ACCESO AUTOMÁTICO POR IDENTIFICACIÓN DE USO ADECUADO DE CUBREBOCAS

AUTOMATIC ACCESS BY IDENTIFICATION OF PROPER USE OF FACE MASKS

Braulio González Esquivel
Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Cuernavaca, México
a01425274@tec.mx

Brian Manuel González Contreras
Universidad Autónoma de Tlaxcala, México
brianmanuel.gonzalez@uatx.mx

Recepción: 24/noviembre/2023 **Aceptación:** 26/abril/2024

Resumen

En el contexto de la recién finalizada pandemia por coronavirus, el uso de los cubrebocas para evitar o disminuir contagios, inclusive de otras enfermedades, sigue siendo de interés en muchas zonas de México y el mundo. Garantizar que las personas lleven cubrebocas para el acceso a ciertos recintos no es posible con personal de vigilancia; en su lugar, los sistemas automáticos son una mejor opción para ayudar a gestionar el comportamiento del público. En este trabajo se propone un control de acceso por reconocimiento facial que integra software y hardware. El primero usando redes neuronales convolucionales desarrolladas en la plataforma Keros-TensorFlow que permiten análisis en tiempo real de imágenes de video provenientes de cámaras de vigilancia. El segundo se realiza con una tarjeta Arduino integrando sensores/actuadores. Las pruebas realizadas muestran efectividad de la propuesta lo que permitiría que el sistema pueda usarse en diferentes instalaciones o aplicaciones en tiempo real.

Palabras Clave: Aprendizaje profundo, Automatización, Detección de cubrebocas, Red neuronal convolucional.

Abstract

In the context of the recently ended coronavirus pandemic, the use of face masks to prevent or reduce transmission, including of other diseases, continues to be of

interest in many areas of Mexico and the world. Ensuring that people wear face masks for access to certain venues is not possible with surveillance personnel; instead, automated systems are a better option to help manage public behavior. This paper proposes an access control scheme that integrates software and hardware. The first using convolutional neural networks developed on the Keros-TensorFlow platform that allow real-time analysis of video images from surveillance cameras. The second is performed with an Arduino board integrating actuators/sensors. The tests performed show the effectiveness of the proposal, which would allow the system to be used in different installations or real-time applications.

Keywords**:** *Automation, Convolutional neural network, Deep learning, Face mask detection.*

1. Introducción

Llevar un cubreboca (o mascarilla facial) que cubra la nariz y la boca en un entorno público, así como lavarse las manos con frecuencia o utilizar desinfectantes con un 70% de alcohol como mínimo, es una forma de evitar la exposición a virus [Sánchez-Guevara, 2022], [Leung, 2020]. La detección de rostros con cubrebocas se ha convertido en una aplicación necesaria a raíz de la pandemia de Covid-19, además de mantener el distanciamiento social y utilizar desinfectantes para lavarse las manos. Aunque hasta ahora se han abordado otros problemas de distanciamiento social y desinfección, la cuestión de la detección de cubreboca bien colocados sigue siendo de interés para ayudar en la prevención de enfermedades contagiosas [Leung, 2020], [Umer, 2023].

Los cubrebocas siguen siendo obligatorios para las personas en distintos países. En México, aún hay lugares en donde no se permite la entrada sin ellos, como es el caso de oficinas públicas y hospitales [Sánchez Guevara, 2022]. Sin embargo, debido al gran número de personas que acuden a esos entornos, así como lugares públicos como aeropuertos, estaciones de autobuses y centros comerciales, la inspección manual es muy difícil [Zakariyya, 2023]. La investigación relacionada con la detección e identificación automática de cubrebocas sigue teniendo relevancia, pues puede ayudar en aplicaciones de control y vigilancia durante temporadas de

incidencia viral y brotes sospechosos [Alturki, 2022], [Oliveira-Teixeira, 2021]. Inclusive, a las personas con temperatura corporal elevada no se les permite la entrada en lugares públicos porque corren un alto riesgo de infectarse y propagar el virus, aunado a ello portar cubrebocas es esencial [Leung, 2020]. Por lo anterior, en las entradas de lugares de trabajo, centros comerciales o de hospitales, también aún son solicitados los controles de temperatura, además del cubreboca. Lo anterior motiva el desarrollo de un sistema de control de acceso con un cierto nivel de automatización. Tales sistemas tienen que ser incorporados dentro de esquemas de domótica contemporáneos [Zakariyya, 2023], así como de monitoreo para prevención y reacción rápida ante posibles brotes contagiosos futuros [Alturki, 2022].

En este trabajo se propone un sistema de control de acceso de personas a recintos empleando detección de cubrebocas (incluida su mala colocación) por medio de un algoritmo de aprendizaje profundo para la detección y clasificación de rostros basado en redes neuronales convolucionales. Para la parte de acceso, se utiliza una tarjeta Arduino por medio de la cual: se detecta la presencia de personas que intentan ingresar, se mide la temperatura de la persona, se permite o no el acceso mediante la apertura/cierre de puerta empleando un relé, se realiza la visualización y sonorización indicativa de lo que ocurre.

Revisión bibliográfica

La combinación de las técnicas de *Aprendizaje máquina* o *automático* y *Visión por computadora* permiten obtener esquemas de detección rápidos y precisos, aprovechando los avances del aprendizaje profundo, específicamente, de las redes neuronales profundas [Naseri, 2023], [Umer, 2023]. Las redes neuronales (artificiales) reciben señales de entrada y las procesan a través de sumas de estas, multiplicadas por sus pesos asociados para determinar el “impulso nervioso” que recibe la neurona [Chollet, 2022]. Este valor, se procesa mediante una función de activación que devuelve un valor que se envía como salida de la neurona. Una red neuronal artificial está compuesta por neuronas agrupadas en capas. La primera capa, llamada capa de entrada, recibe los datos reales que alimentan a la red. La

última capa, llamada capa de salida, produce el resultado visible de la red. Las capas intermedias, llamadas capas ocultas, procesan los datos entre la entrada y la salida [Chollet, 2022]. El aprendizaje profundo se refiere al uso de muchas capas ocultas en una red neuronal [Goodfellow, 2016].

Una red neuronal convolucional (CNN, por sus siglas en inglés) es una forma de red neuronal artificial que se construye específicamente para interpretar la entrada de píxeles y se utiliza principalmente para el reconocimiento y análisis de imágenes [Goodfellow, 2016], en el que cada capa se aplica a un conjunto diferente de filtros. Alrededor de 100 a 1000 filtros se combinan para dar un resultado final y luego la salida obtenida se envía a la siguiente capa de esta red neuronal [Chollet, 2022]. En la clasificación basada en CNNs, los modelos detectores de rostros aprenden directamente de los datos del usuario y luego aplican los algoritmos de aprendizaje profundo sobre ellos. La red toma como entrada los pixeles de una imagen, entonces si se tiene una imagen de 170x170 pixeles (alto x ancho), eso equivale a 28 900 neuronas en un color (escala de grises). Con color se requieren 3 canales, por lo que la red debería ser de 86 700 neuronas en la entrada. La convolución implica una combinación matemática de dos funciones para obtener la tercera función. Funciona mediante un mecanismo de ventana deslizante que ayuda a extraer características de una imagen. Esto ayuda a generar mapas de características durante su entrenamiento, en donde a través de la retropropagación ajusta los pesos dentro de las funciones de activación [Goodfellow, 2016].

Por otro lado, la visión por computadora es el área de la ciencia que desarrolla teorías y métodos enfocados a la extracción automática de información útil contenida en imágenes [Militante, 2020].

En este sentido, dentro de la plataforma de programación de Pyhton se cuenta con lo que hace cada una de las siguientes librerías [Sandler, 2018], [Umer, 2023]:

- *TensorFlow*: es una plataforma de aprendizaje automático que se utiliza para el aprendizaje profundo.
- *Keras*: es una API (interfaz de programación de la aplicación, por sus siglas en inglés) de redes neuronales que se utiliza para la construcción del modelo de aprendizaje profundo.

- *OpenCV* (Open Computer Vision Library): es una librería de visión por computadora que se utiliza para la detección de caras y la detección de objetos en imágenes.
- *MobileNetV2*: es una arquitectura de CNN que se utiliza para la detección y clasificación de objetos en imágenes, de poco consumo de recursos computacionales.

El proceso de detección de cubrebocas incluye dos tareas: identificación y clasificación [Militante, 2020]. En la primera tarea se detectan las características faciales y se ignora el resto de la imagen, es decir, se identifican los rostros. A continuación, en la segunda tarea, se trata de decidir si llevan cubrebocas o no. La primera tarea es un área de investigación ampliamente explorada en el campo de la visión por computadora, ya que la comunidad investigadora ha desarrollado muchas técnicas de detección de rostros [Naseri, 2023]. La segunda tarea es más compleja pues requiere establecer un vínculo entre el aprendizaje sobre la detección y la clasificación para discernir si se porta o no el cubrebocas.

Para complementar aplicaciones prácticas y de interés general, se han establecido esquemas de acceso que normalmente son propios de la domótica. Sin embargo, para aplicaciones especiales, como las de acceso a entornos por motivos de higiene, la incorporación de reconocimiento facial-cubrebocas representa un nuevo reto [Zakariyya, 2023], [Umer, 2023]. El estudio presentado en [Alturki, 2022] muestra comparativos entre diferentes técnicas para detectar cubrebocas, sin embargo, solo se enfoca a la parte del uso de la inteligencia artificial para ese propósito sin hablar acerca de su incorporación en sistemas automáticos. Similarmente ocurre con el trabajo presentado en [Azouji, 2022]. En [Banik, 2023] se muestra un esquema de detección de cubrebocas similar al presentado en este proyecto, pero no se indican aplicaciones directas como es el caso de este trabajo.

Propuesta conceptual

La detección de cubrebocas en este trabajo de investigación se apoya en el trabajo de [Rosebrock, 2020], en donde básicamente se contemplan cuatro

elementos principales para realizar dicha acción: Keras, TensorFlow, OpenCV y MobileNetV2, que más adelante se describirán. Otros trabajos en este sentido se han desarrollado en [Oliveira-Teixeira, 2021] y [Miranda-Ramos, 2022], en donde se utilizan otros algoritmos de redes neuronales para la clasificación de imágenes. Sistemas similares para reconocimiento facial se han presentado en [Vega-Luna, 2020]. Sin embargo, la aportación de este trabajo radica en incorporar la detección de cubrebocas (colocado de manera correcta cubriendo boca y nariz) para automatizar el acceso a algún recinto de interés. La figura 1 da una idea de una posible aplicación de esta propuesta. En ella se consideran los elementos principales básico de la aplicación, en donde se cuenta con un sensor de presencia para que se inicie el proceso de detección de cubrebocas a través del registro en tiempo real de video para la toma de imágenes.

Fuente: elaboración propia.

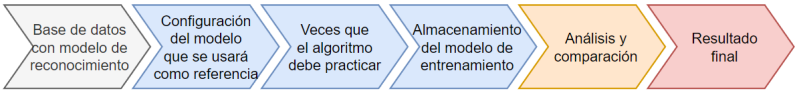
Figura 1 Descripción introductoria de la propuesta.

El control de acceso de este trabajo toma en cuenta la siguiente secuencia de funcionamiento: el sensor PIR (infrarrojo pasivo, por sus siglas en inglés) detecta presencia de una persona dentro del área de visión de la cámara y comienza un conteo de 8 s, entonces se enciende un LED (diodo emisor de luz, por sus siglas en inglés) rojo y se presenta un conteo de 5 s en el visualizador; el algoritmo de detección de cubrebocas determina si se porta o no. En caso de portarlo de forma adecuada, al finalizar esos 8 s se enciende el LED amarillo (se apaga el rojo) durante 5 s indicando que ya está por terminar el turno, al finalizar ese tiempo se

enciende el LED verde (y se apaga el amarillo) durante 5 s indicando que puede pasar otra persona, en ese momento se abre la puerta de acceso durante esos mismos 5 s; después se reinicia el proceso. El proceso se repite para la persona que sigue. En caso de no portar el cubrebocas de manera adecuada o no se porte, se le solicita a la persona (a través de un texto en el visualizador) que se retire y vuelva después portando el cubrebocas, o bien, a que lo porte adecuadamente, asimismo el LED permanecerá en rojo, sonará el zumbador y la puerta no se abre. Durante el proceso anterior, en paralelo, se mide a distancia la temperatura corporal de la persona. Si esta es inferior a los 37 °C, el proceso previo sigue su curso de la manera indicada. En caso contrario, se emite un sonido y se enciende el LED rojo a manera de alarma audiovisual, que indicará que la persona está con temperatura alta, por lo tanto, no le será posible acceder. El proceso descrito para cada persona lleva un tiempo total de 18 s. Este proceso es básico y general en el contexto de este proyecto, sin embargo, puede modificarse y mejorarse en función de las necesidades del entorno a donde se desee acceder. El proceso consta de tres etapas: detección de presencia de persona, reconocimiento facial con portación correcta/incorrecta del cubrebocas y el control de acceso.

2. Métodos

En esta sección se describen los pasos seguidos para desarrollar este trabajo, la recopilación de datos y el uso de los resultados de entrenamiento. El flujo del método propuesto se muestra en la figura 2.



Fuente: elaboración propia.

Figura 2 Pasos seguidos para el desarrollo de la detección de cubreboca.

Tras la recogida de datos que contienen grupos con máscara y sin máscara, las imágenes se preprocesan utilizando cuatro etapas de procesamiento de imágenes (se definen su altura y ancho predeterminados, de modo que todas tengan el mismo tamaño en pixeles, eliminar el ruido de la imagen obtenida en el proceso de captura,

aplicar filtros de suavizado y resaltado). Como las variaciones en la complexión y los efectos de iluminación hacen que la detección de máscaras faciales sea una tarea más difícil, se utilizan pasos de procesamiento de imágenes para normalizar las imágenes de entrada. Finalmente, se emplea un modelo de aprendizaje profundo personalizado para detectar si hay o no un cubreboca en los rostros de las imágenes de prueba.

Elementos utilizados

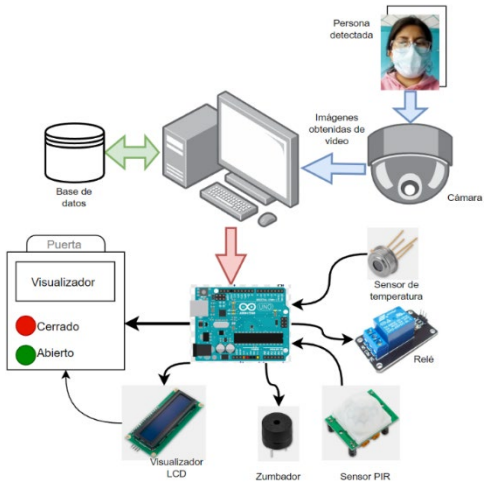
TensorFlow es una biblioteca de software de aprendizaje automático de código abierto y gratuita. Puede realizar grandes cálculos numéricos sin tener en cuenta el aprendizaje profundo, sin embargo, puede utilizarse para una variedad de actividades, destacando en la inferencia y el entrenamiento de redes neuronales profundas. Es compatible con el aprendizaje automático tradicional, así como con TensorFlow de Google lo que permite el cálculo numérico rápido al incorporarse como una biblioteca de Python. Puede simplificar el proceso de uso y desarrollo de otras bibliotecas diseñadas para redes neuronales. En este proyecto esto se aprovecha para poder utilizar MobileNetV2.

MobileNetV2 es una arquitectura de CNN que busca un buen rendimiento en dispositivos móviles. Se basa en una estructura residual invertida en la que las conexiones residuales se encuentran entre las capas cuello de botella [Sandler, 2018], [Umer, 2023]. La capa de expansión intermedia utiliza convoluciones ligeras en profundidad para filtrar características como fuente de no linealidad. En conjunto, la arquitectura de MobileNetV2 contiene la capa inicial de convolución completa con 32 filtros, seguida de 19 capas residuales de cuello de botella. Este módulo toma como entrada una representación comprimida de baja dimensión que primero se expande a alta dimensión y se filtra con una convolución ligera en profundidad. Posteriormente, las características se proyectan de nuevo a una representación de baja dimensión con una convolución lineal.

Para desarrollar materialmente este proyecto se consideró el uso de una tarjeta de desarrollo Arduino UNO con la cual se programa el sistema de control de acceso para personas en fila con un intervalo de tiempo de 18 s por persona (incluyendo

registro de número de personas accediendo al recinto), junto con este microcontrolador se utilizó un visualizador en el cual se va mostrando el tiempo que transcurre, así como la temperatura, para la detección de presencia se ocupó un sensor PIR, mientras que para la toma de temperatura se usa el sensor MLX90614. Los equipos materiales utilizados (Figura 2) en este trabajo son:

- Tarjeta Arduino UNO.
- Cámara de vigilancia de resolución selectiva 1080 p a 30 fps, 720 p a 60 fps y 640x480p 60/90. En este caso se puede configurar en la mínima resolución.
- Sensor infrarrojo MLX90614, que actúa como un lector de temperatura por infrarrojos que realiza la medición sin entrar en contacto con los objetos.



Fuente: elaboración propia.

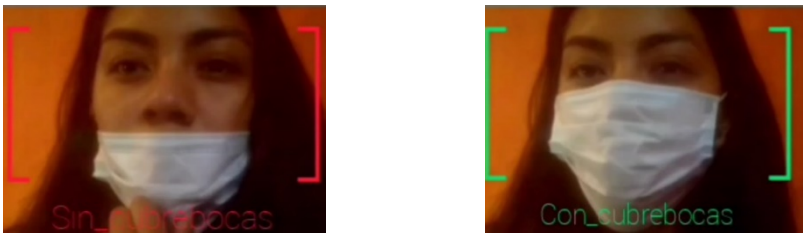
Figura 3 Elementos utilizados en la integración de la propuesta.

- Sensor PIR modelo HC-SR, que permite detectar el movimiento de una persona basándose en la radiación infrarroja que emite en un radio de 6 m.
- Cerradura electromagnética de doble acción.
- Módulo de relé para la placa controladora para interconectar circuitos o módulos eléctricos externos de hasta 5 A.

Integración de elementos

La incorporación de todos los elementos se da de acuerdo con el esquema general que se muestra en la figura 3 y con la siguiente descripción:

- Preparación de la detección de cubrebocas: se realiza en la computadora asumiendo que la CNN ha sido entrenada, entonces solo se corre el programa que se encarga de la detección. En cámara sólo se presentan los datos de “con cubrebocas” o “sin cubrebocas” (Figura 4). Internamente se envía el dato de 0 o 1 para fines del control de acceso.



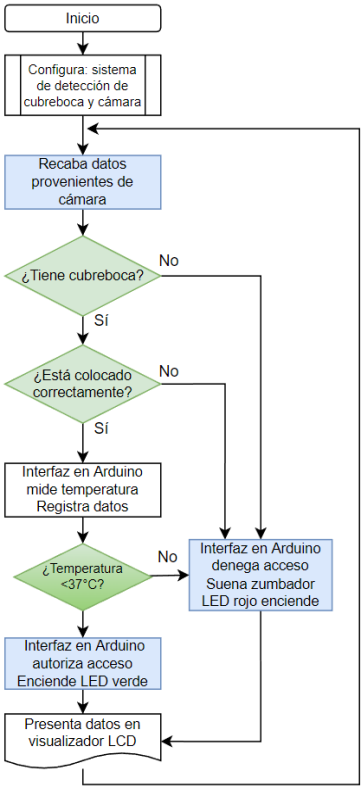
a) Cubrebocas mal colocado o no colocado. *b) Cubrebocas bien colocado.*

Fuente: elaboración propia.

Figura 4 Detección de cubrebocas y mensaje en monitor.

- La placa Arduino (en este caso el modelo UNO) se conecta a través del puerto serie de la computadora para que se esté enviando la información correspondiente entre placa y computadora. La computadora se encarga de procesar el algoritmo de reconocimiento de cubrebocas y ese valor se procesa en la programación del Arduino para que se puedan ejecutar las acciones de control de acceso.
- La placa Arduino genera los datos de apertura-cierre de puerta, indicadores visuales de paso tipo semáforo, zumbador de alarma, detección de presencia, medición de temperatura.
- Los elementos auxiliares indicados en el punto anterior se pueden adaptar de acuerdo con las necesidades especiales de cada entorno, por ello se programaron de manera tal que pueden ser reemplazables.
- Pantalla de cristal líquido (LCD, por sus siglas en inglés), que permite la visualización de mensajes cortos o dibujos sencillos a través de la interfaz I2C (circuito inter-integrado, por sus siglas en inglés). Presenta los datos de temperatura para que los observe el usuario. Esta información también se envía hacia la computadora (vía la interfaz del mismo Arduino).

- El esquema básico considerando la detección de cubrebocas y que desencadena la funcionalidad descrita se muestra en el diagrama de flujo de la figura 5.



Fuente: elaboración propia.

Figura 5 Diagrama de flujo funcional del control de acceso.

3. Resultados

El entrenamiento se ha llevado a cabo en una computadora con sistema operativo Windows 10 de 64 bits, con una CPU Intel Core-i5 8265U a 1.60 GHz y 16 GB de RAM. Como lenguaje de desarrollo de la aplicación se ha utilizado Python 3.7. El modelo fue desarrollado y entrenado utilizando *Keras* como *backend* y la plataforma *Tensor-Flow*. Para generar el conjunto de datos de entrada del modelo detector de cubrebocas y afinación del modelo de reconocimiento, se utiliza *MobileNetV2* a través del script Python de entrenamiento como lo muestra la figura 6a. Es importante indicar que el entrenamiento ya se ha realizado según lo presentado en [Rosebrock, 2020], utilizando la base de datos propuesta, es decir, preparar la base

de conocimiento para la detección en tiempo real. Esto se muestra en la figura 6b. El entrenamiento utiliza 1 376 imágenes: 690 con cubrebocas (bien colocado) y 686 sin cubrebocas (o mal colocado).

```
[2]: from tensorflow.keras.preprocessing.image import ImageDataGenerator
     from tensorflow.keras.applications import MobileNetV2
     from tensorflow.keras.layers import AveragePooling2D
     from tensorflow.keras.layers import Dropout
     from tensorflow.keras.layers import Flatten
     from tensorflow.keras.layers import Dense
     from tensorflow.keras.layers import Input
     from tensorflow.keras.models import Model
     from tensorflow.keras.optimizers import Adam
     from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
     from tensorflow.keras.preprocessing.image import img_to_array
     from tensorflow.keras.preprocessing.image import load_img
     from tensorflow.keras.utils import to_categorical
     from sklearn.preprocessing import LabelBinarizer
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import classification_report
     from imutils import paths
     import matplotlib.pyplot as plt
     import numpy as np
     import argparse
     import os
```

```
[6]: INIT_LR = 1e-4
     EPOCHS = 2
     BS = 32
     print("[INFO] loading images...")
     imagePaths = list(paths.list_images('/Users/Difha/OneDrive/Documentos/red neuro
     data = []
     labels = []
     # loop over the image paths
     for imagePath in imagePaths:
         # extract the class label from the filename
         label = imagePath.split(os.path.sep)[-2]
         # load the input image (224x224) and preprocess it
         image = load_img(imagePath, target_size=(224, 224))
         image = img_to_array(image)
         image = preprocess_input(image)
         # update the data and labels lists, respectively
         data.append(image)
         labels.append(label)
     # convert the data and labels to NumPy arrays
     data = np.array(data, dtype="float32")
     labels = np.array(labels)
     print(labels)

     [INFO] loading images...
     ['con_mascara' 'con_mascara' 'con_mascara' ... 'sin_mascara' 'sin_mascara'
      'sin_mascara']
```

a) Librerías para detección de cubreboca. *b) Entrenamiento empleando datos predefinidos.*

Fuente: elaboración propia.

Figura 6 Carga de librerías en Python.

El entrenamiento se realiza a través de imágenes predefinidas que se encuentran disponibles para ese fin. De hecho, los archivos de aprendizaje pueden obtenerse sin necesidad de entrenar el algoritmo. Sin embargo, puede realizarse un entrenamiento utilizando imágenes propias. Esto último no se realizó en virtud de que no se cuenta con una cantidad suficiente de imágenes que permitan un entrenamiento efectivo. Utilizando la base de datos de imágenes sugeridas en [Rosebrock, 2020] y [Varshini, 2021], con imágenes de personas con cubrebocas y sin cubrebocas, el algoritmo aprende a detectar esas diferencias a través de un proceso de aprendizaje supervisado, ya que de antemano se conoce qué imágenes llevan cubrebocas y cuáles no. Posteriormente, el algoritmo será capaz de detectar si una persona tiene o no el cubrebocas colocado, basándose en la generalización del aprendizaje adquirido, clasificando así las nuevas imágenes que nunca han sido vistas anteriormente.

La resolución de las imágenes debe ser de al menos 600x600 pixeles que, con la cámara indicada previamente, se logran sin problema, siempre y cuando se encuentren en el área de visión. El algoritmo funciona adecuadamente, detectando

la portación de cubrebocas de personas que no fueron consideradas en la etapa de entrenamiento, esto se presenta en la figura 7.

Fuente: elaboración propia.

Figura 7 Detección de cubrebocas: mal colocado, bien colocado, ausencia.

El algoritmo está elaborado utilizando herramientas de aprendizaje automático dentro de las librerías de visión por computadora propias de OpenCV. El objetivo es que la aplicación que contiene este algoritmo pueda detectar si una persona porta o no el cubrebocas (incluso si está mal colocado). Los pasos detallados para implementar el algoritmo de detección de cubrebocas son los siguientes:

- Descargar e instalar Python y OpenCV.
- Descargar el conjunto de datos de entrenamiento de reconocimiento facial.
- Entrenar un modelo de reconocimiento facial utilizando el conjunto de datos descargado.
- Crear un detector de caras utilizando OpenCV.
- Crear un detector de cubrebocas utilizando OpenCV.
- Combinar el detector de caras y el detector de cubreboca para determinar si una persona porta o no el cubrebocas, esto a través del clasificador MobileNetV2.

Parte de lo indicado previamente se muestra en figura 8a, implementación del código utilizando MobileNetV2. Otra manera de observar la funcionalidad de la detección del cubrebocas es a través de la generación de un historial de entrenamiento con curvas de precisión/pérdida, figura 8b. Lo que siempre se busca es disminuir el error (pérdida o *loss* en la gráfica) y maximizar la precisión durante las fases de entrenamiento y validación.

```
# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)
# partition the data into training and testing splits using 80% of
# the data for training and the remaining 20% for testing
(trainX, testX, trainY, testY) = train_test_split(data, labels,
    test_size=0.20, stratify=labels, random_state=42)
# construct the training image generator for data augmentation
aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")
    metrics=["accuracy"])
[INFO] compiling model...
# load the MobileNetV2 network, ensuring the head FC layer sets are
# left off
baseModel = MobileNetV2(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))
# construct the head of the model that will be placed on top of the
# the base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)
# place the head FC model on top of the base model (this will become
# the actual model we will train)
model = Model(inputs=baseModel.input, outputs=headModel)
# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
    layer.trainable = False
```

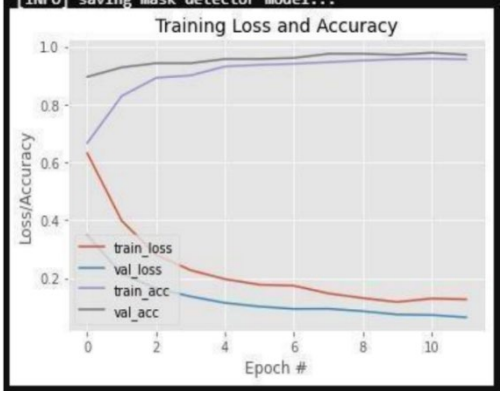
a) Código para el etiquetado y clasificación de imágenes.

```
[INFO] evaluating network...
              precision    recall  f1-score   support

con_cubrebocas       0.97      0.97      0.97       138
sin_cubrebocas       0.97      0.97      0.97       138

    accuracy                           0.97       276
   macro avg       0.97      0.97      0.97       276
weighted avg       0.97      0.97      0.97       276

[INFO] saving mask detector model...
```



b) Curvas de aprendizaje.

Fuente: elaboración propia.

Figura 8 Implementación y evaluación de la red neuronal.

Las cuatro curvas presentan una evolución por épocas: después de 10 épocas se logra un error de 10% para la validación y 18% para el entrenamiento, mientras que para precisión se logra un 97%. Estos valores se calculan con las métricas clásicas para este tipo de métodos [Umer, 2023], [Chollet, 2022], [Goodfellow, 2016]. Las métricas se definen en términos de cuatro términos básicos: verdadero positivo (VP), falso positivo (FP), verdadero negativo (VN) y falso negativo (FN). VP se refiere a la persona que porta cubrebocas y el modelo lo determina como "con cubrebocas", mientras que FP son aquellas personas que no llevan cubrebocas, pero se determina como "con cubrebocas". De manera similar, VN se refiere a las personas que no llevan cubrebocas y se determinan como "sin cubrebocas " y FN son las que llevan cubrebocas, pero se determinan como "sin cubrebocas". Las fórmulas correspondientes se muestran en las ecuaciones 1 a 4.

$$P = \frac{VP + VN}{VP + VN + FP + FN}, \tag{1}$$

$$E = \frac{VP}{VP + FP}, \tag{2}$$

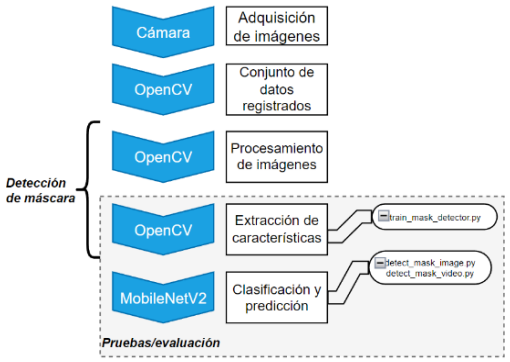
$$R = \frac{VP}{VP + FN}, \tag{3}$$

$$F_{1-score} = 2\frac{PR}{P+R}. \quad (4)$$

Donde P es la precisión, E es la exactitud, R es el recobro (exhaustividad) y $F_{1-score}$ es la métrica de rendimiento.

En resumen, las actividades realizadas para determinar la portación o no del cubreboca (Figura 9) son:

- Adquisición de imágenes: se necesitan imágenes de mediana calidad de la persona predominantemente de frente (80-100%) posando con y sin máscara facial, provenientes de la cámara de seguridad o de vigilancia.
- Colección de datos registrados: un conjunto de datos basado en el conocimiento se crea etiquetando adecuadamente de las imágenes capturadas con clases únicas.



Fuente: elaboración propia.

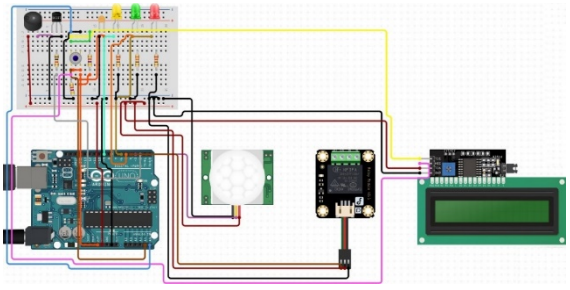
Figura 9 Pasos seguidos para el desarrollo de la detección de cubreboca.

- Procesamiento de imágenes: las imágenes obtenidas que se someterán a un paso de preprocesamiento se mejoran aún más con el fin de resaltar características específicas durante el procesamiento (cara y después cubrebocas). El proceso de segmentación divide las imágenes en varios segmentos y se utilizan en la extracción de las áreas cubiertas por el cubrebocas en la cara de la persona desde el fondo de la imagen.
- Extracción de características: esta sección incluye las capas convolutivas que obtienen las características de la imagen a partir de las imágenes redimensionadas y también se unen después de cada convolución. La

agrupación máxima y media de la extracción de características disminuye el tamaño. En última instancia, tanto las capas convolucionales como las de agrupamiento actúan como depuradores para generar características patrón de la imagen.

- Clasificación de las imágenes: es la parte final para entrenar modelos de aprendizaje profundo junto con las imágenes etiquetadas. El entrenamiento se enfoca en reconocer y clasificar imágenes de acuerdo con los patrones visuales aprendidos.

Una vez funcional el esquema de detección, este genera unos o ceros en función de la detección de cubrebocas o cubrebocas mal colocado/ausente, respectivamente, tal como se describió en la sección de “Integración de elementos”. Este valor se procesa en la programación del Arduino para que se puedan ejecutar las acciones que dependen de la portación del cubreboca. El diagrama general del circuito con las conexiones se muestra en la figura 10.



Fuente: elaboración propia.

Figura 10 Elementos conectados a la placa Arduino UNO.

4. Discusión

El tiempo propuesto para realizar el control de acceso descrito en la introducción (propuesta conceptual) es adecuado desde el punto de vista del tiempo de procesamiento de cada imagen para determinar si una persona porta o no el cubrebocas. Es importante indicar que esos lapsos pueden modificarse fácilmente para adaptarlos a las necesidades del entorno. Las pruebas realizadas para verificar la eficacia de esta propuesta muestran resultados satisfactorios asumiendo la

funcionalidad indicada, sin embargo, las bases de datos utilizadas para el entrenamiento son genéricas, por lo que sería interesante utilizar bases de datos endémicas para que la detección sea del 100%. En ese sentido, la base de datos empleada para el entrenamiento presenta imágenes *ad hoc* para que el entrenamiento sea exitoso, esto se discute a detalle en [Varshini, 2021].

5. Conclusiones

Este proyecto presentó una solución hardware y software para la detección de cubrebocas (o máscaras faciales) en tiempo real, incluye la implementación de un sistema de control de acceso a entornos de recepción de personas. El sistema integra, a través de técnicas de aprendizaje profundo por medio de redes neuronales convolucionales y de una tarjeta de desarrollo Arduino UNO, un sistema de alarma que detecta la presencia de personas en el área de acceso, determina si portan cubrebocas, mide su temperatura y permite su acceso. El proyecto desarrollado proporciona resultados funcionales y rápidos para la detección de cubrebocas. Los resultados de las pruebas muestran una tasa de precisión adecuada en la detección de personas con y sin cubrebocas. El modelo entrenado utiliza conjuntos de datos obtenidos de fuentes públicas que presentan rostros con y sin cubreboca, el cual fue capaz de llevar a cabo su tarea utilizando el modelo de CNN apoyado por OpenCV, Tensor Flow y Keras en un ambiente de programación Python. Los resultados ofrecen una alternativa de apoyo en la lucha contra la propagación del virus estacionarios o brotes repentinos de contagios mediante la detección de una persona que lleva cubreboca o no y establece el acceso tomando en cuenta su temperatura. También puede verse como una herramienta útil para mejorar accesos y control a determinados entornos públicos.

Los trabajos futuros incluyen la integración del distanciamiento físico, en el que la cámara detecte si la persona porte o no una mascarilla facial y, al mismo tiempo, mida la distancia entre cada persona para generar alarmas cuando el distanciamiento físico no se observe correctamente. Igualmente se trabaja para realizar comparaciones entre diferentes algoritmos de clasificación, de forma que su integración permita generar modelos con mayor precisión y rendimiento durante el

entrenamiento y así aumentar el rendimiento en la detección y el reconocimiento de personas que porten cubrebocas. En este sentido, utilizar esta propuesta para generar un conjunto de datos propios de forma que el entrenamiento sea independiente. Además, se investiga sobre la implementación en plataformas de internet de las cosas y tarjetas de mayor capacidad para poder aprovechar de mejor manera las capacidades de las CNNs.

6. Bibliografía y Referencias

[1] Alturki, R., Alharbi, M., AlAnzi, F. & Albahli, S. Deep learning techniques for detecting and recognizing face masks: A survey. Frontiers in Public Health, Vol. 10, 1-24, 2022.

[2] Banik, D., Rawat, S., Thakur, A., Parwekar, P. & Chandra, S.S. Automatic approach for mask detection: effective for COVID-19. Soft Computing, Vol. 27, 7513–7523, 2023.

[3] Chollet, F. Deep learning with python. Manning Publications. 2022.

[4] Goodfellow, I., Bengio, Y. & Courville, A. Deep Learning. MIT Press. 2016.

[5] Leung, N. H. L., Chu, D. K. W., Shiu, E. Y. C., Chan, K.-H., McDevitt, J. J., Hau, B. J. P., Yen, H.-L., Li, Y., Ip, D. K. M., Peiris, J. S. M., Seto, W.-H., Leung, G. M., Milton, D. K., & Cowling, B. J. Respiratory virus shedding in exhaled breath and efficacy of face masks. Nature Medicine, Vol. 26, 676–680, 2020.

[6] Militante, S. V., & Dionisio, N. V. Real-Time Facemask Recognition with Alarm System using Deep Learning. 11th IEEE Control and System Graduate Research Colloquium (ICSGRC). Shah Alam, Malaysia, August 2020.

[7] Miranda-Ramos, M. M., Hallon, J. O., & Suriaga, J. D. Sistema de cámaras para la detección de mascarillas con Python y Raspberry PI con comunicación inalámbrica a través del sistema global de comunicación móvil (GSM). Información tecnológica, Vol. 33, No. 3, 23–30, 2022.

[8] Naseri, R. A. S., Kurnaz, A., & Farhan, H. M. Optimized face detector-based intelligent face mask detection model in IoT using deep learning approach. Applied Soft Computing, Vol. 134, No. 109933, 1-23, 2023.

[9] Oliveira-Teixeira, F., Donadon-Homem, T. P., & Pereira-Junior, A. Aplicación de inteligencia artificial para monitorear el uso de mascarillas de protección. Revista Científica General José María Córdova, Vol. 19, No. 33, 205–222, 2021.

[10] Rosebrock, A. COVID-19: Face Mask Detector with OpenCV, Keras/TensorFlow, and Deep Learning. May 4, 2020. https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/.

[11] Sánchez-Guevara, I., & Vite-Cristóbal, R. La modelación del COVID-19 en México a partir del modelo SIR. Política y Cultura, no. 58, 151–178, 2022.

[12] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 4510-4520, 2018.

[13] Umer, M., Sadiq, S., Alhebshi, R. M., Alsubai, S., Al Hejaili, A., Eshmawi, A. A., Nappi, M., & Ashraf, I. Face mask detection using deep convolutional neural network and multi-stage image processing. Image and Vision Computing, Vol. 133, No. 104657, 1-10, 2023.

[14] Varshini, B., Yogesh, H., Pasha, S. D., Suhail, M., Madhumitha, V., & Sasi, A. IoT-Enabled smart doors for monitoring body temperature and face mask detection. Global Transitions Proceedings, Vol. 2, No. 2, 246–254, 2021.

[15] Vega-Luna, J. I., Salgado-Guzmán, G., Sánchez-Rangel, F. J., Cosme-Aceves, J. F., Tapia-Vargas, V. N. Sistema de control de acceso usando reconocimiento facial con una Raspberry pi 4 y Opencv. Pistas Educativas, No. 136, 1011-1028, 2020.

[16] Zakariyya, I., Kalutarage, H., & Al-Kadri, M. O. Towards a Robust, Effective and Resource Efficient Machine Learning Technique for IoT Security Monitoring, Computers and Security, p. 103388, 2023.

[17] Azouji, N., Sami, A. & Taheri, M. Efficient Mask-Net for face authentication in the era of COVID-19 pandemic, Signal, Image and Video Processing, Vol. 16, 1991–1999, 2022.