# importing required libraries:

import numpy as np

import cv2

import os

import random

import matplotlib.pyplot as plt

import pickle

# setting the path to our eye dataset:

Directory =r’C:\Users\akash\AppData\Local\Programs\Python\Python38\DriverDrowsyness\train’

# specify two categories on which we want to train our data:

CATEGORIES = ['Closed' , 'Open']

#setting image size:

img\_size = 24

data = []

#iterating over each image and get the image in array form,

for category in CATEGORIES:

folder = os.path.join(Directory,category)

label = CATEGORIES.index(category)

for img in os.listdir(folder):

img\_path = os.path.join(folder, img)

img\_arr = cv2.imread(img\_path)

img\_arr = cv2.cvtColor(img\_arr, cv2.COLOR\_BGR2GRAY)

img\_arr = cv2.resize(img\_arr,(img\_size, img\_size),1)

data.append([img\_arr , label])

# see the length of data:

len(data)

# we shuffle the data to get random images of open eyes and closed eyes:

random.shuffle(data)

# dividing features and label for training the model:

X = []

Y = []

for features,label in data:

X.append(features)

Y.append(label)

#covert them into array:

X = np.array(X)

Y = np.array(Y)

# save the data into system:

pickle.dump(X , open('X.pkl' , 'wb'))

pickle.dump(Y , open('Y.pkl' , 'wb'))

# normalize the image array:

X = X/255

# reshape the X array to (24,24,1)

img\_rows,img\_cols = 24,24

X = X.reshape(X.shape[0],img\_rows,img\_cols,1)

X.shape

# creating model:

model = Sequential()

model.add(Conv2D(64 , (3,3) , activation = 'relu' , input\_shape= X.shape[1:]))

model.add(MaxPooling2D((1,1)))

model.add(Conv2D(64 , (3,3) , activation = 'relu'))

model.add(MaxPooling2D((1,1)))

model.add(Conv2D(64 , (3,3) , activation = 'relu'))

model.add(MaxPooling2D((1,1)))

model.add(Flatten())

model.add(Dense(128, activation = 'relu'))

model.add(Dense(2, activation = 'softmax'))

# compile model that we have created

model.compile(optimizer = 'adam' , loss = 'sparse\_categorical\_crossentropy' , metrics = ['accuracy'])

# fit X , Y to the model to see accuracy of model:

model.fit(X, Y, epochs = 5 , validation\_split = 0.1 , batch\_size = 32)

# save model and architecture to single file

model.save("custmodel.h5")