**STUDENT MANAGEMENT SYSTEM USING COMPUTER VISION AND DEEP LEARNING**

**CODE:**

*import* os

*import* cv2

*import* numpy *as* np

*from* keras.models *import* Sequential

*from* keras.layers *import* Conv2D, MaxPooling2D, Flatten, Dense

*import* json

*import* time

*import* random

*# Function to register a person and generate a JSON record*

def register\_person(*person\_name*):

    data = {

        ‘name’: *person\_name*,

        ‘ID’: random.randint(2016000, 2017000)

    }

*return* data

*# Path to the folder containing training images*

*# C:\Users\ayush\OneDrive – Graphic Era University\Desktop\FOLDERS\MY FOLDER\COLLEGE\SUBJECTS\IMAGE PROCESSING & COMPUTER VISION\PROJECT\images*

train\_path = r’C:\\Users\\ayush\\OneDrive – Graphic Era University\Desktop\\FOLDERS\\MY FOLDER\\COLLEGE\SUBJECTS\\IMAGE PROCESSING & COMPUTER VISION\\PROJECT\\images’

*# Get the list of people from the training folder*

people = os.listdir(train\_path)

*# Prepare empty lists to store the training data and labels*

X\_train = []

y\_train = []

*# Load and preprocess training images*

*for* person\_id, person *in* enumerate(people):

    person\_path = os.path.join(train\_path, person)

*for* image\_name *in* os.listdir(person\_path):

        image\_path = os.path.join(person\_path, image\_name)

        image = cv2.imread(image\_path)

        image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)  *# Convert to RGB*

        image = cv2.resize(image, (128, 128))  *# Resize to a fixed size*

        X\_train.append(image)

        y\_train.append(person\_id)

*# Create a dictionary to map label indices to person names*

label\_to\_person = {i: person *for* I, person *in* enumerate(people)}

*# Convert the training data and labels to NumPy arrays*

X\_train = np.array(X\_train)

y\_train = np.array(y\_train)

*# Normalize the pixel values to the range of [0, 1]*

X\_train = X\_train.astype(‘float32’) / 255.0

*# Create a CNN model*

model = Sequential()

model.add(Conv2D(32, (3, 3), *activation*=’relu’, *input\_shape*=(128, 128, 3)))  *# Updated input shape*

model.add(MaxPooling2D((2, 2)))

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

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), *activation*=’relu’))

model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

model.add(Dense(128, *activation*=’relu’))

model.add(Dense(len(people), *activation*=’softmax’))

*# Compile the model*

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

*# Train the model*

model.fit(X\_train, y\_train, *epochs*=200, *batch\_size*=32)

*# Load the face detection classifier*

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + ‘haarcascade\_frontalface\_default.xml’)

*# Initialize video capture from the default camera*

video\_capture = cv2.VideoCapture(0)

*# Variable for tracking time*

last\_detection\_time = time.time()

*# Initialize the person\_detected variable*

person\_detected = False

*# List to hold all student records*

student\_records = []

*while* True:

*# Read the current frame from the video stream*

    ret, frame = video\_capture.read()

*# Convert the frame to grayscale for face detection*

    gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

*# Detect faces in the frame*

    faces = face\_cascade.detectMultiScale(gray, *scaleFactor*=1.3, *minNeighbors*=5, *minSize*=(100, 100))

*# Reset the person\_detected variable at the start of each loop iteration*

    person\_detected = False

*# Iterate over detected faces*

*for* (x, y, w, h) *in* faces:

*# Extract the face region from the frame*

        face = frame[y:y+h, x:x+w]

*# Preprocess the face image*

        face = cv2.resize(face, (128, 128))

        face = cv2.cvtColor(face, cv2.COLOR\_BGR2RGB)

        face = np.expand\_dims(face, *axis*=0)

        face = face.astype(‘float32’) / 255.0

*# Make prediction on the face*

        prediction = model.predict(face)

        label = np.argmax(prediction)

*# Update the person\_detected variable if a person is detected*

        person\_detected = True

*# last\_detection\_time = time.time()*

*# Draw bounding box and label on the frame*

        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

        cv2.putText(frame, people[label], (x, y-10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 255, 0), 2)

*# If a person is detected for 3-5 seconds, register them*

*if* person\_detected:

*if* time.time() – last\_detection\_time >= 5:

            student\_name = label\_to\_person[label]

            student\_record = register\_person(student\_name)

*# Check if the record already exists in student\_records*

*if* student\_record[“name”] not in student\_records:

                student\_records.append(student\_record)

            last\_detection\_time = time.time()

*# Display the resulting frame*

    cv2.imshow(‘Face Recognition’, frame)

*# Break the loop if ‘q’ is pressed*

*if* cv2.waitKey(1) & 0xFF == ord(‘q’):

*break*

*# Save all student records to a JSON file*

json\_path = ‘student\_records.json’

*with* open(json\_path, ‘w’) *as* json\_file:

    json.dump(student\_records, json\_file)

*# Release the video capture and close all windows*

video\_capture.release()

cv2.destroyAllWindows()

**CONFIGURATIONS:**

1. Conda package with following Python package installations:  
 a. Tensorflow  
 b. Keras  
 c. OpenCV  
 d. json  
 e. time

2. The system setup must:  
 a. possess a camera input (embedded/external)  
 b. Min. 8 GB RAM with shared or dedicated GPU  
 c. Windows 8 or above installation

3. Python version must be 3+