Ejercicio: Reconocimiento de Rostros con FaceNet y Python

Objetivo

Desarrollar una aplicación en Python que utilice una red neuronal preentrenada (**FaceNet**) para detectar y reconocer rostros en imágenes.

El alumno aprenderá a:

- Utilizar modelos preentrenados de Deep Learning.
- Extraer embeddings faciales.
- Comparar rostros mediante distancia euclidiana.
- Implementar reconocimiento facial básico.

Introducción Teórica

¿Qué es FaceNet?

FaceNet es una red neuronal profunda desarrollada por Google para reconocimiento facial.

FaceNet no clasifica personas directamente.
En lugar de ello:

1. Recibe una imagen facial.
2. Genera un vector numérico llamado **embedding facial**.
3. Los embeddings de la misma persona son similares.
4. Los embeddings de personas distintas son diferentes.

Flujo General del Sistema

```
Imagen → Detección de rostro → FaceNet → Embedding → Comparación → Resultado
```

Requisitos

Instalación de librerías

```
pip install keras-facenet mtcnn opencv-python matplotlib numpy
```

Estructura del Proyecto

```
reconocimiento_rostros/
│
├── personas/
│   ├── juan/
│   │   ├── foto1.jpg
│   │   ├── foto2.jpg
│   │   └── foto3.jpg
│   │
│   ├── maria/
│   │   ├── foto1.jpg
│   │   ├── foto2.jpg
│   │   └── foto3.jpg
│
├── prueba.jpg
└── reconocimiento.py
```

Explicación del Dataset

Cada carpeta representa una persona distinta.

Ejemplo:

```
personas/
   juan/
   maria/
```

Cada carpeta contiene varias fotografías del rostro.

Puedes usar fotografías propias o datasets públicos como:

- Labeled Faces in the Wild (LFW)
- Kaggle Face Dataset

Código Completo

Archivo: `reconocimiento.py`

```
import os
import cv2
import numpy as np
from mtcnn import MTCNN
from keras_facenet import FaceNet

# Inicializar detector y modelo
detector = MTCNN()
embedder = FaceNet()

# Diccionario para almacenar embeddings
base_datos = {}

# ----------------------------------------
# FUNCIÓN PARA EXTRAER ROSTRO
# ----------------------------------------
def extraer_rostro(ruta_imagen):

    imagen = cv2.imread(ruta_imagen)

    if imagen is None:
        return None

    imagen_rgb = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)

    resultados = detector.detect_faces(imagen_rgb)

    if len(resultados) == 0:
        return None

    x, y, ancho, alto = resultados[0]['box']

    x, y = abs(x), abs(y)

    rostro = imagen_rgb[y:y+alto, x:x+ancho]

    rostro = cv2.resize(rostro, (160, 160))

    return rostro


# ----------------------------------------
# GENERAR EMBEDDING
# ----------------------------------------
def generar_embedding(rostro):

    rostro = rostro.astype('float32')

    rostro = np.expand_dims(rostro, axis=0)

    embedding = embedder.embeddings(rostro)

    return embedding[0]


# ----------------------------------------
# CARGAR BASE DE DATOS
```

```
# ----------------------------------------
ruta_personas = "personas"

for persona in os.listdir(ruta_personas):

    carpeta_persona = os.path.join(ruta_personas, persona)

    embeddings_persona = []

    for archivo in os.listdir(carpeta_persona):

        ruta = os.path.join(carpeta_persona, archivo)

        rostro = extraer_rostro(ruta)

        if rostro is not None:

            embedding = generar_embedding(rostro)

            embeddings_persona.append(embedding)

    if len(embeddings_persona) > 0:

        promedio = np.mean(embeddings_persona, axis=0)

        base_datos[persona] = promedio

print("Base de datos cargada correctamente")


# ----------------------------------------
# RECONOCIMIENTO
# ----------------------------------------
imagen_prueba = "prueba.jpg"

imagen = cv2.imread(imagen_prueba)

imagen_rgb = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)

resultados = detector.detect_faces(imagen_rgb)

for resultado in resultados:

    x, y, ancho, alto = resultado['box']

    x, y = abs(x), abs(y)

    rostro = imagen_rgb[y:y+alto, x:x+ancho]

    rostro_red = cv2.resize(rostro, (160, 160))

    embedding_prueba = generar_embedding(rostro_red)

    nombre_detectado = "Desconocido"

    distancia_minima = 999
```

```
    for nombre, embedding_bd in base_datos.items():

        distancia = np.linalg.norm(embedding_prueba - embedding_bd)

        if distancia < distancia_minima:

            distancia_minima = distancia
            nombre_detectado = nombre

    # Umbral de reconocimiento
    if distancia_minima > 1.0:
        nombre_detectado = "Desconocido"

    # Dibujar rectángulo
    cv2.rectangle(imagen, (x, y), (x+ancho, y+alto), (0,255,0), 2)

    texto = f"{nombre_detectado} ({distancia_minima:.2f})"

    cv2.putText(
        imagen,
        texto,
        (x, y-10),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.8,
        (0,255,0),
        2
    )

# Mostrar imagen
cv2.imshow("Reconocimiento Facial", imagen)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

Explicación del Código

1. Detección Facial

Se utiliza:

```
MTCNN()
```

para localizar rostros en la imagen.

---

2. Extracción del Embedding

FaceNet genera un vector de 512 características:

```
embedding = embedder.embeddings(rostro)
```

---

3. Comparación de Rostros

Se usa distancia euclidiana:

$$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Si la distancia es pequeña, los rostros pertenecen a la misma persona.

Resultado Esperado

El sistema debe:

- Detectar rostros.
- Dibujar un rectángulo.
- Mostrar el nombre identificado.

Ejemplo:

```
Juan (0.45)
María (0.62)
Desconocido (1.35)
```

Actividades para el Alumno

Parte 1

Ejecutar el sistema con 2 personas distintas.

Parte 2

Agregar mínimo 5 imágenes por persona.

Parte 3

Modificar el umbral:

```
if distancia_minima > 1.0:
```

Probar:

- 0.7
- 0.9
- 1.2

Analizar resultados.

Parte 4

Agregar reconocimiento en tiempo real usando webcam.

Ejemplo Webcam en Tiempo Real

```
cap = cv2.VideoCapture(0)

while True:

    ret, frame = cap.read()

    if not ret:
        break

    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    resultados = detector.detect_faces(rgb)

    for resultado in resultados:

        x, y, w, h = resultado['box']

        x, y = abs(x), abs(y)

        rostro = rgb[y:y+h, x:x+w]

        rostro = cv2.resize(rostro, (160,160))

        embedding = generar_embedding(rostro)
```

```
            nombre = "Desconocido"
            dist_min = 999

            for persona, emb_bd in base_datos.items():

                dist = np.linalg.norm(embedding - emb_bd)

                if dist < dist_min:
                    dist_min = dist
                    nombre = persona

            if dist_min > 1.0:
                nombre = "Desconocido"

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

            cv2.putText(
                frame,
                nombre,
                (x,y-10),
                cv2.FONT_HERSHEY_SIMPLEX,
                0.8,
                (0,255,0),
                2
            )

        cv2.imshow("Webcam", frame)

        if cv2.waitKey(1) == 27:
            break

cap.release()
cv2.destroyAllWindows()
```

Preguntas de Análisis

1. ¿Qué es un embedding facial?
2. ¿Por qué FaceNet no clasifica directamente?
3. ¿Qué sucede si el umbral es muy pequeño?
4. ¿Qué sucede si el umbral es muy grande?
5. ¿Qué ventajas tiene usar modelos preentrenados?
6. ¿Qué problemas éticos existen en el reconocimiento facial?

Mejoras Posibles

- Guardar embeddings en una base de datos.
- Reconocimiento en video.

- Entrenamiento personalizado.
- Uso de GPU.
- Implementar interfaz gráfica.
- Integrar con sistemas de asistencia.

Conclusión

En esta práctica se implementó un sistema de reconocimiento facial utilizando:

- Python
- OpenCV
- MTCNN
- FaceNet

Se aplicaron conceptos de:

- Deep Learning
- Visión por computadora
- Redes neuronales convolucionales
- Embeddings faciales

El sistema permite identificar personas mediante comparación de características faciales generadas por una red neuronal preentrenada.