Egypt-Japan University of Science and Technology
Computer Science and Engineering Department
Project Based Learning

Face Recognition

Author:
OSAMA M. TAHA

Supervised by:
Prof. Walid Gomaa

June 2024

Contents

1 Introduction **1**
1.1 Objective . . . 1
1.2 Background . . . 1

2 Methodology **1**
2.1 Dataset . . . 1
2.2 Preprocessing . . . 1
2.3 Feature Extraction . . . 2
2.4 Model . . . 2

3 Implementation **2**
3.1 Environment Setup . . . 2
3.2 Code Explanation . . . 2
3.2.1 Dataset Loading and Preprocessing . . . 2
3.2.2 Face Detection . . . 2
3.2.3 Model Training . . . 3
3.2.4 Classification . . . 3
3.2.5 Face Verification . . . 4

4 Results **4**
4.1 Evaluation Metrics . . . 4
4.2 Performance . . . 4

5 Conclusion **5**
5.1 Summary . . . 5
5.2 Future Work . . . 5

6 References **5**

1 Introduction

1.1 Objective

The objective of this project is to develop a Python-based solution for comparing two celebrity photos to determine if they depict the same person or different individuals. This involves facial recognition techniques using a deep learning approach to extract and compare facial features.

1.2 Background

Facial recognition technology has gained significant attention due to its applications in security systems, social media, and identity verification. This project leverages deep learning models to perform face verification tasks, contributing to the broader field of computer vision and machine learning.

Face recognition technology works by identifying or verifying a person's identity using their facial features. The process begins with face detection, where an algorithm identifies and locates faces within an image or video. Techniques like the Multi-task Cascaded Convolutional Networks (MTCNN) are commonly used due to their accuracy in detecting faces under various conditions. Once a face is detected, it is aligned and cropped to ensure consistency in orientation and size, which is crucial for the subsequent steps.

The next step is feature extraction, where unique features of the detected face are captured and transformed into a numerical representation known as an embedding. Modern methods employ deep learning models such as FaceNet, which are trained on vast datasets to recognize facial features with high precision. These embeddings are high-dimensional vectors that encapsulate the distinctive characteristics of a person's face, making it possible to distinguish between different individuals even with variations in expressions, lighting, and angles.

Finally, face matching is performed by comparing these embeddings. The similarity between two faces is determined by calculating the distance between their embeddings using metrics like Euclidean distance. If the distance falls below a predefined threshold, the faces are considered a match. This process can be used for both identification, where the system finds the closest match from a database of known faces, and verification, where the system checks if a given face matches a claimed identity. Despite challenges such as varying lighting conditions, occlusions, and aging, face recognition technology has proven to be a powerful tool with applications in security, authentication, and social media.

2 Methodology

2.1 Dataset

The dataset used consists of celebrity photos sourced from an available dataset, containing multiple samples per individual. This variety makes it suitable for training and testing facial recognition models.

2.2 Preprocessing

The preprocessing steps include:

- Loading and converting images to RGB.
- Detecting faces within the images.
- Extracting and resizing the detected faces to a standard size (160x160 pixels).

2.3 Feature Extraction

Facial features are extracted using a pre-trained FaceNet model, which is known for its accuracy in generating embeddings representing the unique characteristics of a face.

2.4 Model

The model architecture used is based on the Siamese Neural Network principle, which is effective for tasks involving comparisons. However, for this implementation, a Support Vector Machine (SVM) classifier is used for the final classification, utilizing the face embeddings generated by the FaceNet model.

3 Implementation

3.1 Environment Setup

Tools and libraries used include:

- Python 3.12
- Google Colab for execution
- Libraries: OpenCV, TensorFlow, Keras, MTCNN, and others

3.2 Code Explanation

3.2.1 Dataset Loading and Preprocessing

The dataset is loaded from Google Drive, and necessary libraries are installed.

```
from google.colab import drive
drive.mount('/content/drive')

!pip install keras-facenet
!pip install tensorflow
!pip install keras
!pip install preprocessing
!pip install image
!pip install mtcnn
```

3.2.2 Face Detection

The MTCNN library is used for face detection, extracting faces from images for further processing.

```
from mtcnn.mtcnn import MTCNN
from keras_facenet import FaceNet
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

detector = MTCNN()
```

```
embedder = FaceNet()

# Load and preprocess an image
image = Image.open('./img_dataset/AL-Limby/El-Lemby004.jpg')
image = image.convert('RGB')
pixels = np.asarray(image)

# Detect faces
results = detector.detect_faces(pixels)
x1, y1, width, height = results[0]['box']
x1, y1 = abs(x1), abs(y1)
x2, y2 = x1 + width, y1 + height
face = pixels[y1:y2, x1:x2]
image = Image.fromarray(face).resize((160, 160))
face_array = np.asarray(image)
```

3.2.3 Model Training

The dataset is split into training and testing sets. The faces are converted into embeddings using the FaceNet model.

```
from sklearn.model_selection import train_test_split

trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.2,
    shuffle=True, random_state=42)
embedder = FaceNet()

def get_embedding(model, face):
    face = face.astype('float32')
    mean, std = face.mean(), face.std()
    face = (face - mean) / std
    sample = np.expand_dims(face, axis=0)
    yhat = model.predict(sample)
    return yhat[0]

emdTrainX = [get_embedding(embedder.model, face) for face in trainX]
emdTestX = [get_embedding(embedder.model, face) for face in testX]

emdTrainX = np.asarray(emdTrainX)
emdTestX = np.asarray(emdTestX)
```

3.2.4 Classification

An SVM classifier is trained on the embeddings to classify the faces.

```
from sklearn.preprocessing import Normalizer, LabelEncoder
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

```

```
in_encoder = Normalizer()
emdTrainX_norm = in_encoder.transform(emdTrainX)
emdTestX_norm = in_encoder.transform(emdTestX)

out_encoder = LabelEncoder()
out_encoder.fit(trainy)
trainy_enc = out_encoder.transform(trainy)
testy_enc = out_encoder.transform(testy)

model = SVC(kernel='linear', probability=True)
model.fit(emdTrainX_norm, trainy_enc)
yhat_train = model.predict(emdTrainX_norm)
yhat_test = model.predict(emdTestX_norm)

score_train = accuracy_score(trainy_enc, yhat_train)
score_test = accuracy_score(testy_enc, yhat_test)
print('Accuracy: train=%.3f, test=%.3f' % (score_train*100, score_test*100))
```

3.2.5 Face Verification

Face verification is performed by comparing embeddings of two faces and computing the distance.

```
ii = "./img_dataset/AL-Limby/El-Lemby007.jpg"
jj = "./img_dataset/AL-Limby/El-Lemby001.jpg"

detect_ii = embedder.extract(ii, threshold=0.95)
detect_jj = embedder.extract(jj, threshold=0.95)
dd = embedder.compute_distance(detect_ii[0]['embedding'],
    detect_jj[0]['embedding'])

if dd < 0.5:
    print('Same Person')
else:
    print('Not Same Person')
print(dd)
```

4 Results

4.1 Evaluation Metrics

The model's performance is evaluated using accuracy on the training and testing datasets. The classifier achieves an accuracy of `train=xx.x%, test=xx.x%`, indicating its ability to correctly classify faces.

4.2 Performance

The face verification task correctly identifies if two images depict the same person based on a threshold distance between their embeddings.

5 Conclusion

5.1 Summary

This project successfully demonstrates the use of deep learning techniques for facial recognition and verification. The combination of MTCNN for face detection and FaceNet for embedding generation provides an effective solution for comparing celebrity photos.

5.2 Future Work

Future improvements could include using a larger and more diverse dataset, exploring different neural network architectures, and implementing real-time face recognition systems. Further optimization and fine-tuning could also enhance the model's accuracy and robustness.

6 References

1. StoicStatic. (n.d.). Face recognition dataset [Data set]. Kaggle. Retrieved June 7, 2024, from https://www.kaggle.com/datasets/stoicstatic/face-recognition-dataset/data

2. Patel, V. (n.d.). Face recognition dataset [Data set]. Kaggle. Retrieved June 7, 2024, from https://www.kaggle.com/datasets/vasukipatel/face-recognition-dataset