Categoría prototipado: Ingeniería.

Sistema de detección de mascarilla mediante visión artificial

Mask detection system by artificial vision

Edwil Aguilar[1], Wing Ng[1], Anthony Romero[1], Eloy Lezcano[1]*
[1]Facultad de Ingeniería de Sistemas Computacionales, Centro Regional de Chiriquí, Universidad Tecnológica de Panamá

Resumen Este documento proporciona información sobre nuestro proyecto de visión artificial, que aplican lo estudiado en el curso de "Herramientas Aplicadas para la Inteligencia Artificial". La planificación y diseño del proyecto tuvo en cuenta las condiciones de salud que hemos presenciado en el mundo debido a la rápida propagación del COVID-19. La bioseguridad ha producido una respuesta absoluta en el mundo, que no se puede ignorar. En gran medida, el problema radica en la falta de comprensión de la multitud de alrededor, que se opone al correcto uso de la mascarilla. Para ello, diseñamos un modelo de IA enfocada en la visión artificial para cooperar con las entidades de salud en la detección del uso de las mascarillas en las personas manteniendo un control en el cumplimiento de las normas de uso obligatorio de las mismas, pudiendo así detectar de una forma bastante precisa si el individuo está utilizando la mascarilla o no. Además, nuestro sistema puede ser implementando en pequeños y grandes establecimientos ayudando a cumplir con las normas sanitarias impuestas por el Ministerio de Salud.

Palabras clave Conjunto de datos**,** entrenamiento**,** IA, TensorFlow, visión artificial**.**

Abstract This document provides information about our computer vision project, which applies what we studied in the "Applied Tools for Artificial Intelligence" course. The planning and design of the project considered the health conditions we have witnessed in the world due to the rapid spread of COVID-19. Biosecurity has produced an absolute response in the world, which cannot be ignored. To a large extent, the problem lies in the lack of understanding of the surrounding crowd, which opposes the correct use of the mask. For this, we designed an AI model focused on artificial vision to cooperate with health entities in detecting the use of masks in people by keeping a check on compliance with the rules of mandatory use of masks, thus being able to detect in a fairly accurate way whether the individual is using the mask or not. In addition, our system can be implemented in small and large establishments helping to comply with the sanitary norms imposed by the Ministry of Health.

Keywords Dataset, training, AI, TensorFlow, artificial vision.

* Corresponding author: eloy.lezcano@utp.ac.pa

1. Introducción

Desde la aparición del COVID-19 y luego de declarar como una pandemia por parte de la OMS, se han puesto en práctica diferentes medidas de bioseguridad como: el distanciamiento social, restricción de viajes, aislamiento, uso de alcohol en gel, uso de mascarillas para evitar propagar el virus a más personas y así reducir el número de personas contagiadas [7]. Aunque al día las medidas se han reducido por una baja cantidad de casos en comparación a antes, es importante asegurar en los establecimientos que se siguen las medidas para prevenir que el virus se siga propagando y con el avance de la tecnología y la cuarta revolución industrial se facilita mucho esta labor.

La ciencia informática camina actualmente hacia la creación de computadoras cada vez más rápidas, más expertas y autónomas. Una de las empresas más ambiciosas consiste en dotar a los ordenadores de la facultad de relacionarse con su entorno del mismo modo que lo hace un humano: a través de los sentidos. Proporcionar capacidad sensorial a un ordenador es una tarea difícil ya que entran en juego elementos accesorios al microprocesador, como son los sensores, las tarjetas adaptadoras de señal o el ruido del entorno. A pesar de los muchos inconvenientes, existe hoy un interés especial por dotar a los ordenadores de uno de los cinco sentidos del hombre: la habilidad de ver. Mientras que el resto de los sentidos humanos no despierta interés, la visión artificial se aplica ya en procesos industriales en los que la simple detección de presencia no resulta una fuente de información suficiente [5].

La visión artificial o visión por computador intenta emular la capacidad de algunos seres vivos para ver una escena, entenderla y actuar en consecuencia. Existe un crecimiento en

el número y tipo de aplicaciones industriales que demandan el uso de técnicas de visión artificial [3]. Entre los sectores donde más se están utilizando actualmente esta tecnología se encuentran las industrias: auxiliar del automóvil, farmacéutica, embotellado y embalaje, etiquetado, textil, papel, metalúrgica, semiconductores, cerámica, madera, equipos electrónicos, entre otros [5].

La visión artificial lleva asociada una enorme cantidad de conceptos relacionados con hardware, software y también con desarrollos teóricos.

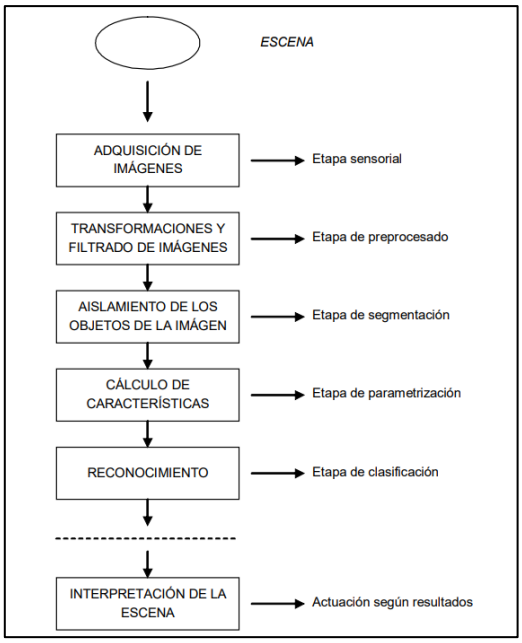

Figura 1. Etapas en un proceso de visión artificial. Fuente: [5]

Hoy en día, se pueden encontrar herramientas que facilitan la creación de sistemas inteligentes de tal manera que no es necesario programarlos desde cero. Uno de los más utilizados y conocidos de código abierto está TensorFlow que es una plataforma de código abierto de extremo a extremo para el aprendizaje automático. Cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos de la comunidad que permite que los investigadores innoven con el aprendizaje automático y los desarrolladores creen e implementen aplicaciones con tecnología de aprendizaje automático fácilmente [6].

Por lo tanto, nuestro proyecto emplea la visión artificial y TensorFlow para poder diseñar un sistema capaz de detectar si una persona lleva puesta la mascarilla o no, también cabe resaltar que este sistema es adaptable a cualquier modelo que permita reconocer categorías que sean de interés para el usuario.

2. Antecedentes

Se han realizado diversos trabajos relacionados al reconocimiento de objetos, entre los que coincidían con nuestra idea de proyecto podemos mencionar al proyecto realizado por Montesdeoca Ordoñez [7], que entrenó cuatro redes neuronales con diferentes parámetros a los cuales evaluó y aplicó métricas para seleccionar la red neuronal más eficiente, y desarrolló una página web en la que el usuario puede observar e interactuar con la red neuronal. Las librerías que utilizó fueron las que ofrece Python, OpenCV y TensorFlow.

Otro trabajo similar sería el realizado por Madelaine Chóez y César Palma [8] que diseñaron un prototipo web que funciona como un detector de mascarilla mediante RTMP y visión artificial para un mejor control dentro del transporte público en tiempo de pandemia. Sin embargo, para nuestro proyecto se tiene contemplado la integración de software con hardware por lo que no se realizará un prototipo web.

Después de la investigación realizada, se decidió basarse en el proyecto realizado por Onur Bölükbaş [1], que empleó TensorFlow para el aprendizaje automático de su sistema. Sin embargo, consideramos utilizar otro modelo para realizar el entrenamiento, en vez del Faster-RCNN-Inception-V2-COCO que fue el modelo que empleó Bölükbaş para realizar 10,000 épocas que le demoró un total de 4 horas el entrenamiento.

3. Diseño

Para el desarrollo del proyecto se contemplaron componentes de software y hardware.

3.1 Software

Se utilizaron las siguientes plataformas, herramientas y librerías:

- Anaconda3: Para la creación de entornos virtuales en Python.
- Google Colab: Para ejecutar el entrenamiento del modelo en un entorno de ejecución en la nube.
- Arduino IDE: Para la programación de la placa de desarrollo ESP32.
- Librerías:
  - TensorFlow: Para el aprendizaje automático.
  - OpenCV: Para la implementación de la visión artificial.
  - Numpy: Librería de Python para realizar los cálculos y acceder a funciones matemáticas.
  - OS: Para el acceso a funciones básicas del sistema operativo.
  - CUDA: Para la computación en las unidades de procesamiento gráfico (GPU) de NVIDIA.
  - cuDNN: Para trabajar con redes neuronales profundas.

3.2 Hardware

Se utilizaron los siguientes elementos de hardware:

- ESP32: Placa de desarrollo utilizado para la programación del servidor web.
- Cámara OV2640: Elemento compatible con la board esp32 para la detección en tiempo real.
- Webcam: Dispositivo externo de obtención de imagen para la detención en tiempo real.

Figura 2. Fotografía del ESP32 + Cámara OV2640. Fuente: Los autores.

4. Procedimiento Metodológico

4.1 Paso 1 – Elaboración y recopilación del dataset

En el primer paso de nuestro proyecto se incluye la elaboración y recopilación del conjunto de datos que usaremos para el entrenamiento de la red neuronal con nuestro modelo. Estos fueron obtenidos de la plataforma Kaggle que es una comunidad en línea de científicos de datos y profesionales del aprendizaje automático.

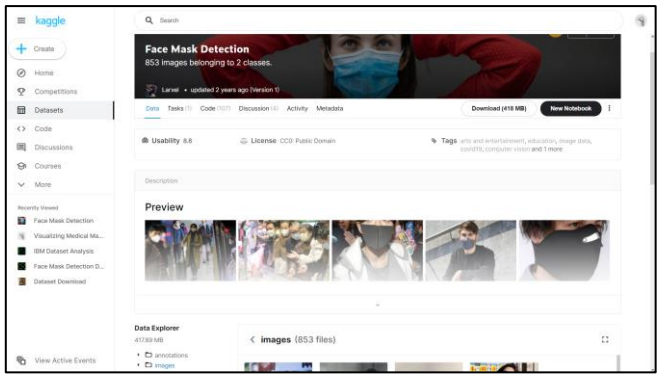

Figura 3. Dataset para la detección de mascarilla de la plataforma Kaggle. Fuente: Los autores.

4.2 Paso 2 – Anotaciones de las imágenes del dataset

El objetivo del segundo paso de nuestro proyecto es obtener las anotaciones correspondientes a cada imagen del dataset utilizado para luego ser cargadas en el entorno de entrenamiento.

4.3 Paso 3 – Entrenamiento en Google Colab

En el tercer paso, nos centramos en el entrenamiento para esto utilizaremos como modelo pre-entrenado ssd_mobilnet_v1 y Google Colab PRO como entorno de entrenamiento en el cual se realizaron 5 mil épocas con un aproximado de 7 horas.

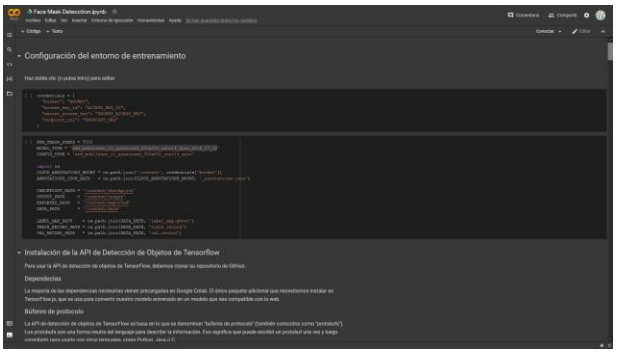

Figura 4. Código del entrenamiento en Google Colab. Fuente: Los autores.

4.4 Paso 4 – Prueba en video, webcam e imagen

Una vez finalizado el entrenamiento en Google Colab, esto nos permitirá exportar archivos fundamentales, como el modelo congelado y el archivo de nombre de clases que usaremos para testear nuestro modelo con Python.

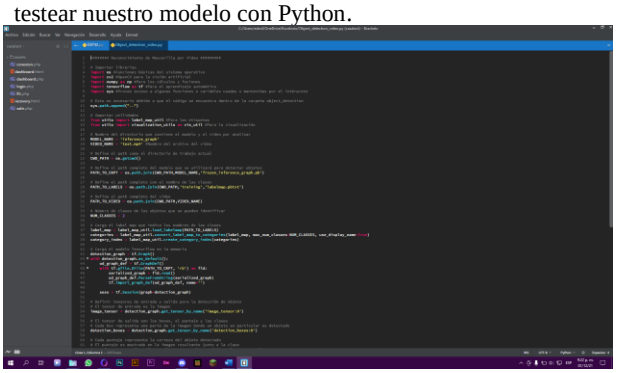

Figura 5. Código para testear el modelo utilizando video en python. Fuente: Los autores.

4.5 Paso 5 – Prueba utilizando el ESP32-CAM

En este paso estaremos utilizando la placa de desarrollo ESP32 y la cámara OV2640 compatible con nuestra board. Lo siguiente es programar el ESP32 con el IDE de Arduino para que funcione como servidor y cámara IP, para trasmitir fotogramas a través de una dirección URL para luego ser decodificada en Python utilizando la librería urllib para la detección de los objetivos con OpenCV.

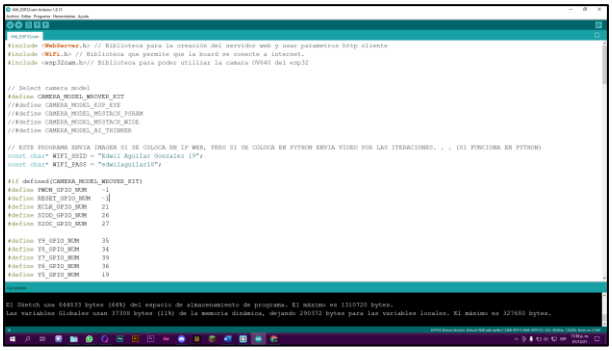

Figura 6. Código para programar el servidor web en el ESP32 con el IDE de Arduino. Fuente: Los autores.

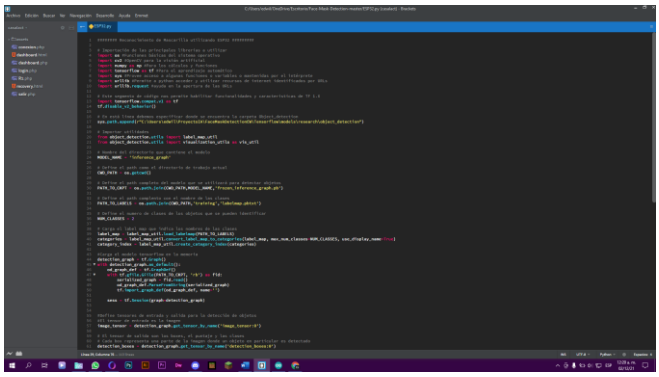

Figura 7. Código para testear el modelo generado decodificando los fotogramas enviados por el ESP32 a través de una URL. Fuente: Los autores.

5. Resultados

Para el desarrollo del proyecto se contemplaron componentes de software y hardware.

5.1 Archivo de las clases, modelo y gráfico congelado

Durante el proceso de entrenamiento en el entorno de Google Colab, generará varios archivos, los más importante serán los archivos que exportaremos. El labelmap.pbtxt archivo de mapa de etiquetas contiene el nombre de la clase de los objetos que se está entrenando para reconocer. Al mismo tiempo, se genera un archivo saved_model.pb. Este archivo contiene el gráfico completo de todos los pesos de entrenamiento y que el modelo puede utilizar para continuar entrenando y no está serializado. Además, exportaremos nuestro frozen_inference_graph.pb, que es un gráfico congelado que no se puede entrenar y está serializado.

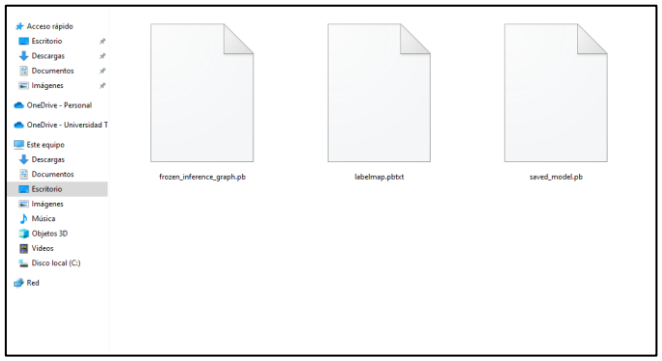

Figura 8. Archivos generados por el entorno de entrenamiento de Google Colab para la probar nuestro modelo en python. Fuente: Los autores.

5.2 Detección con imagen

Para la detección con imagen se ejecuta el módulo para ello. Solo se puede analizar una imagen por cada ejecución por lo es necesario modificar en el código el nombre del archivo de la imagen a detectar. Los boxes verdes indica que la persona lleva mascarilla y los boxes amarillos indican que no porta una mascarilla. Realizó sin problemas y con un buen porcentaje de certeza la detección de las imágenes. Cabe resaltar que se puede realizar la detección de mascarillas de múltiples objetivos.

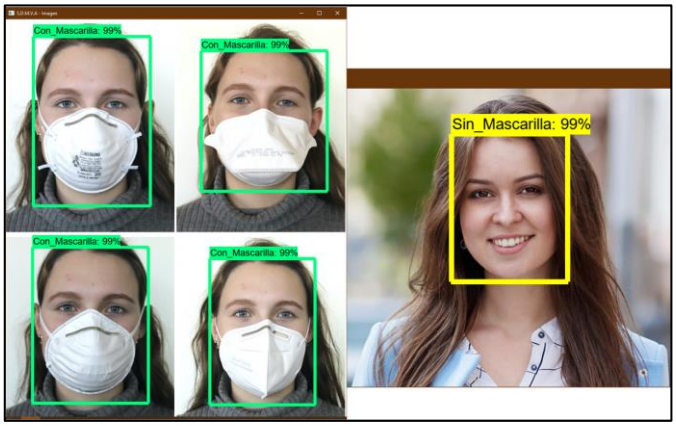

Figura 9. Resultados obtenidos de la detección de imagen. En la parte izquierda se observa que detecta las mascarillas con un alto porcentaje de certeza, lo mismo ocurre en la parte derecha a diferencia de que es sin mascarilla. Fuente: Los autores

5.3 Detección con video

Para la detección con video, el módulo que se ejecuta carga el video y realiza la detección poco a poco, por lo que puede demorar el proceso. Cuando el video termina, se finaliza la ejecución del programa y se cierra la ventana de visualización. Como se observa en la Figura 6, el programa reconoce con buena precisión cuando se quita la mascarilla y cuando se tiene puesta.

Figura 10. Fragmento del video resultante con la detección de mascarilla.

5.4 Detección en tiempo real

En la detección con la webcam, no es necesario proporcionarle algún archivo para que empiece la detección. Lo importante es contar con una webcam que sea reconocible. Como se puede observar en la Figura 11, el sistema pudo detectar con éxito cuando se lleva puesta la mascarilla y cuando no. Cabe resaltar que cuando se realiza la detección en tiempo

real, puede fallar en distinguir si se lleva la mascarilla o no debido a que la precisión disminuye y se requiere de bastantes recursos computacionales para el procesamiento.

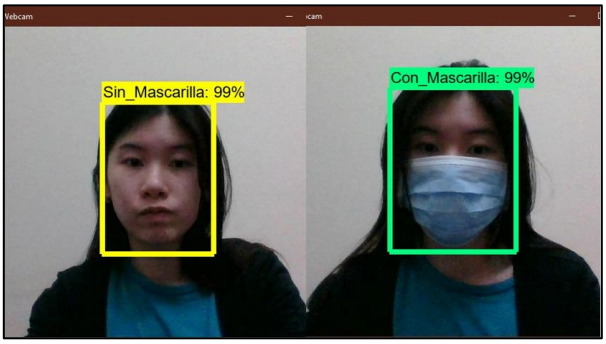

Figura 11. Captura del resultado en tiempo real utilizando la webcam.

5.5 Detección con el ESP32

Para poder usar el ESP32 para la detección de objetos debemos de programar con anticipación y utilizando las librerías necesarias para nuestro sistema, y así poder levantar nuestro servidor web para enviar los frames capturados a nuestro cliente. En este caso, nuestro código en Python será el responsable de la decodificación de la información suministrada por el ESP32. La detección por este ultimo se da de forma similar a la detección por webcam. Además, dado que se puede conectar a la red WIFI y utilizar varias fuentes de alimentación externa, el uso del dispositivo no depende de la conexión con el cliente (PC) esto da la posibilidad de colocar en diferentes puntos estratégicos para realizar la detección permitiendo enviar la información de manera remota al cliente.

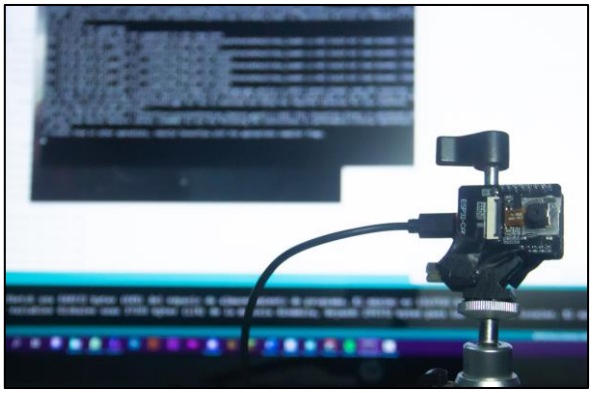

Figura 12. Fotografía del ESP32 en funcionamiento. Fuente: Los autores.

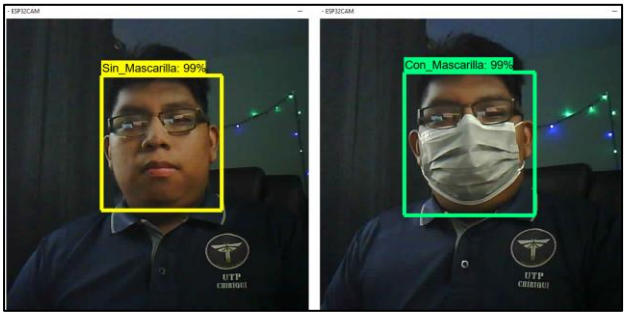

Figura 13. Captura de resultado en tiempo real utilizando el ESP32. Fuente: Los autores.

6. Conclusiones

Nuestro proyecto del sistema de detección de mascarilla mediante visión artificial permite a los usuarios analizar en tiempo real o mediante imágenes y videos, si las personas portan su mascarilla o no con un buen grado de certeza, demostrando que el entrenamiento de 5,000 épocas dio su fruto. Además, permite integrarlo a un dispositivo como se demostró con el ESP32. No obstante, a pesar de que se realizó el entrenamiento con una alta cantidad de épocas y un dataset de 853 imágenes, el sistema aún tiene sus fallas. También, se recomienda utilizar Google Colaboratory u otro servicio que permita utilizar recursos computacionales de la nube para ejecutar programas. Para trabajos futuros se recomienda que se actualice la versión de Python y las librerías, que en este trabajo se utilizó Python 3.6 y las versiones de las librerías que fueran compatibles con ello. Este es un aspecto muy importante porque hay muchas funciones de librerías que se vuelven obsoletas con el tiempo.

Algo que se podría implementarse que no se pudo en este proyecto por falta de tiempo, sería la detección de varias imágenes en una sola ejecución y que se guarden los resultados en un directorio. Una de las ventajas que proporciona este sistema es su adaptabilidad debido a que permite que el usuario cree su propio dataset con IBM Cloud Annotations con la cantidad de clases que desee y por ello también permite aplicarse a un sinnúmero de aplicaciones que para este proyecto se podría mencionar el control de las medidas sanitarias como la principal aplicación.

Sistemas como el realizado en este trabajo facilitaría el trabajo en muchos sectores automatizando los procesos con la inteligencia artificial, por lo que se debe aprovechar de las tecnologías con los que se cuentan para obtener el mayor beneficio posible.

REFERENCIAS

[1] Bölükbaş, O., 2021. GitHub - onurbolukbas/Face-Mask-Detection: Face Mask😊 Detection using TensorFlow. [online] GitHub. Available at:

<https://github.com/onurbolukbas/Face-Mask-Detection> [Accessed 15 November 2021].

[2] GitHub. 2021. GitHub - tensorflow/models at r1.13.0. [online] Available at: <https://github.com/tensorflow/models/tree/r1.13.0> [Accessed 22 November 2021].

[3] I. García and V. Caranqui, "La visión artificial y los campos de aplicación", Tierra Infinita, vol. 1, no. 1, 2015. Available: https://revistasdigitales.upec.edu.ec/index.php/tierrainfinita/article/view/76. [Accessed 23 September 2021].

[4] Bourdakos, N., 2021. GitHub - cloud-annotations/google-colab-training: A notebook for training an object detection model. [online] GitHub. Available at: <https://github.com/cloud-annotations/google-colab-training> [Accessed 21 November 2021].

[5] A. González et al., Técnicas y Algoritmos Básicos de Visión Artificial. España: Universidad de La Rioja, 2006.

[6] "TensorFlow", TensorFlow, 2021. [Online]. Available: https://www.tensorflow.org/. [Accessed: 05- Oct- 2021].

[7] M. Ordoñez, "Implementación de un sistema de reconocimiento del uso de mascarillas como medida de precaución contra el Covid19 usando deep learning", Universidad Técnia de Machala, 2021. Available: http://repositorio.utmachala.edu.ec/bitstream/48000/15890/1/TTFIC-2020-IS-DE00009.pdf. [Accessed 23 September 2021].

[8] M. Chóez and C. Palma, "Prototipo Web detector de mascarilla mediante RTMP y visión artificial para el control dentro del transporte público del norte de Guayaquil en tiempo de pandemia", Universidad de Guayaquil, 2020. Available: http://repositorio.ug.edu.ec/handle/redug/52664. [Accessed 23 November 2021].