RETO DE 'DEEP LEARNING'. RECONOCIMIENTO FACIAL.

El reto consiste en utilizar Python para reconocer, en una fotografía, a las personas de este curso.

Hay que ir realizando las siguientes tareas:

1. Hacer una fotografía del grupo y guardarla en la carpeta temporal para que pueda ser utilizada.
2. Abrir la imagen (PIL/Image.open) y mostrarla en pantalla (pyplot.imshow).
3. Para el reconocimiento facial hay que detectar dónde se encuentran las caras dentro de la imagen. Usar el detector MTCNN de 'facenet_pytorch'.
4. Utilizar el método detect() del modelo MTCNN, que nos devuelve:
    a. Las coordenadas de la esquina superior-izquierda y la esquina inferior-derecha de cada bounding box (recuadros donde están las caras).
    b. Probabilidad de que cada bounding box contenga una cara.
    c. Coordenadas de los ojos, nariz y comisura de la boca en cada cara detectada.
5. Mostrar en pantalla la imagen del grupo con los recuadros de las caras.
6. Extracción de caras: utilizar el método forward(). Si hay más de una cara, al instanciar el modelo MTCNN hay que poner el parámetro 'keep_all' a 'True'. Se devuelve un tensor (array numérico) con el valor de los pixeles del recorte/s [nº caras, canales de color, altura, anchura].
7. Representar en pantalla las distintas caras extraídas (con matplotlib).
8. Una vez identificadas las caras, hay que realizar una transformación numérica que sea capaz de representar los aspectos característicos de cada una. Al vector numérico resultante se le conoce como *embedding* o *encoding*. Para ello, usar el modelo InceptionResnetV1 de facenet_pytorch (con el parámetro pretrained='vggface2'). Obtener el 'encoding' de todas las caras de la imagen.

El objetivo de obtener una representación numérica de las caras (embeddings) es poder cuantificar similitudes entre ellas. Una forma de calcular esta similitud es utilizando la distancia euclídea entre embeddings. Cuanto menor es la distancia, mayor la similitud de las caras.

9. Calcular las Distancias entre embeddings de tres caras de tres personas de clase. Para ello usar el método 'euclidean' del paquete 'scipy.spatial.distance'.

Para poder identificar una cara en una imagen, hay que tener una base de datos que contenga un embedding de referencia de la cara que queremos identificar.

En los apartados anteriores, se ha descrito de forma separada cada uno de los pasos que forman parte del proceso de reconocimiento facial. Sin embargo, para poder aplicarlo en un sistema real, es necesario combinarlos todos para que se ejecuten de forma secuencial. Con este objetivo:

10. Crear una función 'detectar_caras' para Detectar la posición de caras en una imagen empleando un detector MTCNN. La función debe devolver un Numpy array con las bounding box de cada cara detectada (Las bounding box devueltas por el detector ``MTCNN`` están definidas por valores de tipo `float`. Esto supone un problema para la posterior representación con matplotlib, por lo que se convierten a tipo `int`.).
11. Crear una función 'extraer_caras' para extraer las zonas de una imagen contenidas en bounding boxes.
12. Crear una función 'calcular_embeddings' para calcular los encodings de caras utilizando el modelo InceptionResnetV1 de la librería facenet_pytorch.
13. Crear la función 'identificar_caras' para: dado un conjunto de embeddings de una foto y un diccionario de referencia, se calcula la similitud entre cada nuevo embedding y los embeddings de referencias. Si la similitud supera un determinado threshold se devuelve la identidad (nombre) de la persona (que está en el diccionario de referencia).
14. Crear una función 'mostrar_boxes' para mostrar la imagen original con las boundig box de las caras detectadas empleando matplotlib. Si se le pasan las identidades (nombres) de las personas, se muestran sobre cada bounding box. Esta función no devuelve nada.
15. Crear una función 'deteccion_imagen' que integre las funciones previamente definidas, para: detectar_caras, extraer_caras, calcular_embeddings, identificar_caras y mostrar_boxes.
16. Crear la función 'diccionario_referencias' para crear o actualizar un diccionario con embeddings de referencia de personas. La función devolverá un Diccionario con los embeddings de referencia. La clave representará el nombre de la persona y el valor el embedding de su cara.
17. Crear un diccionario de referencia para las personas de clase con la función creada previamente para ello.
18. Utilizar la función 'deteccion_imagen' para realizar el Reconocimiento en imágenes. Uno de los parámetros de entrada será el diccionario de referencia para que la función pueda identificar las personas y poner nombre a cada cara.