is the simplest form of ANN. It consists of a single input layer, one or more hidden layer and finally an output layer. A layer consists of a collection of perceptron. Input layer is basically one or more features of the input data. Every hidden layer consists of one or more neurons and process certain aspect of the feature and send the processed information into the next hidden layer. The output layer process receives the data from last hidden layer and finally output the result.

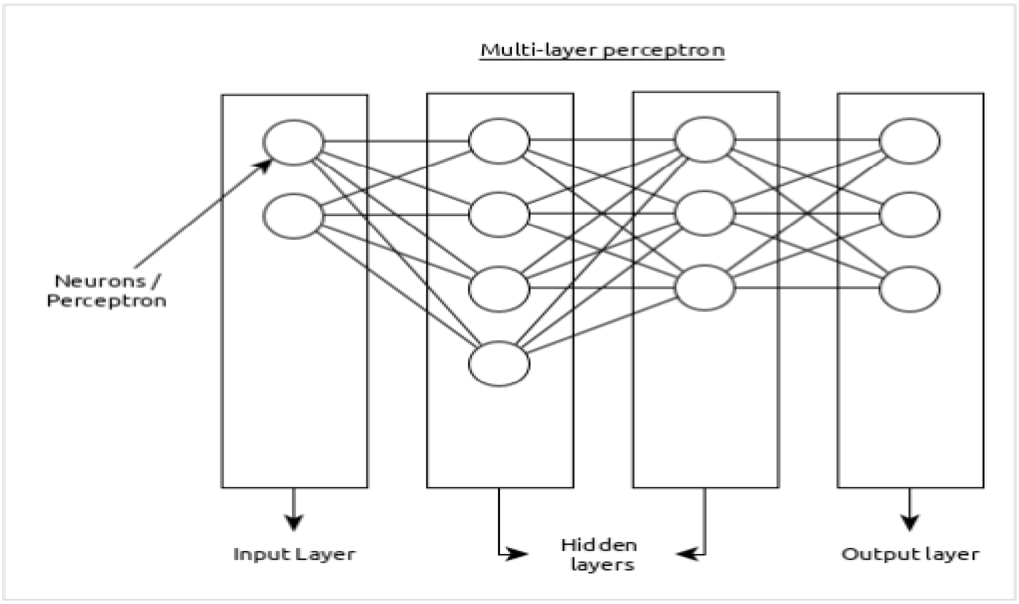


Fig 3.3 Multi-layer perception

Convolutional Neural Network (CNN)

Convolutional neural network is one of the most popular ANN. It is widely used in the fields of image and video recognition. It is based on the concept of convolution, a mathematical concept. It is almost similar to multi-layer perceptron except it contains series of convolution layer and pooling layer before the fully connected hidden neuron layer.

It has three important layers

• Convolution layer It is the primary building block and perform computational tasks based on convolution function.

• Pooling layer It is arranged next to convolution layer and is used to reduce the size of inputs by removing unnecessary information so computation can be performed faster.

• Fully connected layer It is arranged to next to series of convolution and pooling layer and classify input into various categories.

A simple CNN can be represented as below

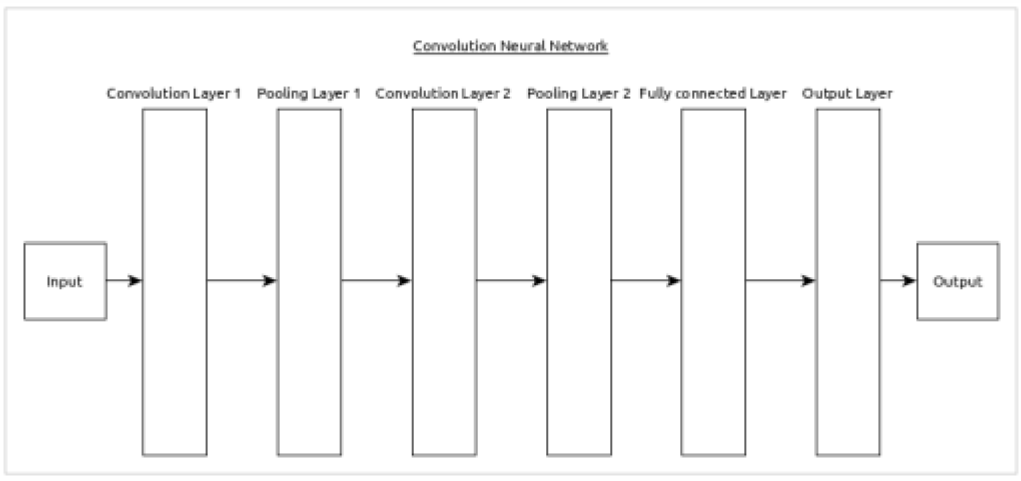


Fig 3.4 Convolutional Neural Network

PYTHON NUMPY

Our Python NumPy Tutorial provides the basic and advanced concepts of the NumPy. Our NumPy tutorial is designed for beginners and professionals.

NumPy stands for numeric python which is a python package for the computation and processing of the multidimensional and single dimensional array elements.

PYTHON PILLOW — OVERVIEW

Some of the most common image processing libraries are OpenCV, Python Imaging Library (PIL), Scikit-image, Pillow. However, in this proposed system used Pillow module and will try to explore various capabilities of this module.

Pillow is built on top of PIL (Python Image Library). PIL is one of the important modules for image processing in Python. It supports wide variety of images such as “jpeg”, “png”, “bmp”, “gif”, “ppm”, “tiff”. Apart from basic image processing functionality, including point operations, filtering images using built-in convolution kernels, and color space conversions.

IMAGE ARCHIVES

The Python Imaging Library is best suited for image archival and batch processing applications. Python pillow package can be used for creating thumbnails, converting from one format to another and print images, etc.

IMAGE DISPLAY

You can display images using Tk PhotoImage, BitmapImage and Windows DIB interface, which can be used with PythonWin and other Windows-based toolkits and many other Graphical User Interface (GUI) toolkits.

For debugging purposes, there is a show () method to save the image to disk which calls the external display utility.

IMAGE PROCESSING

The Pillow library contains all the basic image processing functionality. You can do image resizing, rotation and transformation.

Pillow module allows you to pull some statistics data out of image using histogram method, which later can be used for statistical analysis and automatic contrast enhancement.

PYTHON PILLOW — ENVIRONMENT SETUP

This chapter discusses how to install pillow package in your computer.

Installing pillow package is very easy, especially if you’re installing it using pip.

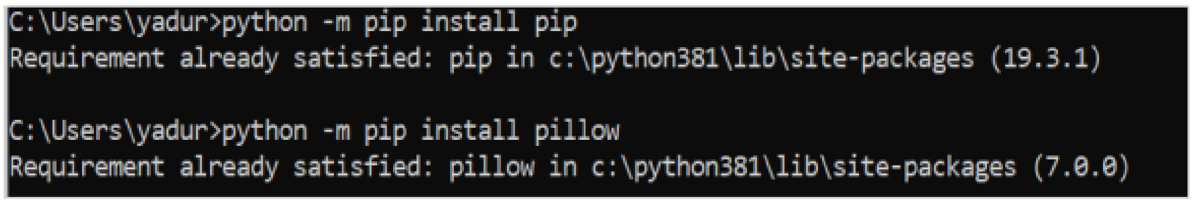
Installing Pillow using pip

To install pillow using pip, just run the below command in your command prompt

python -m pip install pip

python -m pip install pillow

In case, if pip and pillow are already installed in your computer, above commands will simply mention the ‘requirement already satisfied’ as shown below



```
C:\Users\yadur>python -m pip install pip
Requirement already satisfied: pip in c:\python381\lib\site-packages (19.3.1)

C:\Users\yadur>python -m pip install pillow
Requirement already satisfied: pillow in c:\python381\lib\site-packages (7.0.0)
```

Fig 3.5 Installing Pillow using PIP

PYTHON PILLOW — USING IMAGE MODULE

To display the image, pillow library is using an image class within it. The image module inside pillow package contains some important inbuilt functions like, load images or create new images, etc.

Opening, rotating and displaying an image

To load the image, we simply import the image module from the pillow and call the Image.open(), passing the image filename.

Instead of calling the Pillow module, we will call the PIL module as to make it backward compatible with an older module called Python Imaging Library (PIL). That’s why our code starts with “from PIL import Image” instead of “from Pillow import Image”.

Next, we’re going to load the image by calling the Image.open() function, which returns a value of the Image object data type. Any modification we make to the image object can be saved to an image file with the save() method. The image object we received using Image.open(), later can be used to resize, crop, draw or other image manipulation method calls on this Image object.

CHAPTER 4

Project Design

Dataset diagram:

The belove diagram shows the dataset image training on the neural network.

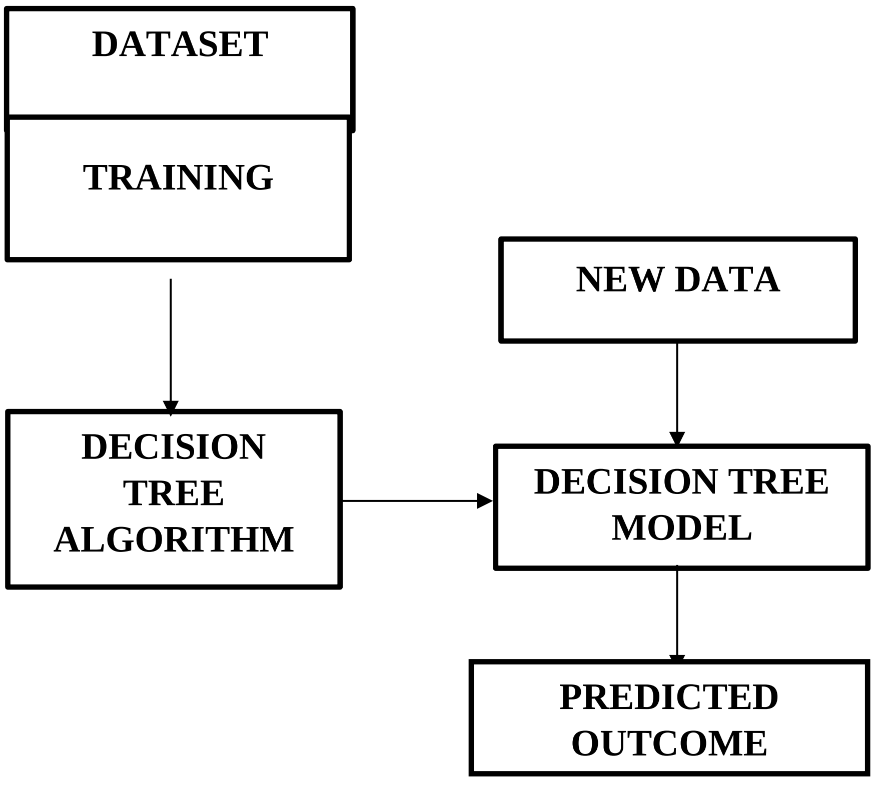


Fig 4.1 dataset training.

BLOCK DIAGRAM

The belove figure 4.2 shows the Block Diagram of the project

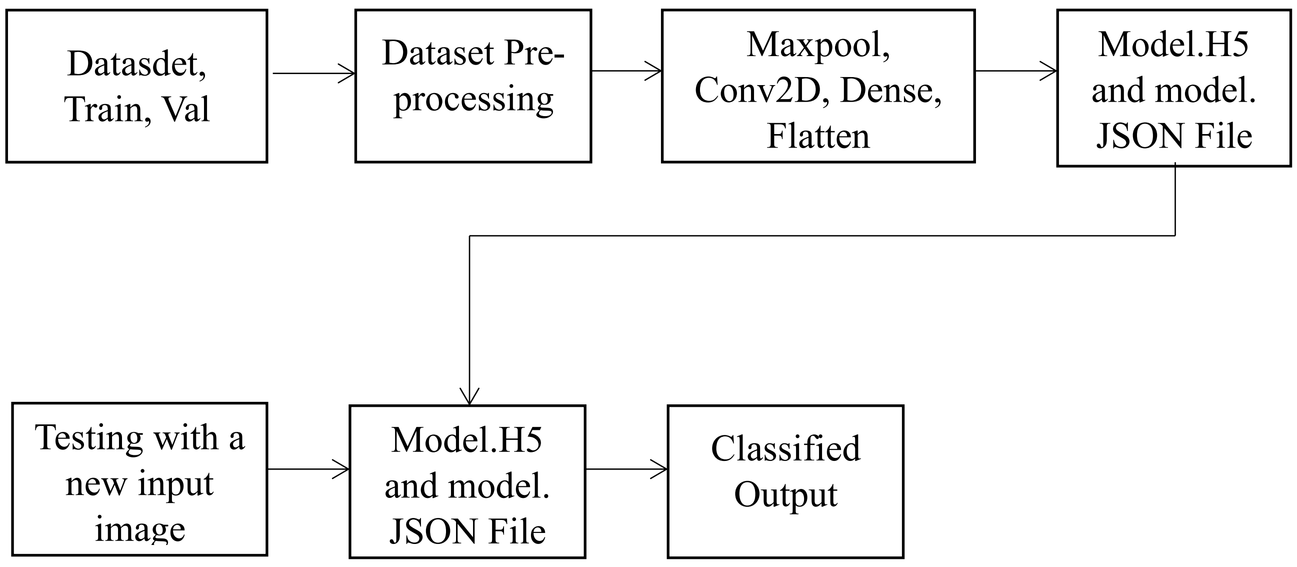
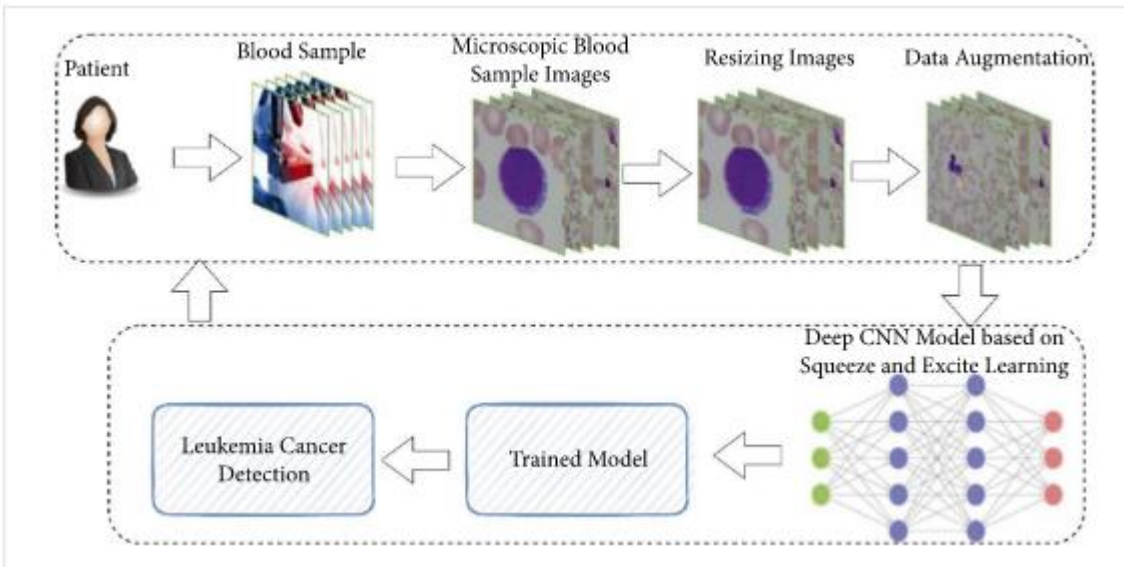

Datasdet, Train, Val
Dataset Pre-processing
Maxpool, Conv2D, Dense, Flatten
Model.H5 and model. JSON File
Testing with a new input image
Model.H5 and model. JSON File
Classified Output

FIG4.2 Block Diagram.

CHAPTER 5

PROGRAM DESIGN

The design overview of the proposed methodology is depicted in below images. The proposed framework begins with the acquisition of microscopic images of blood samples. Later on, the data augmentation techniques are employed to overcome the problem of fewer data since in deep neural networks more data are required for their training and superior performance. Lastly, a deep CNN architecture-based squeeze and excitation learning is proposed to diagnose leukemia from the inputted microscopic images of blood samples. Each step is explained in-depth in the following subsections of methodology.


Patient
Blood Sample
Microscopic Blood Sample Images
Resizing Images
Data Augmentation
Deep CNN Model based on Squeeze and Excite Learning
Trained Model
Leukemia Cancer Detection

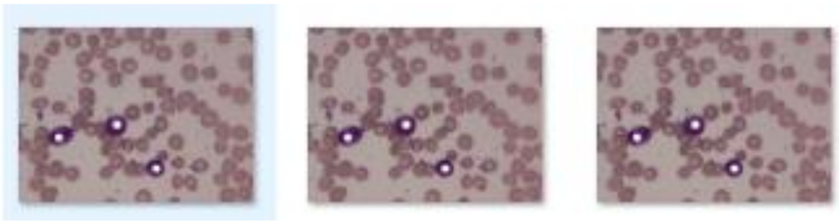


Fig 5.1 Normal Image

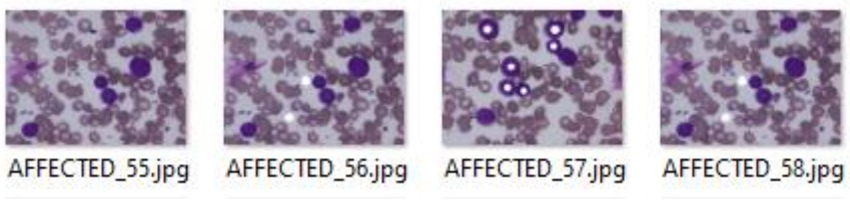


Fig 5.2 Affected image

Proposed CNN Architecture

Convolutional Layers and Max-Pool Layers

Generally, the two fundamental operations of the Convolutional neural networks (CNNs) include the convolution and max-pooling layers mimicking a variety of substantially complex cells in the visual cortex. Besides this, CNNs have a localized perceptive area, hierarchical organization, feature extraction, and classification phase that can automatically learn the appropriate feature and categorization process, and has a significant implication in the domain of computer vision. In the proposed model, microscopic images of blood samples are preprocessed by data augmentation and directly fed into the proposed deep learning model design to craft the localized features. Consider a microscopic blood sample image of a patient of dimension with a kernel size of which is convolved over the image to generate a collection of features maps of dimensions

In equations and , the zero-padding in the direction of both width and height is denoted by and while the value of stride in both vertical and horizontal directions is denoted by and , respectively. The input image of size is provided as an input to the first convolutional layer. Continuing to follow the convolution layers, a pooling layer is also used which contributes to diminishing the computing and spatial necessities of the activation function and makes the proposed model to be translation invariant and functions interdependently on each layer of the input data and spatially downscales it.

DATASET

The dataset used in this study comprises four thousand blood smear acute myeloid samples gathered from two different sources, that is, a reputable tertiary care hospital of Peshawar, Pakistan, and publicly available microscopic peripheral blood images released by Acevedo et al. The dataset consists of 4000 images having 1500 normal monocytes, 1500 abnormal monocytes, and 1000 lymphocytes. The data are preclassified by the experts of the field. The properties of the dataset.

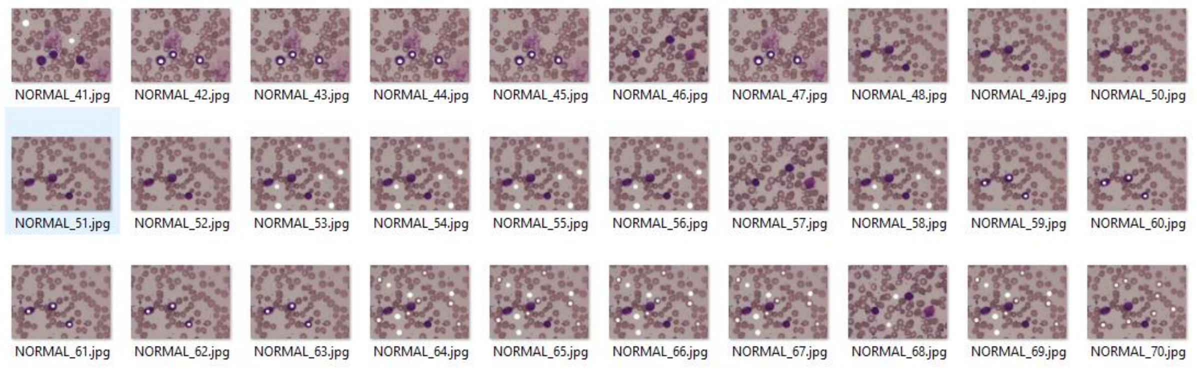


Fig.5.2 Normal images

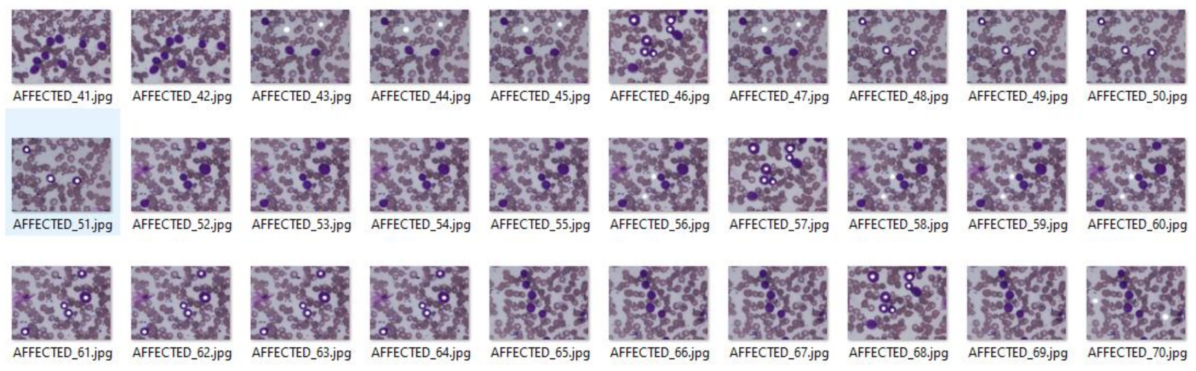


Fig.5.3 Affected images

DATA PREPROCESSING

Data preprocessing is a critical step used to check the data for experiments; before image analysis, the images should be prepared for a good outcome. In our experiments, we resize every image into a 256 × 256 resolution image. Then they are converted to a 227 × 227 pixel image to feed it to the first layer of the CNN model.

DECISION TREE ALGORITHM

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.

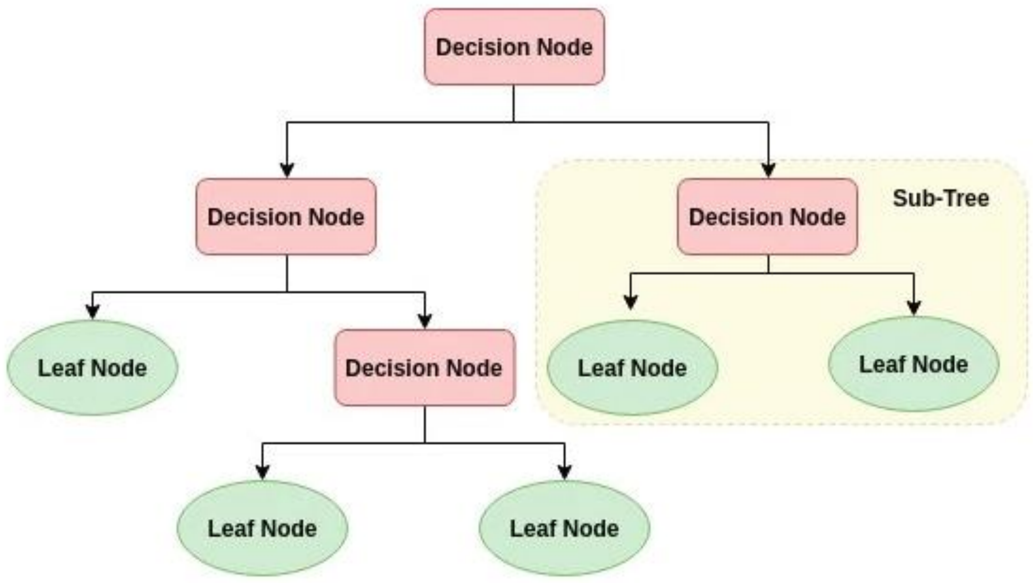


Fig5.4 Decision Node

Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data.

The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy.

How does the Decision Tree algorithm work?

The basic idea behind any decision tree algorithm is as follows

1. Select the best attribute using Attribute Selection Measures(ASM) to split the records.

2. Make that attribute a decision node and breaks the dataset into smaller subsets.

3. Starts tree building by repeating this process recursively for each child until one of the condition will match

   - All the tuples belong to the same attribute value.
   - There are no more remaining attributes.
   - There are no more instances.

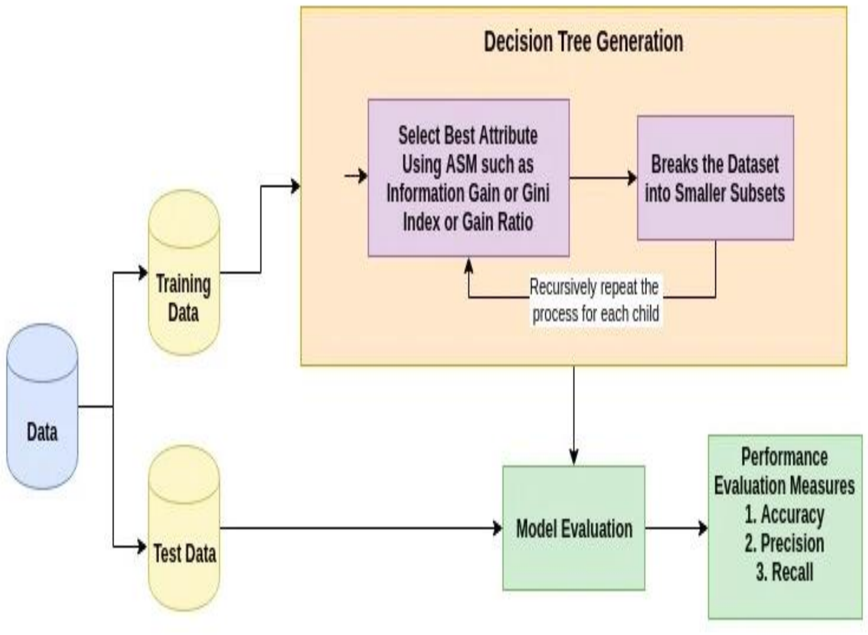

Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

Fig 5.5 Decision Tree Block Diagram

Attribute Selection Measures

Attribute selection measure is a heuristic for selecting the splitting criterion that partition data into the best possible manner. It is also known as splitting rules because it helps us to determine breakpoints for tuples on a given node. ASM provides a rank to each feature(or attribute) by explaining the given dataset. Best score attribute will be selected as a splitting attribute (Source). In the case of a continuous-valued attribute, split points for branches also need to define. Most popular selection measures are Information Gain, Gain Ratio, and Gini Index.

Information Gain

Shannon invented the concept of entropy, which measures the impurity of the input set. In physics and mathematics, entropy referred as the randomness or the impurity in the system. In information theory, it refers to the impurity in a group of examples. Information gain is the decrease in entropy.

Information gain computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values. ID3 (Iterative Dichotomiser) decision tree algorithm uses information gain.

$$\text{Info(D)= - } \sum_{i=1}^{m} pi \log_2 pi$$

Where, Pi is the probability that an arbitrary tuple in D belongs to class Ci.

$$\text{Info}_A(D)=\sum_{j=1}^{V} \frac{|Dj|}{|D|} \quad \text{X Info}(D_j)$$

$$\text{Gain(A)=Info(D)- Info}_A(D)$$

Where,Info(D) is the average amount of information needed to identify the class label of a tuple in D.

- |Dj|/|D| acts as the weight of the jth partition.
- InfoA(D) is the expected informa-tion required to classify a tuple from D based on the partitioning by A.

The attribute A with the highest information gain, Gain(A), is chosen as the splitting attribute at node N().

Gain Ratio

Information gain is biased for the attribute with many outcomes. It means it prefers the attribute with a large number of distinct values. For instance, consider an attribute with a unique identifier such as customer_ID has zero info(D) because of pure partition. This maximizes the information gain and creates useless partitioning. C4.5, an improvement of ID3, uses an extension to information gain known as the gain ratio. Gain ratio handles the issue of bias by normalizing the information gain using Split Info. Java implementation of the C4.5 algorithm is known as J48, which is available in WEKA data mining tool.

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

Where,

- |Dj|/|D| acts as the weight of the jth partition.
- v is the number of discrete values in attribute A.

The gain ratio can be defined as

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

The attribute with the highest gain ratio is chosen as the splitting attribute (Source).

Gini index

Another decision tree algorithm CART (Classification and Regression Tree) uses the Gini method to create split points.

$$\text{Gini}(D)=1-\sum_{i=1}^{m} Pi^2$$

Where, pi is the probability that a tuple in D belongs to class Ci.

The Gini Index considers a binary split for each attribute. You can compute a weighted sum of the impurity of each partition. If a binary split on attribute A partitions data D into D1 and D2, the Gini index of D is

$$\text{Gini}_A(D)=\frac{|D1|}{|D|}\,\text{Gini}(D_1)+\frac{|D2|}{|D|}\,\text{Gini}(D_2)$$

In case of a discrete-valued attribute, the subset that gives the minimum gini index for that chosen is selected as a splitting attribute. In the case of continuous-valued attributes, the strategy is to select each pair of adjacent values as a possible split-point and point with smaller gini index chosen as the splitting point.

$$\Delta Gini(A) = Gini(D) - Gini_A(D).$$

The attribute with minimum Gini index is chosen as the splitting attribute.

CONVOUTIONAL LAYERS

In a CNN, the input is a tensor with a shape (number of inputs) x (input height) x (input width) x (input channels). After passing through a convolutional layer, the image becomes abstracted to a feature map, also called an activation map, with shape (number of inputs) x (feature map height) x (feature map width) x (feature map channels).

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus.[14] Each convolutional neuron processes data only for its receptive field.

Although fully connected feed forward neural networks can be used to learn features and classify data, this architecture is generally impractical for larger inputs such as high resolution images.

It would require a very high number of neurons, even in a shallow architecture, due to the large input size of images, where each pixel is a relevant input feature.

For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for *each* neuron in the second layer.

Instead, convolution reduces the number of free parameters, allowing the network to be deeper.[15] For example, regardless of image size, using a 5 x 5 tiling region, each with the same shared weights, requires only 25 learnable parameters.

Using regularized weights over fewer parameters avoids the vanishing gradients and exploding gradients problems seen during back propagation in traditional neural networks.

Furthermore, convolutional neural networks are ideal for data with a grid-like topology (such as images) as spatial relations between separate features are taken into account during convolution and/or pooling.

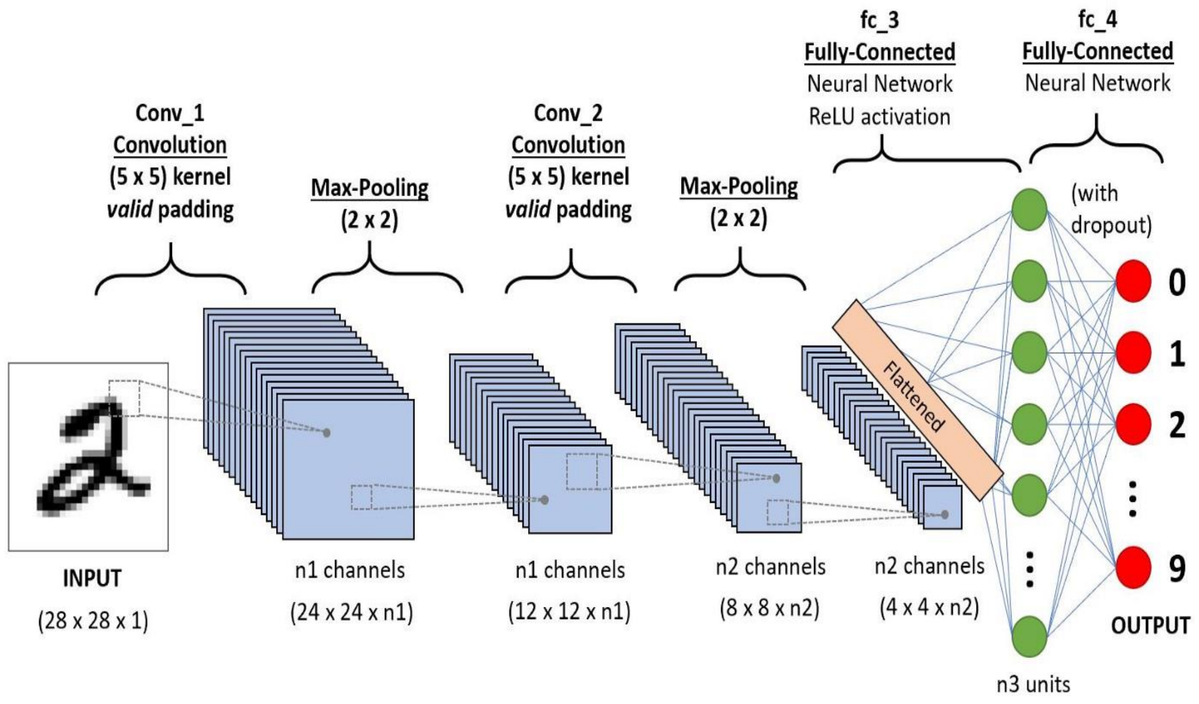


Fig.5.6 CNN Architecture

Layers in CNN

- There are five different layers in CNN
- Input layer
- Convo layer (Convo + ReLU)
- Pooling layer
- Fully connected (FC) layer
- Softmax/logistic layer
- Output layer

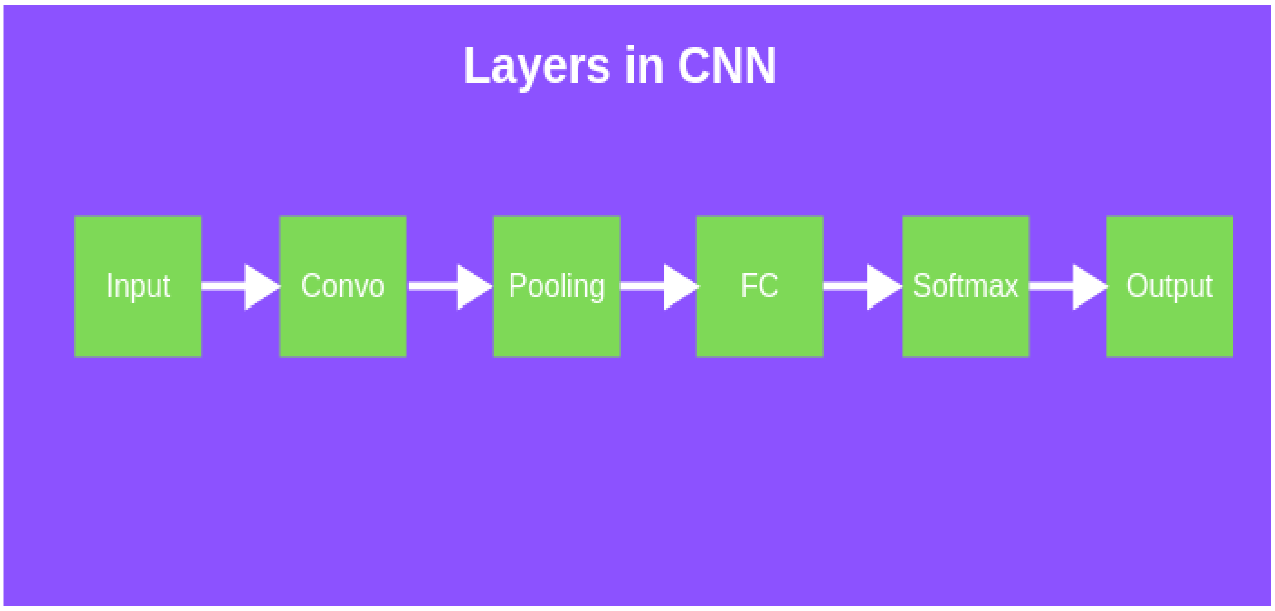


Fig.5.7 Layers in CNN

Different layers of CNN

Input Layer

Input layer in CNN should contain image data. Image data is represented by three dimensional matrix as we saw earlier. You need to reshape it into a single column. Suppose you have image of dimension 28 x 28 =784, you need to convert it into 784 x 1 before feeding into input. If you have "m" training examples then dimension of input will be (784, m).

Convo Layer

Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive field (it is a local region of the input image that has the same size as that of filter) and the filter. Result of the operation is single integer of the output volume. Then we slide the filter over the next receptive field of the same input image by a Stride and do the same operation again. We will repeat the same process again and again until we go through the whole image. The output will be the input for the next layer.

Convo layer also contains ReLU activation to make all negative value to zero.

Pooling Layer

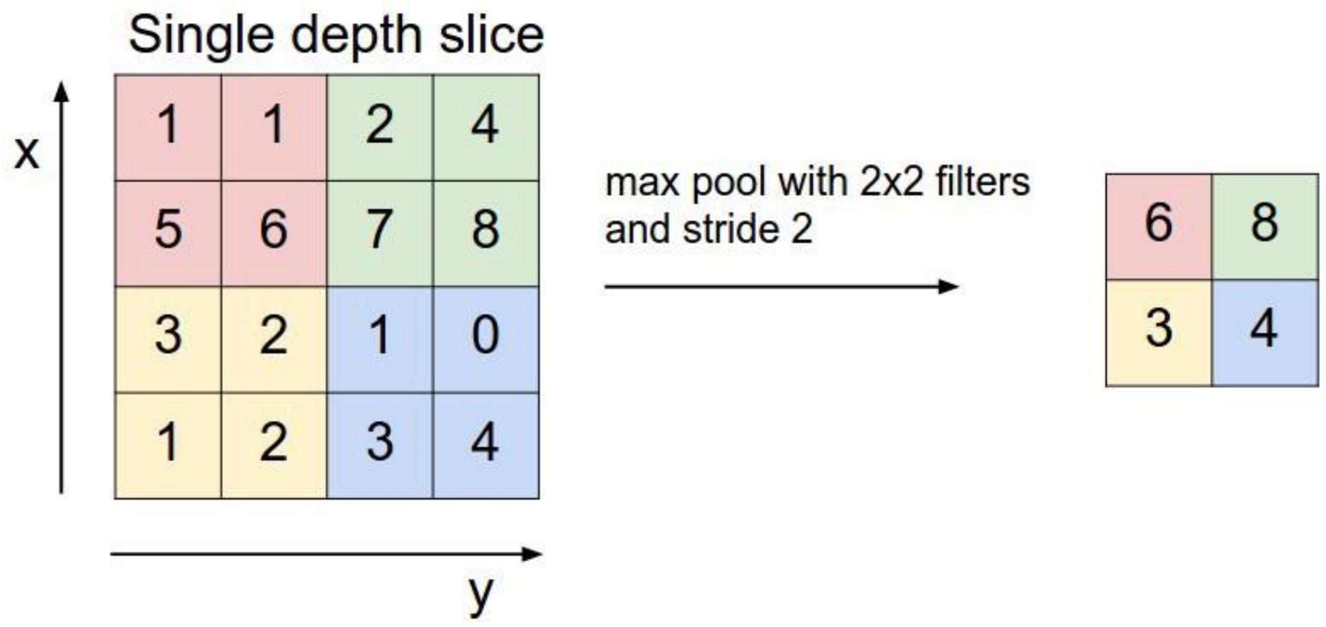


Fig.5.8 Pooling Layer

Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layer. If we apply FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive and we don’t want it. So, the max pooling is only way to reduce the spatial volume of input image. In the above example, we have applied max pooling in single depth slice with Stride of 2. You can observe the 4 x 4 dimension input is reduce to 2 x 2 dimension.

There is no parameter in pooling layer but it has two hyperparameters — Filter(F) and Stride(S).

In general, if we have input dimension W1 x H1 x D1, then

W2 = (W1−F)/S+1

H2 = (H1−F)/S+1

D2 = D1

Where W2, H2 and D2 are the width, height and depth of output.

Fully Connected Layer (FC)

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different categories by training.

Softmax / Logistic Layer

Softmax or Logistic layer is the last layer of CNN. It resides at the end of FC layer. Logistic is used for binary classification and softmax is for multi-classification.

Output Layer

Output layer contains the label which is in the form of one-hot encoded.

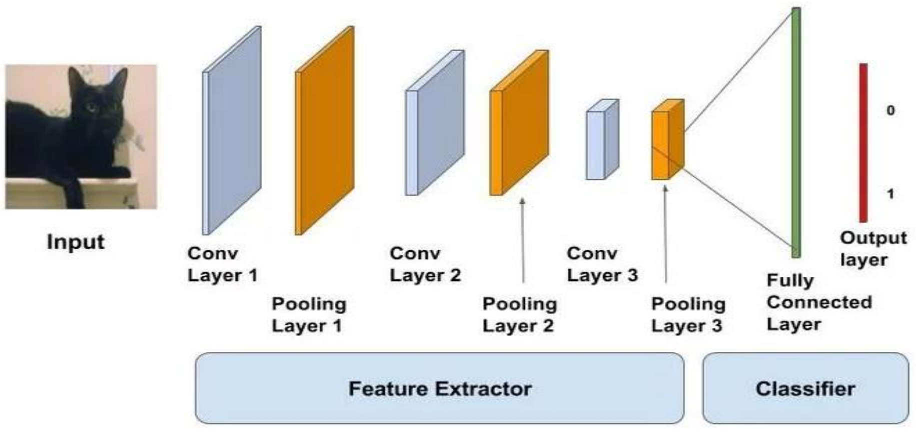


Fig.5.9 Output Layer

EXPLANATION

The proposed system of binary classification will detect whether crack is present or not. There are Training data, Test data and validation present in the proposed data model. With the help of training data, we just going to explain the system to identify the accurate results. In the proposed training process, the dataset is fed to the data preprocessing model. The data preprocessing model helps to eliminate the unwanted images and then given to that neural network layers (Conv2D,Maxpool,dense and flatten layers) used to enhance the pixel quality by convolve 2 dimensional array and max pooling operation is performed on the received input which is identification of highest value in each patch of feature map, dense layer is used to classify image based on output from convolutional layers and finally flatten layer is used to make the multidimensional input into one dimensional flatten layer or fully connected layer. On successful completion of neural network layer process, automatically R CNN file will be generated. The CNN model will analyze the image given input image and predict the correct result using the data pre-trained. Thus the correct desired output is got from CNN model .The output classified using binary classification.

TRAIN AND TESTING A DATA USING CNN ALGORITHM

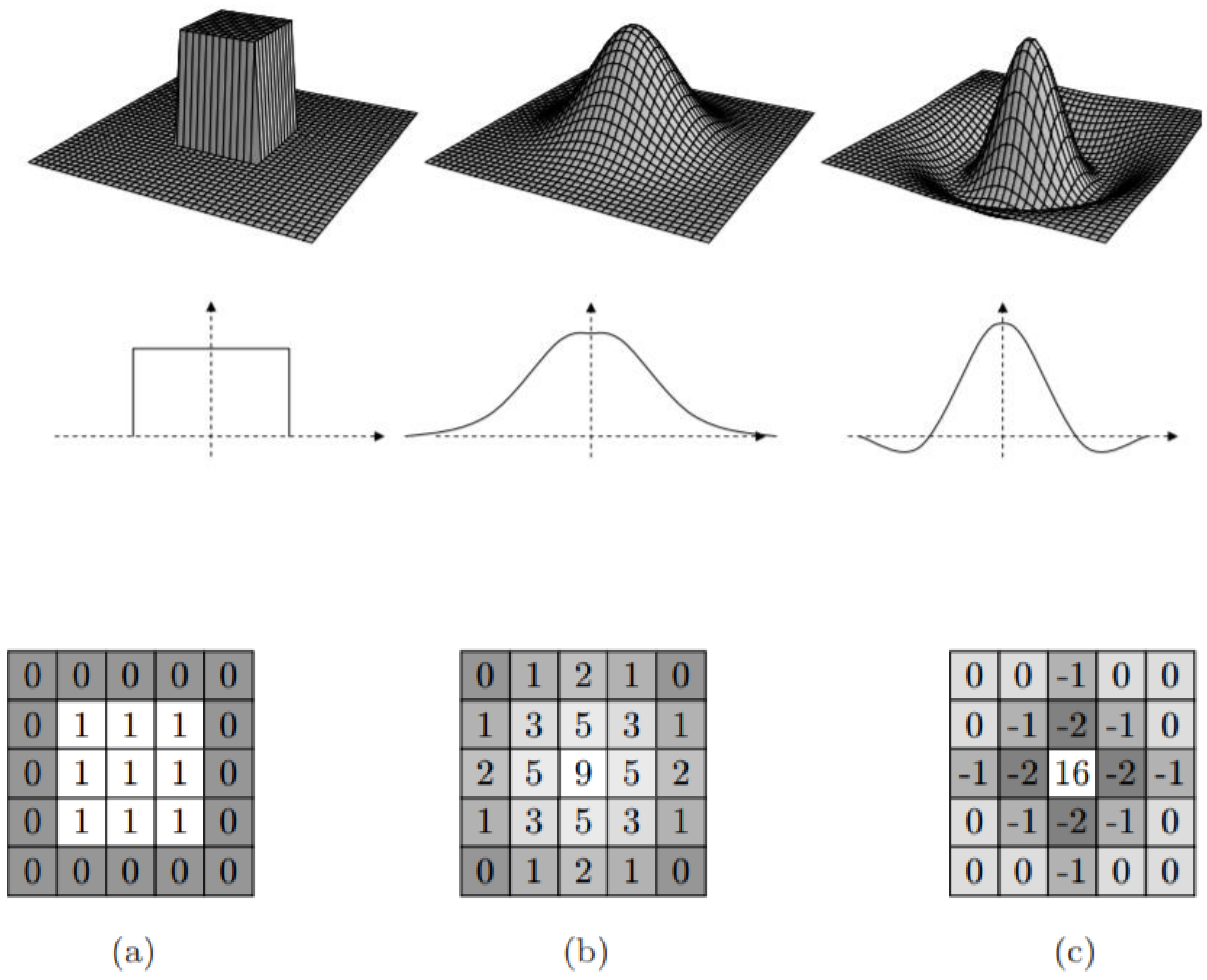
Introduction

Convolutional Neural Networks come under the subdomain of Machine Learning which is Deep Learning. Algorithms under Deep Learning process information the same way the human brain does, but obviously on a very small scale, since our brain is too complex (our brain has around 86 billion neurons).

Why CNN for Image Classification?

Image classification involves the extraction of features from the image to observe some patterns in the dataset. Using an ANN for the purpose of image classification would end up being very costly in terms of computation since the trainable parameters become extremely large.For example, if we have a 50 X 50 image of a cat, and we want to train our traditional ANN on that image to classify it into a dog or a cat the trainable parameters become – (50*50) * 100 image pixels multiplied by hidden layer + 100 bias + 2 * 100 output neurons + 2 bias = 2,50,302

We use filters when using CNNs. Filters exist of many different types according to their purpose.



0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

(a)

0	1	2	1	0
1	3	5	3	1
2	5	9	5	2
1	3	5	3	1
0	1	2	1	0

(b)

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

(c)

Fig.5.10 CNN Classification

Step 1 Choose a Dataset

Choose a dataset of your interest or you can also create your own image dataset for solving your own image classification problem. An easy place to choose a dataset is on kaggle.com.

This dataset contains 12,500 augmented images of blood cells (JPEG) with accompanying cell type labels (CSV). There are approximately 3,000 images for each of 4 different cell types grouped into 4 different folders (according to cell type). The cell types are Eosinophil, Lymphocyte, Monocyte, and Neutrophil

Here are all the libraries that we would require and the code for importing them.

```
from keras.models import Sequential
import tensorflow as tf
import tensorflow_datasets as tfds
tf.enable_eager_execution()
from keras.layers.core import Dense, Activation, Dropout, Flatten
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.optimizers import SGD, RMSprop, adam
from keras.utils import np_utils
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn import metricsfrom sklearn.utils import shuffle
from sklearn.model_selection import train_test_splitimport matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np
import os
import cv2
import randomfrom numpy import *
from PIL import Image
import theano
```

Step 2 Prepare Dataset for Training

Preparing our dataset for training will involve assigning paths and creating categories(labels), resizing our images.

Resizing images into 200 X 200

```
path_test = "/content/drive/My Drive/semester 5 - ai ml/datasetHomeAssign/TRAIN"

CATEGORIES = ["EOSINOPHIL", "LYMPHOCYTE", "MONOCYTE", "NEUTROPHIL"]

print(img_array.shape)IMG_SIZE =200

new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
```

Step 3 Create Training Data

Training is an array that will contain image pixel values and the index at which the image in the CATEGORIES list.

```
training = []def createTrainingData()
  for category in CATEGORIES
    path = os.path.join(path_test, category)
    class_num = CATEGORIES.index(category)
    for img in os.listdir(path)
      img_array = cv2.imread(os.path.join(path,img))
      new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
      training.append([new_array, class_num])createTrainingData()
```

Step 4 Shuffle the Dataset

```
random.shuffle(training)
```

Step 5 Assigning Labels and Features

This shape of both the lists will be used in Classification using the NEURAL NETWORKS.

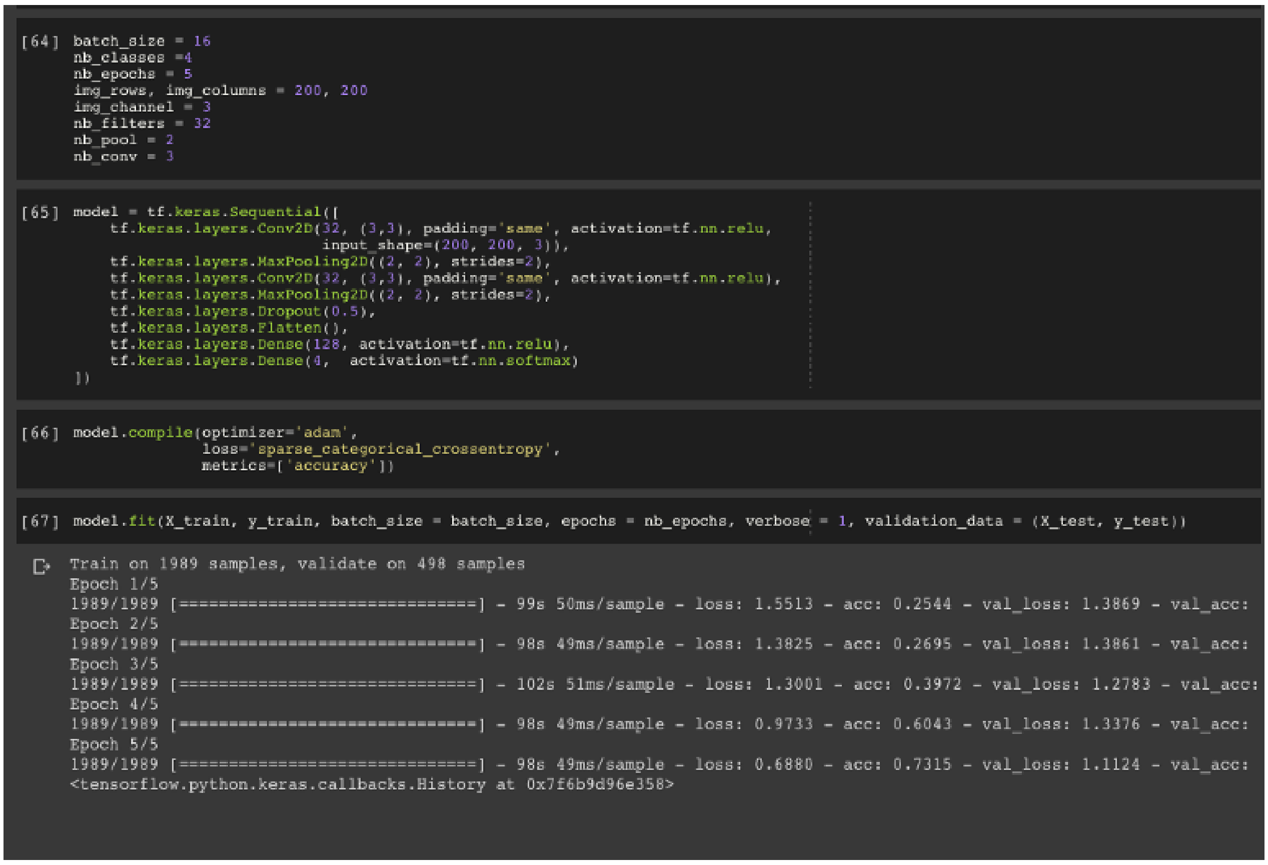
```
X =[]

y =[]for features, label in training

  X.append(features)

  y.append(label)

X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 3)
```

Step 6 Normalising X and converting labels to categorical data

```
X = X.astype('float32')

X /= 255

from keras.utils import np_utils

Y = np_utils.to_categorical(y, 4)

print(Y[100])

print(shape(Y))
```

Step 7 Split X and Y for use in CNN

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 4)
```



```
[64] batch_size = 16
     nb_classes =4
     nb_epochs = 5
     img_rows, img_columns = 200, 200
     img_channel = 3
     nb_filters = 32
     nb_pool = 2
     nb_conv = 3
```

```
[65] model = tf.keras.Sequential([
         tf.keras.layers.Conv2D(32, (3,3), padding='same', activation=tf.nn.relu,
                                input_shape=(200, 200, 3)),
         tf.keras.layers.MaxPooling2D((2, 2), strides=2),
         tf.keras.layers.Conv2D(32, (3,3), padding='same', activation=tf.nn.relu),
         tf.keras.layers.MaxPooling2D((2, 2), strides=2),
         tf.keras.layers.Dropout(0.5),
         tf.keras.layers.Flatten(),
         tf.keras.layers.Dense(128, activation=tf.nn.relu),
         tf.keras.layers.Dense(4,  activation=tf.nn.softmax)
     ])
```

```
[66] model.compile(optimizer='adam',
                   loss='sparse_categorical_crossentropy',
                   metrics=['accuracy'])
```

```
[67] model.fit(X_train, y_train, batch_size = batch_size, epochs = nb_epochs, verbose = 1, validation_data = (X_test, y_test))
```

```
Train on 1989 samples, validate on 498 samples
Epoch 1/5
1989/1989 [==============================] - 99s 50ms/sample - loss: 1.5513 - acc: 0.2544 - val_loss: 1.3869 - val_acc:
Epoch 2/5
1989/1989 [==============================] - 98s 49ms/sample - loss: 1.3825 - acc: 0.2695 - val_loss: 1.3861 - val_acc:
Epoch 3/5
1989/1989 [==============================] - 102s 51ms/sample - loss: 1.3001 - acc: 0.3972 - val_loss: 1.2783 - val_acc:
Epoch 4/5
1989/1989 [==============================] - 98s 49ms/sample - loss: 0.9733 - acc: 0.6043 - val_loss: 1.3376 - val_acc:
Epoch 5/5
1989/1989 [==============================] - 98s 49ms/sample - loss: 0.6880 - acc: 0.7315 - val_loss: 1.1124 - val_acc:
<tensorflow.python.keras.callbacks.History at 0x7f6b9d96e358>
```

Fig.5.11 Data set Design

Step 8 Define, compile and train the CNN Model

batch_size = 16

nb_classes =4

nb_epochs = 5

img_rows, img_columns = 200, 200

img_channel = 3

nb_filters = 32

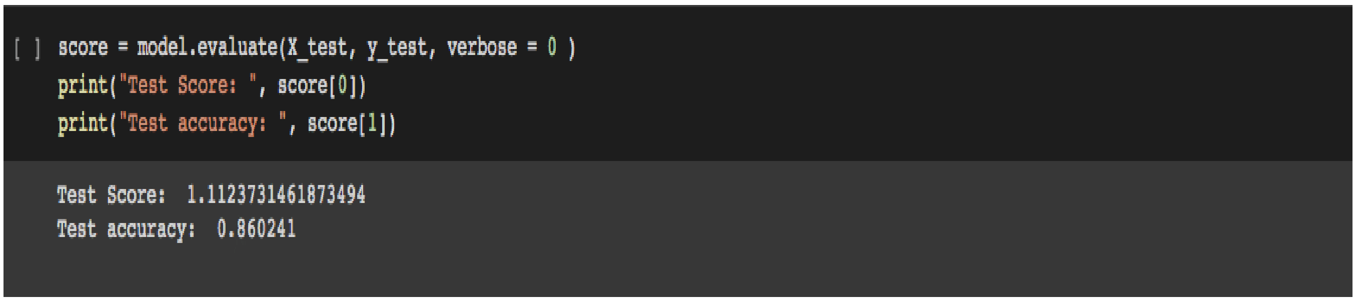
nb_pool = 2

nb_conv = 3

```
model = tf.keras.Sequential([


tf.keras.layers.Conv2D(32, (3,3), padding='same', activation=tf.nn.relu,
                        input_shape=(200, 200, 3)),
    tf.keras.layers.MaxPooling2D((2, 2), strides=2),
    tf.keras.layers.Conv2D(32, (3,3), padding='same', activation=tf.nn.relu),
    tf.keras.layers.MaxPooling2D((2, 2), strides=2),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation=tf.nn.relu),
    tf.keras.layers.Dense(4,  activation=tf.nn.softmax)
])
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics
=['accuracy'])
model.fit(X_train, y_train, batch_size = batch_size, epochs = nb_epochs, verbose
= 1, validation_data = (X_test, y_test))
```

Step 9 Accuracy and Score of model

```
score = model.evaluate(X_test, y_test, verbose = 0 )
print("Test Score ", score[0])
print("Test accuracy ", score[1])
```



```
[ ] score = model.evaluate(X_test, y_test, verbose = 0 )
    print("Test Score: ", score[0])
    print("Test accuracy: ", score[1])

    Test Score:  1.1123731461873494
    Test accuracy:  0.860241
```

Fig.5.12 Dataset Design

In these 9 simple steps, you would be ready to train your own Convolutional Neural Networks model and solve real-world problems using these skills. You can practice these skills on platforms like Analytics Vidhya and Kaggle. You can also play around by changing different parameters and discovering how you would get the best accuracy and score. Try changing the batch_size, the number of epochs or even adding/removing layers in the CNN mode.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

Acute Lymbhoplastic Leukemia (ALL) and Multimyloma (MM) is a hematological disorder and type of cancer that weakens the human immune system by generating malignant WBCs. In this work, we have proposed the CNN-based model to identify ALL and MM in microscopic blood images and compared its performance with the CNN based model in Precision, Recall, Accuracy, and Quadratic Loss. CNN model is composed of two convolutional, two maximum pooling, and three FC layers. In contrast, the CNN model has five convolutional, three maximum pooling, and three FC layers. Based on results, it is concluded that CNN performed well with high accuracy compared to LeNet-5-based model. CNN was able to classify 88.9% images correctly with 87.4% precision and 98.58% accuracy, whereas LeNet-5 correctly identified 85.3% images with 83.6% precision and 96.25% accuracy.

CNN algorithm is a competent and well-known deep learning algorithm that can be used efficiently in significant research areas, especially medical image processing. CNN can analyze and detect important features from different medical images such as CT scans, X-rays, MRI, PET, ultrasound, and hematological images. In the future, we are planning to apply the CNN architecture for other types of leukemia cell detection, such as Acute Lymphatic Leukemia (ALL), to get high accuracy.

CHAPTER 7

REFERENCES

1) Vos, Theo, C. Allen, M. Arora, R.M. Barber,Z. M. Brown,A. Carter, A.Casey et al. "Global, regional, and national incidence, prevalence, and years lived with disability for 310 diseases and injuries, 1990– 2015 a systematic analysis for the Global Burden of Disease Study 2015." The Lancet 388, no. 10053.2016, pp. 1545-1602.
2) Wang, Haidong, M. Naghavi, A. Carter, R. M. Barber , Z. M. Brown , A. Casey et al. "Global, regional, and national life expectancy, allcause mortality, and cause-specific mortality for 249 causes of death, 1980–2015 a systematic analysis for the Global Burden of Disease Study 2015." The lancet 388, no. 10053. 2016, pp. 1459-1544.
3) C. H. Pui, L. S. Frankel, A. J. Carroll, S. C. Raimondi, J. J. Shuster, D. R. Head, et al. "Clinical characteristics and treatment outcome of childhood acute lymphoblastic leukemia with the t (4; 11)(q21; q23) a collaborative study of 40 cases [see comments]". Blood, vol. 77(3). 1991, pp. 440-447.
4) S. Mandal, V. Daivajna, S. Kalsangra, Rajagopalan V, A. Kuchlous, “Computer aided system for automatic detection and marking instance of nuclei”, Proceedings in IEEE ICECCT. 2019, in press.
5) M. M. Amin, S. Kermani, A. Talebi, M. G. Oghli. "Recognition of acute lymphoblastic leukemia cells in microscopic images using kmeans clustering and support vector machine classifier". Journal of medical signals and sensors.2015, vol. 5(1), p.49.
6) S. Mohapatra, D. Patra, S. Satpathy. “An ensemble classifier system for early diagnosis of acute lymphoblastic leukemia in blood microscopic images”. Neural Computing and Applications. 2014, 24(7-8), pp. 1887-1904.

CHAPTER 8

APPENDIX

```
 from PyQt5 import QtCore, QtGui, QtWidgets
import numpy as np
from keras.preprocessing import image
from keras.models import Sequential
from keras.layers import Dense
from keras.models import model_from_json
import os
import cv2
import pandas as pd



class Ui_MainWindow(object)
    def setupUi(self, MainWindow)
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1081, 593)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(520, 30, 391, 31))
self.label.setObjectName("label")
self.IMAGESHOW = QtWidgets.QLabel(self.centralwidget)
self.IMAGESHOW.setGeometry(QtCore.QRect(540, 100, 241, 151))
self.IMAGESHOW.setFrameShape(QtWidgets.QFrame.Box)
self.IMAGESHOW.setText("")
self.IMAGESHOW.setObjectName("IMAGESHOW")
self.SELECTIMAGE = QtWidgets.QPushButton(self.centralwidget)
self.SELECTIMAGE.setGeometry(QtCore.QRect(520, 290, 111, 31))
self.SELECTIMAGE.setObjectName("SELECTIMAGE")
self.CLASSIFY = QtWidgets.QPushButton(self.centralwidget)
self.CLASSIFY.setGeometry(QtCore.QRect(690, 290, 111, 31))
self.CLASSIFY.setObjectName("CLASSIFY")
self.PREDICTION = QtWidgets.QLabel(self.centralwidget)
self.PREDICTION.setGeometry(QtCore.QRect(530, 360, 91, 21))
self.PREDICTION.setObjectName("PREDICTION")
self.PREDICT_OUTPUT = QtWidgets.QLabel(self.centralwidget)
self.PREDICT_OUTPUT.setGeometry(QtCore.QRect(670, 350, 151, 31))
self.PREDICT_OUTPUT.setFrameShape(QtWidgets.QFrame.Box)
self.PREDICT_OUTPUT.setText("")
self.PREDICT_OUTPUT.setObjectName("PREDICT_OUTPUT")
self.PERCENTAGE = QtWidgets.QLabel(self.centralwidget)
self.PERCENTAGE.setGeometry(QtCore.QRect(530, 420, 91, 21))
```

```
        self.PERCENTAGE.setObjectName("PERCENTAGE")
        self.PERCENTAGE_OUTPUT = QtWidgets.QLabel(self.centralwidget)
        self.PERCENTAGE_OUTPUT.setGeometry(QtCore.QRect(670, 410, 151,
31))
        self.PERCENTAGE_OUTPUT.setFrameShape(QtWidgets.QFrame.Box)
        self.PERCENTAGE_OUTPUT.setText("")

self.PERCENTAGE_OUTPUT.setObjectName("PERCENTAGE_OUTPUT")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1081, 21))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)


        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)
        self.SELECTIMAGE.clicked.connect(self.loadImage)
        self.CLASSIFY.clicked.connect(self.classifyFunction)
```

```
def loadImage(self)
    fileName, _ = QtWidgets.QFileDialog.getOpenFileName(None, "Select Image", "", "Image Files (*.png *.jpg *jpeg *.bmp);;All Files (*)") # Ask for file
    if fileName # If the user gives a file
        print(fileName)
        self.file=fileName
        pixmap = QtGui.QPixmap(fileName) # Setup pixmap with the provided image
        pixmap = pixmap.scaled(self.IMAGESHOW.width(), self.IMAGESHOW.height(), QtCore.Qt.KeepAspectRatio) # Scale pixmap
        self.IMAGESHOW.setPixmap(pixmap) # Set the pixmap onto the label
        self.IMAGESHOW.setAlignment(QtCore.Qt.AlignCenter) # Align the label to center



def classifyFunction(self)
    json_file = open('model.json', 'r')
    loaded_model_json = json_file.read()
    json_file.close()
    model = model_from_json(loaded_model_json)
    model.load_weights("model.h5")
    print("Loaded model from disk")
    try
```

```
path2=self.file
test_image = image.load_img(path2, target_size = (512, 512))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis=0)
result = model.predict(test_image)


if result[0][0] == 1
    prediction = 'LUKEAMEA'
    PREDICTION = prediction
    self.PREDICT_OUTPUT.setText(PREDICTION)
    img = cv2.imread(path2)
    dataset = pd.read_csv("leukemia.csv")
    print(dataset)
    x = dataset.iloc[,-1] #independent
    y = dataset.iloc[,-1] #dependent
    from sklearn.model_selection import train_test_split
    X_train, X_test, Y_train, Y_test = train_test_split(x,y, test_size=0.80,
random_state=0)
    print(X_train)
    print(Y_train)
    print(X_test)
    print(Y_test)
```

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, Y_train)
Y_predict = classifier.predict(X_test)
img = cv2.resize(img,(400,400))
#cv2.imshow("Original Frame",img)
## convert to hsv
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
#cv2.imshow("hsv",hsv)
## mask of red (36,0,0) ~ (70, 255,255)
mask1 = cv2.inRange(hsv, (0,0,100), (0,0,255)) #red
#cv2.imshow("mask1",mask1)
red= cv2.countNonZero(mask1)
print("red = ",red)
img = cv2.GaussianBlur(img,(5,5),2)
im_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(im_gray,127,255,0)
count = cv2.countNonZero(thresh)
#print(count)
RED=((red+count)/2)*0.001000
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
```

```
            for contour in contours
                cv2.drawContours(im_gray, contours, -1, (0,255,0), 6)
                #cv2.imshow("contour",im_gray)
            output = classifier.predict([[red]])
            print("Predicted New Output = ",output)
            if output == 1
                print("Affected")
            RED=int(RED)
            if RED <11
                RED= int(RED)
            PERCENTAGE=str(RED)
            self.PERCENTAGE_OUTPUT.setText(PERCENTAGE)
        else
            prediction = 'No LUKEAMEA'
            PREDICTION = prediction
            self.PREDICT_OUTPUT.setText(PREDICTION)
    except Exception as e
        print(e)


def retranslateUi(self, MainWindow)
    _translate = QtCore.QCoreApplication.translate
```

```
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.label.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" font-size10pt; font-weight600;\">LEUKEMIA DETECTION USING DEEP LEARNING</span></p></body></html>"))
        self.SELECTIMAGE.setText(_translate("MainWindow", "SELECT IMAGE"))
        self.CLASSIFY.setText(_translate("MainWindow", "CLASSIFY"))
        self.PREDICTION.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" font-size10pt; font-weight600;\">PREDICTION</span></p></body></html>"))
        self.PERCENTAGE.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" font-size10pt; font-weight600;\">PERCENTAGE</span></p><p><br/></p></body></html>"))


if __name__ == "__main__"
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

CHAPTER 9

OUTPUT

Output for normal image:

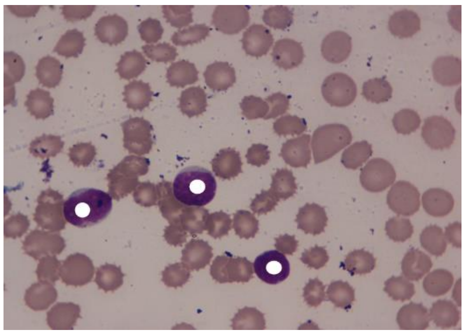
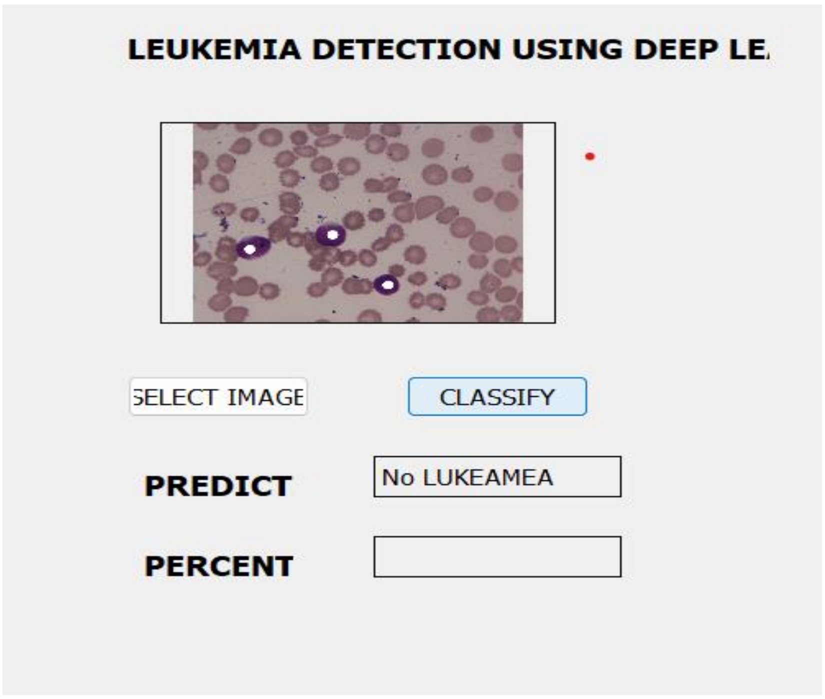


Fig.9.1 Normal Cell



LEUKEMIA DETECTION USING DEEP LE

PREDICT No LUKEAMEA

PERCENT

Fig.9.2 Output Image for Normal Cell

Output for Affected image:

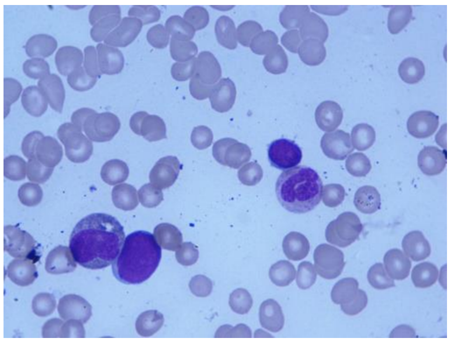
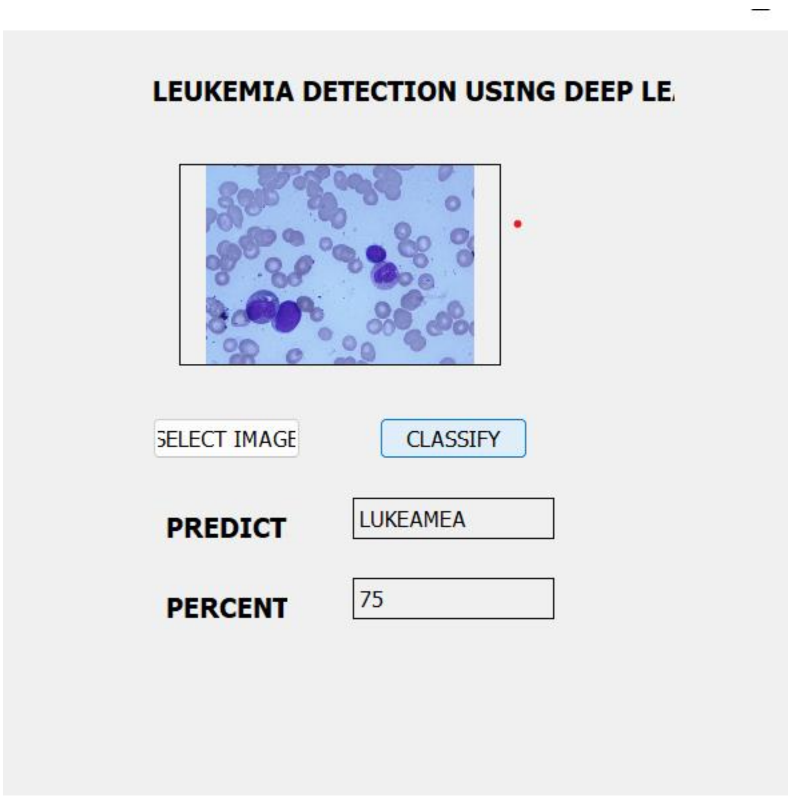


Fig.9.3 Affected Cell



PREDICT LUKEAMEA

PERCENT 75

Fig.9.4 Output Image for Affected Cell.