Detector de personas con o sin mascarillas

Jordi Sebastián Valencia Cartagena

Este algoritmo de detección de uso de máscaras en tiempo real se crea utilizando herramientas de aprendizaje automático en una disciplina conocida como visión artificial. El objetivo es que la aplicación que contiene este algoritmo pueda detectar si una persona no lleva mascarilla. Algo que podría ser una tarea bastante simple para nuestros cerebros adquiere cierta complejidad. En términos de proceso, primero le enseñamos a nuestro algoritmo a "aprender" (por eso se llama inteligencia artificial) y a poder distinguir si lo que ve es una persona y luego si una persona lleva una mascarilla o no.

I. Introducción

Se recomienda el uso de mascarilla durante la pandemia. La OMS originalmente recomendaba usarlo solo para personas enfermas, pero recientemente cambió su posición y recomienda su uso más amplio. Por eso hay la idea de desarrollar un algoritmo de inteligencia artificial para detectar el uso de mascarillas en tiempo real, en tiempos de pandemia. Este algoritmo se puede instalar en una computadora con una simple cámara web en la entrada de una oficina, tienda, lugar público o cualquier lugar donde se quiera verificar el uso de este dispositivo de protección personal.

II. Desarrollo de Contenidos

Open Source Computer Vision (OpenCV) es una biblioteca de programas informáticos de código abierto desarrollada para admitir aplicaciones que utilizan la visión artificial. Ofrece cientos de funciones para capturar, analizar y manipular datos visuales y puede eliminar algunos de los problemas que enfrentan los programadores cuando desarrollan aplicaciones basadas en visión por computadora. Partes de la biblioteca también brindan una interfaz de usuario y capacidades de reconocimiento de patrones. OpenCV se ha utilizado en aplicaciones prácticas y creativas, incluidos vehículos autónomos y nuevas formas de arte digital [1].

MediaPipe, una herramienta para teléfonos Android que usa la cámara para aprender movimientos y formas de las manos a través del aprendizaje automático, por lo que los gestos del lenguaje de señas pueden decirle a Google lo que desea transferir [2].

En el proyecto se hizo uso de entornos virtuales estos nos permitirán crear entornos de python con sus paquetes estándar, permitiendo instalar los paquetes que necesitemos en cada uno de ellos sin que estos interfieran con otros entornos creados, ni con los paquetes ya instalados en todo el sistema. Es decir que estos nos ayudarán a organizar los paquetes que usamos para nuestros proyectos, y de tal modo trabajaremos eficientemente.

Para la realización de este proyecto necesitaremos tener instalados en python: Numpy, OpenCV Y MediaPipe.

Para empezar vemos un dataset con dos clases, por un lado tenemos imágenes a color de rostros con mascarillas y por otro índice tenemos imágenes de rostros de personas sin mascarillas. Estas imágenes se obtuvieron con mediapipe después de aplicar la detección de rostros para ajustarlas a una altura y ancho específicos.

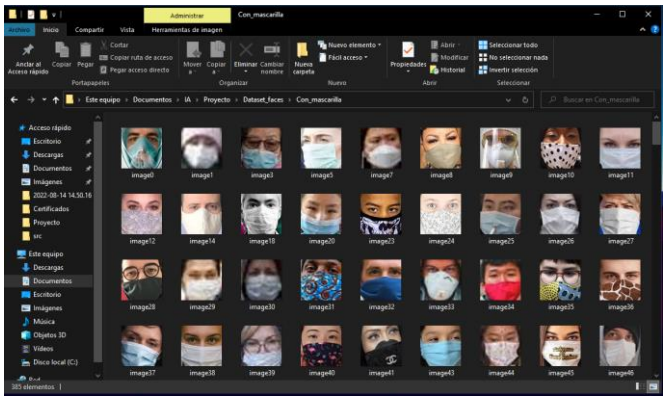

Fig. 1 Dataset.

Este dataset fue construido tomando imágenes de https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset?select=Face+Mask+Dataset.

En un principio las imágenes fueron obtenidas de la fuente anteriormente mencionada, luego se volvió a aplicar detección facial con MediaPipe face detection. Una vez ubicado el rostro en la imagen se procedió a recortar esta área y se la redimensionó a 72x72 pixeles.

Los procesos de aprendizaje supervisado en machine learning requieren de entrenamiento, el cual se realiza a través de imágenes. En este caso, fotografías de personas con mascarillas y sin mascarillas. Normalmente se utiliza el mismo número de imágenes de un tipo y de otro tipo. En este módulo, enseñaremos a nuestro algoritmo a reconocer estas diferencias a través de un proceso de aprendizaje supervisado (ya que le decimos de antemano qué imágenes tienen máscara y cuáles no).

Después de descargar el conjunto de datos, entrenaremos un modelo usando Local Binary Pattern Histogram Face Recognizer. De esta forma, experimentaremos si este método puede ayudarnos a ver si una persona lleva mascarilla o no, ya que este método suele utilizarse para el reconocimiento facial. Para ello, creamos el siguiente script llamado train.py, como en Fig. 2.

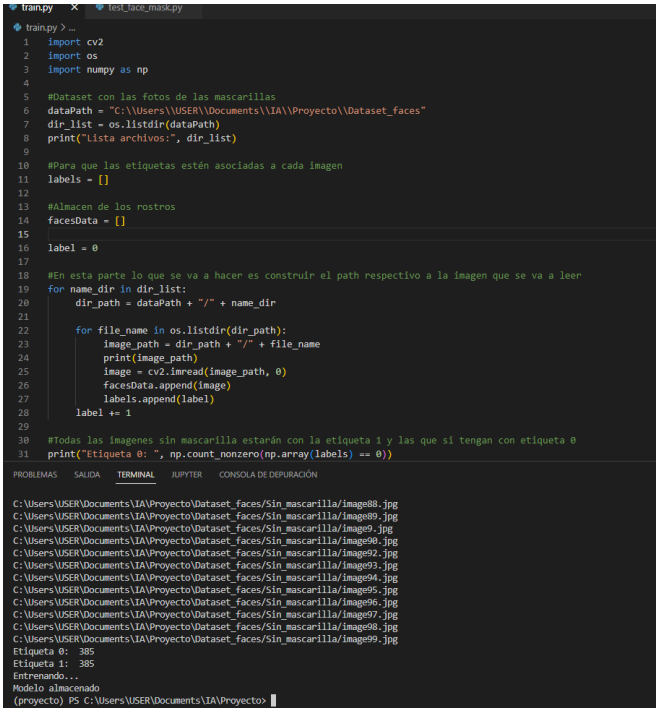

Fig. 2 Entrenamiento del modelo.

Finalmente, probaremos nuestro modelo entrenado, para esto necesitamos aplicar la detección de rostros con Mediapipe en las imágenes de entrada (que se aplicó para obtener las imágenes de entrenamiento).

A continuación, debemos recortar la imagen de la cara, aplicar la escala de grises y cambiar su tamaño. De esta forma, aplicaremos los mismos procedimientos tanto a las imágenes que se tenía para entrenar como a aquellas con las que testearemos nuestro modelo.

Finalmente probaremos el algoritmo para verificar su funcionamiento.

Fig. 3 muestra un caso donde la persona se encuentra sin mascarilla, mientras que Fig. 4 muestra a la persona con mascarilla.

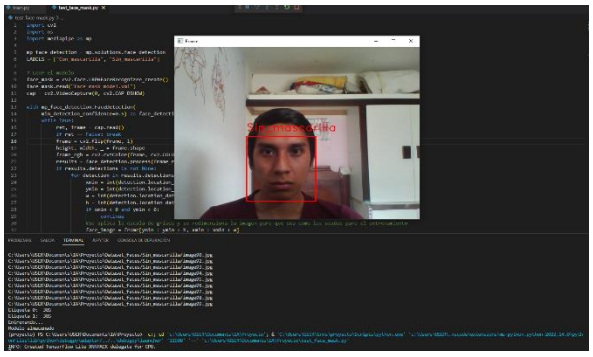

Fig. 3 Captura de la imagen sin mascarilla

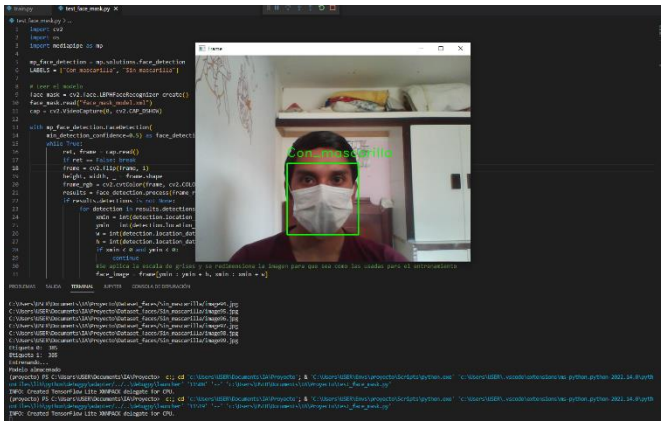

Fig. 4 Captura de la imagen con mascarilla

CONCLUSIONES

El programa cumple con los requisitos pero no es 100% efectivo debido a que se entrenó con imágenes de personas de frente a la cámara y aquí es donde presenta dificultades al identificar las mascarillas si se esta de lado o mirando hacia arriba o muy abajo.

Una forma de mejorar la efectividad sería agregando más imágenes al dataset para que el algoritmo aprenda varias posiciones de la cara.

Taparse la boca con la mano también hace que el algoritmo lo detecte como mascarilla.

REFERENCIAS

[1] ¿Qué es OpenCV? website. [Online]. Available: https://spiegato.com/es/que-es-opencv

[2] MediaPipe, la tecnología de Google website. [Online]. Available: https://difusionnorte.com/mediapipe-google-senas/