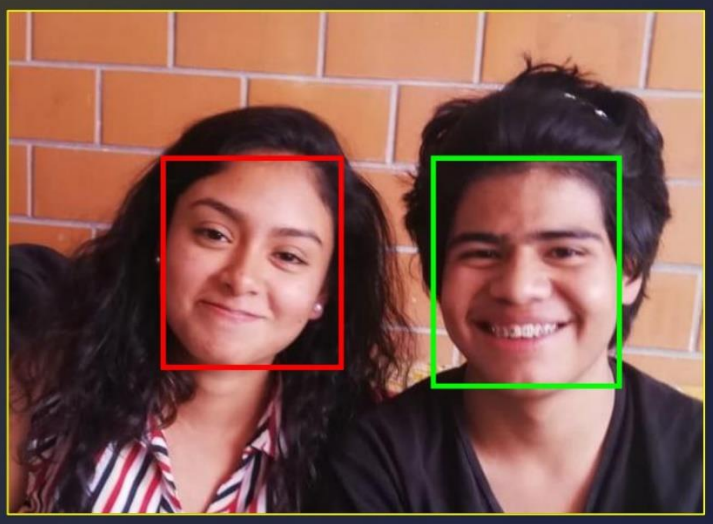


Reconocimiento facial con Python

Octubre de 2021

Christian Amauri Amador Ortega

¿Qué hace?

Este script toma imágenes de rostros guardados en el directorio 'knowns' para calcular sus embeddings y, de esta manera, reconocerlos. Luego, compara los embeddings de los rostros desconocidos almacenados en el directorio 'unknowns' con los embeddings de los rostros conocidos para determinar si hay una coincidencia. Finalmente, dibuja un rectángulo verde sobre los rostros que coinciden y guarda copias de esas nuevas imágenes en un tercer directorio llamado 'results'.

¿Cómo funciona?

Este script utiliza las siguientes librerías con los siguientes propósitos:

- **TensorFlow**: Crear y entrenar modelos de aprendizaje automático y redes neuronales profundas.
- **OpenCV**: Procesar y analizar imágenes y videos en tiempo real.
- **NumPy**: Realizar operaciones matemáticas y manipulación de arreglos multidimensionales.
- **os**: Proporciona funciones para interactuar con el sistema operativo, como manejo de archivos y directorios.
- **keras_facenet**: Librería que implementa el modelo de reconocimiento facial Facenet utilizando Keras para la extracción de características faciales.

MobileNet es un modelo de red neuronal convolucional (CNN), diseñado para dispositivos con recursos limitados (como teléfonos móviles). Se emplea en tareas de clasificación de imágenes, pero también se puede usar para detección de objetos, por ejemplo rostros.

Se utiliza **TensorFlow** para cargar **MobileNet** (mobilenet_graph.pb en el directorio principal). y crear un gráfico computacional con él *(tf.import_graph_def(graph_def,name=''))*. Este gráfico contiene la red neuronal de MobileNet, que ya ha aprendido a identificar características relevantes en las imágenes para detectar objetos, en este caso, rostros.

Luego, el modelo se usa dentro de una sesión de TensorFlow para hacer predicciones sobre nuevas imágenes (“unknowns”). El script utiliza las salidas de la red (coordenadas de los bounding boxes y las scores de confianza) para detectar las ubicaciones de los rostros en la imagen. Estas predicciones ayudan a identificar qué partes de la imagen corresponden a rostros, permitiendo luego extraer y procesar esas áreas específicas.

FaceNet transforma un rostro (una imagen de una cara) en un vector numérico de alto dimensionalidad (un embedding facial). Cada embedding es una representación compacta y

única de un rostro, que captura sus características distintivas. (en versiones anteriores, se usaba el archivo facenet_keras.h5, disponible en el directorio fuente).

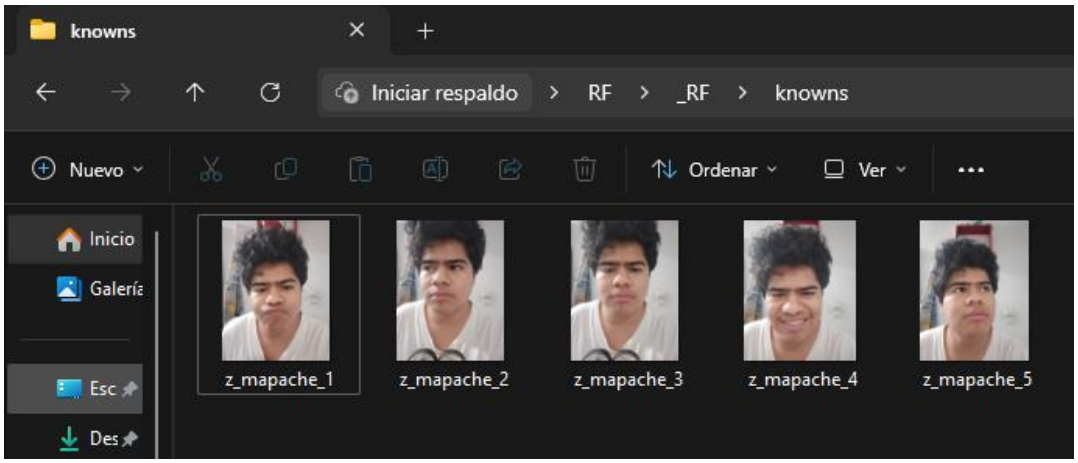
Una vez que el script detecta los rostros en las imágenes (usando MobileNet y TensorFlow), los recorta y redimensiona a un tamaño adecuado. Luego, se pasa cada rostro recortado al modelo FaceNet (**keras_facenet**). Este modelo genera un embedding para cada rostro.

OpenCV se utiliza para leer y convertir las imágenes de los rostros en formatos compatibles, y para dibujar los bounding boxes que rodean a los rostros detectados. Primero, se utiliza para leer las imágenes de los rostros tanto de los directorios 'knowns' como 'unknowns' a través de la función *cv2.imread()*. Después, OpenCV convierte las imágenes de su formato original (BGR) al formato RGB usando *cv2.cvtColor()*, (necesario para el procesamiento en TensorFlow). Luego, se encarga de dibujar los bounding boxes alrededor de los rostros detectados utilizando la función *cv2.rectangle()*. Finalmente OpenCV se usa para guardar las imágenes con los rostros reconocidos en el directorio 'results' mediante *cv2.imwrite()*.

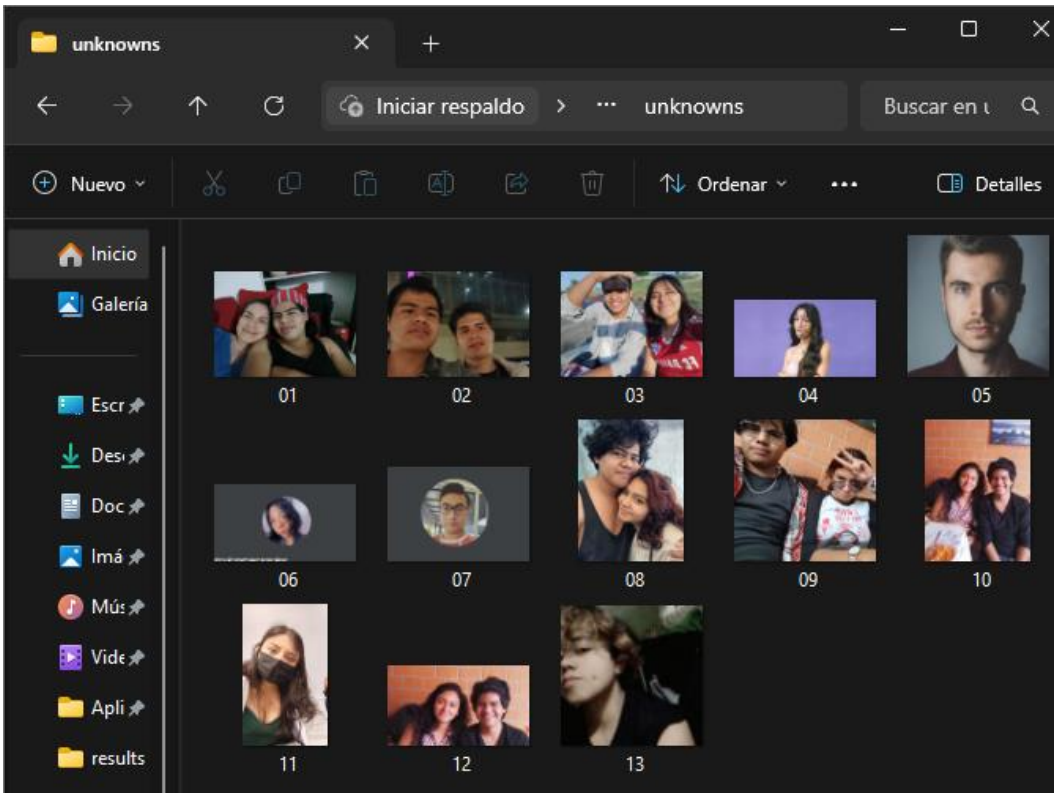
NumPy, por su parte, realiza operaciones matemáticas necesarias para manejar y transformar los datos de las imágenes, como redimensionar, normalizar o cambiar el formato de las matrices de píxeles. Cuando se extraen los rostros, NumPy se usa para ajustar las dimensiones de la imagen recortada antes de pasarla a FaceNet. También es usado para realizar la normalización de los píxeles de los rostros (restando la media y dividiendo por la desviación estándar) antes de enviarlos al modelo FaceNet. Y finalmente, NumPy facilita la comparación de rostros y la detección de coincidencias. al utilizar *np.linalg.norm()* para calcular la distancia euclidiana entre los embeddings.

(las versiones específicas de las librerías usadas, están indicadas en la hora requirements.txt en el directorio fuente).

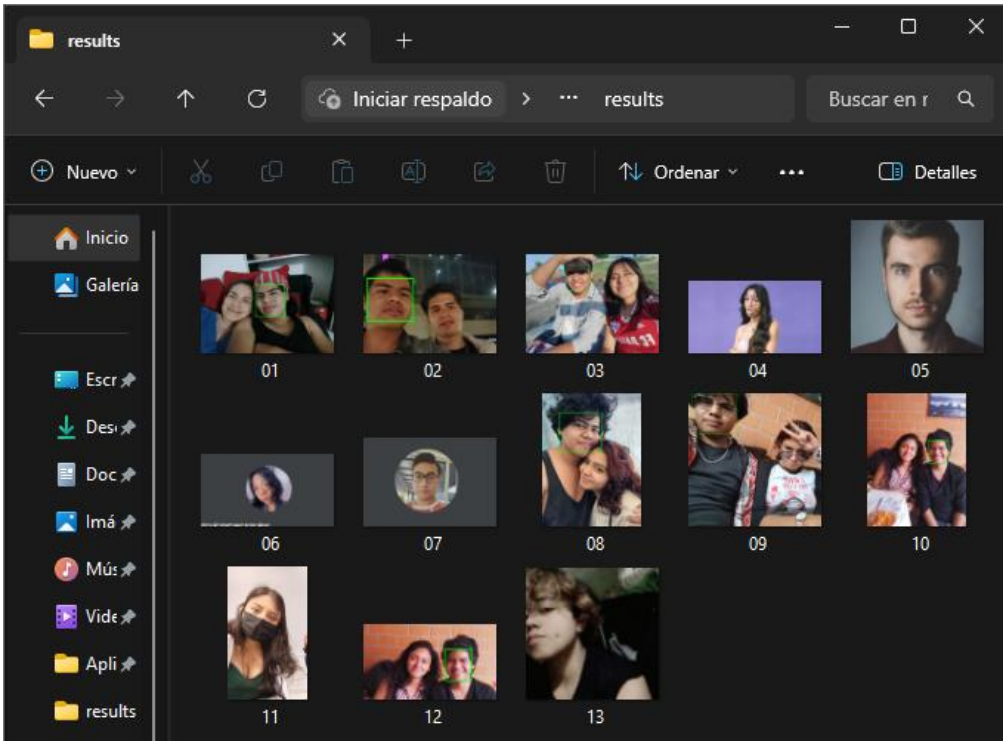
Resultados



Directorio "knowns"



Directorio "unknowns"



Directorio "results"

Rostro reconocido y rostros no reconocidos.