B5 -IRIS RECOGNITION USING MACHINE LEARNING TECHNIQUE

In [1]:

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
import numpy as np
import matplotlib.pyplot as plt
import os
from keras.utils.np_utils import to_categorical
from keras.layers import  MaxPooling2D
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D
from keras.models import Sequential
from keras.models import model_from_json
import pickle
import cv2
from keras.preprocessing import image
from skimage import data, color
from skimage.transform import hough_circle, hough_circle_peaks
from skimage.feature import canny
from skimage.draw import circle_perimeter
from skimage.util import img_as_ubyte

main = tkinter.Tk()
main.title("Iris Recognition using Machine Learning Technique") #designing main sc
main.geometry("1300x1200")

global filename
global model

def getIrisFeatures(image):
    global count
    img = cv2.imread(image,0)
    img = cv2.medianBlur(img,5)
    cimg = cv2.cvtColor(img,cv2.COLOR_GRAY2BGR)
    circles = cv2.HoughCircles(img,cv2.HOUGH_GRADIENT,1,10,param1=63,param2=70,minI
    if circles is not None:
        height,width = img.shape
        r = 0
        mask = np.zeros((height,width), np.uint8)
        for i in circles[0,:]:
            cv2.circle(cimg,(i[0],i[1]),int(i[2]),(0,0,0))
            cv2.circle(mask,(i[0],i[1]),int(i[2]),(255,255,255),thickness=0)
            blank_image = cimg[:int(i[1]),:int(i[1])]

            masked_data = cv2.bitwise_and(cimg, cimg, mask=mask)
            _,thresh = cv2.threshold(mask,1,255,cv2.THRESH_BINARY)
            contours = cv2.findContours(thresh,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_S
            x,y,w,h = cv2.boundingRect(contours[0][0])
            crop = img[y:y+h,x:x+w]
            r = i[2]
        cv2.imwrite("test.png",crop)
    else:
        count = count + 1
        miss.append(image)
```

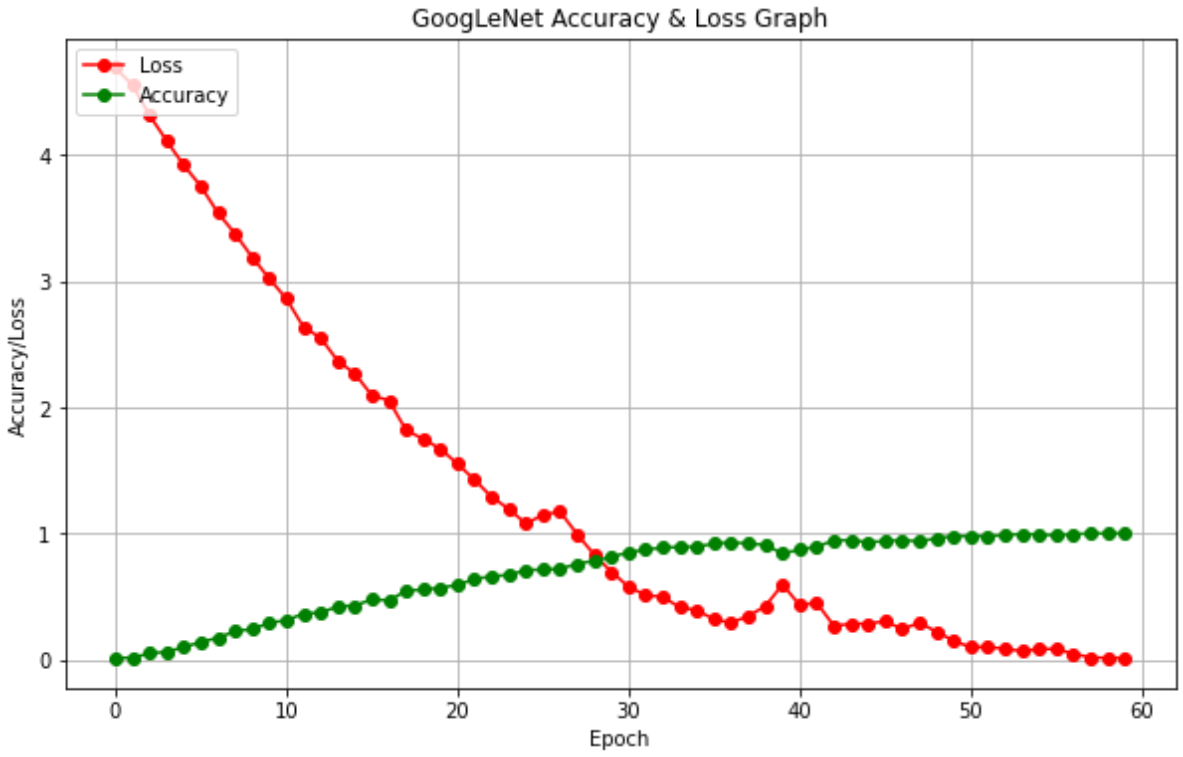
```
    return cv2.imread("test.png")

def uploadDataset():
    global filename
    filename = filedialog.askdirectory(initialdir=".")
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n\n");

def loadModel():
    global model
    text.delete('1.0', END)
    X_train = np.load('model/X.txt.npy')
    Y_train = np.load('model/Y.txt.npy')
    print(X_train.shape)
    print(Y_train.shape)
    text.insert(END,'Dataset contains total '+str(X_train.shape[0])+' iris images
    if os.path.exists('model/model.json'):
        with open('model/model.json', "r") as json_file:
            loaded_model_json = json_file.read()
            model = model_from_json(loaded_model_json)
        model.load_weights("model/model_weights.h5")
        model._make_predict_function()
        print(model.summary())
        f = open('model/history.pckl', 'rb')
        data = pickle.load(f)
        f.close()
        acc = data['accuracy']
        accuracy = acc[59] * 100
        text.insert(END,"CNN Model Prediction Accuracy = "+str(accuracy)+"\n\n")
        text.insert(END,"See Black Console to view CNN layers\n")
    else:
        model = Sequential()
        model.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation =
        model.add(MaxPooling2D(pool_size = (2, 2)))
        model.add(Convolution2D(32, 3, 3, activation = 'relu'))
        modeladd(MaxPooling2D(pool_size = (2, 2)))
        model.add(Flatten())
        model.add(Dense(output_dim = 256, activation = 'relu'))
        model.add(Dense(output_dim = 108, activation = 'softmax'))
        print(model.summary())
        model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metri
        hist = model.fit(X_train, Y_train, batch_size=16, epochs=60, shuffle=True,
        model.save_weights('model/model_weights.h5')
        model_json = classifier.to_json()
        with open("model/model.json", "w") as json_file:
            json_file.write(model_json)
        f = open('model/history.pckl', 'wb')
        pickle.dump(hist.history, f)
        f.close()
        f = open('model/history.pckl', 'rb')
        data = pickle.load(f)
        f.close()
        acc = data['accuracy']
        accuracy = acc[59] * 100
        text.insert(END,"CNN Model Prediction Accuracy = "+str(accuracy)+"\n\n")
        text.insert(END,"See Black Console to view CNN layers\n")

def predictChange():
    filename = filedialog.askopenfilename(initialdir="testSamples")
    print(filename)
    image = getIrisFeatures(filename)
    img = cv2.resize(image, (64,64))
    im2arr = np.array(img)
    im2arr = im2arr.reshape(1,64,64,3)
```

```
    img = np.asarray(im2arr)
    img = img.astype('float32')
    img = img/255
    preds = model.predict(img)
    predict = np.argmax(preds) + 1
    print(predict)
    img = cv2.imread(filename)
    img = cv2.resize(img, (600,400))
    img1 = cv2.imread('test.png')
    img1 = cv2.resize(img1, (400,200))
    cv2.putText(img, 'Person ID Predicted from Iris Recognition is : '+str(predict
    cv2.imshow('Person ID Predicted from Iris Recognition is : '+str(predict), img
    cv2.imshow('Iris features extacted from image', img1)
    cv2.waitKey(0)



def graph():
    f = open('model/history.pckl', 'rb')
    data = pickle.load(f)
    f.close()

    accuracy = data['accuracy']
    loss = data['loss']
    plt.figure(figsize=(10,6))
    plt.grid(True)
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy/Loss')
    plt.plot(loss, 'ro-', color = 'red')
    plt.plot(accuracy, 'ro-', color = 'green')
    plt.legend(['Loss', 'Accuracy'], loc='upper left')
    #plt.xticks(wordloss.index)
    plt.title('GoogLeNet Accuracy & Loss Graph')
    plt.show()

def close():
    main.destroy()

font = ('times', 16, 'bold')
title = Label(main, text='Iris Recognition using Machine Learning Technique')
title.config(bg='goldenrod2', fg='black')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 12, 'bold')
text=Text(main,height=20,width=150)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=50,y=120)
text.config(font=font1)


font1 = ('times', 13, 'bold')
uploadButton = Button(main, text="Upload Iris Dataset", command=uploadDataset, bg=
uploadButton.place(x=50,y=550)
uploadButton.config(font=font1)

modelButton = Button(main, text="Generate & Load CNN Model", command=loadModel, bg=
modelButton.place(x=240,y=550)
modelButton.config(font=font1)

graphButton = Button(main, text="Accuracy & Loss Graph", command=graph, bg='#ffb3f
graphButton.place(x=505,y=550)
```

```
graphButton.config(font=font1)

predictButton = Button(main, text="Upload Iris Test Image & Recognize", command=pre
predictButton.place(x=730,y=550)
predictButton.config(font=font1)

exitButton = Button(main, text="Exit", command=close, bg='#ffb3fe')
exitButton.place(x=1050,y=550)
exitButton.config(font=font1)


main.config(bg='SpringGreen2')
main.mainloop()
```

```
(683, 64, 64, 3)
(683, 108)
```

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\hi\anaconda3\lib\tkinter\__init__.py", line 1892, in __call__
    return self.func(*args)
  File "C:\Users\hi\AppData\Local\Temp\ipykernel_7332\302105068.py", line 78, in l
oadModel
    model._make_predict_function()
AttributeError: 'Sequential' object has no attribute '_make_predict_function'
C:\Users\hi\AppData\Local\Temp\ipykernel_7332\302105068.py:149: UserWarning: color
is redundantly defined by the 'color' keyword argument and the fmt string "ro-" (-
> color='r'). The keyword argument will take precedence.
  plt.plot(loss, 'ro-', color = 'red')
C:\Users\hi\AppData\Local\Temp\ipykernel_7332\302105068.py:150: UserWarning: color
is redundantly defined by the 'color' keyword argument and the fmt string "ro-" (-
> color='r'). The keyword argument will take precedence.
  plt.plot(accuracy, 'ro-', color = 'green')
```




```
C:/Users/hi/Downloads/B5 -IRIS RECOGNITION USING MACHINE LEARNING TECHNIQUE/B5 -IR
IS RECOGNITION USING MACHINE LEARNING TECHNIQUE/6.IRIS RECOGNITION USING MACHINE L
EARNING TECHNIQUE/SOURCE CODE/IrisRecognition/testSamples/a.jpg
```

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\hi\anaconda3\lib\tkinter\__init__.py", line 1892, in __call__
    return self.func(*args)
  File "C:\Users\hi\AppData\Local\Temp\ipykernel_7332\302105068.py", line 117, in
predictChange
    image = getIrisFeatures(filename)
  File "C:\Users\hi\AppData\Local\Temp\ipykernel_7332\302105068.py", line 43, in g
etIrisFeatures
    cv2.circle(cimg,(i[0],i[1]),int(i[2]),(0,0,0))
cv2.error: OpenCV(4.6.0) :-1: error: (-5:Bad argument) in function 'circle'
> Overload resolution failed:
>  - Can't parse 'center'. Sequence item with index 0 has a wrong type
>  - Can't parse 'center'. Sequence item with index 0 has a wrong type
```

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\hi\anaconda3\lib\tkinter\__init__.py", line 1892, in __call__
    return self.func(*args)
  File "C:\Users\hi\AppData\Local\Temp\ipykernel_7332\302105068.py", line 117, in
predictChange
    image = getIrisFeatures(filename)
  File "C:\Users\hi\AppData\Local\Temp\ipykernel_7332\302105068.py", line 35, in g
etIrisFeatures
    img = cv2.medianBlur(img,5)
cv2.error: OpenCV(4.6.0) D:\a\opencv-python\opencv-python\opencv\modules\imgproc\s
rc\median_blur.dispatch.cpp:283: error: (-215:Assertion failed) !_src0.empty() in
function 'cv::medianBlur'
```

In [ ]:

In [ ]:

In [ ]: