

Práctica 1

Face Swapping con Python



Profesor:
David Campoy Miñarro



¿Qué es el Face Swapping?

El *Face Swapping* es una técnica de edición de imágenes y videos que se utiliza para reemplazar una cara en una imagen o clip de video por otra.

Face Swapping plantea preocupaciones éticas y de privacidad, ya que puede utilizarse para crear imágenes y videos falsos que parecen auténticos, lo que puede llevar a la difusión de información engañosa o dañina.



Es fundamental utilizar esta tecnología con responsabilidad y tener en cuenta sus implicaciones éticas y legales.



Insightface

InsightFace es una biblioteca de Python desarrollada para tareas de reconocimiento facial.

<https://github.com/deepinsight/insightface>

InsightFace permite no solo el reconocimiento facial, sino también la extracción de características faciales, lo que puede ser útil en aplicaciones de análisis y clasificación de rostros.



InsightFace es compatible con CPUs y GPUs, lo que permite acelerar el proceso de reconocimiento facial si se dispone de hardware compatible.

Preparando el entorno en Python

Necesitarás importar:


- `!pip install insightface`
- `!pip install onnxruntime`

¿Qué es onnxruntime?

ONNX Runtime (*ONNX Runtime*, a veces abreviado como *ORT*) es una biblioteca de código abierto desarrollada por Microsoft que se utiliza para ejecutar modelos de aprendizaje automático en el formato *Open Neural Network Exchange* (ONNX)

Modelo entrenado:

- `inswapper_128.onnx` (528 Megas)



<https://github.com/facefusion/facefusion-assets/releases>

Posibles errores en Windows

Si al instalar la librería Insightface te aparece un error en Windows, posiblemente necesites actualizar la versión de Microsoft C++ Build Tools.

<https://visualstudio.microsoft.com/es/visual-cpp-build-tools/>



Paso 1. Detectar caras

Puedes utilizar el entorno Jupyter Notebook o Google Colab.

```
#!/pip install onnxruntime
#!/pip install insightface
import numpy as np
import os
import glob
import cv2
import matplotlib.pyplot as plt

import insightface
from insightface.app import FaceAnalysis
from insightface.data import get_image as ins_get_image
```

- `import numpy as np`: Se utiliza para trabajar con matrices.
- `import os`: Importa la biblioteca "os", que proporciona funciones para interactuar con el sistema operativo, como trabajar con archivos y directorios.
- `import glob`: Se utiliza para buscar archivos y directorios.
- `import cv2`: OpenCV (Open Source Computer Vision Library), se utiliza para procesar imágenes y videos.
- `import matplotlib.pyplot as plt`: Para crear gráficos y visualizaciones, en este caso, se importa el módulo para crear gráficos en una ventana o en un archivo de imagen.

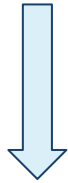
- `import insightface`: Tareas de reconocimiento facial y análisis de imágenes.
- `from insightface.app import FaceAnalysis`: Esta clase es utilizada para realizar análisis facial, incluyendo la detección de rostros y la extracción de características faciales.
- `from insightface.data import get_image as ins_get_image`: Cargar imágenes.



Mostramos la versión de *insightface*.

```
# *****
```

```
print("insightface,", insightface.__version__)  
print("numpy", np.__version__)
```



```
insightface, 0.7.3  
numpy 1.24.3
```

Seleccionamos el modelo entrenado

```
app = FaceAnalysis(name='buffalo_l')
app.prepare(ctx_id=0, det_size=(640,640))

img = ins_get_image("t1")
plt.imshow(img[:, :, ::-1])
plt.show()
```

Esta arquitectura es especialmente eficaz para detectar rostros en una variedad de condiciones, incluyendo diferentes tamaños, ángulos y poses.

Capaz de procesar 10 Gigaflops. Un gigaflop representa mil millones (10^9) de operaciones de punto flotante por segundo.

Name	Detection Model	Recognition Model	Alignment	Attributes	Model-Size
antelopev2	RetinaFace-10GF	ResNet100@Glint360K	2d106 & 3d68	Gender&Age	407MB
buffalo_l	RetinaFace-10GF	ResNet50@WebFace600K	2d106 & 3d68	Gender&Age	326MB
buffalo_m	RetinaFace-2.5GF	ResNet50@WebFace600K	2d106 & 3d68	Gender&Age	313MB
buffalo_s	RetinaFace-500MF	MBF@WebFace600K	2d106 & 3d68	Gender&Age	159MB
buffalo_sc	RetinaFace-500MF	MBF@WebFace600K	-	-	16MB

Modelos entrenados:

https://github.com/deepinsight/insightface/tree/master/model_zoo





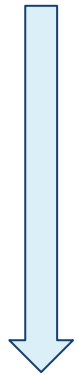
Vamos a buscar las 6 caras de esta imagen.

Detecta las caras

```
faces=app.get(img)
```

```
len(faces)
```

```
faces[0].keys()
```



Nº de caras:

6

```
dict_keys(['bbox', 'kps', 'det_score', 'landmark_3d_68', 'pose', 'landmark_2d_106',  
'gender', 'age', 'embedding'])
```

- `faces = app.get(img)`: La función `app.get(img)` se encarga de realizar el análisis facial, lo que puede incluir la detección de rostros y la extracción de características faciales.
- `len(faces)`: La longitud de esta lista indica cuántas caras se han detectado en la imagen.
- `faces[0].keys()`: Las claves representan las diferentes características o propiedades de la cara detectada, como la ubicación, las coordenadas, la confianza en la detección, entre otras. Esto permite inspeccionar las propiedades de la primera cara detectada.

Mostramos las caras

```
img = ins_get_image("t1")
fig, axs = plt.subplots(1,6, figsize=(12, 5))

for i, face in enumerate(faces):
    bbox = face["bbox"]
    bbox = [int(b) for b in bbox]
    axs[i].imshow(img[bbox[1]:bbox[3],bbox[0]:bbox[2],::-1])
    axs[i].axis("off")
```

dict_keys(['bbox', 'kps', 'det_score', 'landmark_3d_68', 'pose', 'landmark_2d_106', 'gender', 'age', 'embedding'])

Applied providers: ['CPUExecutionProvider'], with options: {'CPUExecutionProvider': {}}

inswapper-shape: [1, 3, 128, 128]



Ahora vamos a intercambiar las caras.

Cargamos el modelo que realizará el intercambio de caras:

```
swapper = insightface.model_zoo.get_model("./inswapper_128.onnx", download=False, download_zip=False)
```

Recuerda, tendrás que tener el modelo en tu ordenador o en Google Colab.



The screenshot shows the Google Colab interface. On the left, the 'Archivos' (Files) pane is open, displaying a directory structure with a folder named 'sample_data' and a file named 'inswapper_128.onnx'. The file 'inswapper_128.onnx' is highlighted with a red rectangular box. On the right, the main workspace area shows a video player with a red play button and a duration of 7 seconds. The video frame displays a poker table with various cards and chips. Below the video player, there is a code editor with the following text:

```
+ Código + Texto Se han guardado todos los cambios
```

7 s

700
800

0 200 400 600 800 1000 1200

/usr/local/lib/python3.10/dist-packages/insightface/utils/transform.py:6
To use the future default and silence this warning we advise to pass `rc`
P = np.linalg.lstsq(X_homo, Y)[0].T # Affine matrix. 3 x 4

Cogemos una cara cualquiera, por ejemplo la 5.

El siguiente código recorta la cara de la imagen según las coordenadas almacenadas en la clave "bbox".

```
source_face = faces[5]
bbox = source_face["bbox"]
bbox = [int(b) for b in bbox]
plt.imshow(img[bbox[1]:bbox[3],bbox[0]:bbox[2],::-1])
plt.show()
```



```
# *****
```

```
res= img.copy()
```

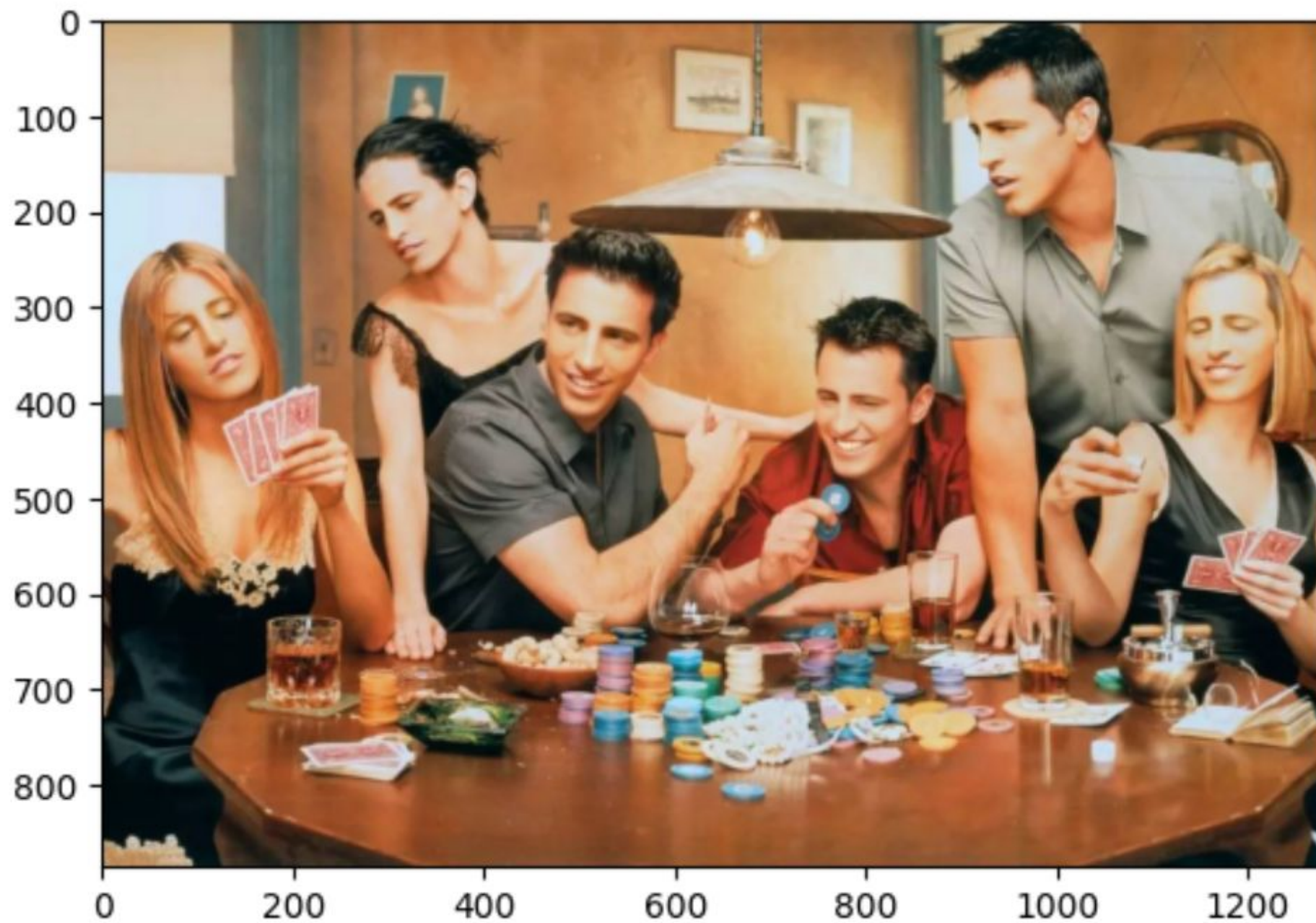
```
for face in faces:
```

```
    res = swapper.get(res, face, source_face, paste_back=True)
```

```
# *****
```

```
plt.imshow(res[:,::-1])
```

```
plt.show()
```



Obtenemos las caras intercambiadas

```
res = []  
for face in faces:  
    _img, __ = swapper.get(img, face, source_face, paste_back=False)  
    res.append(_img)  
res = np.concatenate(res, axis=1)  
fig, ax = plt.subplots(figsize=(15,5))  
ax.imshow(res[:, :, ::-1])  
ax.axis('off')  
plt.show()
```



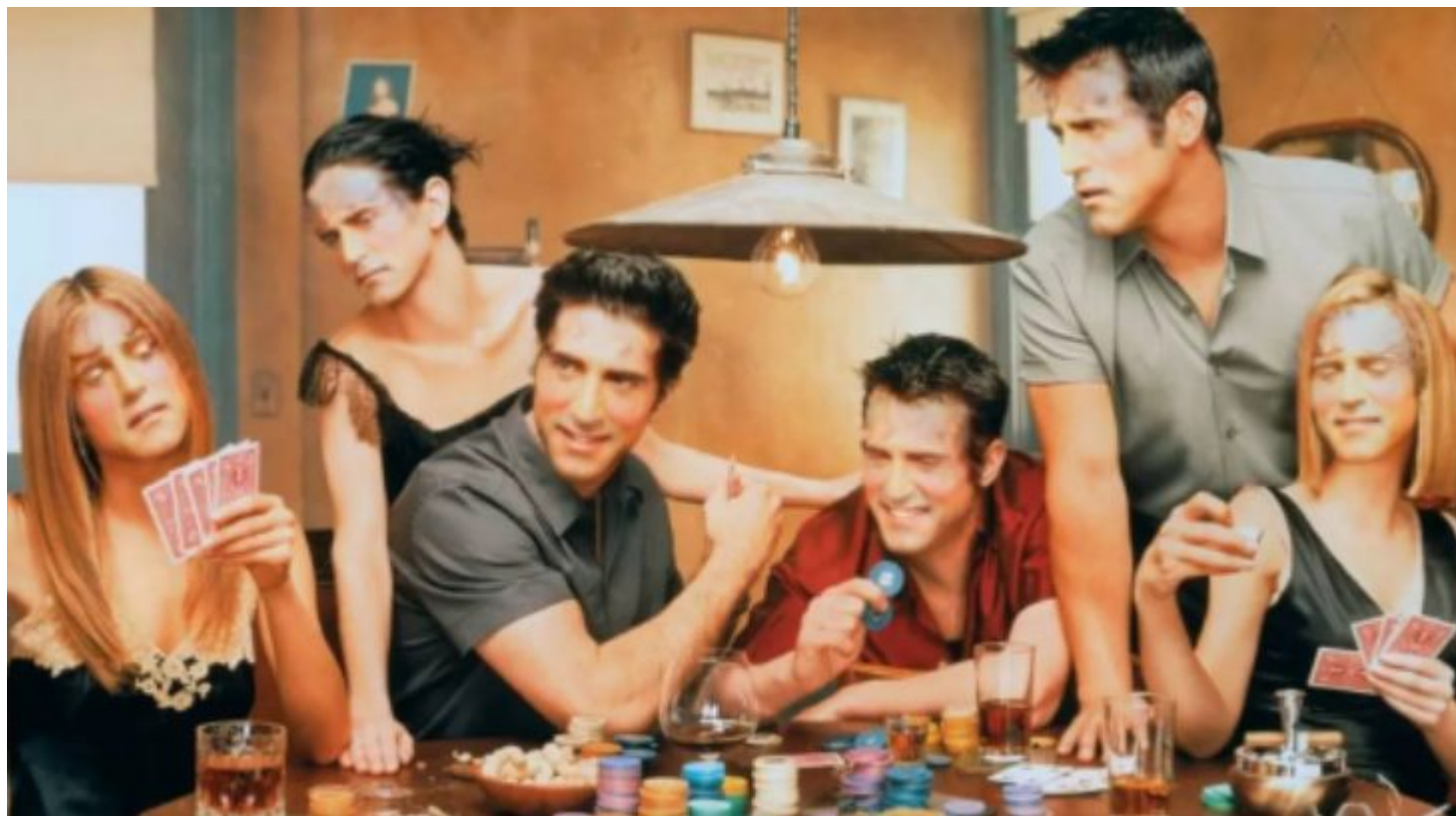
Para poner una foto nuestra:

```
rob = cv2.imread("./pant04.jpg")  
plt.imshow(rob[:, :, ::-1])  
plt.show()
```



y por último realizamos el intercambio de caras

```
rob_faces = app.get(rob)
rob_face = rob_faces[0]
res = img.copy()
for face in faces:
    res = swapper.get(res, face, rob_face, paste_back=True)
fig, ax = plt.subplots()
ax.imshow(res[:, :, ::-1])
ax.axis("off")
plt.show()
```



y por último ... ¿y si queremos obtener las caras de una foto cualquiera?

```
foto = cvs.imread("aqui-otra-foto.jpg")
plt.imshow(foto[:, :, ::-1])
plt.show()

faces = app.get(foto)
res = foto.copy()
for face in faces:
    res = swapper.get(res, face, rob_face, paste_back=True)
fig, ax = plt.subplots()
ax.imshow(res[:, :, ::-1])
ax.axis("off")
plt.show()
```