Proyecto Biométrica: Reconomiento Facial

Task 1.1: Read the DiveFace database and obtain the embeddings of at least 500 face images (1 image per subject) from the 6 demographic groups (500*6=3000 embeddings in total). DiveFace contains face images from 3 demographic groups (3 ethnicity and 2 gender)

Para poder obtener 500 imágenes de personas distintas por cada uno de los 6 grupos demográficos, lo primero que tenemos que hacer es descargar el archivo comprimido proporcionado, el cual, es la base de de datos llamada "DiveFace", y descomprimirlo mediante el comando: "unrar x DiveFace4K-120.rar".

Tras ello, se ha analizado la estructura de carpetas de la base de datos y se ha creado un script en Python para la extracción automática del objetivo anteriormente anunciado:

```
DATA_PATH = "imagenes_task1-2/"
ORIGINAL_RAR = "4K_120/"
AMOUNT_OF_IMAGES = 500

demographic_groups = ut.create_directories_per_demographic_group(
    data_path=DATA_PATH + ORIGINAL_RAR,
    output_path=DATA_PATH,
    create=True,
)
subdirectories_dict = ut.get_given_amount_of_subdirectories(
    data_path=DATA_PATH + ORIGINAL_RAR,
    demographic_groups=demographic_groups,
    amount=AMOUNT_OF_IMAGES
)
ut.copy_images_per_demographic_group(
    data_path=DATA_PATH + ORIGINAL_RAR,
    output_path=DATA_PATH,
    demographic_groups=demographic_groups,
    subdirectories_dict=subdirectories_dict,
    copy=True
)
```

Figure 1: Script extracción de las imágenes deseadas

Así, los pasos lógicos seguidos han sido:

1. Crear una subcarpeta para cada grupo demográfico:
    - Hombre blanco (HB4K-120)
    - Hombre negro (HN4K-120)
    - Hombre chino (HA4K-120)
    - Mujer blanca (MB4K-120)
    - Mujer negra (MN4K-120)
    - Mujer china (MA4K-120)
2. Obtener los subdirectorios requeridos (500 por grupo) para obtener imágenes diferentes de la base de datos original.
3. Copiar una imagen por subdirectorio del segundo paso a la carpetas creadas en el primer paso.

Para más detalle del funcionamiento del código, mirar las funciones creadas en el archivo "Utils.py" conjunto a las constantes definidas en el archivo "Enum.py".

Task 1.2: Using t-SNE, represent the embeddings and its demographic group. Can you differentiate the different demographic groups? Explain it.

En primer lugar, definiendo qué es t-SNE (t-Distributed Stochastic Neighbor Embedding) se tiene que es una popular técnica de reducción dimensional utilizada para visualizar datos de alta dimensión en un espacio de menor dimensión, la cual, fue introducida por Laurens van der Maaten y Geoffrey Hinton en 2008.

El objetivo principal de t-SNE es captar la estructura y las relaciones subyacentes en los datos representándolos en un formato visualmente más comprensible. Esto resulta especialmente útil para explorar y comprender conjuntos de datos complejos en los que métodos tradicionales como el PCA (Análisis de Componentes Principales) pueden no ser tan eficaces.

Más en detalle, t-SNE construye para cada punto una distribución de probabilidad que mide la similitud entre ese punto y todos los demás del conjunto de datos. Posteriomente, construye un espacio de menor dimensión, normalmente en 2D o 3D, y optimiza dicha representación iterativamente mediante descenso por gradiente para minimizar la divergencia de Kullback-Leibler entre las similitudes por pares tanto en el espacio de altas dimensiones como en el espacio de dimensiones inferiores.

De tal manera, dicha técnica es útil para la tarea que se quiere realizar. El código utilizado es el siguiente:

```
list_of_emmbedings = list()

for root, _, files in os.walk("imagenes_task1-2", topdown=True):
    for name in files:
        embedding = ut.get_embedding_from_image(
            path_to_image=os.path.join(root, name)
        )
        list_of_emmbedings.append(embedding)

tsne = TSNE(n_components=2)
tsne_result = tsne.fit_transform(np.array(list_of_emmbedings))

ut.plot_tsne_graphic(tsne_result)
```

Figure 2: Script para la representación t-SNE de las imágenes

Arrojando los siguientes resultados:

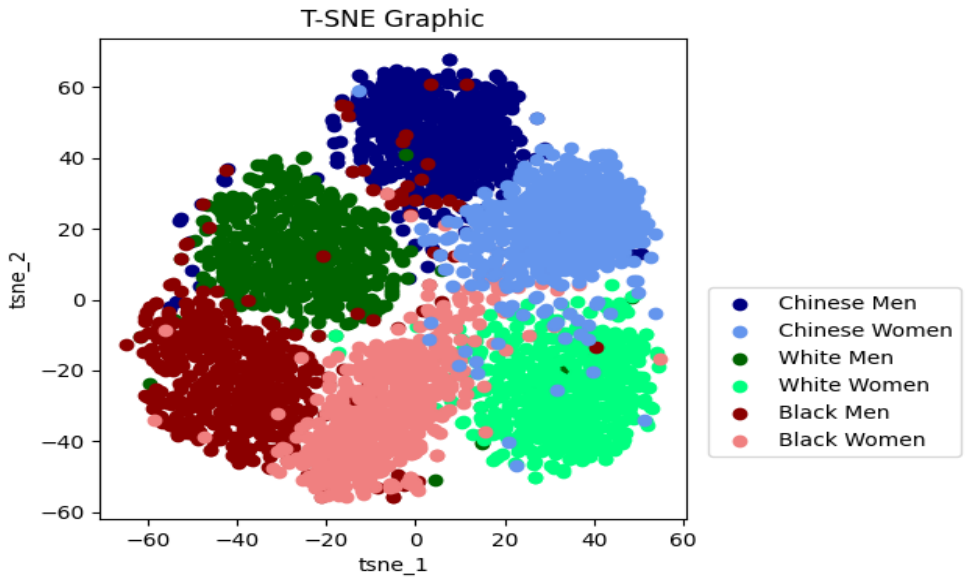

Figure 3: Resultado t-SNE del conjunto de datos

Como se puede observar t-SNE ha conseguido obtener su objetivo, es decir, pasar de un conjunto de embeddings de imágenes de caras de alta dimensionalidad a una representación en la que a simple vista:

1. Se diferencian las etnias (blanca, negra y china).
2. Se ven las diferencias entre hombres y mujeres de la misma etnia.
3. Se ven las diferencias entre hombres y mujeres independientemente de la etnia.

Task 1.3: Using the ResNet-50 embedding (freeze the model), train your own attribute classifiers (ethnicity and gender). Recommendation: use a simple dense layer with a softmax output. Divide DiveFace into train and test.

En este apartado pasaremos a construir la arquitectura de nuestro modelo, el cual, es un clasificador binario (hombre / mujer) que será entrenado tanto para la Task 2.1 como para la Task 2.3. :

```
def baseline_model():
    model = Sequential()
    model.add(Dense(1, activation='sigmoid'))
    model.compile(
        loss='binary_crossentropy',
        optimizer='adam',
        metrics=['accuracy']
    )
    return model
```

Figure 4: Modelo base para clasificación binaria

El flujo de trabajo será explicado de forma específica en las mencionadas tareas.

Task 2.1: Train 3 different Gender Classifiers (previous Task 1.4) using images from same ethnic group: Model A (only Asian), Model B (only Black), Model C (only Caucasian)

Así, el flujo de trabajo es el siguiente:

1. Elección de la cantidad adecuada de datos de "DiveFace", en este caso serán 750 por etnia/sexo. Mismo procedimiento que en la Figura 1 pero cambiando la constante "AMOUNT OF IMAGES".
2. Extracción de caraterísticas de dichas imágenes faciales mediante el modelo "Resnet50".

3. Hacer el split de los embeddigns por etnia/sexo de forma que hayan 500 para training y 250 para test. Asignando también a cada embedding la etiqueta correspodiente "Hombre" o "Mujer".

4. Entrenar un clasificador binario de sexo para cada etnia (3), juntando los 500 embeddings de cada sexo, es decir, cada modelo será entrenado con un total de 1000 datos.

Task 2.2: Evaluate the 3 Gender Classifiers (previous Task 2.1) using images from each of the three ethnic groups. Explain the performance obtained for each model and each group.

El accuracy de cada uno de los modelos es testeado tanto con los datos de su propia etnia como con respecto a las otras dos. De tal manera, el resultado es una matriz 3x3:

	Chinese Data	Black Data	White Data
Chinese Model	0.992	0.962	0.861
Black Model	0.956	0.984	0.883
White Model	0.814	0.875	0.994

Figure 5: Resultados de los modelos por etnia

.

Como era de esperar, los resultados de la diagonal principal de la matriz son los mejores con respecto a su correspondiente fila. En otras palabras, el modelo entrenado con datos de una etnia reconoce mejor los imágenes faciales de dicha etnia.

Sin embargo, destaca que el modelo de la etnia blanca generaliza significativamente peor que el resto, cuando a priori se podría formular la siguiente hipótesis a la vista de la Figura 3:

"Como hay menor distancia entre la etnia blanca y la etnia china que la que hay entre la etnia negra y la etnia china el modelo entrenado con datos de etnia negra es el que generalizara peor."

Task 2.3: Train one Gender Classifiers (previous Task 1.4) using images from all three ethnic groups.

En este caso, el flujo de trabajo es el siguiente:

1. Aprovechando el trabajo previo de extracción de características, se reorganizan los datos de forma que tenemos un total de 3000 embeddings de todas las etnias juntas.

2. Se entrena un único clasificador binario con dichos datos.

Task 2.4: Evaluate the Gender Classifier (previous Task 2.3) using images from each of the three ethnic groups. Explain the performance obtained for each group.

El accuracy del modelo es testeado con los datos de las distintas etnias (500 por cada una). De tal manera, el resultado es una matriz 1x3:

	Chinese Data	Black Data	White Data
Model	0.992	0.98	0.99

Figure 6: Resultado del modelo general

.

Como era de esperar, dado que el modelo general ha sido entrenado con una cantidad homogéna de datos de las distintas etnias, los resultados son muy parecidos entre las tres. Esto demuestra que se pueden generar modelos de inteligencia artifical sin sesgo étnico, haciendo el esfuerzo de conseguir datos lo suficientemente heterogéneos y balanceados.

Por último, destacar que el código de las tareas 1 se encuentra disponible en "Tasks1.py" y el de las tareas 2 se encuentra en "Tasks2.py". Además, recordar de nuevo la existencia del archivo "Utils.py" para las funciones auxiliares y del archivo "Enum.py" para la definición de constantes.