UNIVERSIDAD
FRANCISCO DE VITORIA

PRÁCTICA 3

Asignatura: Procesamiento Multimedia
Profesor: Eusebio Daniel Rodrigues

31 DE DICIEMBRE DE 2023

Autor: Enrique Ramos García y Pablo Martos García
4º INGENIERÍA MATEMÁTICA

ÍNDICE

Introducción

El proyecto realizado consiste en el desarrollo de un programa de detección facial en vídeos y posteriormente la clasificación de la cara detectada como una de dos opciones: hombre o mujer

Motivación

Nuestra motivación a la hora de realizar este proyecto se ha basado en buscar la manera de relacionar nuestros conceptos aprendidos sobre procesamiento de vídeos y en OpenCV unidos a la idea de introducir y aplicar nuestros conocimientos en el área del machine learning combinándolos así para crear un algoritmo de clasificación con una primera detección mediante clasificadores Haar Cascade.

Técnicas utilizadas

A continuación, se explican brevemente las técnicas utilizadas para el procesamiento de vídeo. Estas se verán, posteriormente, en el desarrollo del código.

1. Clasificación de género: utilizada para identificar el género de las personas en un vídeo. Se entrena un modelo de clasificación de género y se utiliza para analizar cada rostro detectado en los fotogramas del vídeo.

2. Detección de rostros: se utiliza un clasificador en cascada de Haar para identificar y extraer los rostros en cada fotograma del vídeo.

3. Procesamiento de imágenes: varias operaciones de procesamiento de imágenes, como la conversión a escala de grises, redimensionamiento y normalización, son aplicadas a las imágenes para poder ser procesadas por el modelo de clasificación de género.

Desarrollo e implementación del código

A continuación, se explica el desarrollo del proyecto, así como una explicación de las diversas partes del código y su utilidad en el mismo.

Red Neuronal de Clasificación

Para el desarrollo del proyecto, este se ha dividido en dos partes. En la primera se ha diseñado y entrenado una red neuronal para clasificar caras en hombre o mujer. Para esto se ha utilizado el modelo preentrenado MobileNetV2 de TensorFlow, al cual se le han añadido varias capas adicionales para adaptar la red a nuestras necesidades específicas y mejorar su rendimiento en la tarea de clasificación de género. Todo este proceso se encuentra documentado a mayor profundidad en el Jupyter Notebook de la carpeta del proyecto (*GenderClassification_OpenCV&Keras.ipynb*). Cabe mencionar que este proceso se ha realizado en Google Colab por conveniencia y, posteriormente, se ha descargado el modelo de clasificación para su aplicación en la segunda parte del proyecto.

Clasificador

Continuando con el proyecto se ha realizado un programa que dado un vídeo es capaz de detectar caras humanas de frente mediante detección Haar Cascade con un archivo XML preentrenado y posteriormente clasificar estas caras en hombre o mujer. Una vez clasificadas dibuja un rectángulo azul o rojo alrededor de la cara en función del género. A continuación, se explica este proceso en mayor profundidad.

Implementación (código y explicación)

A continuación, se explica el funcionamiento y desarrollo del código. Primero se explicarán las librerías y su objetivo. Posteriormente, se explicarán las funciones desarrolladas. Finalmente, se explicará el flujo de trabajo del código.

Librerías

El desarrollo de la práctica se ha realizado en Python. Además, se han utilizado las siguientes librerías:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
```

1. **Cv2:**

OpenCV es una librería de programación open source que se centra en el procesamiento de imágenes y *computer vision* en tiempo real. Esta proporciona diferentes funciones y herramientas para cargar, guardar, mostrar y manipular imágenes y videos, así como implementar *computer vision algorithms* (detección de bordes, seguimiento de objetos…).

2. **Numpy:**

NumPy es una biblioteca de Python que permite trabajar con matrices y funciones matemáticas de alto nivel. En este proyecto, se utiliza NumPy para trabajar con imágenes como matrices y realizar operaciones con estas.

3. **Matplotlib:**

Matplotlib es una biblioteca de Python utilizada para crear gráficos y visualizaciones de datos. En este proyecto, se utiliza pyplot, un submódulo de Matplotlib, para mostrar imágenes y gráficos.

4. **Tensorflow:**

TensorFlow es una biblioteca de aprendizaje automático de código abierto desarrollada por Google. En este proyecto, se utiliza TensorFlow para cargar y utilizar el modelo de clasificación de género previamente entrenado.

Funciones

- **Función *process_video()***

```python
def process_video(video_path, model, n_frames):
    cap = cv2.VideoCapture(video_path)
    frame_count = 0

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        if frame_count % n_frames == 0:
            processed_frame = checking(frame, model)
            cv2.imshow('Processed Frame', processed_frame)
            # Press 'q' to close the window
            if cv2.waitKey(10) & 0xFF == ord('q'):
                break

        frame_count += 1
    cap.release()
    cv2.destroyAllWindows()
```

- Parámetros de la función:

video_path: ruta del archivo de video a procesar
model: modelo preentrenado de clasificación que se utilizará para clasificar las caras
n_frames: integer indica cada cuántos fotogramas se realizará el procesamiento

- Variables:

cap: objeto de OpenCV utilizado para leer el video
frame_count: un contador de fotogramas utilizado para saber cuándo procesar un fotograma
ret: valor booleano que controla si lectura del frame es correcta

frame: matriz que representa el frame (imágen)
processed_frame: matriz que representa el frame tras detección, clasificación y procesamiento

- Operaciones:
cv2.VideoCapture: abre el archivo de video y crea un objeto de captura de video
cap.isOpened(): verifica si el archivo de video está abierto y listo para leer
cap.read(): lee un fotograma del video
cv2.imshow(): muestra el fotograma procesado en una ventana
cv2.waitKey(): espera a que se presione la tecla 'q' para cerrar la ventana
cap.release(): libera los recursos del objeto de captura de video
cv2.destroyAllWindows(): cierra todas las ventanas abiertas por OpenCV

- Salida: la función no devuelve nada, pero va mostrando los fotogramas procesados

- **Función *checking()***

```
def checking(img, model):
    label = {0: "female", 1: "male"}
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    faces = cascade.detectMultiScale(gray, 1.1, 7)

    for x, y, w, h in faces:
        face = img[y:y+h, x:x+w]
        # Check if the face image is not empty
        if face.size == 0:
            continue
        new_width = 160
        new_height = 160
        face_resized = cv2.resize(face, (new_width, new_height))

        face_scaled = face_resized / 255.0
        reshape = np.reshape(face_scaled, (1, new_height, new_width, 3))
        result = np.argmax(model.predict(reshape), axis=-1)

        if result == 0:
            cv2.rectangle(img, (x-10, y), (x+w, y+h), (255, 0, 0), 4)
            cv2.rectangle(img, (x-10, y-50), (x+w, y), (255, 0, 0), -1)
            cv2.putText(img, label[0], (x, y-10), cv2.FONT_HERSHEY_SIMPLEX,
                        0.5, (255, 255, 255), 2)

        elif result == 1:
            cv2.rectangle(img, (x-10, y), (x+w, y+h), (0, 0, 255), 4)
            cv2.rectangle(img, (x-10, y-50), (x+w, y), (0, 0, 255), -1)
            cv2.putText(img, label[1], (x, y-10), cv2.FONT_HERSHEY_SIMPLEX,
                        0.5, (255, 255, 255), 2)
    return img
```

- Parámetros de la función:
img: fotograma del video a procesar

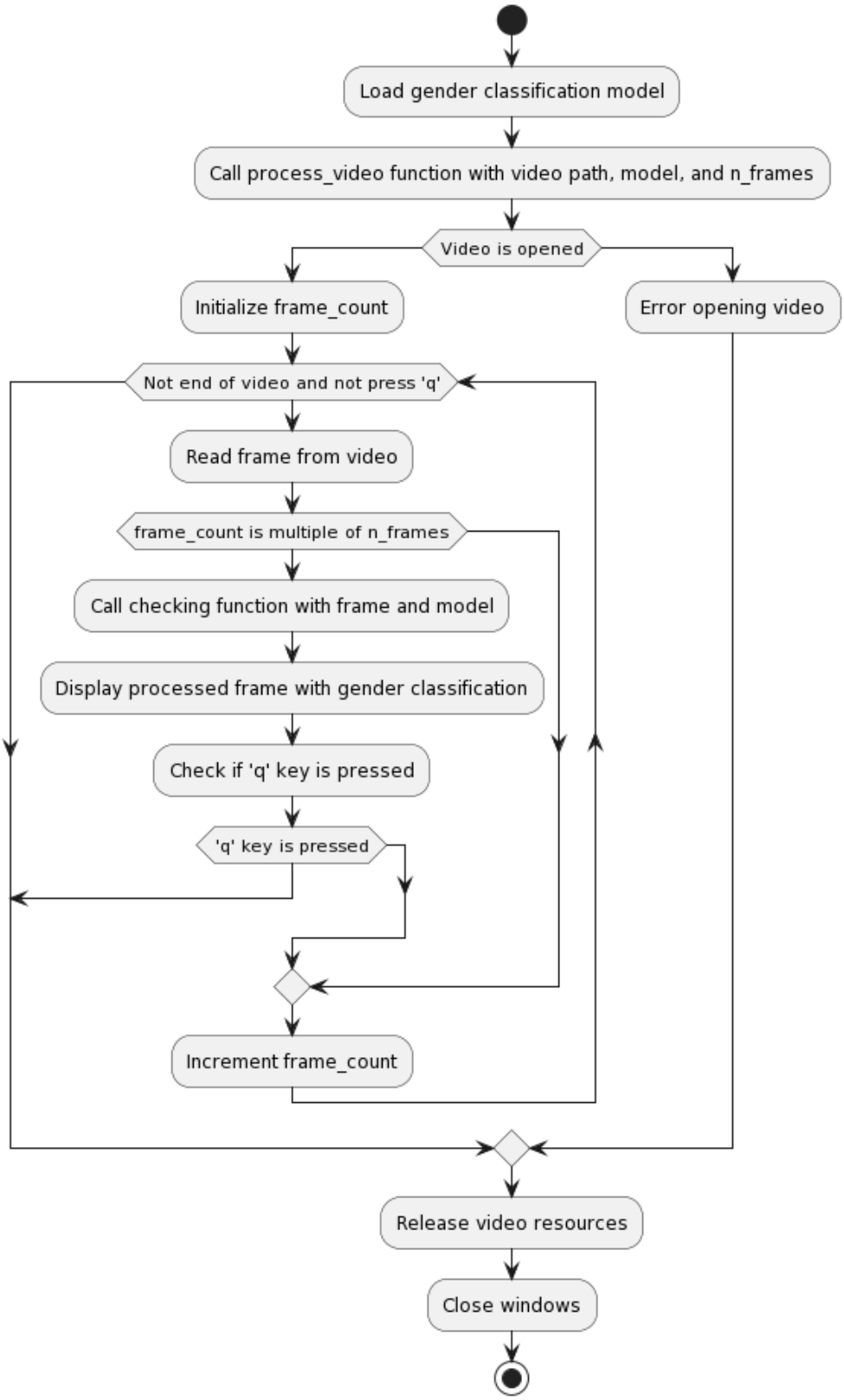
model: red neuronal de clasificación de género entrenado en el Google Colab

- Variables:
label: etiquetas numéricas a las categorías 'female' y 'male'.
gray: matriz que representa la versión en escala de grises de la imagen de entrada para su detección de rostros mediante HaarCascadeClassifier
cascade: objeto CascadeClassifier de OpenCV utilizado para detectar caras de frente en la imagen
faces: una lista que contiene las coordenadas y dimensiones de las caras detectadas en la imagen.
new_width, new_height: números que se utilizarán para redimensionar las imágenes (esto es necesario puesto que la red neuronal de clasificación está diseñada para inputs de imágenes 160x160)
result: etiqueta predicha por el modelo para la cara

- Operaciones:
cv2.cvtColor(): convierte la imagen a escala de grises.
cv2.CascadeClassifier(): crea CascadeClassifier a partir de un archivo
cascade.detectMultiScale(): detecta caras en la imagen
cv2.resize(): redimensiona la imagen
model.predict(): realiza predicción utilizando el modelo preentrenado
np.argmax(): etiqueta numérica predicha por el modelo (0 o 1)
cv2.rectangle(): dibuja rectángulos alrededor de las caras detectadas

- Salida: devuelve la imagen procesada con rectángulos y etiquetas dibujadas alrededor de las caras detectadas

Diagrama de flujo de trabajo

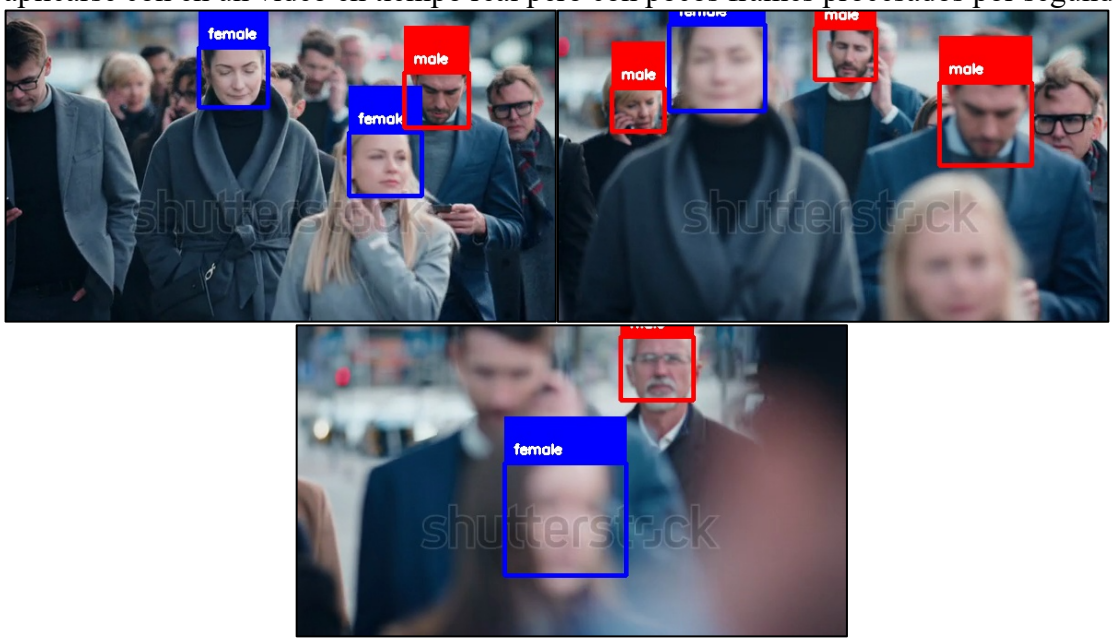


Resultados obtenidos

A continuación, se comentan los resultados obtenidos en el desarrollo del proyecto en función a los objetivos que se tenían marcados.

1. **Clasificación de género**: El modelo de clasificación de género con redes neuronales entrenado se podría considerar exitoso. Esto se debe a que se ha obtenido una precisión del 86% para los datos de test y de un 87% para los de validación
2. **Detección de rostros**: Se ha conseguido implementa correctamente la detección de rostros en un video utilizando *'haarcascade_frontalface_default.xml'*. Los rostros son detectados eficientemente en tiempo real en cada frame del video.
3. **Visualización de resultados**: El proyecto muestra los resultados de clasificación de género en tiempo real en una ventana de OpenCV, con etiquetas y rectángulos alrededor de cada rostro detectado y un color determinado para cada categoría.
4. Rendimiento: Por último, pero más importante para una implementación real del proyecto se encuentra el rendimiento. En este proyecto se ha obtenido un rendimiento adecuado, pero no a tiempo real. Los frames tardan alrededor de 42ms en ser procesados. Una posible mejora a futuro sería intentar mejorar el rendimiento del proyecto.

En resumen, los resultados obtenidos en este proyecto demuestran que el modelo de clasificación de género y la detección de rostros funcionan correctamente y pueden aplicarse con en un video en tiempo real pero con pocos frames procesados por segundo.



Los resultados pueden observarse tanto en el vídeo de los frames procesados en la carpeta ***outputvideo*** como en la carpeta ***processedframes*** para ver los frames uno a uno en imágenes. Además, en el Jupyter Notebook también hay ciertas comprobaciones.

Conclusión

El proyecto de detección facial, clasificación de género y seguimiento de objetos en video ha permitido obtener valiosas conclusiones sobre la efectividad del detector facial utilizado, la precisión en la asignación de género, y la coherencia en el seguimiento de objetos. Se observó la capacidad del sistema para manejar múltiples caras simultáneamente, evaluando la eficiencia del procesamiento de video y la optimización de recursos. Los resultados proporcionan información clave sobre la aplicabilidad del sistema en diversas situaciones, como conteo de género en eventos o análisis demográfico en entornos públicos, permitiendo posibles ajustes para mejorar su desempeño y versatilidad en aplicaciones práctica.