Demo - Reconocimiento facial

Mediante la librería de Python **face-recognition** es posible reconocer y manipular rostros bajo modelos pre-entrenados de Deep learning. Permitiendo:

* Encontrar rostros en una imagen
* Manipular características faciales de los rostros en una imagen.
* Identificar rostros en imágenes

**Requerimientos:**

Para el reconocimiento facial es necesario tener instalado lo siguiente:

* Python 3.X (Preferible v3.6.8)
* Face\_recognition
* OpenCv
* Imutils

**Instalación del entorno de trabajo:**

En Anaconda, estos son los comandos necesarios para instalar el entorno de trabajo.

1. Creación y activación del entorno con Python en la versión 3.6.8

*conda create -n test\_env python=3.6.8*

*conda activate test\_env*

1. Instalación de librerías

*pip install cmake*

*pip install numpy*

*pip install scipy*

*pip install matplotlib*

*pip install pandas*

1. Instalación de DLIB y wheels

*python -m pip install https://files.pythonhosted.org/packages/8c/23/848298cccf8e40f5bbb59009b32848a4c38f4e7f3364297ab3c3e2e2cd14/wheel-0.34.2-py2.py3-none-any.whl*

*python -m pip install https://files.pythonhosted.org/packages/0e/ce/f8a3cff33ac03a8219768f0694c5d703c8e037e6aba2e865f9bae22ed63c/dlib-19.8.1-cp36-cp36m-win\_amd64.whl#sha256*=794994fa2c54e7776659fddb148363a5556468a6d5d46be8dad311722d54bfcf

1. Desde python probamos que la instalación de dlib esté correcta con *import dlib*.
2. Instalamos face\_recognition

*python -m pip install https://files.pythonhosted.org/packages/3f/ed/ad9a28042f373d4633fc8b49109b623597d6f193d3bbbef7780a5ee8eef2/face\_recognition-1.2.3-py2.py3-none-any.whl*

1. Instalación Imutils y OpenCv

*pip install opencv-python*

*pip install imutils*

**ESTRUCTURA DEL PROYECTO**

Dentro del proyecto se encuentran tres carpetas con la siguiente estructura:

* **Dataset**: Almacena las imágenes conocidas con las que se entrena un modelo de reconocimiento facial, ordenadas en subcarpetas por cada usuario.
  + **Usuario01**
    - Imagen\_01.jpg
    - Imagen\_02.jpg
    - Imagen\_n.jpg
  + **Usuario02**
    - Imagen\_01.jpg
    - Imagen\_02.jpg
    - Imagen\_n.jpg
* **Test**: Se almacena las imágenes o videos no conocidos con los que se realizará las pruebas de reconocimiento facial.
  + **Imagen**
    - Imagen\_01.jpg
    - Imagen\_n.jpg
  + **Video**
    - Video\_01.mp4
    - Video\_n.mp4
* **Output**: Aquí se almacenarán los videos de la webcam o de archivos que se ingresaron para su reconocimiento.
  + Webcam\_output.avi

Los archivos escritos en Python permiten:

* **Encode\_faces.py**: serializa los rostros conocidos obtenidos desde el dataset, según el modelo seleccionado,y los almacena en el archivo *encodings.pickle*.
* **Encondings.pickle**: Contiene la serialización de los rostros conocidos y nombres de estos para luego compararlos con los rostros no conocidos que se ingresen para el reconocimiento facial.
* Los archivos **recognize\_faces\_image.py**, **recognize\_faces\_video.py** y **recognize\_faces\_video\_file.py** contienen la lógica del reconocimiento facial. Para el reconocimiento facial en video, se hace uso del modelo HOG (Histograma de gradientes orientadas).

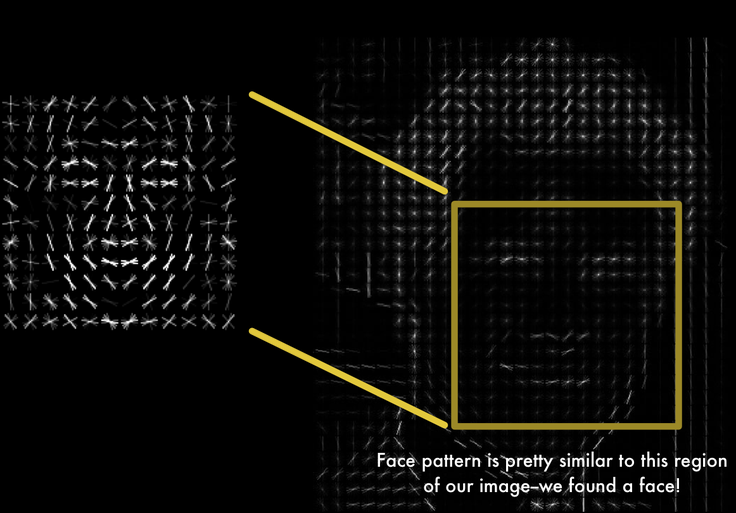
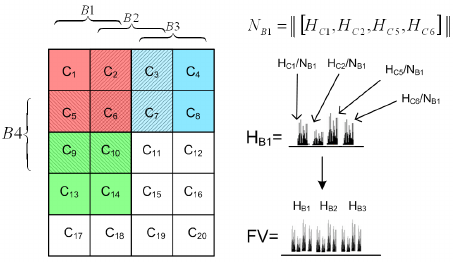


Ilustración 1. Patrón de un rostro usando HOG

**FLOWCHART**

Dataset

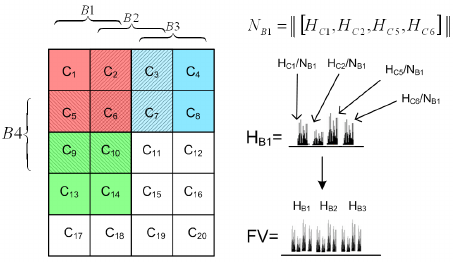
Encodings



Extracción de características



Video Test



Extracción de características



Búsqueda de coincidencias

Clasificación

**OUTPUT**

**MODELO DE RECONOCIMIENTO FACIAL**

La librería face\_recognition (<https://pypi.org/project/face-recognition/>) permite reconocer y manipular rostros con el lenguaje Python, provee un modelo **pre-entrenado** de Deep-learning . Para este caso, se hace uso del modelo HOG (<https://www.researchgate.net/publication/283543555_Facial_expression_recognition_and_histograms_of_oriented_gradients_a_comprehensive_study>) o CNN[[1]](#footnote-1)(Redes neuronales Convolucionales), mismo que se especifica en tanto al generar el archivo que contiene las serializaciones de los rostros conocidos, como al realizar las pruebas de reconocimiento facial, por lo tanto:

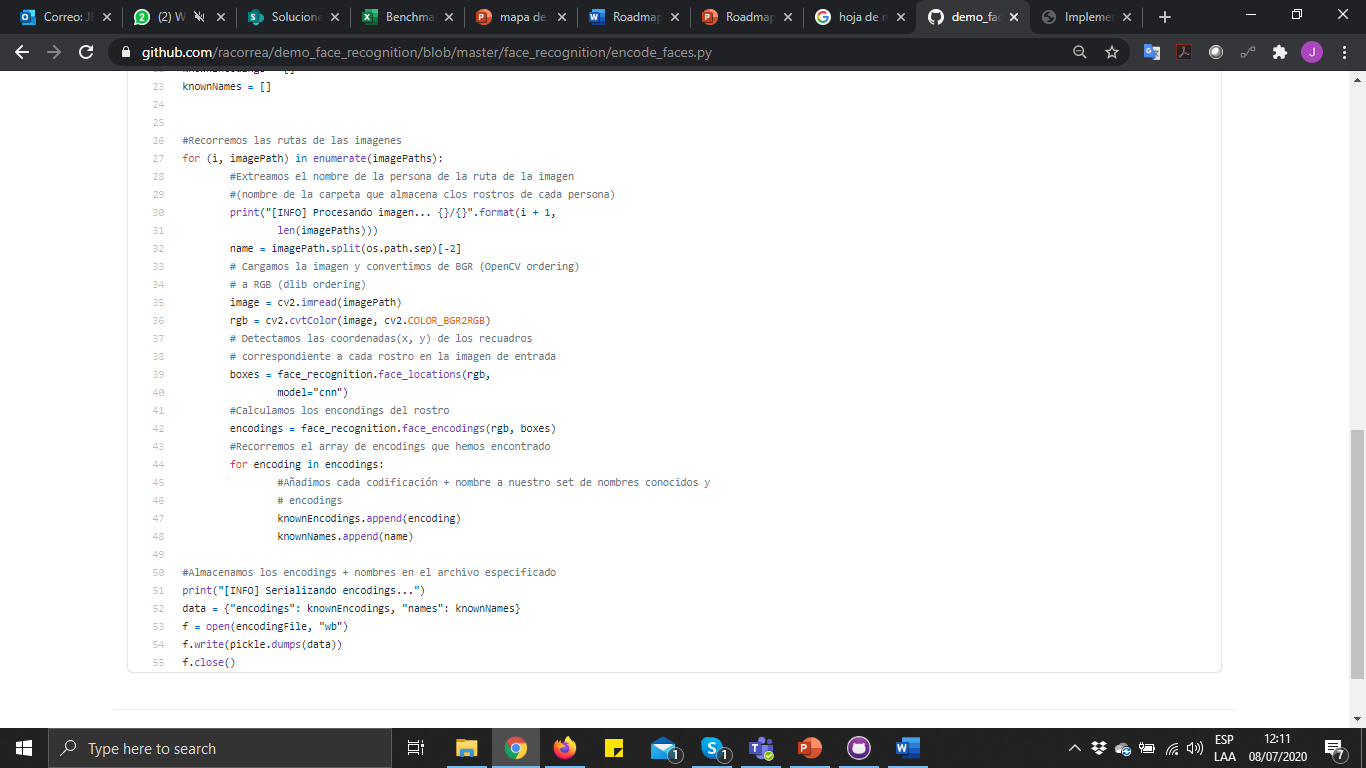
Primero se transforma la imagen a escala de grises mediante la siguiente línea de código

rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

Luego, en el archivo **encode\_faces.py**, en la línea 39, donde se detecta las coordenadas de los recuadros de cada rostro, se envía la imagen y se determina el modelo a usar [‘hog’ ‘cnn’]:

*boxes = face\_recognition.face\_locations(rgb, model="hog")*

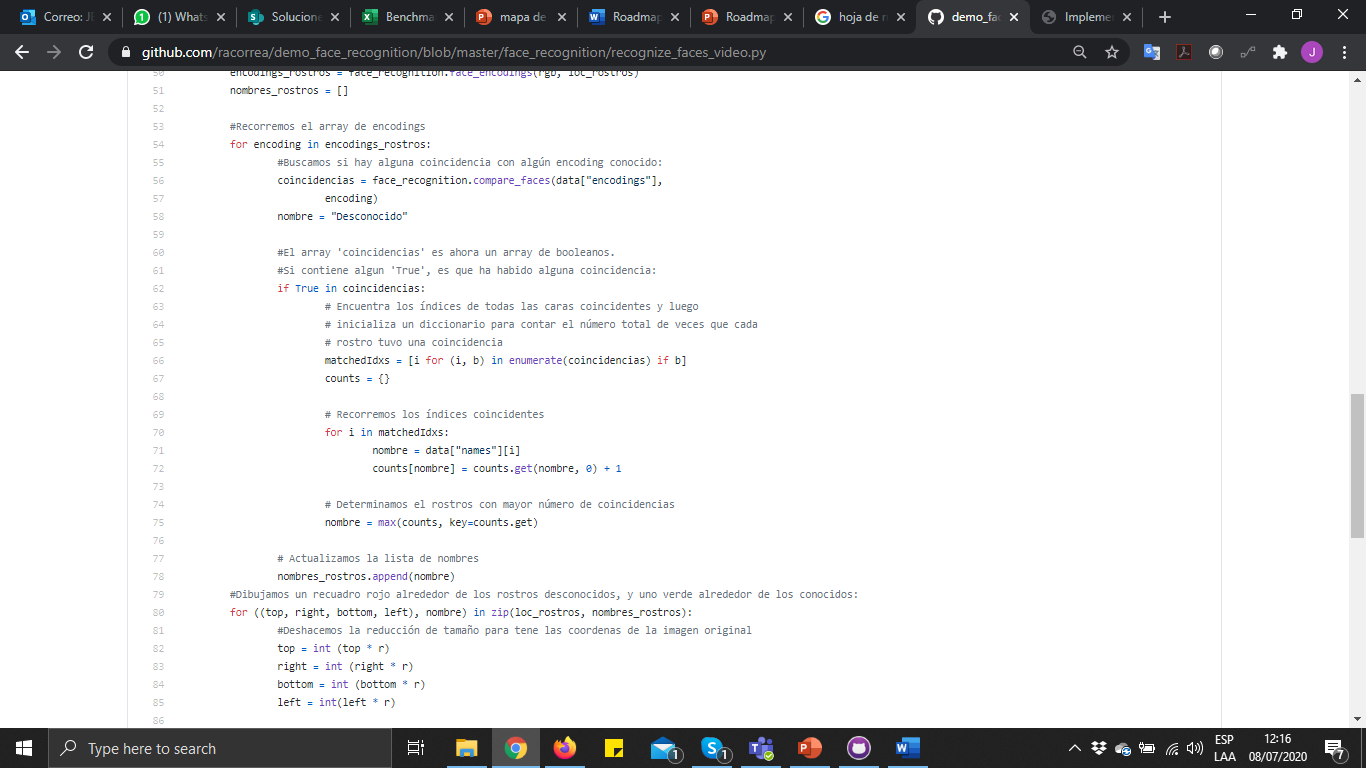
Posterior a ello, se realiza el cálculo de las características de los rostros conocidos:



En los archivos **recognize\_faces\_image.py**, **recognize\_faces\_video.py** y **recognize\_faces\_video\_file.py**, se realiza la misma configuración para el modelo a utilizar en los rostros no conocidos:

*Loc\_rostros = face\_recognition.face\_locations(rgb, model="hog")*

Para luego comparar las características obtenidas de los rostros conocidos con los nuevos rostros y ver si existen coincidencias:



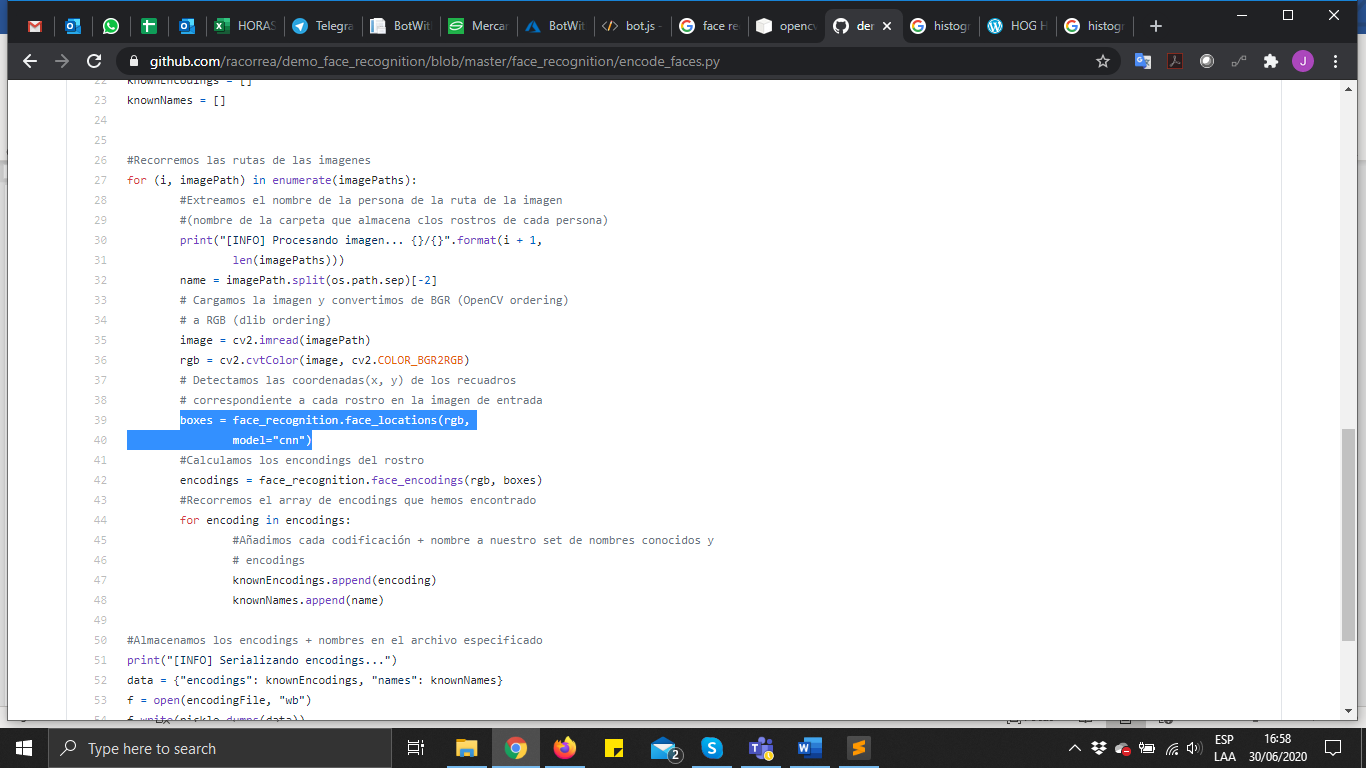
**EJECUCIÓN DEL PROYECTO**

Como primer paso, se debe cargar las imágenes de los rostros conocidos en el dataset según la estructura indicada con los nombres de las personas, estos servirán para entrenar el modelo.

En el cmd nos ubicamos en el directorio del proyecto y ejecutamos el archivo encode\_faces.py para generar las serializaciones de los rostros que hemos ingresado en el dataset, de la siguiente manera:

*python encode\_faces.py*

Para ver las diferencias existentes entre un modelo HOG y CNN (redes neuronales convolucionales), en el archivo encode\_faces.py se puede modificar el valor del modelo “hog” por “cnn”.



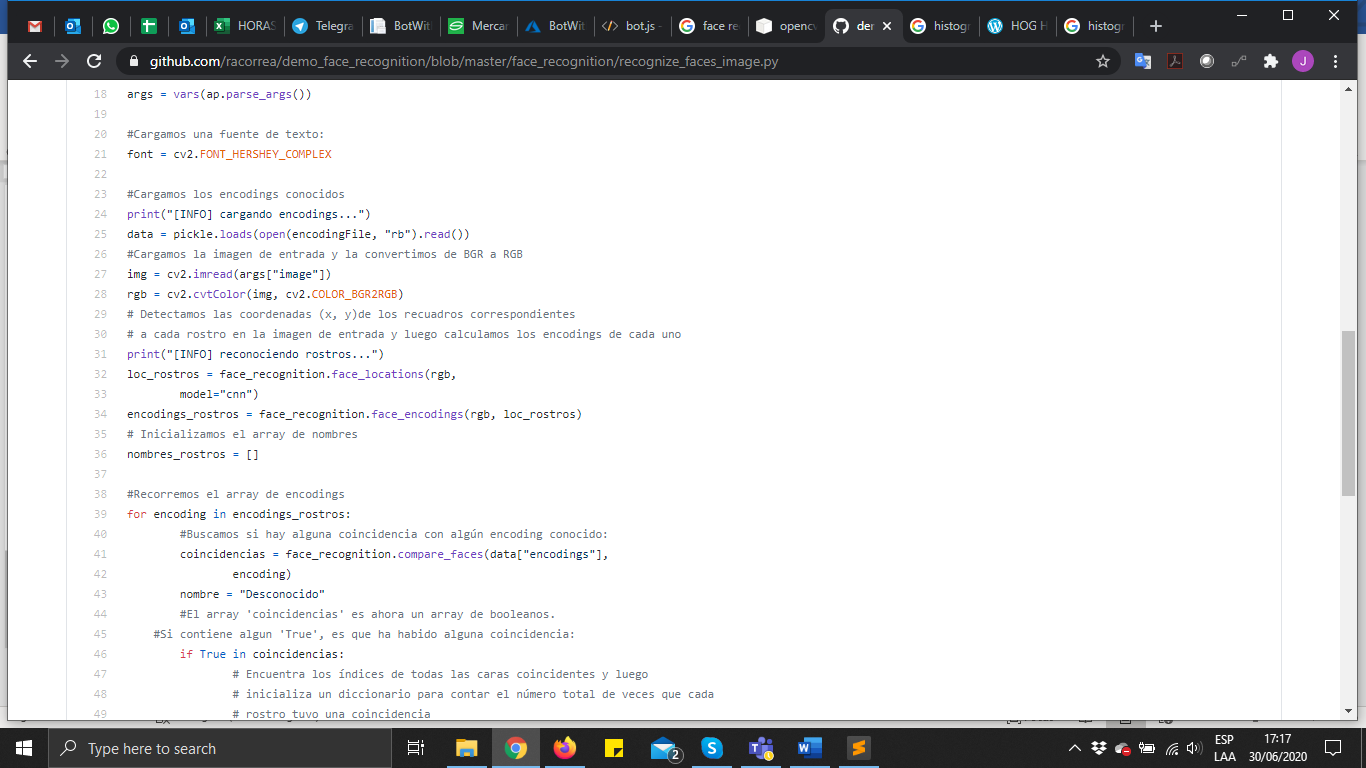
Procedemos a ejecutar los archivos de reconocimiento facial, según lo que se requiera:

* **Para detección en Imagen:**

Requiere como parámetros de entrada la imagen sobre la cual se realizará el reconocimiento facial, para ello, ubicarse en el directorio del proyecto y ejecutar:

*python recognize\_face\_image.py -i test\image\image\_01.jpg*

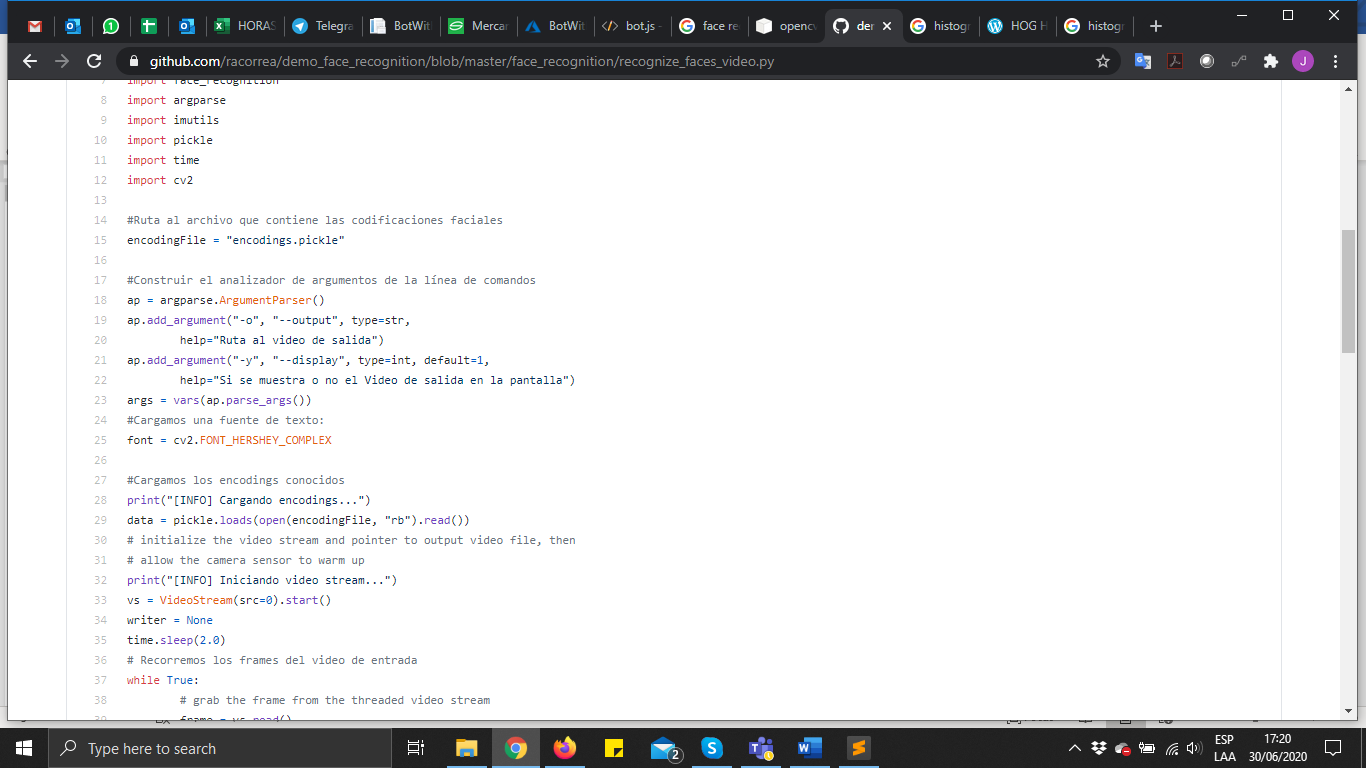
Para el reconocimiento facial en imágenes, se está haciendo uso del modelo CNN para mayor precisión, sin embargo, se puede probar con el modelo HOG.



* **Para detección desde webcam:**

Para realizar el reconocimiento facial mediante webcam, es opcional los siguientes parámetros:

* + Ruta al video de salida, en caso de querer almacenarlo
  + Definir si se muestra o no el video de salida en pantalla, por defecto se muestra.



Para su ejecución, ubicarse en el directorio del proyecto y existen las siguientes posibilidades:

* + Si se desea guardar el video final y que no se muestre en pantalla:

*python recognize\_face\_video.py -o output\webcam\_output.avi -y 0*

* + Si no se desea guardar el video final

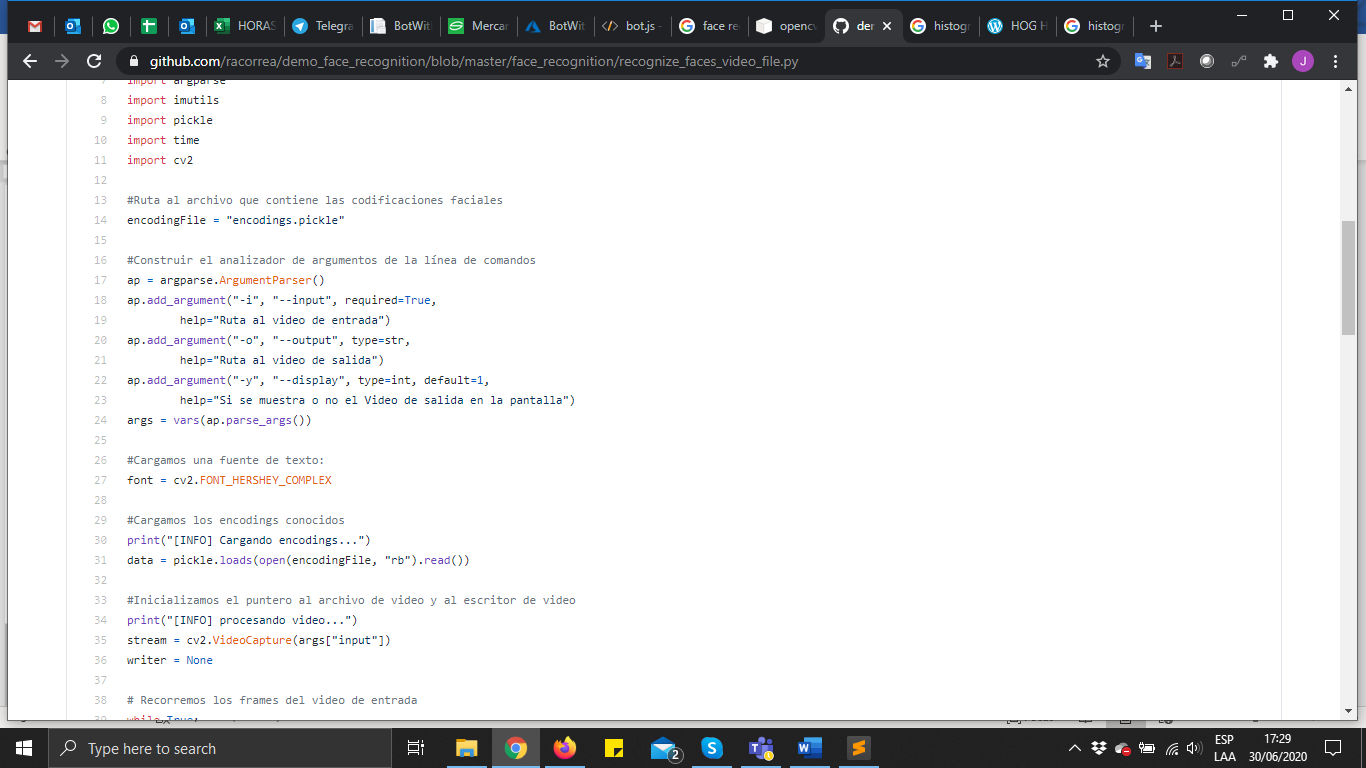
*python recognize\_face\_video.py*

Se debe tener en cuenta que se utiliza el modelo HOG, por temas de procesamiento.

* **Para detección desde archivo de video:**

Para la detección de rostros mediante archivos de video, los parámetros son:

* + Ruta al video de entrada, **parámetro requerido**.
  + Ruta al video de salida, en caso de querer almacenarlo
  + Definir si se muestra o no el video de salida en pantalla, por defecto se muestra.



Para su ejecución, ubicarse en el directorio del proyecto:

*python recognize\_face\_video\_file.py -i output\video.mp4 -o output\video.avi -y 1*

Se debe tener en cuenta que se utiliza el modelo HOG, por temas de procesamiento.

Para **detener** la ejecución del reconocimiento facial en la webcam o archivos de video, pulsar la tecla “**q**”.

1. Note: GPU acceleration (via nvidia’s CUDA library) is required for good performance with this model. You’ll also want to enable CUDA support when compliling dlib [↑](#footnote-ref-1)