Objectives: our project is made for , I.preprocess the image: i)read image ii)resize image iii)remove noice iv)segmentation v)morphology II.applying the CNN algorithm to the dataset:

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.

The dataset I'm going with can be found here.

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) Step 8: Define, compile and train the CNN Model 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])

III.how deep neural network detect the disease: neural networks provide a mapping between input -such as an image of a diseased plant-to an output -such as an crop-disease pair. IV.how to find the acuracy of the model: Balanced accuracy = (Sensitivity + Specificity) / 2. Balanced accuracy = (0.75 + 9868) / 2. Balanced accuracy = 0.8684 V.buid web application using flask framework: Step 1 — Installing Flask. … Step 2 — Creating a Base Application. … Step 3 — Using HTML templates. … Step 4 — Setting up the Database. … Step 5 — Displaying All Posts. … Step 6 — Displaying a Single Post