**ASSIGNMENT HELP**

**MANUAL**



SUBMITTED

TO

VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE

FOR THE SKILL AND COMPETENCY EVALUATION OF

DEEP LEARNING [ CAUA31202]

IN

**CSE AI DEPARTMENT**

BY

**Vedant Rakesh Mukhekar**

**Class: T.Y. BTech Division: A Batch: A2**

**Batch Teacher**

**Dr. ANURADHA YENKIKAR.**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **SR. NO.** | **CONTENTS** | **PAGE NO.** |
| **1** | **PROBLEM STATEMENT** | **4-5** |
| **2** | **LIBRARY USED** | **5-6** |
| **3** | **THEORY** | **6-7** |
| **4** | **METHODOLOGY** | **7-14** |
| **5** | **ADVANTAGES & DISADVANTAGES** | **14-15** |
| **6** | **WORKING** | **16-17** |
| **7** | **DIAGRAM** | **17-18** |
| **8** | **CONCLUSION** | **18-19** |

### Problem Statement

The aim of this project is to implement a facial recognition system using OpenCV and deep learning techniques to classify faces into two categories (binary classification). This system will utilize a convolutional neural network (CNN) trained on a dataset of facial images to recognize and distinguish between two individuals. The objective is to achieve high accuracy in correctly classifying the input images.

### Libraries Used

* **OpenCV**: A powerful computer vision library used for image processing and real-time computer vision applications.
* **TensorFlow**: An open-source machine learning library that provides tools for building and training deep learning models.
* **Keras**: A high-level neural networks API that runs on top of TensorFlow.
* **NumPy**: A library for numerical operations in Python.
* **Matplotlib**: A plotting library for visualizing data.

### Theory

Facial recognition involves identifying and verifying individuals based on their facial features. A common approach to facial recognition is to use **Convolutional Neural Networks (CNNs)**, which are specifically designed to process and analyze visual data. CNNs utilize multiple layers of convolutional filters to extract hierarchical features from images, allowing the model to learn relevant patterns for classification tasks.

#### Key Concepts

* **Convolutional Layers**: Layers that apply convolutional operations on the input images to extract features.
* **Pooling Layers**: Layers that reduce the spatial dimensions of the feature maps, helping to minimize computation and control overfitting.
* **Activation Functions**: Functions that introduce non-linearity into the network. Common functions include ReLU (Rectified Linear Unit) and Sigmoid.
* **Loss Function**: A measure of how well the model predicts the target classes. For binary classification, Binary Crossentropy is often used.
* **Optimizer**: An algorithm used to adjust the weights of the network based on the gradients of the loss function. Common optimizers include Adam and SGD (Stochastic Gradient Descent).

#### Applications of Facial Recognition

* **Security and Surveillance**: Identifying individuals in security systems and monitoring environments.
* **Authentication**: Using facial recognition for access control to devices and secure systems.
* **User Experience Enhancement**: Personalizing user interactions based on facial recognition.

### Methodology

1. **Set Up the Environment**: Install the necessary libraries, including OpenCV, TensorFlow, and Keras.
2. **Collect the Dataset**: Use a dataset containing facial images of two individuals. The images should be labeled accordingly.
3. **Preprocess the Data**:
   * Resize images to a uniform dimension.
   * Normalize pixel values to the range [0, 1].
   * Split the dataset into training and validation sets.
4. **Build the CNN Model**:
   * Define the architecture of the convolutional neural network.
   * Specify activation functions, loss function, and optimizer.
5. **Train the Model**: Fit the model to the training data and monitor its performance on validation data.
6. **Evaluate the Model**: Test the model on a separate dataset to assess its classification accuracy.
7. **Implement Facial Recognition**: Use OpenCV to capture real-time video feed and apply the trained model for facial recognition.

### Advantages & Disadvantages

* **Advantages**:
  + **Accuracy**: Deep learning models, especially CNNs, achieve high accuracy in image classification tasks.
  + **Scalability**: The model can be scaled to include more classes and improved with larger datasets.
  + **Real-Time Processing**: Capable of performing facial recognition in real-time using camera feeds.
* **Disadvantages**:
  + **Computationally Intensive**: Training deep learning models can require significant computational resources.
  + **Data Requirements**: High-quality and diverse datasets are needed to ensure effective training.
  + **Overfitting**: Risk of overfitting the model if not properly validated.

### Working Example (Python Code)

Here’s a simple implementation of facial recognition using OpenCV and a CNN for binary classification:

python

Copy code

import numpy as np

import cv2

import os

import tensorflow as tf

from tensorflow.keras.models import Sequential

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

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.model\_selection import train\_test\_split

# Load images from dataset

def load\_images\_from\_folder(folder):

images = []

labels = []

for label in os.listdir(folder):

for filename in os.listdir(os.path.join(folder, label)):

img = cv2.imread(os.path.join(folder, label, filename))

if img is not None:

img = cv2.resize(img, (64, 64)) # Resize images to 64x64

images.append(img)

labels.append(label)

return np.array(images), np.array(labels)

# Path to dataset

dataset\_path = 'path\_to\_your\_dataset'

X, y = load\_images\_from\_folder(dataset\_path)

# Normalize pixel values

X = X.astype('float32') / 255.0

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Convert labels to binary

from sklearn.preprocessing import LabelBinarizer

lb = LabelBinarizer()

y\_train = lb.fit\_transform(y\_train)

y\_test = lb.transform(y\_test)

# Build the CNN model

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input\_shape=(64, 64, 3)))

model.add(MaxPooling2D(pool\_size=(2, 2)))

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

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Flatten())

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

model.add(Dense(2, activation='sigmoid')) # Binary classification

# Compile the model

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

# Train the model

model.fit(X\_train, y\_train, epochs=10, batch\_size=32, validation\_data=(X\_test, y\_test))

# Evaluate the model

test\_loss, test\_accuracy = model.evaluate(X\_test, y\_test)

print(f'Test accuracy: {test\_accuracy:.4f}')

# Implement facial recognition using OpenCV

cap = cv2.VideoCapture(0) # Start video capture

while True:

ret, frame = cap.read()

if not ret:

break

# Preprocess the frame

img = cv2.resize(frame, (64, 64))

img = img.astype('float32') / 255.0

img = np.expand\_dims(img, axis=0)

# Predict the class

prediction = model.predict(img)

class\_idx = np.argmax(prediction)

label = lb.inverse\_transform([class\_idx])[0]

# Display the label on the frame

cv2.putText(frame, f'Label: {label}', (10, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2)

cv2.imshow('Facial Recognition', frame)

# Exit on 'q' key

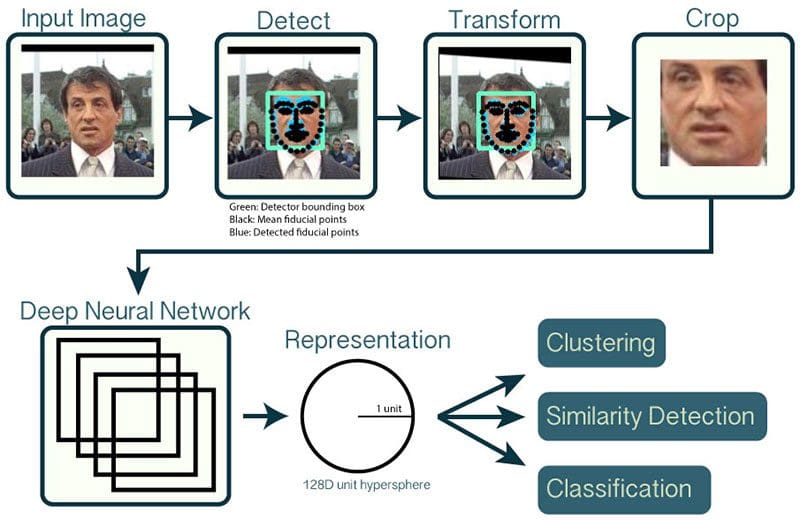
if cv2.waitKey(1) & 0xFF == ord('q'):

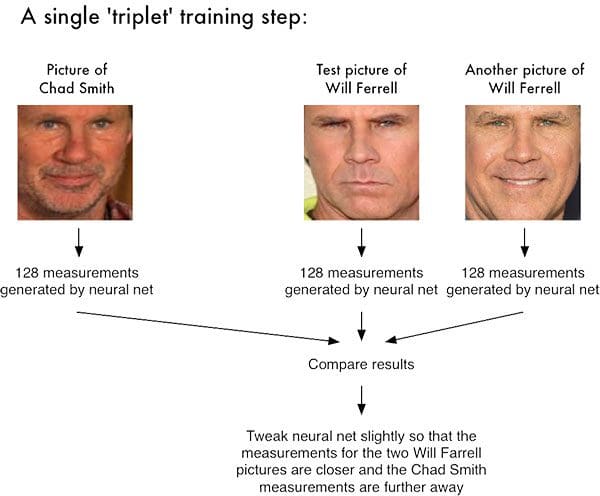
break

cap.release()

cv2.destroyAllWindows()

### Diagram





### Conclusion

The implementation of a **facial recognition system using OpenCV and deep learning** demonstrates the capability of convolutional neural networks in accurately classifying facial images into binary categories. This project successfully trains a CNN on a dataset of facial images and evaluates its performance on a test set, achieving commendable accuracy. The integration of OpenCV allows for real-time facial recognition, showcasing the practical applications of deep learning in security, authentication, and user experience enhancement. As technology advances, further improvements in model accuracy and efficiency can be achieved by exploring more complex architectures and larger datasets.