ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

Sistema para la detección de uso de mascarilla utilizando técnicas de redes neuronales convolucionales.

Chiza Llambo, Juan Carlos

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnologías

Centro de Posgrados

Maestría en electrónica mención redes industriales

Trabajo de Titulación, Previo a la Obtención del Título de Magister en Electrónica Mención Redes Industriales

Ing. Eddie Galarza Zambrano Mgs.

28/ Abril/ 2022

Latacunga

Reporte de verificación del contenido

Tesis_Mascarillas_Juan_Chiza_feb2.docx

Scanned on: 15:16 February 2, 2022 UTC

Identical Words	512
Words with Minor Changes	210
Paraphrased Words	103
Omitted Words	0

Firma:

.....

Ing. Eddie Galarza Zambrano Mgs.

DIRECTOR

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnología

Centro de Posgrados

Certificación

Certifico que el trabajo de titulación: "**sistema para la detección de uso de mascarilla utilizando técnicas de redes neuronales convolucionales**" fue realizado por el señor **Chiza Llambo, Juan Carlos;** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Latacunga, febrero del 2022

Firma:

.....

Ing. Eddie Galarza Zambrano Mgs.

Director

C.C.: 1303128514

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnología

Centro de Posgrados

Responsabilidad de Autoría

Yo **Chiza Llambo, Juan Carlos**, con cédula n° 1804801940, declaro que el contenido, ideas y criterios del trabajo de titulación: **"sistema para la detección de uso de mascarilla utilizando técnicas de redes neuronales convolucionales"** es de mí autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Latacunga, febrero del 2022

.....

Chiza Llambo, Juan Carlos

C.C.:1804801940

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnología

Centro de Posgrados

Autorización de publicación

Yo **Chiza Llambo, Juan Carlos**, con cédula n° 1804801940, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: "**Sistema para la detección de uso de mascarilla utilizando técnicas de redes neuronales convolucionales**" en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi/nuestra responsabilidad.

Latacunga, febrero del 2022

.....

Juan Carlos Chiza Llambo

C.C.: 1804801940

Dedicatoria

" Siempre parece imposible hasta que se hace”

Nelson Mandela

Más que nada, agradezco a Dios por darme la fortaleza, el razonamiento, haberme cuidado y mantenerme con salud y vida a lo largo de esta enfermedad pandémica que azoto al mundo.

A la Universidad de las Fuerzas Armadas ESPE, que me brindó la posibilidad de participar en este programa de formación académica el mismo que me dejará enriquecer mis ocupaciones profesionales y particulares.

A mi tutor del proyecto de titulación, Ing. Eddie Galarza Zambrano Mgs., por su apoyo, guía y amistad que me brindo a lo largo de este proyecto de titulación por eso, hoy logro alcanzar este objetivo en mi formación.

A mi madre que me brindo el apoyo, la guía, los consejos necesarios y la fortaleza para seguir a lo largo de la vida y la carrera a pesar de las dificultades.

Juan C. Chiza Ll.

Agradecimiento

"La calidad nunca es un accidente, siempre es resultado de un esfuerzo de la inteligencia"

John Ruskiin

Este trabajo final, lo quiero dedicar a mi Padre Dios por estar siempre a mi lado cuidando, bendiciendo y brindando esos dones espirituales que me ha llevado al camino del éxito.

Le agradezco de una manera muy especial a mi madre Beatriz por ser un ejemplo de persona luchadora que siempre busca cumplir sus planes y metas trazadas, más que mi vieja es padre y madre, amiga y lo más importante en mi vida.

A mi tutor de investigación Ing. Eddie Galarza Mgs., director de este proyecto por su entrega incondicional a la docencia e investigación y su valiosa guía en el transcurso de mi formación académica

Juan C. Chiza Ll.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Resumen

Con el auge de la digitalización y el proceso de automatización en el reconocimiento de imágenes, en estos días es atractivo para el apalancamiento de realizar tareas que son problemáticas para el ser humano realizarlo manualmente (Jain, 2019). En la actualidad debido a la pandemia mundial y el uso obligatorio de la mascarilla recomendado por la organización mundial de la salud (OMS) surge la necesidad de realizar un proceso de reconocimiento de uso de mascarilla en las personas para evitar la propagación y el contagio.Para el desarrollo de esta investigación nace la inminente necesidad de desarrollar técnicas de preprocesamiento, clasificador de imágenes y el método de aprendizaje profundo mediante redes neuronales convolucionales (DCNN) para que la computadora reconozca patrones para el cual será entrenado previamente y distinga dos formas puntuales y clasifique si utiliza o no mascarilla. Por lo cual se propone la implementación de un sistema de reconocimiento de uso de mascarillas programado en software libre Python y los resultados serán interpretados utilizando la biblioteca de aprendizaje automático Tensorflow desarrollado por Google.Esto permitiría aprovechar al máximo la aplicabilidad de la inteligencia artificial junto al procesamiento de imágenes y la necesidad actual de la detección de mascarillas. Además, serviría como base para futuras investigaciones en aplicaciones móviles.

Palabras Clave: clasificador de imágenes, redes neuronales convolucionales, procesamiento de imágenes, aprendizaje profundo, tensorflow, python.

Abstract

With the rise of digitalization and the process of automation in image recognition, these days it is attractive for the leverage of performing tasks that are problematic for the human being to do it manually (Jain, 2019).Currently due to the global pandemic and the mandatory use of the mask recommended by the World Health Organization (OMS) arises the need to carry out a process of recognition of use of the mask in people to prevent the spread and contagion.For the development of this research there is an imminent need to develop preprocessing techniques, image classifier and the deep learning method through the use of deep convolutional neural networks (DCNN) so that the computer recognizes patterns for which it will be previously trained and will distinguish two points and classifies whether a human is using or not a mask. herefore, this works implement a system of recognition of use of masks that was programmed using Python free software and the results will be interpreted using the Tensorflow Machine Learning Library developed by Google. It is possible to take the full advantages of the applicability of artificial intelligence alongside image processing and the current need for mask detection. In addition, it would serve as a basis for future research in mobile applications.

Keywords: 3d virtual environment, convolutional neural networks, image processing, deep learning , tensorflow, python.

Capítulo I

Contenido general

Antecedentes

Las redes neuronales convolucionales (DCNN) se han convertido en una de las técnicas más populares dentro del campo de inteligencia artificial, que es muy usada en el reconocimiento de imágenes. El avance de las DCNNs se encuentra evolucionando en varios ámbitos desde la rama medicinal para la clasificación de tejido invasivo de cáncer de seno (Ramírez & Cruz, 2018), la detección y clasificación de la mitosis (Ciseran, 2013), detección de bacterias Escherichia Coli en verduras frescas la cual provoca infecciones en la salud a cualquier persona (Rodríguez, 2019) de igual forma en aplicaciones en otras áreas como el reconocimiento de caracteres manuscritos implementados en Python (Trujillo, 2017).

Las redes neuronales convolucionales son un modelo para encontrar características principales de una imagen basado en el clasificador de imagen, en la clasificación de imágenes en las últimas décadas ha presentado cambios importantes en los métodos y técnicas a utilizar para brindar un mejor resultado desde filtros de base Haar y clasificadores en casada utilizado para la detección de rostros en imágenes digitales (Guevara, 2008).

Así mismo otro factor que en los últimos tiempos ha presentado cambios y avances significativos en el procesamiento y reconocimiento de características de una imagen de entrada es la técnica de Deep Learning o aprendizaje profundo (DL) donde uno de los métodos de DL que se emplea para el análisis de imágenes son las redes neuronales convolucionales la que permite detectar esquinas, bordes e incluso características complejas mediante el entrenamiento previo con grandes cantidades de datos de entrada debidamente etiquetadas y una variable de salida ocasionando un mayor consumo computacional, pero gracias al desarrollo tecnológico que en los últimos años se ha presentado junto a la potente capacidad

de unidad de procesamiento gráfico (GPU), todo esto es viable a pesar que estás arquitecturas pueden ser tediosas y demandar demasiado tiempo (Chanampe, 2019).

La DCNN es una técnica de aprendizaje profundo aplicado para la reducción de imágenes atenuando el tiempo de procesamiento y consumo de memoria computacional debido a que solo analizará la información necesaria como bordes, esquinas y algunos detalles complejos pero importantes al momento de identificar las características de la imagen, además aplica la capa rectificante lineal (RELU) para aumentar la no linealidad de la red neuronal para una mejor adaptación a nuevas imágenes de ingreso (Kuo, 2016).

Planteamiento del problema

En la actualidad, por la pandemia del COVID - 19 que a nivel mundial afecta la salud de todos los habitantes, la Organización Mundial de la Salud (OMS), recomienda el uso de un cubre bocas que es apropiado para prevenir la propagación o transmisión del virus en lugares públicos, al sociabilizar con personas ajenas o cuando el distanciamiento social no lo permite. De igual forma recomienda el cambiar frecuentemente el cubre boca cuando esté húmedo o deteriorado con el propósito de mantener su eficacia, adicionalmente otras medidas de bioseguridad.

El uso de la mascarilla está recomendado como una barrera simple que ayuda a evitar que las macropartículas que salen de la boca y nariz al momento de toser, estornudar o hablar viajen por el aire hasta otra persona que se encuentre cerca, el uso de esta barrera simple pero efectiva ayuda a precautelar la salud del público en general, incluso del grupo más vulnerable como son los adultos mayores o personas con enfermedades de salud grave.

Pero a pesar de esta recomendación emitida por la OMS, en lugares públicos o de gran afluencia en donde es imposible mantener el distanciamiento social y sobre todo en lugares de poca ventilación existen personas donde el uso de la mascarilla no lo ven como una opción de

salvaguardar su salud ni de los demás, provocando malestar, miedo e inconformidad, esto genera los problemas en la sociedad que se detallan a continuación.

Primero, provoca histeria colectiva al sentirse amenazados de una posible transmisibilidad del agente COVID-19 y por tanto el daño potencial que genera esta enfermedad. Segundo, la incomodidad del público en general que se encuentran alrededor por la inseguridad, por el simple hecho que las partículas puedan fluir por el aire libremente al no anteponerse una barrera de protección.

Los problemas anteriormente descritos generan la necesidad de desarrollar un sistema de detección de uso de mascarilla utilizando técnicas de redes neuronales convolucionales que mediante un sistema de alerta ofrezca al público en general la posibilidad de superar estos inconvenientes.

Justificación, importancia y alcance del proyecto (preguntas de investigación)

En lugares públicos, debido al poco control de uso de mascarilla en las personas es evidente observar que transitan libremente sin una medida de precaución, además es limitado el uso de alguna tecnología al momento de controlar y mitigar este tipo de inconvenientes en comparación a países del primer mundo. Por tal motivo el presente trabajo tiene como finalidad diseñar un sistema que permita verificar en las personas el uso de mascarillas con ayuda de visión artificial y redes neuronales convolucionales, por esta razón y a las condiciones presentes de la pandemia, desarrollar un sistema que identifique el uso de mascarillas es necesario e importante.

La importancia de este proyecto es estudiar la posibilidad de utilizar la clasificación de imágenes y el método de aprendizaje profundo (DL) para clasificar y distinguir dos posibilidades básicas si están o no utilizando mascarillas.

El alcance del proyecto está limitado al desarrollo de un algoritmo para la detección de uso de mascarillas, en base a un previo entrenamiento de las capas de las redes neuronales y el ajuste de los pesos para el sistema a diseñar. La arquitectura de las redes neuronales convolucionales se basará en modelamientos realizados en estudios previos, que serán acoplados a las necesidades del presente estudio.

Objetivos

Objetivo General

Diseñar e implementar un sistema de detección de uso de mascarillas utilizando técnicas de redes neuronales convolucionales.

Objetivos Específicos

- Obtener información necesaria de redes neuronales convolucionales.
- Adquirir conocimientos sobre el procesamiento y las características de las imágenes digitales.
- Estudiar el funcionamiento de la librería aprendizaje profundo DL en entorno de programación de software libre.
- Recolectar imágenes de rostros con y sin mascarillas para la Data base de entrenamiento, prueba y validación con su respectivo etiquetado.
- Diseñar la red neuronal convolucional.
- Entrenar al sistema de detección de uso de mascarillas para que pueda reconocer el uso de tapa boca.
- Evaluar por medio de testeo respectivo el funcionamiento óptimo de la aplicación

Hipótesis de investigación

La implementación del sistema de detección de objetos en imágenes utilizando técnicas de redes neuronales convolucionales permitirá identificar si una persona utiliza tapa boca.

Categorización de las variables de investigación.

En efecto, de la hipótesis planteada se identifican dos elementos que a continuación se detalla:

- **Variable Independiente:** Sistema de detección de objetos en imágenes.
- **Variable Dependiente:** Identificación de personas que utilizan tapa bocas.

La operacionalización de las variables se muestra en la tabla 1:

Tabla 1

Detalle de las Variables

VARIABLES	TIPO	DEFINICIÓN CONCEPTUAL	DEFINICIÓN OPERACIONAL	DIMENSIONES	INDICADORES
Identificación de personas utilizando tapa bocas.	Dependiente	En la identificación se determinará el número de personas que estén utilizando tapa bocas.	En la identificación se determinará el número de personas que estén utilizando tapa bocas	-Personas detectadas con tapa bocas.	- Porcentaje de detección de personas con tapa bocas
Sistema de detección de objetos en imágenes.	Independiente	Los sistemas de detección se centran en determinar la presencia de objetos en la escena de análisis, independientemente del tamaño o la posición. El fin es el de determinar si una imagen contiene o no, uno o varios objetos concretos	Es un sistema que permitirá identificar si una persona utiliza o no mascarilla, dentro de las imágenes digitales.	-Detección de objetos. -Tiempo de detección.	- Porcentaje de detección de objetos. - Porcentaje de reducción de tiempo de detección

Capítulo II

Marco teórico

Marco Legal

- **Constitución de la República del Ecuador**

En el artículo 42 establece que “El Estado garantizará el derecho a la salud, su promoción y protección, por medio del desarrollo de la seguridad alimentaria, la provisión de agua potable y saneamiento básico, el fomento de ambientes saludables en lo familiar, laboral y comunitario, y la posibilidad de acceso permanente e ininterrumpido a servicios de salud, conforme a los principios de equidad, universalidad, solidaridad, calidad y eficiencia."

En el artículo 389 establece que: "El Estado protegerá a las personas, las colectividades y la naturaleza frente a los efectos negativos de los desastres de origen natural o antrópico mediante la prevención ante el riesgo, la mitigación de desastres, la recuperación y mejoramiento de las condiciones sociales, económicas y ambientales, con el objetivo de minimizar la condición de vulnerabilidad

- **Ley Orgánica de la salud**, (Registro Oficial Suplemento 423 de 22-dic.-2006).

  Art 6.- inciso 3 establece “Diseñar e implementar programas de atención integral y de calidad a las personas durante todas las etapas de la vida y de acuerdo con sus condiciones particulares.”
- **Art. 9.-** Corresponde al Estado garantizar el derecho a la salud de las personas, para lo cual tiene, entre otras, las siguientes responsabilidades: a) Establecer, cumplir y hacer cumplir las políticas de Estado, de protección social y de aseguramiento en salud a favor de todos los habitantes del territorio nacional; b) Establecer programas y acciones de salud pública sin costo para la población; c) Priorizar la salud pública sobre los intereses comerciales y económicos; d) Adoptar las medidas necesarias para garantizar en caso de

emergencia sanitaria, el acceso y disponibilidad de insumos y medicamentos necesarios para afrontarla, haciendo uso de los mecanismos previstos en los convenios y tratados internacionales y la legislación vigente; e) Establecer a través de la autoridad sanitaria nacional, los mecanismos que permitan a la persona como sujeto de derechos, el acceso permanente e ininterrumpido, sin obstáculos de ninguna clase a acciones y servicios de salud de calidad.

Marco Conceptual

Este capítulo describe los fundamentos teóricos para comprender el trabajo. Inicialmente se analiza la importancia de la mascarilla para combatir la propagación del virus SARS COV-19, el procesamiento y clasificación de imágenes, redes neuronales partiendo desde el análisis funcional biológico para finalmente abordar la red neuronal convolucional en el contexto de aprendizaje profundo.

Covid-19 y la importancia de la mascarilla

El creciente auge de la pandemia mundial COVID-19 en todo el planeta ha planteado en la humanidad desafíos importantes en adoptar e implementar estrategias para mitigar la transmisión del virus en el entorno. Según la Organización mundial de la salud OMS los principales síntomas reportados por COVID – 19 son: el 93% fiebre, 82% tos, 55% disnea y el 29% tienen alguna dificultad para respirar, en base a las investigaciones actuales del virus, sugiere que la razón principal de propagación del virus en el entorno se debe potencialmente a las gotas infectadas que se transmiten cuando una persona que presentada un estado de infección estornuda, tose o simplemente conversa sin un medio de barrera de protección. Los mecanismos preventivos para combatir la propagación del virus que ha demostrado ser altamente contagiosa son: el distanciamiento social, lavado de mano con frecuencia, vacunación, uso de gel y adicional a lo mencionado es el de utilizar mascarillas en público

siendo esta ultima una de las medidas de prevención y control más importantes implementadas por los gobiernos del mundo, tanto a nivel social como en los establecimientos de salud, para limitar la propagación del virus. En el ámbito sanitario, para prevenir la transmisión de agentes infecciosos aerotransportados en frecuencia se utiliza la mascarilla. Las mascarillas médicas, también llamadas mascarillas quirúrgicas, están diseñadas para evitar que las gotas de un paciente infectado entren en contacto con las membranas mucosas de la nariz y la boca de los trabajadores de la salud que las usan proporcionando una mayor protección. (Bonilla, 2020)

Como podemos ver en la Figura 1, el uso de una mascarilla quirúrgica para prevenir el contagio del virus Covid-19

Figura 1

Mascarilla Quirúrgica

Nota. Recuperado de Bonilla (2020).

Procesamiento de imágenes

El procesamiento de imágenes digitales son un conjunto de técnicas que se aplica a las imágenes adquiridas por diferentes fuentes de ingreso las mismas que pueden ser fotografías o fotogramas de un video cuyo objetivo es de mejorar la calidad de dichas imágenes mejorando el resultado de la imagen para una aplicación en específico, mejorando características de la imagen mediante filtros de suavizado, eliminación de ruido, detección de bordes, etc. El procesamiento de imágenes es, por tanto, la operación matemática aplicada a los píxeles para resaltar información o mejorar la calidad. (Esqueda J, 2004).

Algunas operaciones adicionales a las mencionadas anteriormente y las más frecuentes en un procesamiento de imágenes digitales se encuentran correcciones de color (intensidad, contrastes y saturación), interpolación, alineación y modificaciones geométricas. El procesamiento de imágenes depende del campo a aplicarse estas pueden ser programas espaciales, medicina, biología, industrial o científica (NLS Palomino, 2009).

La visión artificial tiene como objetivo principal la detección e identificación de objetos a través del procesamiento de imágenes digitales, las cuales son adquiridas de diferentes formas como escaneo, cámaras digitales o simplemente fotogramas obtenidos de un video, con el objetivo de crear un sistema mecánico computacional semejante al ser humano, las etapas de un procesado de imagen se detallan a continuación y se presentan en la Figura 2.

Figura 2

Etapas de procesamiento de imagen

Nota. Recuperado de NLS Palomino (2009)

Adquisición de la imagen: Consiste en obtener retratos o fotogramas de un video por medio de un dispositivo que registre imágenes estáticas o en movimiento.

Preprocesamiento: Conjuntos de técnicas aplicadas a una imagen captada por diversas fuentes de ingresos, cuya finalidad es reducir el ruido para añadir ciertos detalles como brillo, color, contraste, etc.

Segmentación: Proceso de dividir a una imagen digital en varias partes formando grupos de pixeles

Representación y descripción: Etapa donde se obtiene puntos relevantes de una imagen como estructura, forma, tamaño las cuales les diferencia con otros objetos.

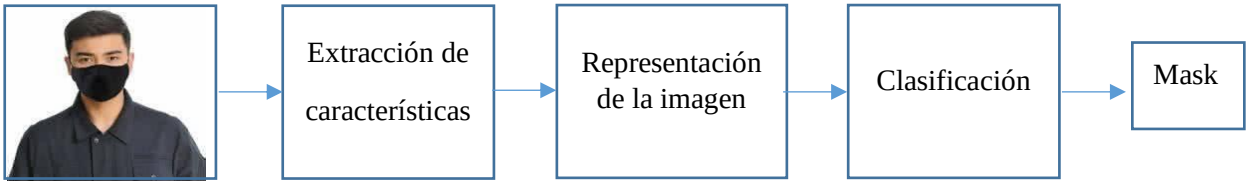
Reconocimiento e interpretación: Proceso donde se identifica y se le asocia un significado propio.

Clasificación de una imagen

Las criaturas vivientes tienen el instinto básico de reconocer y clasificar objetos en el entorno en función de sus sentidos. Cuando se trata de visión por computadora, especialmente en la clasificación de imágenes, el propósito es asignar una categoría o etiqueta a una clase y se pueden utilizar métodos matemáticos estadísticos para distinguir los principales atributos de la imagen. En otras palabras, la clasificación se refiere al proceso de asignar imágenes a las etiquetas correspondientes. Existen varios métodos matemáticos asistidos por computadora para clasificar una imagen, que incluyen: árbol de decisión, redes neuronales. A continuación, las etapas de clasificación general de una imagen se describen en la Figura 3.

Figura 3

Etapa general de clasificación de una imagen



Extracción de características: La extracción de características de una imagen es la información asociada a los aspectos relevantes de un objeto como tamaño, color, forma, etc. Además, aporta la reducción espacial de la dimensionalidad del objeto. Extraer este tipo de información es importante para lograr un buen desempeño al momento de clasificar y reconocer una imagen, normalmente este tipo de información se encuentra localizada en los bordes del objeto o en el cambio de contraste (Kumar, 2014).

Representación de la imagen: enfocándose desde lo estadístico, se lleva a cabo una sola representación de la imagen completa como un arreglo numérico. Esta representación es el

resultado de combinar o sumar cada uno de los vectores numéricos que especifican las regiones de interés de la extracción de características.

Clasificación: para esta etapa la imagen de entrada se clasifica en función de la representación única de la imagen. En este punto, el sistema de clasificación aprende del database para discernir imágenes de diferentes clases.

El conjunto de imágenes (base de datos) que se utiliza en este trabajo es previamente clasificado por personas en diferentes carpetas (máscara y sin máscara), a las que llamaremos clases, estas se pueden observar en la Figura 4.

Figura 4

Carpetas de imágenes con y sin mascarilla

Nota. El gráfico representa dos clases que contienen las respectivas imágenes con y sin mascarillas las mismas que se utilizara para el entrenamiento de la red neuronal

Aprendizaje profundo

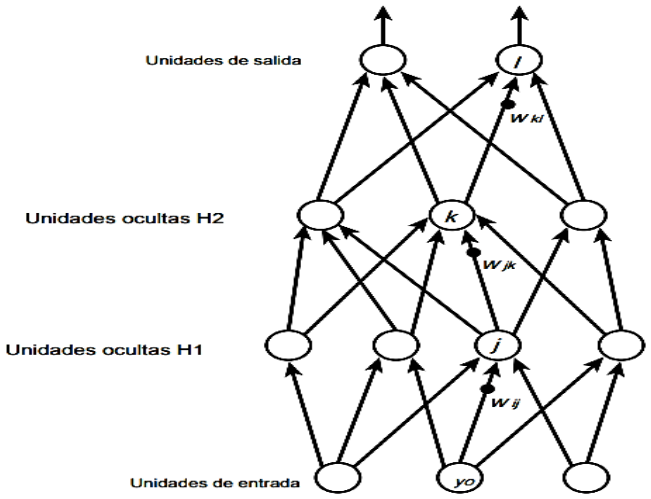
La inteligencia artificial debe su auge gracias al aprendizaje profundo (DL) basado en la analogía del cerebro de los mamíferos para adquirir conocimientos, las personas aprendemos debido a las experiencias adquirida y mientras más variada es proporcional a la cantidad de aprendizaje. Del mismo modo las máquinas alcanzan su experiencia mediante los datos que adquieren, la cantidad y la calidad de información determina su aprendizaje y eso a su vez

influye en la detección o clasificación. Los métodos de aprendizaje profundo son métodos que se componen de múltiples capas obtenidas por la composición de módulos no lineales, debido al conjunto de métodos aplicados en el DL puede aprender datos complejos. Para ciertas tareas, como la clasificación de objetos, las múltiples capas refuerzan los aspectos relevantes de la entrada para la diferenciación y, a su vez, suprimen la información irrelevante, por ejemplo, cuando una imagen se digitaliza, se representa mediante un conjunto de valores en una matriz en la que la primera capa aprende características relevantes, por lo general la ausencia de bordes y ciertos puntos en la imagen, la segunda capa normalmente reconoce ciertas disposiciones en los bordes, mientras que la tercera capa reconoce diferencias y combina motivos de clasificación con capas posteriores. La clave del aprendizaje profundo es que las capas anteriores no están diseñadas para el aprendizaje de objetos de una sola vez, por el contrario, contienen un algoritmo de propósito general donde el proceso de aprendizaje se basa en los datos. (Hinton, 2015).

En la Figura 5, se muestra la estructura de una red neuronal de aprendizaje profundo con diversas capas diseñada para propósito general.

Figura 5

Red neuronal multicapa



Nota. Recuperado de Hilton (2015).

Redes neuronales

Antes de proceder con un estudio detallado de las redes neuronales convolucionales (RNC) y el impacto que pueden tener en los estudios de diferentes imágenes, es conveniente y útil conocer las principales características de la neurona de una forma más exhaustiva, desde la anatomía hasta los parámetros de convolución de una red.

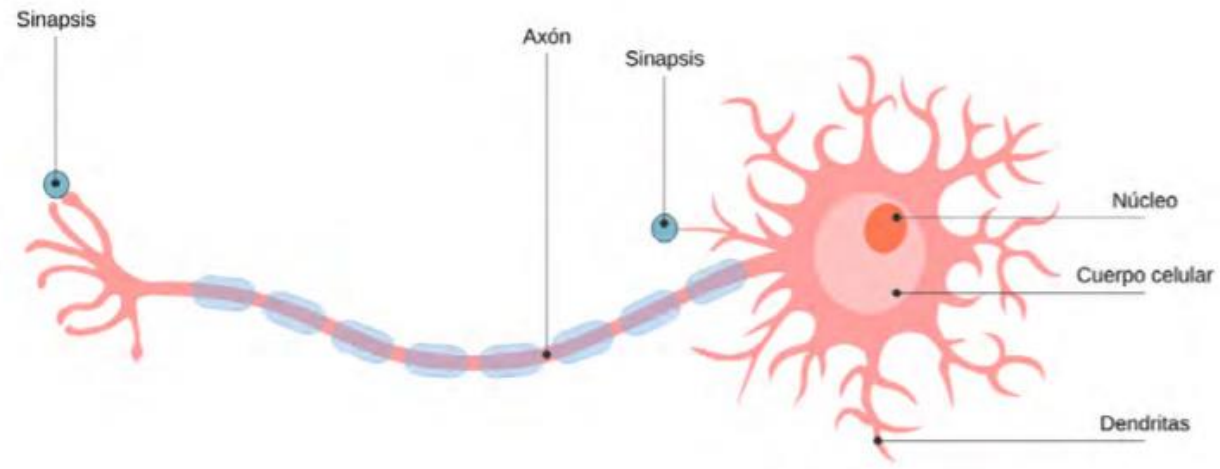
Anatomía de una neurona

Las partes que conforman una neurona son las dendritas, el cuerpo celular, el axón y la sinapsis. (Freeman, 1993). El funcionamiento de una neurona es la siguiente: capta señales de otras neuronas mediante un conjunto de estructuras llamadas dendritas y se comunican entre ellas mediante la sinapsis que sirve como unión entre cada una de ellas. La información luego pasa a través del cuerpo celular para ser procesada. Este cuerpo celular produce resultados utilizando los axones, que sirven para enviar información a otra neurona. (López-Saca, 2019)

A continuación, en la Figura 6, se muestra la estructura de una neurona biológica o celular neuronal.

Figura 6

Neurona Biológica



Nota. Recuperado de López-Saca (2019).

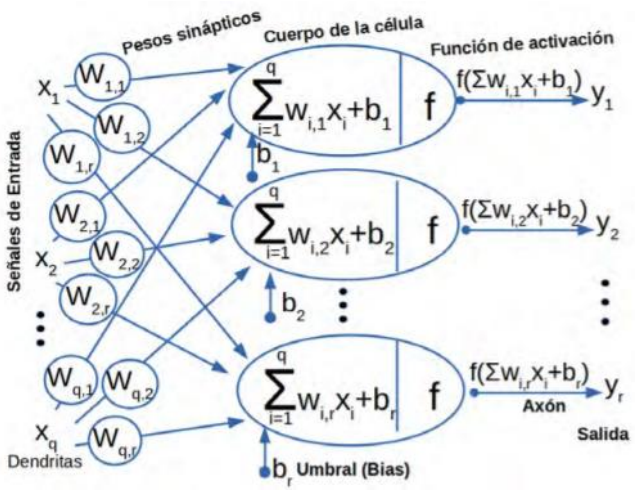
Redes neuronales artificiales

La Red Neural Artificial (RNA) o también conocida como sistema conectado es un modelo computacional inspirado en la analogía del cerebro de los mamíferos porque tiene la capacidad de almacenar información basada en la experiencia. Las RNA constan de un conjunto de unidades de cómputo simples, llamadas neuronas artificiales, que se encuentran interconectadas entre sí para transmitir señales, permitiendo crear un sistema que aprende, este conocimiento lo adquiere mediante un proceso de aprendizaje (entrenamiento), mostrando a la red neuronal cual debe ser la salida dada cierta información de entrada la misma que se somete a diversas operaciones cuando atraviesa las capas de las RNA, hasta que esta sea capaz de acumular la información necesaria para inferir en su salida (Jiménez, 2019).

Como se puede apreciar en la Figura 7, Cada una de las neuronas están conectadas entre sí, también llamadas sinapsis en su contraparte biológica, en estas conexiones el valor de la neurona anterior se multiplica por un valor de peso, esto permite incrementar o suprimir la activación de neuronas vecinas también en cada neurona tiene una función de activación que se lleva a cabo imponiendo un umbral de activación, su función es establecer un umbral antes de que se propague a la siguiente neurona.

Figura 7

Red neuronal artificial



Nota. Recuperado de López-Saca (2019).

En la Figura 7 podemos apreciar:

- Un **conjunto de entradas (*x*)** información o datos para la predicción o clasificación.
- Un **conjunto de sinapsis (*w*)** o pesos sinápticos, cada uno de estos pesos se caracteriza por un peso prestablecido. Contrariamente a una sinapsis biológica cerebral, el peso de una sinapsis neuronal artificial puede asumir valores negativos como positivos.

- Una **función sumatoria** ∑ su objetivo es de realizar la suma de los pesos sinápticos de las señales de entradas.
- La **función de activación (f)** A la salida de la neurona existe una funcionalidad limitante o umbral, que cambia el valor del resultado o introduce un umbral que debe cruzarse para continuar a otra neurona.
- La **salida (*y*)** normalmente el valor que se toma en bases al estado de activación y por lo general la amplitud de dicha salida está comprendida en el rango de [0,1] o, de manera alternativa [-1,1]. En este modelo de la red la salida está definida en la ecuación 1.

$$y = f\left(\sum_{j=1}^{m} w_i . x_i\right) \qquad (1)$$

Funciones de activación: La función de activación expresada con las letras $(y_1, y_2, .., y_n)$ definen las salidas de la red neuronal dada una entrada o conjunto de entradas como se observa en la figura 7. En general, cada una de las capas que componen la red neuronal tiene una función de activación que permite reconstruir o predecir el objeto. Además, se debe considerar que se utilizará una función no lineal en la red neuronal porque permite que el modelo se adapte para trabajar con una database considerable. (Artola, 2019).

Dependiendo de la tarea a realizar se escoge la función de activación. Las principales funciones se clasifican en: lineal y no lineales

Función lineal: a su vez función identidad su desempeño se basa en que la entrada sea la misma que la salida. Entonces, si tengo una red neuronal multicapa y aplico una función lineal, se llama regresión lineal. Por tanto, esta función de activación lineal se utiliza cuando se

requiere una regresión lineal en la salida y de esta forma la red neuronal a la que se aplica la función produce un valor único (Sánchez, 2015).

La función lineal, está definida en la ecuación 2

$$f(x) = x \qquad (2)$$

Funciones no lineales: Para este tipo de funciones a diferencia de la anterior no posee una tasa de cambio constante y por lo tanto sus gráficas no son líneas rectas, las principales funciones no lineales son detalladas a continuación.

Sigmoide: Es una función que ya no es utilizada por las desventajas que presenta al momento de realizar las operaciones, desventajas como la saturación de la salida y desvanecimiento de la gradiente de la backpropagation generando un mal entrenamiento. Está definida por la siguiente función matemática, Ecuación 3.

$$f(x) = \frac{1}{1 + e^{-x}} \qquad (3)$$

Tangente hiperbólica: Esta función presenta mejores resultados que la *sigmoide,* mencionada anteriormente, debido a que se encuentra focalizada a cero, pero a pesar de eso continúa presentando saturación a la salida, está definida por la Ecuación 4:

$$f(x) = \tanh(x) \qquad (4)$$

Escalón: También conocida como escalón, el rango de respuesta es $\{0, +1\}$ esta función presenta el esquema de activación en el cual si la entrada es un valor menor que cero la neurona será cero, pero cuando el valor es mayor que cero dará la salida de 1. Está definida por la Ecuación 5:

$$f(x) = \mathrm{H}(x) \qquad (5)$$

Rectified Linear Unit: La unidad rectificada lineal (ReLU), es la más utilizada en las redes de aprendizaje profundo por que incrementa la velocidad de aprendizaje y a diferencia de las funciones *sigmoide* y *tangente hiperbólica* evita la desaparición de la gradiente a pesar del aumento de la cantidad de capas, una de los aspectos negativos de (ReLU) suele presentarse que al momento de aprendizaje algunas neuronas dejan de funcionar, eso se debe cuando la gradiente modifica el valor de los pesos inactivando algunas neuronas, pero este problema puede ser evitado al variar en pequeños valores la función de activación. (Pacheco, 2017).

La unidad rectificada lineal (RelU), está definida comola siguiente Ecuación 6:

$$f(x) = \max(0, x) \qquad (6)$$

Softmax: Para la clasificación de imágenes se recomienda utilizar esta función de activación, que es la extensión de la función sigmoidea pero para varias clases, el valor de las neuronas en las capas de salida debe convertirse a valores de probabilidad entre 0 y 1, para ello se utiliza la función SoftMax, en esta función se busca que todas las neuronas reflejen valores cercanos a cero en la salida, excepto la clase correcta donde su valor debe ser cercano a uno. (Peralta, 2005)

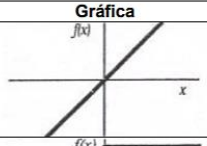
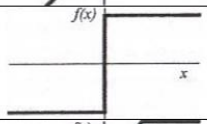
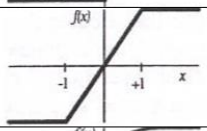
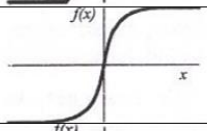
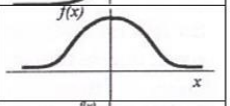
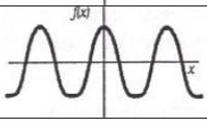
Softmax, está definida por la siguiente Ecuación 7.

$$f(x) = \frac{e^x}{\sum_{j=1}^{m} e^x}, j = 1, \dots, m \qquad (7)$$

En la figura 8, se ilustran las diversas funciones de activación además el rango y sus respectivas gráficas, pero, generalmente las funciones típicas empleadas son la lineal, sigmoide, escalón, sin embargo, para la clasificación de imágenes se utiliza la función SoftMax.

Figura 8

Funciones de activación

	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, \infty]$	
Escalón	$y = \begin{cases} 1, & si\ x \geq 0 \\ 0, & si\ x < 0 \end{cases}$ $y = \begin{cases} 1, & si\ x \geq 0 \\ -1, & si\ x < 0 \end{cases}$	$[0, 1]$ $[-1, 1]$	
Lineal a tramos	$y = \begin{cases} 1, & si\ x > 1 \\ x, & si - 1 \leq x \leq 1 \\ -1, & x < -1 \end{cases}$	$[-1, 1]$	
Sigmoidea	$y = \frac{1}{1 + e^{-x}}$ $y = \tanh(x)$	$[0, 1]$ $[-1, 1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0, 1]$	
Sinusoidal	$y = A\sin(wx + \varphi)$	$[-1, 1]$	

Nota. Recuperado de Sánchez (2015)

Redes neuronales convolucionales

Al trabajar con imágenes Full HD (1980 x 1080 pixeles) de alta calidad en redes neuronales convencionales surge un problema computacional debido a que cada neurona está totalmente conectada con un pixel de entrada esto vuelve a los tiempos de muestreo y entrenamiento enormes, por lo que es evidente la necesidad de utilizar algún otro tipo de sistema que permita solucionar este problema.

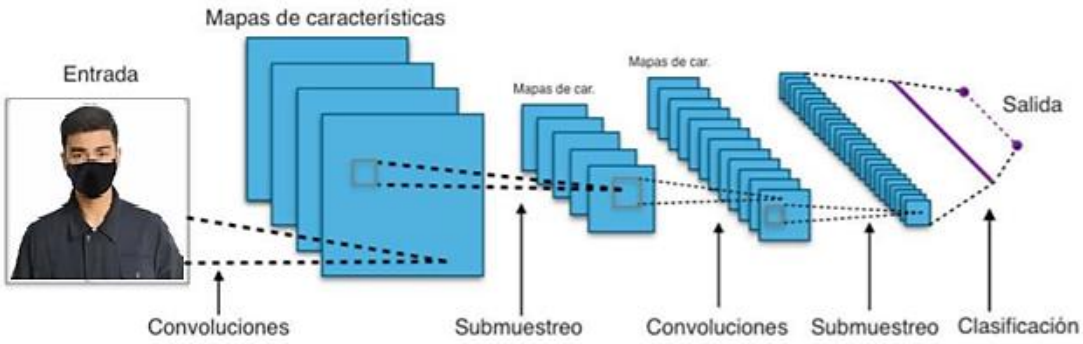
Las redes neuronales convolucionales su principal característica es que presentan una arquitecta multicapa, en donde capa está sujetada a una cantidad determinada de convoluciones con activaciones no lineales (Relu, Tanh o sigmoidal) y pooling (submuestreo) al final, tiene una serie de capas completamente conectadas como una red de perceptrón multicapa. Para obtener resultados como respuesta a datos de entrada, estas entradas pueden

ser una imagen $m\ x\ m\ x\ r$, donde m es tanto altura como ancho y r son los canales de la imagen. Este tipo de conexiones multicapas convolucionales es típicamente una analogía de la corteza cerebral de un mamífero, en la cual la respuesta independiente de cada una de las neuronas puede ser representadas matemáticamente como una convolución. Las capas convolucionales disponen de k filtros denominados *kernel* de dimensión $n\ x\ n\ x\ q$ en donde los valores $n\ x\ q$ son establecidas por el programador. Cada filtro después de convolucionar genera un mapa característico de tamaño $(m-n+1)\ x\ (m-n+1)\ x\ p$, luego el mapa de características se reduce en el paso de agrupación (*pooling*) con la operación de agrupación máxima (*max-pooling*) en las regiones de interés, para evitar el sobre entrenamiento antes o después se aplican funciones de activación no lineal como se observa en la Figura 8. (Suarez,2017)

Como se representa en la Figura 9, la disposición general de la estructura de una red neuronal convolucional se describe a continuación.

Figura 9

Arquitectura general red neuronal convolucional



Nota. Recuperado de Suárez (2017)

Capa de entrada: Una capa de entrada, también llamada sensorial o partida, consta de neuronas que reciben datos que alimentan la red neuronal. En la Figura 10 se observa los datos que alimentan a la red de dimensión $m\ x\ m\ x\ r$, donde m es el ancho y altura mientras tanto que la r son los números de canales.

Figura 10

Imagen de entrada

a. Mask b. NoMask

Nota. En el lado derecho de la imagen se observa a una persona con mascarilla mientras que en la izquierda sin la mascarilla, estas serán las categorías para el cual el sistema será desarrollado.

Capa convolucional: Esta capa es el centro de las DCNNs, en estas capas se realizan los cálculos de convoluciones de la información de entrada (Imágenes) en conjunto con un número determinado de filtros de aprendizaje, cada filtro debe convolucionar con las neuronas adyacentes para generar un mapa de características en su salida. En general, el mapa de características viene representado matemáticamente por la siguiente Ecuación 8(Pacheco, 2017).

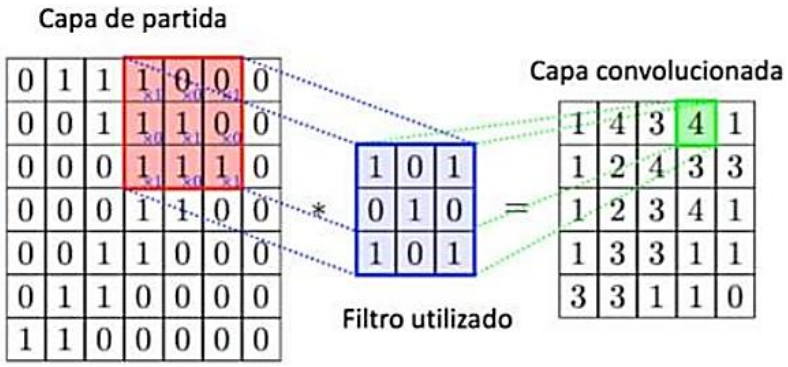
$$y_j^{(l)} = f\left(\sum_{i \in M_j} y_i^{(l-1)} k_{ji}^{(l-1)} + b_j\right) \qquad (8)$$

Donde, la variable l es la capa actual y el tamaño de entrada está representada por M_j, la ventaja principal de la operación de convolución es independientemente al tamaño de la entrada, además otro aspecto importante es disminuir la carga computacional del sistema, para lograr este objetivo, la capa de convolución restringe el número de conexiones posibles entre las neuronas de la capa oculta y los elementos de la imagen de entrada, el objetivo es que cada neurona de la capa oculta esté vinculada a un pequeño subconjunto de los datos de entrada de la imagen total.. Este procedimiento está basado en el sistema visual biológico en donde las neuronas responden a ciertas áreas específicas de la imagen de ingreso.

Para comprender a fondo el comportamiento interno de esta etapa de convolución se describe lo siguiente, ingresa una imagen de entrada con dimensiones $m\ x\ m\ x\ r$, a esta imagen se superpone un filtro *kernel* de dimensión $n\ x\ n\ x\ q$, los elementos de la imagen como la del filtro convolucionan mediante operaciones de productos y sumas arrojando resultados y almacenando en la matriz de características de dimensión $(m-n+1)\ x\ (m-n+1)\ x\ p$ cómo se puede observar en la Figura 11.(Rodriguez,2019)

Figura 11

Proceso de convolución

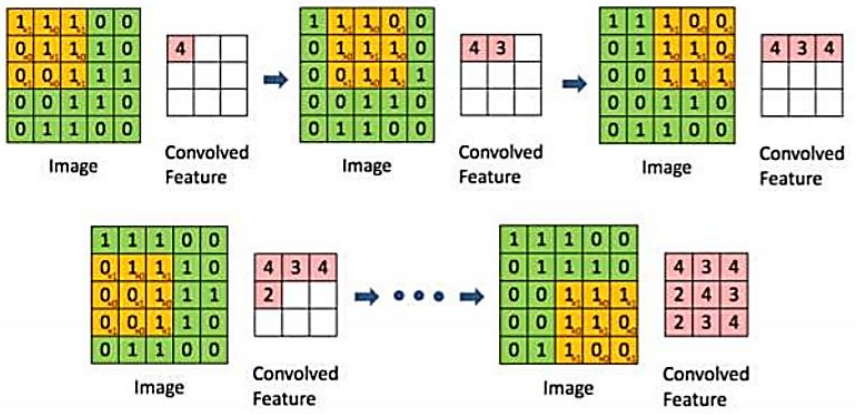


Nota. Recuperado de Rodríguez (2019).

De esta forma, el proceso se repite sobre toda la imagen, moviéndose linealmente de forma horizontal y bajando una unidad vertical hasta llegar al borde de la imagen. Una vez que se cubra la imagen completa, obtendrá la matriz de activación completa que incluye la información relevante. Este proceso se puede observar en la Figura 12.

Figura 12

Matriz de características



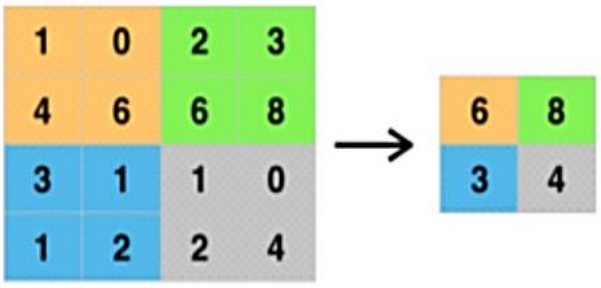
Nota. Recuperado de Suarez (2017).

Capa de submuestreo: Siguiendo el esquema gráfico de la Figura 9, inmediatamente después de recibir el mapa característico de una imagen de entrada mediante operaciones de convolución, la siguiente capa es la capa de submuestreo que tiene como objetivo reducir el tamaño espacial, aunque conlleva desventajas como la pérdida de información, pero también conlleva ventajas para la red, como reducir la carga computacional de las capas posteriores y también reduce la sobreadaptación en la red. Una de las operaciones más utilizadas al momento de realizar la etapa de submuestreo es la denominada *max-pooling,* su desempeño está basado en la segmentación de la entrada en rectángulos y de cada rectángulo conserva el mayor valor o calcula el valor medio a lo largo de la región de la imagen, a continuación, en la Figura 13, se observa como una matriz 4x4 se transforma a una de 2x2 después de haber

realizado la operación de *max-pooling,* con los valores máximos de cada segmentación rectangular creada en la matriz de entrada, obteniendo una matriz resultante de dimensiones considerables menores que la matriz de característica obtenida después de la capa de convolución. (Montoya, 2018).

Figura 13

Operación de submuestreo



Nota. Recuperado de Pacheco (2017)

Al realizar el proceso de *max-pooling, la* salida tiene la siguiente Ecuación 9 matemática:

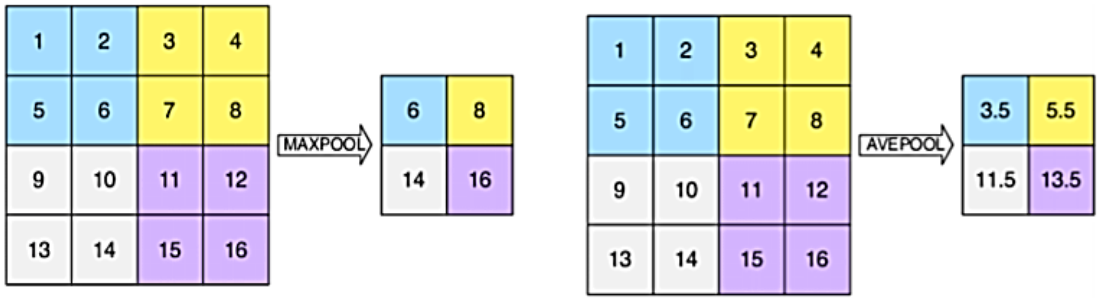
$$y_j^{(l)} = f\left(\beta_j down\left(y_j^{(l-1)}\right) + b_j\right) \qquad (9)$$

Donde, la operación $down$ representa una función de submuestreo

Es importante destacar que en la etapa de *pooling* existe dos tipos de submuestreo: *Max-pooling* (valor máximo) y *average-pooling* (valor promedio) en la siguiente Figura 14, se detalla la diferencia de aplicar un método u otro.

Figura 14

Clasificación de pooling



Nota. Recuperado de Suarez (2017).

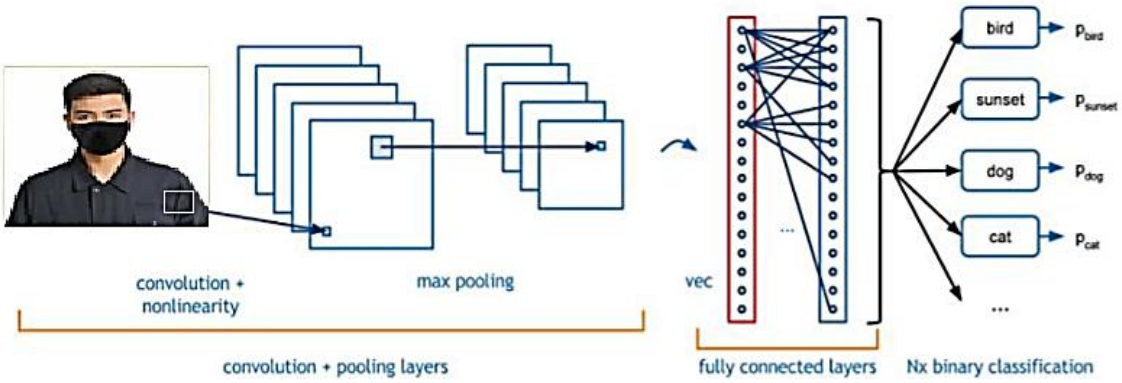
Por lo tanto, esta capa tiene como finalidad reducir aún más la carga computacional en el sistema mientras ayuda a caracterizar la imagen preservando los recursos predominantes en ella.

Capa completamente conectada (full-connected): Después de combinar las capas de convolución y agrupación máxima, la capa completamente conectada se usa generalmente al final de las redes neuronales convolucionales, que consisten en redes neuronales artificiales y el número total de neuronas corresponderá directamente al número de objetos que desea predecir en general, esta capa está orientada a la clasificación de imágenes para dos clases con y sin máscaras. (Pacheco, 2017).

Por otro lado, queremos mencionar en este apartado la implementación de la red convolucional con la red neuronal. La red del proyecto consta en particular de las siguientes capas: entrada, convolución, agrupación, conexión completa. Todo esto forma parte de la red neuronal convolucional orientado para la clasificación de imágenes. En la Figura 15 se observa el esquema general de la red implementada.

Figura 15

Diagrama de Full-connected



Nota. Recuperado de Suarez (2017)

En general, podemos mencionar que DCNNs es un método de deep learning que toma un input y lo envuelve de un conjunto de filtros, para obtener pesos que representen confiabilidad al otorgar una decisión como salida. Además de los pesos, las redes convolucionales cuentan con capas de max-pooling cuyo objetivo es reducir las dimensiones de los datos obtenidos en las capas de convolución y, finalmente, las capas están completamente conectadas, las cuales son las encargadas de calcular los resultados en la decisión final de cada una de las clases que quieres reconocer. (Montoya, 2018).

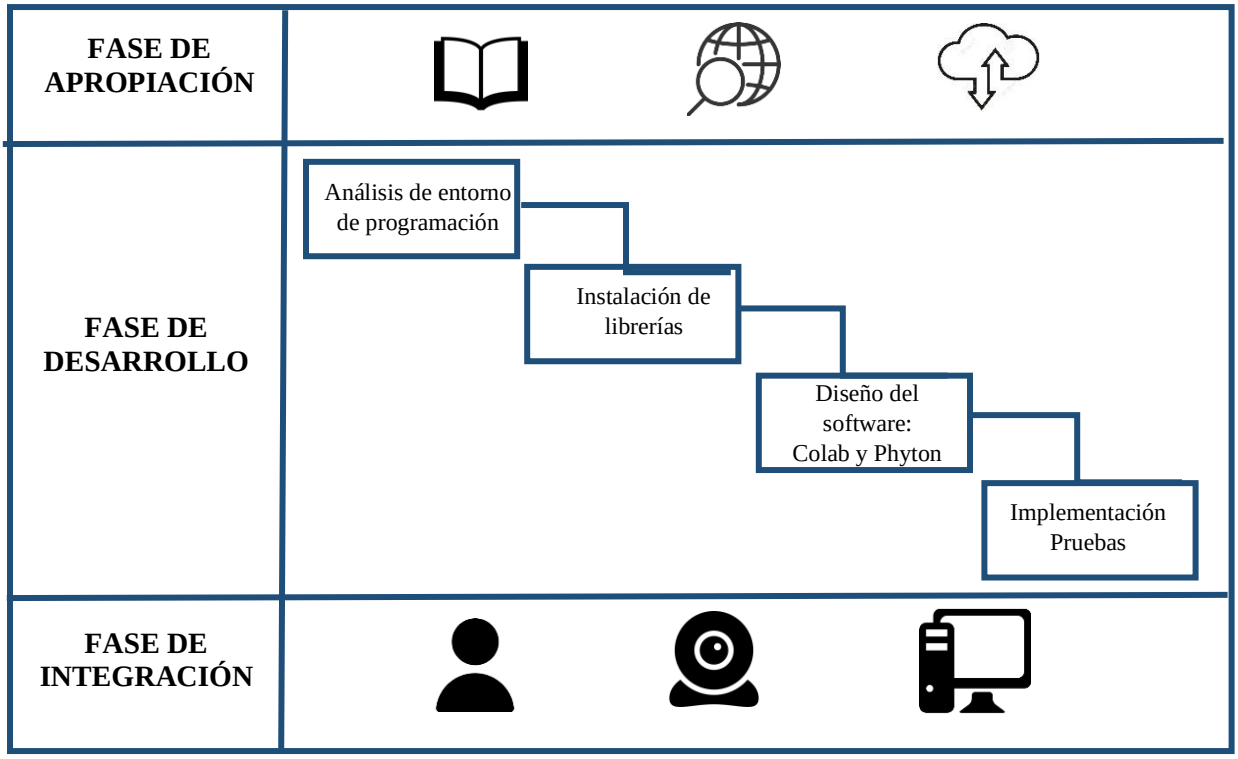
Capítulo III

Metodología

Para llevar a cabo este proyecto en forma rápida, se utilizó la metodología de desarrollo incremental, este método permite dividir el proyecto en diferentes bloques temporales las mismas que serán validadas al final de cada una de las etapas. Por lo tanto, al trabajar de esta forma es posible verificar que los resultados del proyecto son los esperados sin la necesidad de tener el desarrollo de la programación terminada y así poder identificar oportunamente errores en sintaxis, versiones de librerías o mejoras al proyecto en cada fase. En la Figura 16 se observa el esquema general de la metodología implementada en el desarrollo del proyecto. (Cobo,2017)

Figura 16

Metodología de desarrollo incremental



Nota. La planificación del proyecto está dividida en tres fases: formación (investigación requerida para el proyecto), desarrollo (consiste de cuatro iteraciones necesarias para el desarrollo del software de reconocimiento de mascarillas) e integración (consiste en la integración del software y una cámara web en tiempo real).

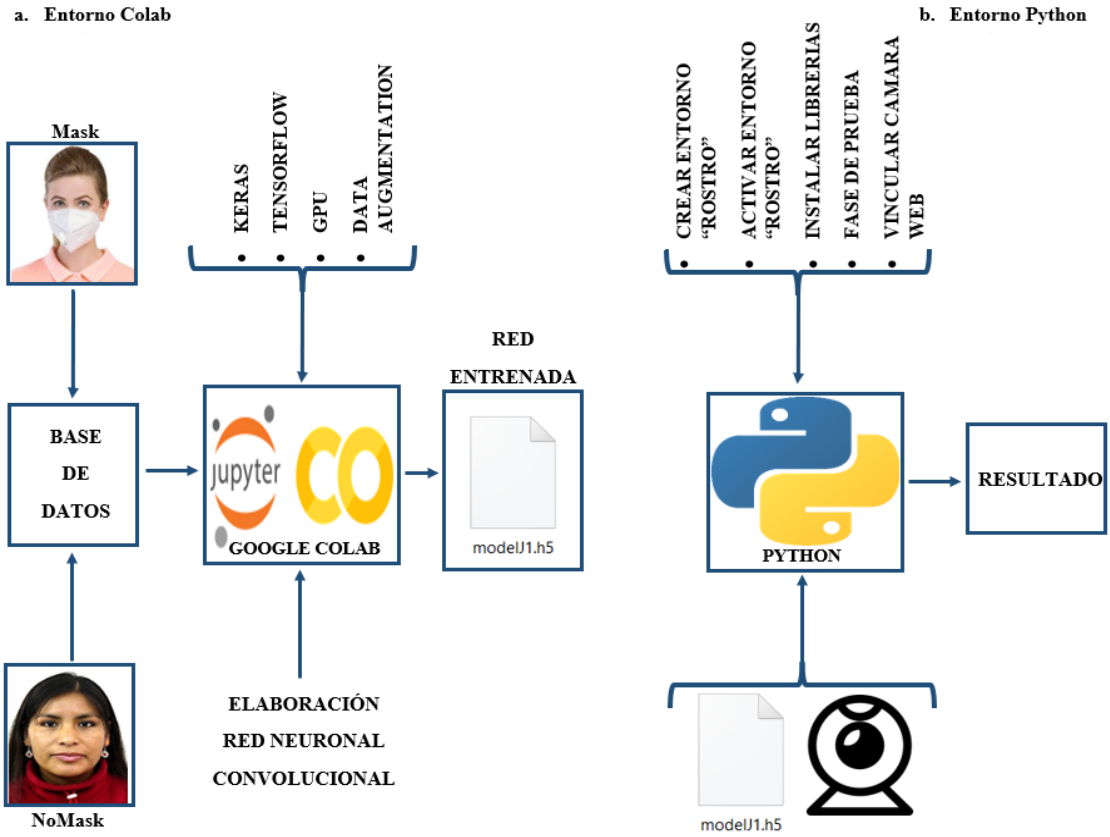
Diseño del sistema de reconocimiento de uso de mascarilla usando RNC

Para que una imagen sea analizada correctamente en las etapas posteriores del sistema, es importante realizar un proceso de identificación y compensación de puntos de interés que son importantes para el entrenamiento de la red neuronal, a esto se lo define como preprocesamiento de imágenes. El objetivo del estudio de visión por computadora e inteligencia artificial es de dotar al ordenador la facultad de predecir algo, que comúnmente un ser humano lo puede realizar. Para el reconocimiento de uso de mascarilla de este proyecto está basado principalmente en la aplicación de redes neuronales convolucionales que son un derivado de la Deep Learning descrito anteriormente, ya que esta inteligencia artificial es utilizada para la clasificación y reconocimiento de imágenes.

En este apartado se describen los pasos para la instalación de los programa, librerías y versiones de Python usados para llevar a cabo este proyecto, como se observa en la Figura 17.

Figura 17

Esquema general del proyecto



Nota. Esquema general para el desarrollo del proyecto, en el apartado (a) se detalla el entorno y las herramientas necesarias para la elaboración de la red neuronal(.h5) mientras que en el apartado (b) se detalla la segunda parte del proyecto para vincular con el archivo .h5 y una cámara web.

Base de Datos

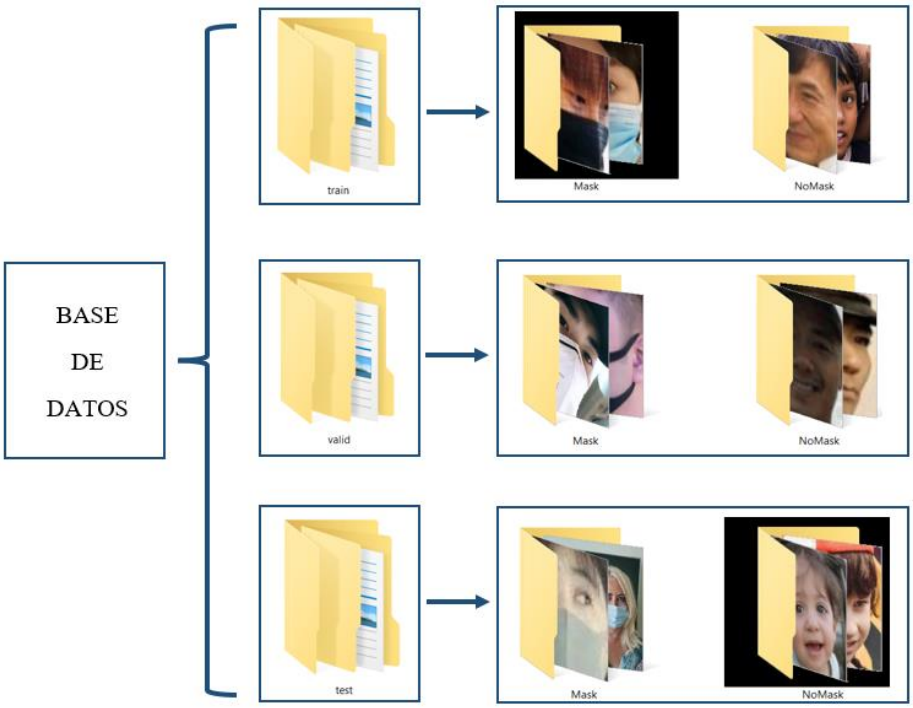
Para iniciar con el desarrollo del sistema de detección de mascarilla es esencial recopilar imágenes con y sin mascarilla las mismas que deben ser clasificadas y almacenadas en las siguientes carpetas:

- NoMask
- Mask

Para un entrenamiento adecuado y así reducir el nivel de falsos positivos se recopila 1597 imágenes en cada una de las carpetas, esto brinda un total de 3194 imágenes con y sin mascarilla, pero debido a la arquitectura del comando de entrenamiento de la red neuronal convolucional en Google Colab es necesario crear tres carpetas adicionales y dividir la totalidad de imágenes en cada una de ellas, estas carpetas están denominadas de la siguiente forma: Train, Valid y Test las mismas que contiene el 70%, 20% y 10% de imágenes con y sin mascarilla respectivamente. En la Figura 18 se observa el esquema general de las carpetas.

Figura 18

Base de Datos



Nota. Esquema general de la clasificación de la base de datos.

Una vez realizada la descarga, clasificación y división de las imágenes por carpetas, se procede a subir la base de datos a Google Drive, la cual se enlazará posteriormente con Google Colab para la elaboración del sistema de detección de mascarilla.

Entorno de Google Colab

Google Colab o Google Colaboratory es un entorno de programación en línea que permite a cualquier cliente digitar y ejecutar comandos de Python en el navegador. Es un entorno adecuado para labores de aprendizaje profundo (Deep Learning), estudio de datos y educación. A partir de un criterio más técnico, Colab es un servicio portátil de Jupyter Notebook en línea abierto que no necesita instalación y que da ingreso gratuito a recursos informáticos como procesadores de imágenes, ideal para el entrenamiento de una red neuronal, estos entornos de ejecución o acelerador de hardware pueden ser: GPUs (Graphics Processor Unit) o TPUs (Tensor Processing Unit), los mismos que acortan el tiempo de entrenamiento a diferencia de usar una tarjeta convencional CPUs (Central Processing Unit).

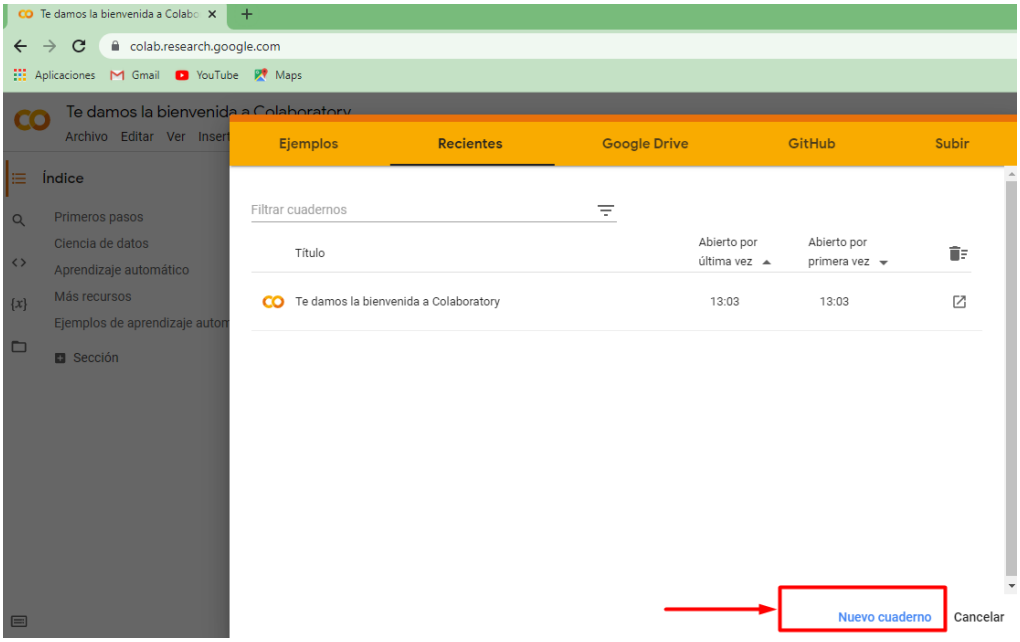
Creación y configuración de entorno

Para la creación de un nuevo proyecto en Google Colab se recomienda seguir las siguientes instrucciones.

Inicialmente se debe ingresar a un navegador de internet (Google Chrome) y buscar https://colab.research.google.com e inmediatamente registrarse con una cuenta de Google, posterior al registro, se apertura una ventana de bienvenida y un botón de crear un nuevo cuaderno de proyecto, como se observa en la Figura 19, después de presionar el botón es necesario colocar un nombre.

Figura 19

Página principal de Google Colab

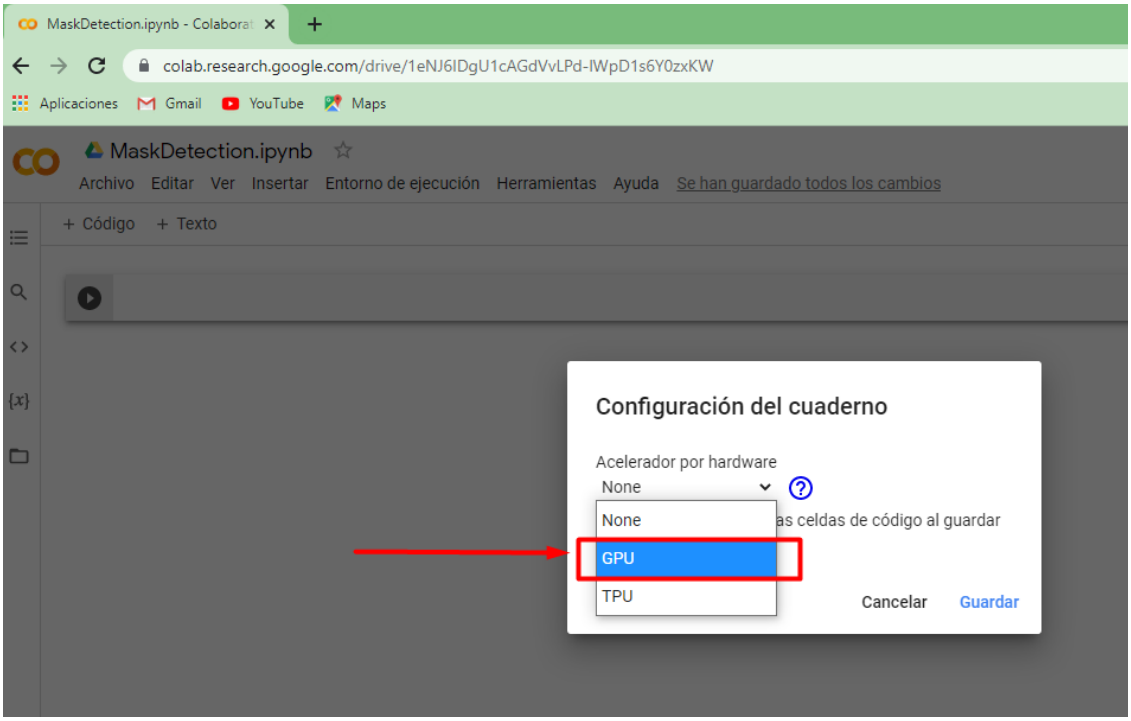


Nota. Creación de un nuevo cuaderno Jupyter Notebook.

Una vez creado el cuaderno de jupyter notebook, se despliega la ventana de entorno de programación con el nombre que se dio inicialmente. Antes de realizar la programación es importante realizar el siguiente paso, verificar que el entorno de ejecución esté activado en GPU, para cambiar el entorno se puede elegir *cambiar tipo de entorno de ejecución* después elegir *acelerador por Hardware* y por último seleccionar *GPU,* como se observa en la Figura 20, el objetivo de realizar esta configuración en un entorno de programación como Google Colab es aprovechar las tarjetas de procesamiento de video de los servidores de Google y así de esta manera brindar más potencia computacional al momento de realizar cálculos de multiplicaciones matriciales, descenso de gradientes o el Backpropagation en las redes neuronales, esto sumado a la mejora de la velocidad de procesamiento convirtiendo un entrenamiento que lleva días a horas.

Figura 20

Configuración del acelerador de Hardware



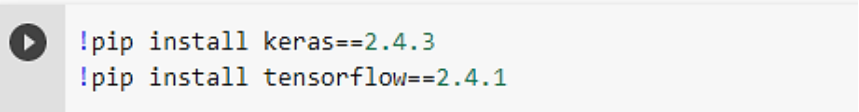
Nota. Creación de un nuevo cuaderno Jupyter Notebook y selección del entorno de ejecución.

Instalación de librerías

Una vez que se realizó el paso anterior, se deben instalar las librerías precargadas de Keras y Tensorflow para la creación de modelos de Deep Learning. La combinación de las dos librerías posibilita edificar redes neuronales complicadas de forma fácil, mediante la especificación de todos los recursos que la conforman, desde la composición física de la red en capas hasta las funciones de activación, los criterios de evaluación y otras funciones, para instalar las librerías en jupyter notebook se utiliza una sintaxis especial (**!pip**) la cual permite ejecutar comandos de instalación de paquetes de Python según sea la aplicabilidad del proyecto, es esencial instalar las librerías correspondientes para la creación de la red como se muestra en la Figura 21.

Figura 21

Librerías keras y tensorflow



Nota. Librerías con las versiones adecuadas.

Enlazar con Google Drive

Se debe recordar que Google Colab es un entorno de programación virtual lo cual implica que la base de datos también debe estar sujeta a un medio de almacenamiento virtual como Google Drive, la ventaja principal de enlazar estos entornos es que permite subir o descargar archivos. Para montar Google Drive en la máquina escribimos la sintaxis que se muestra en la Figura 22.

Figura 22

Sintaxis de unidad Drive

Nota. Sintaxis.

Esto abre una URL que exige un código para autorizar a Colab a vincular con Drive, una vez copiado la dirección, se debe pegar en la autorización de código para tener acceso a los archivos, como se observa en la Figura 23.

Figura 23

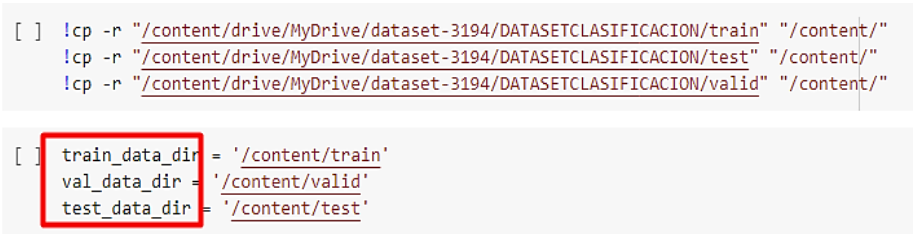
Montaje de unidad Drive

Nota. Solicitud de código.

Una vez que el proceso acaba, Google Drive se ha montado en la dirección indicada mediante la sintaxis de programación: (/content/drive) y de esta manera se puede acceder a los archivos almacenados en la nube. Para direccionar a Colab con las carpetas especificas donde están almacenadas las imágenes (Data Base) en Drive es necesario realizar previamente la escritura de los siguientes comandos como se observa en la Figura 24.

Figura 24

Dirección especifica de la Data Base en Drive



Nota. Dirección especifica de las carpetas de imágenes, asignación a nuevas variables locales para el entrenamiento

Con esto se asegura que los archivos específicos para el entrenamiento del sistema estén almacenados en las variables locales de Colab las cuales se indican encerradas en un rectángulo rojo de la Figura 24.

Generador de imágenes

Después de almacenar los archivos de train, valid y test cada una de ellas con subcarpetas de Mask y NoMask de Drive en las variables locales: *train_data_dir, val_data_dir* y *test_data_dir*, se debe redimensionar las imágenes a un mismo tamaño para su posterior preprocesamiento y así evitar una sobre carga computacional por el tamaño variado de cada foto del data base, para este sistema se establecieron las imágenes de entrada en una dimensión de $224\ x\ 224$ pixeles correspondiente al ancho y alto respectivamente.

El siguiente paso es generar imágenes adicionales, para eso la librería Keras cuenta con la clase *ImageDataGenerator* compatible con los siguientes formatos: png, bmp y jpeg. Su funcionamiento se basa en que cada época las imágenes de Train y Valid sufren ciertas transformaciones que incluyen rotación, zoom, escalado y desplazamiento horizontal o vertical de tal manera que está creando nuevos datos (Data augmentation) de ingreso sin variar la cantidad real de la data base y sin generar imágenes totalmente diferentes a las originales. Así, el modelo aprendido podrá ser más robusto y preciso, debido a que está entrenado en diferentes variaciones de la misma imagen.

A continuación, en la Figura 25, se observa la sintaxis de preprocesamiento de las variables *train_data_dir* y *val_data_dir* y las nuevas localidades de almacenamiento de las imágenes denominadas: *train_generator* y *val_generator* después de aplicar ImageDataGenerator.

Figura 25

Uso de la clase ImageDataGenerator de Keras


```
# DATA GENERADOR DE IMAGENES

from keras.preprocessing.image import ImageDataGenerator
from keras.applications.imagenet_utils import preprocess_input

train_datagen = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.2,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=False,
    preprocessing_function=preprocess_input)

val_datagen = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.2,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=False,
    preprocessing_function=preprocess_input)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(width_shape, height_shape),
    batch_size=batch_size,
    #save_to_dir='',
    class_mode='categorical',shuffle=False)

val_generator = val_datagen.flow_from_directory(
    val_data_dir,
    target_size=(width_shape, height_shape),
    batch_size=batch_size,
    #save_to_dir='',
    class_mode='categorical',shuffle=False)
```

Nota. Los cuadros rojos indican los parámetros que están sujetos las imágenes como rotación, zoom, etc, mientras que el azul indica la nueva variable en donde están almacenados los datos para el posterior entrenamiento

Diseño de red

Una vez cumplido con los pasos anteriores, es momento de diseñar la red y para cumplir con este propósito se necesita de una red preentrenada que fue diseñada para solucionar problemas de carácter general. Existen varios tipos de redes neuronales con sus respectivas arquitecturas empezando desde la YoloNet pasando por la Vgg16 – Vgg19 hasta la ShuffleNet. Un plan para lidiar en redes neuronales profundas es utilizar redes antes preentrenadas con bases de datos gigantes y orientar a solucionar el problema planteado de interés. Después de un análisis de las diferentes arquitecturas de las redes profundas y su dependencia para la clasificación de imágenes compatibles con Keras y Tensorflow se obtiene la siguiente lista: (López, 2018)

- Xception
- InceptionV3
- ResNet50
- VGG16
- VGG19
- MobileNet

Para este proyecto se decide por la red VGG 16, la cual está formada por 16 capas y preentrenada con la base de datos de ImageNet para una clasificación de 1000 clases, la red VGG16 a diferencia de las demás arquitecturas reemplaza los grandes filtros de Kernels por un conjunto de filtros de $3\ x\ 3$, además de las siguientes ventajas: (Pérez, 2020)

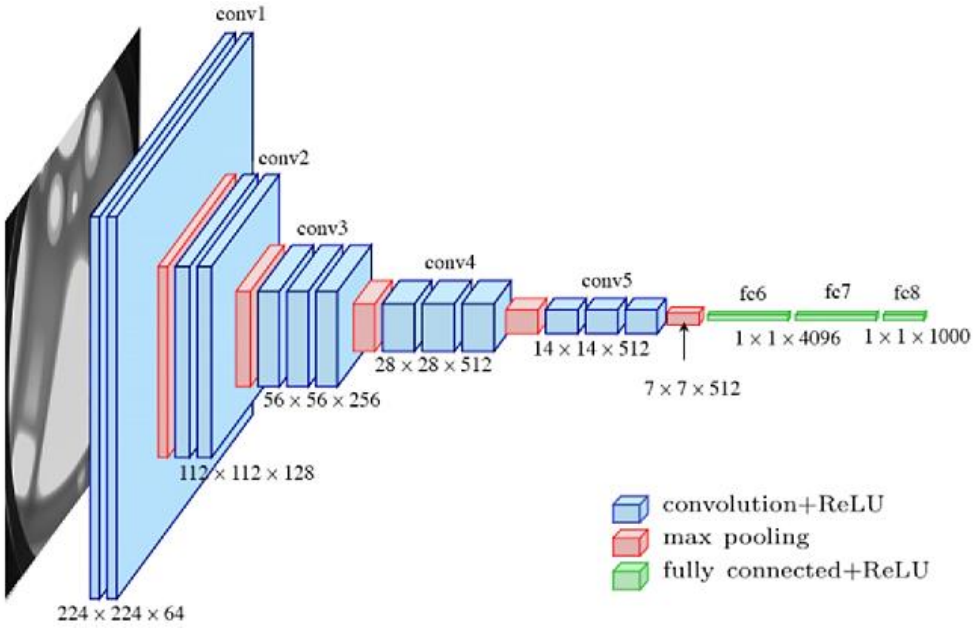
- Posee una arquitectura fácil de comprender

- Contiene pocas capas convolucionales: 3 capas densas y 13 capas convolucionales, en total 16 capas.
- La red está disponible para Keras
- Una red con menos capas de cálculo lo cual implica menos memoria y requerimientos computacional.

En la siguiente Figura 26, se aprecia la arquitectura de la Red VGG16.

Figura 26

Arquitectura de la Red VGG16



Nota. Recuperado de Pérez (2020)

En la Figura 27 se muestra la sintaxis para la transferencia de aprendizaje de la red VGG16 al entorno de programación de Google Colab. Antes de proceder a entrenar al sistema se debe realizar configuraciones previas las cuales permiten adaptar la arquitectura de la red al proyecto. Para empezar, se establecen las dimensiones de $224\ x\ 224$ pixeles de entrada. Para

homologar el tamaño de imágenes, este apartado se analizó en la sección generador de imágenes, seguido de la asignación del modelo a una variable local *model2*, una vez cargado el modelo se deben editar las últimas capas (capas densas) que son responsables de combinar propiedades de las capas convolucionales y esto ayuda en la clasificación final. Entonces, una vez que el modelo VGG16 se utiliza en otro grupo de datos, es viable que tengamos que sustituir cada una de las capas densas. En esta situación añadimos otra capa densa y una capa de caída para eludir un sobreajuste y de esta forma solo optimizamos los pesos de las nuevas capas de clasificación agregadas.

Figura 27

Sintaxis de programación de la red VGG16

```
from keras.applications.vgg16 import VGG16

image_input = Input(shape=(width_shape, height_shape, 3))
model2 = VGG16(input_tensor=image_input, include_top=True,weights='imagenet')
#model2.summary()
last_layer = model2.get_layer('block5_pool').output
x= Flatten(name='flatten')(last_layer)
x = Dense(128, activation='relu', name='fc1')(x)
x = Dense(128, activation='relu', name='fc2')(x)
out = Dense(num_classes, activation='softmax', name='output')(x)
model = Model(image_input, out)
#model.summary()
# congelar todas las capas excepto las densas
for layer in model.layers[:-3]:
  layer.trainable = False
model.summary()

model.compile(loss='categorical_crossentropy',optimizer='adadelta',metrics=['accuracy'])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467904/553467096 [==============================] - 8s 0us/step
Model: "model"
```

En la Figura 28 se observa la arquitectura de la red neuronal artificial, la misma que está formada por capas de convolución, MaxPooling y aplanamiento (Flatten). Se debe mencionar que la función de activación para esta red es la Función Softmax ideal para clasificación de imágenes la cual podemos observar en la Figura 27, adicionalmente en el rectángulo rojo se puede observar que la última capa se define el número de salidas correspondientes a Mask y NoMask respectivamente.

Figura 28

Arquitectura de la Red Neuronal VGG16

Model: "model_2"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 128)	3211392
fc2 (Dense)	(None, 128)	16512
output (Dense)	(None, 2)	258

Nota. Arquitectura de la red,.

Entrenamiento del Sistema

Posteriormente a la configuración de los parámetros de la red, se deben agregar nuevas capas densas y definir el número de clases a la cual la red va estar sujeta para la clasificación de imágenes, a continuación se procede el entrenamiento del sistema de detección de mascarilla para lo cual se aplican las líneas de programación que se detallan en la Figura 29., después de acciones de prueba y error se define el número de épocas (ciclo de entrenamiento) en 100 para mejorar la precisión y acercar la curva de predicción al valor de 1 que es equivalente al 100% como se observa en la Figura 30. Es necesario mencionar que el sistema alcanza una predicción de validación del 0.9903 equivalente al 99.03%, como se observa en la Tabla 2 de la sección 4.

Figura 29

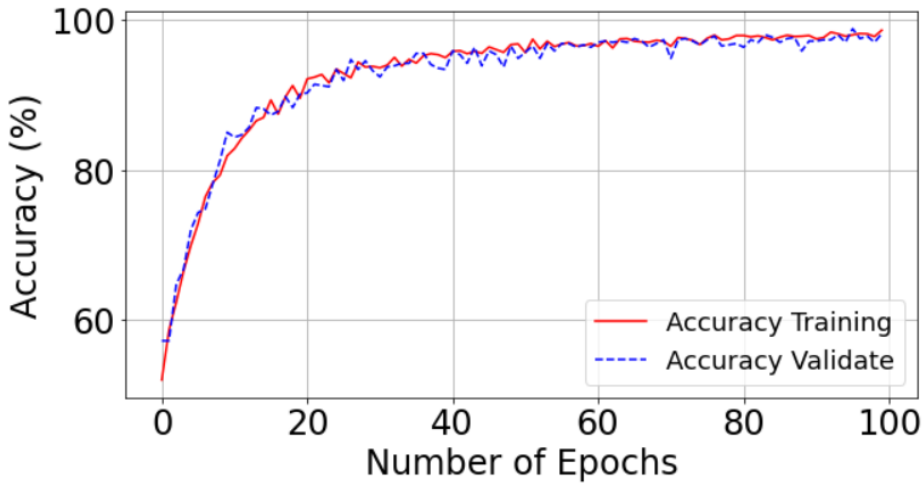
Sintaxis de programación para entrenamiento

```
model_history = model.fit_generator(
    train_generator,
    epochs=epochs,
    validation_data=val_generator,
    steps_per_epoch=nb_train_samples//batch_size,
    validation_steps=nb_val_samples//batch_size,
    callbacks=[tensorboard_callback])   #pasos entre cada epoca
```

Nota. Sintaxis de épocas y pasos por epocas.

Figura 30

Curva de entrenamiento



Nota. Curva para una cantidad de 100 epocas.

Instalación de Python 3.7

Python es un entorno de programación que se encuentra presente en diversas aplicaciones y para diferentes plataformas como iOS, Android, Linux, Windows o Mac. Python es un lenguaje de programación orientado a objetos, funcional e imperativo, por lo que se denomina un lenguaje multiparadigma y multiplataforma en donde predomina por su código legible y limpio. Una de las causas de su éxito es que cuenta con una licencia de código abierto que posibilita su implementación en cualquier escenario.

Para el desarrollo del sistema de detección de mascarilla se utiliza Python el cual es un lenguaje multiplataforma y de código abierto gratuito: A continuación, se detallan los pasos para su instalación y configuración.

Para iniciar la instalación de Python se ingresa a la página de oficial que se indica en la Figura 31 https://www.python.org/downloads/windows/, y se descarga la aplicación Python versión 3.7.

Figura 31

Página principal de Python

Nota. Descarga de Python Versión 3 para un sistema operativo de Windows.

Los pasos siguientes para la instalación se los realiza como cualquier otra aplicación aceptando los términos y condiciones del fabricante, pero previamente en el asistente de instalación debe activarse el casillero de agregar Python como una variable de entorno PATH a CMD (Command Prompt) y esto permite que desde la terminal de Windows se pueda acceder a Python, como se observa en la Figura 32. Con todos los pasos realizados de una manera correcta se finaliza la instalación del entorno de programación.

Figura 32

Asistente de instalación

Nota. Activación del casillero para agregar Python 3.7 como variable de entorno en Windows.

Creación de entorno virtual ¨Face¨

En Python un entorno virtual no es más que un entorno de trabajo aislado del sistema principal o de otros espacios. La creación de uno o varios entornos nos posibilita realizar una cierta aplicación, sin influir el sistema en otros procesos. En ciertas ocasiones la necesidad de tener diferentes espacios de trabajos, con diversas variantes del intérprete y diversas librerías, requiere de una búsqueda de soluciones frecuentando por las más comunes que son utilizar diversas computadoras o máquinas virtuales, cada una con una instalación distinta. Sin embargo, Python provee de una herramienta sencilla y eficaz para facilitar aquel trabajo, sin necesidad de recurrir a resoluciones más complejas. Hay muchas posibilidades de espacios virtuales accesibles, pero algunos de los más reconocidos son: poetry, conda, pipenv, virtualenv

Para realizar el proyecto se inclina por utilizar el instrumento que se denomina *conda*. Por consiguiente, *conda* es un medio que posibilita producir espacios virtuales de Python. Un ámbito virtual que consta de un intérprete (versión de Python) en compañía de todos los módulos requeridos a instalar.

Para comenzar creando un entorno virtual en Python, se accede primeramente a comands prompts (Cmd) para posteriormente escribir la siguiente sintaxis:

Instalación de un nuevo entorno en Python:

conda create -n Rostro anaconda python=3.7.7

Activación de entorno:

conda activate Face

Instalación de ipykernel: Este paquete proporciona el kernel de Python para Jupyter notebook

conda install ipykernel

Activación de ipykernel en notebook jupyter:

python -m ipykernel install --user --name Face --display-name "Face"

Instalación de librerías requeridas para el proyecto:

pip install tensorflow==2.4.1

pip install jupyter

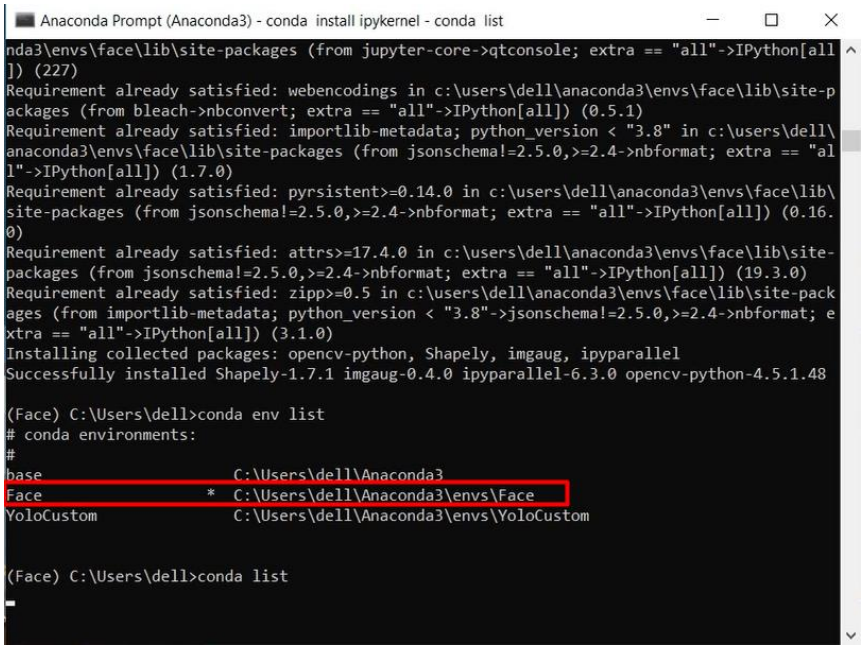
pip install keras==2.4.3

pip install numpy scipy Pillow cython matplotlib scikit-image opencv-python h5py imgaug IPython[all]

Una vez finalizado todos los pasos de una manera correcta, se dirige a la pantalla de Anaconda y se digita el siguiente comando *conda env list*, el cual presenta una lista con los entornos virtuales creados, como se observa en la Figura 33 en el rectángulo rojo.

Figura 33

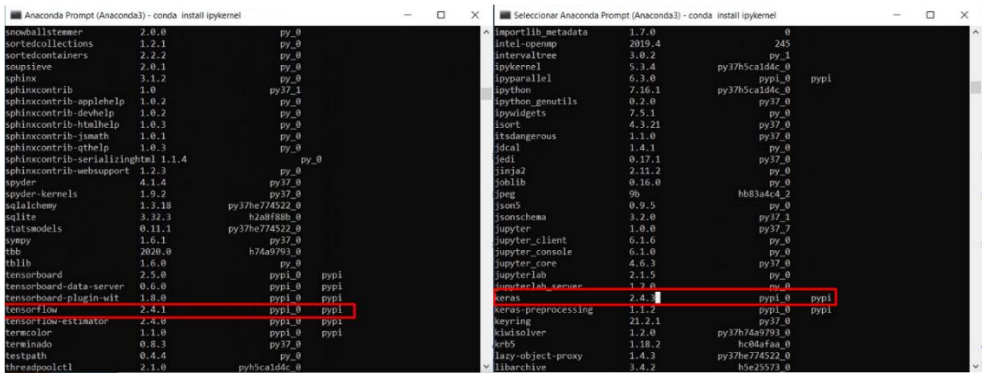
Lista de entornos virtuales en Python



En la Figura 34, se puede observar los paquetes instalados en el entorno virtual ¨Face¨ cómo las versiones requeridas para el desarrollo del sistema de detección de mascarilla.

Figura 34

Librerías instaladas en el entorno Face



Nota. Versiones de librerías instaladas en el entorno Face.

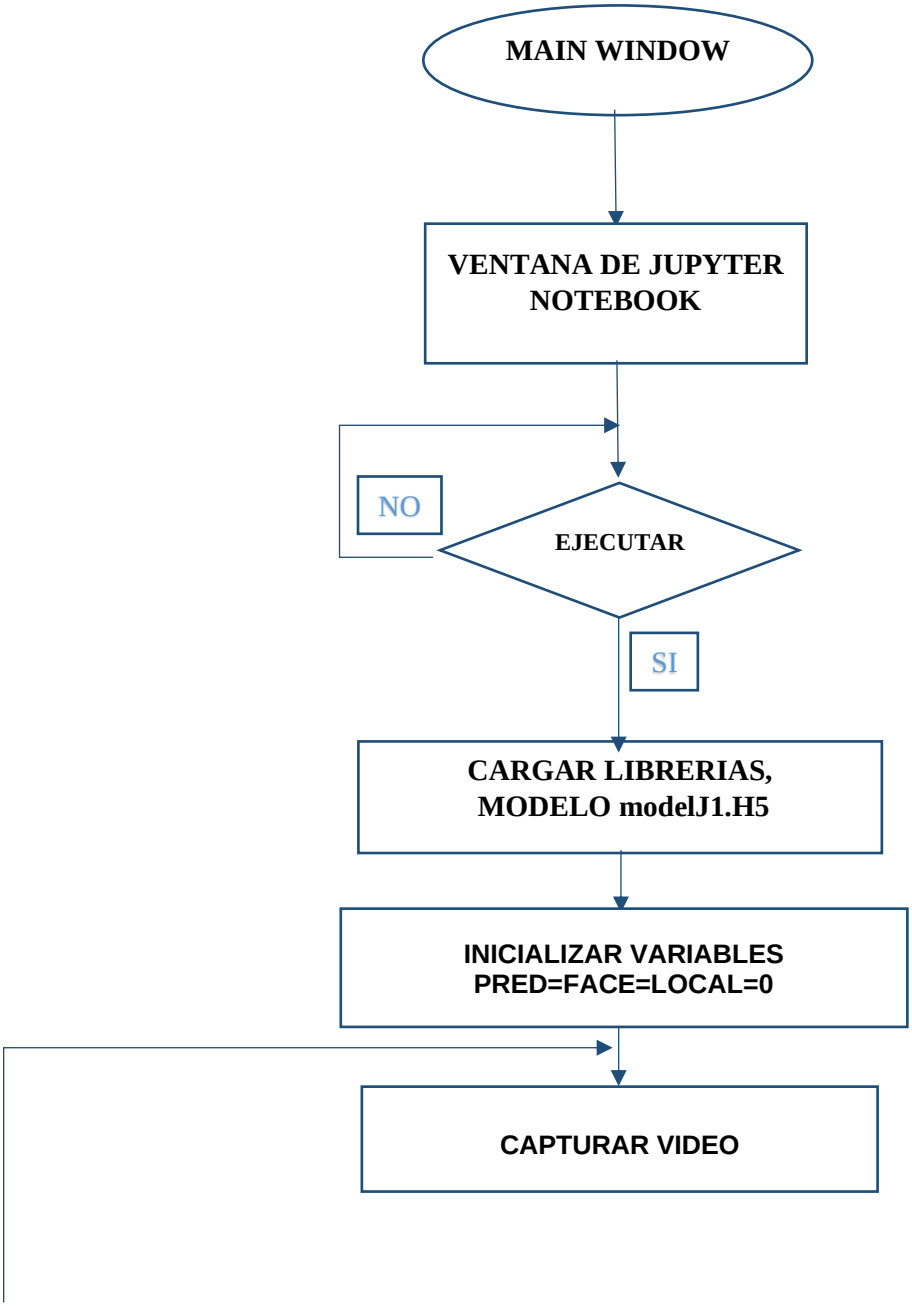
Diagrama del sistema del sistema de detección

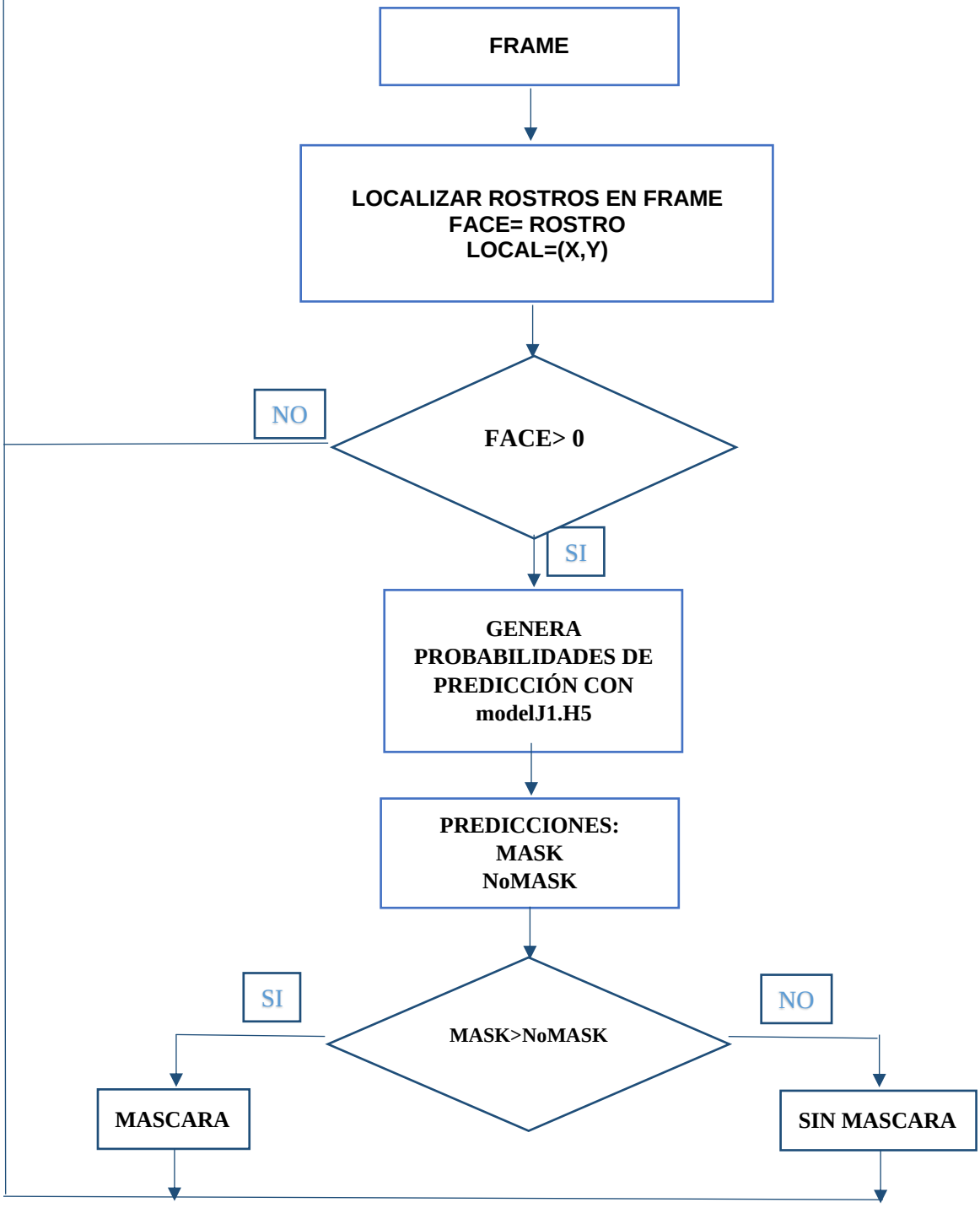
La Figura 35 muestra el diagrama de flujo de la programación en jupyter notebook, para iniciar el procesamiento y detección de mascarillas se debe pulsar el botón de ejecutar, una vez pulsado se cargará automáticamente las librerías, la apertura de la cámara y el archivo *modelj1.h5* el cual contiene la red neuronal entrenada en Google Colab, posteriormente cargado los prerrequisitos, se captura un fotograma del video que es transmitido en tiempo real para posteriormente a través del algoritmo, recortar las regiones de interés como es el rostro de cada frame. Con estas secciones recortadas y la red neuronal cargada inicialmente se realiza una predicción entre el fotograma y el modelo entrenado de la red, inmediatamente se compara las probabilidades de predicción si la de Mask es mayor que la NoMask se etiqueta como ***MASCARA*** caso contrario ***SIN MASCARA.*** Estos mensajes se mostrarán como texto en la esquina superior izquierda de la pantalla de transmisión del video además el programa realiza un seguimiento del rostro encerrándolo en un cuadro verde o rojo según sea la predicción del sistema de detección de mascarilla.

Adicionalmente se colocó en la misma región de la pantalla un texto en color azul el cual indica la cantidad de fotogramas que el sistema procesa por segundo, es importante mencionar que debido a las especificaciones técnicas de hardware de la computadora en donde se desarrolla la fase de integración, se limita la fluidez de streaming.

Figura 35

Diagrama del flujo del sistema de detección de mascarillas





Se presenta en las figuras 36, 37 y 38, la respuesta del sistema de detección de mascarillas a diferentes imágenes, independientemente del color de la mascarilla, posición y cantidad de rostros captados por la cámara web. En la Figura 36 se indica que la respuesta del sistema es correcta a diferentes tipos de mascarillas, en la Figura 37 se aprecia el

comportamiento del sistema para más de una persona en la cual dos de ellas no utilizan mascarillas, mientras que en la Figura 38 se cambia la posición de los rostros y adicionalmente en uno de ellos se coloca una mascarilla diferente al otro para observar el comportamiento del sistema.

Figura 36

Respuesta del sistema a diferentes mascarillas

Figura 37

Respuesta del sistema para más de una persona

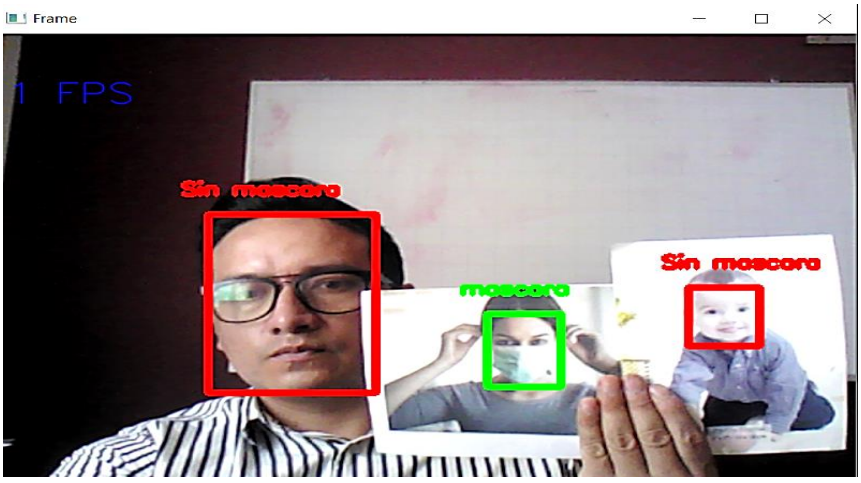
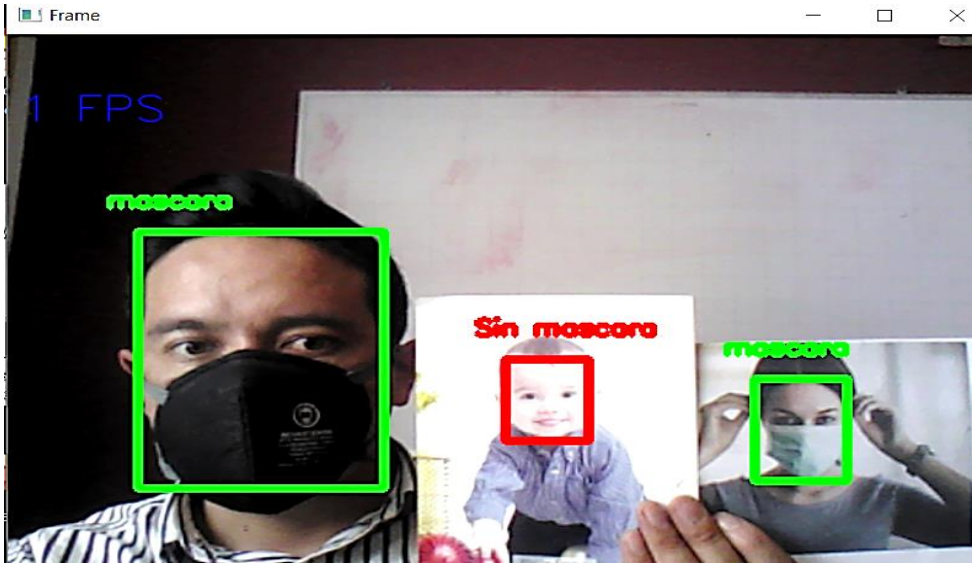


Figura 38

Respuesta del sistema a diferentes mascarillas



Capítulo IV

Pruebas y resultados

En este capítulo se describe la evaluación de la red neuronal convolucional con la arquitectura descrita en el capítulo anterior, la matriz de confusión y las pruebas llevadas a cabo para determinar la eficiencia del sistema de detección de uso de mascarillas utilizando redes neuronales convolucionales. Las pruebas se realizan con diversas imágenes simultáneamente y con modelos entrenados para 25, 50 y 100 épocas en Google Colab descrita en el capítulo anterior, con el fin de revisar el correcto funcionamiento del sistema y verificar el desempeño del algoritmo.

Análisis de la red neuronal convolucional

En esta sección se presenta la evaluación del entrenamiento de la red neuronal convolucional con la arquitectura descrita en el capítulo anterior, adicional se muestra una matriz de confusión usando imágenes diferentes a las utilizadas en el entrenamiento, con las cuales se examina variables de funcionamiento de la red como: precisión, sensibilidad, exactitud y f1-score.

En el campo del aprendizaje profundo y la inteligencia artificial una matriz de confusión es un instrumento que posibilita visualizar el rendimiento de un algoritmo. Esta matriz está formada por columnas y filas las cuales detallan características relevantes como el número de predicción de cada clase mientras que las filas referencia a las instancias en la clase real, en otras palabras, la matriz de confusión nos facilita observar de forma resumida qué tipos de aciertos y errores está teniendo el modelo al momento de pasar por el proceso de entrenamiento. (López, 2018)

Para determinar las variables de la red neuronal mediante de la matriz de confusión es necesario explicar con anticipación ciertos conceptos y cuya relación se observa en la Figura 39.

Figura 39

Esquema de la matriz de confusión

Nota. Matriz de confusión binaria, Recuperado de López (2018)

Verdaderos Positivos = True Positives (TP): Casos correctos clasificados como positivos.

Verdaderos Negativos=True Negative (TN): Casos correctos clasificados como negativos.

Falsos Positivos = False Positives (FP): Casos erróneamente clasificados como positivos.

Falsos Negativos=False Negative (FN): Casos erróneamente clasificados como negativos.

En base a estos 4 principales parámetros, surgen las siguientes métricas de la matriz de confusión como: Acurracy, Precisión, Recall (sensibilidad) y F1-Score las cuales se detallan a continuación:

- Acurracy: Representa el porcentaje de casos predichos correctamente frente al total, se calcula con la Ecuación 10.

$$Acurracy = \frac{TP + TN}{TP + FP + FN + TN} \qquad (10)$$

- Precisión: Representa lo próximo que el resultado está de una predicción del valor verdadero, se calcula con la Ecuación 11.

$$Presición = \frac{TP}{TP + FP} \qquad (11)$$

- Sensibilidad: Representa la cantidad de verdaderos positivos, es la relación entre los casos positivos correctamente clasificados por la red con respecto al total de positivos, se calcula con la Ecuación 12.

$$Recall = \frac{TP}{TP + FN} \qquad (12)$$

- F1-Score: Es la ponderación entre la sensibilidad y la precisión, se calcula con la Ecuación 13.

$$F1 - Score = \frac{2 * (Recall * Precisión)}{Recall + Precisión} \qquad (13)$$

Al finalizar el entrenamiento de la red neuronal con un data base de 3194 imágenes divididas en dos subconjuntos de 1582 por clase se obtuvo una respuesta del sistema que tiene una exactitud (acurracy) del 0.9903% y una pérdida(loss) menos del 0.1 %. El comportamiento de la red para acurracy y loss durante el proceso de entrenamiento se puede observar en las

Figuras 40 y 41 respectivamente. Para corroborar con los valores de la red entrenada se puede también observar en las Figuras 42 y 43 respectivamente.

Figura 40

Evolución de acurracy durante 100 épocas de entrenamiento

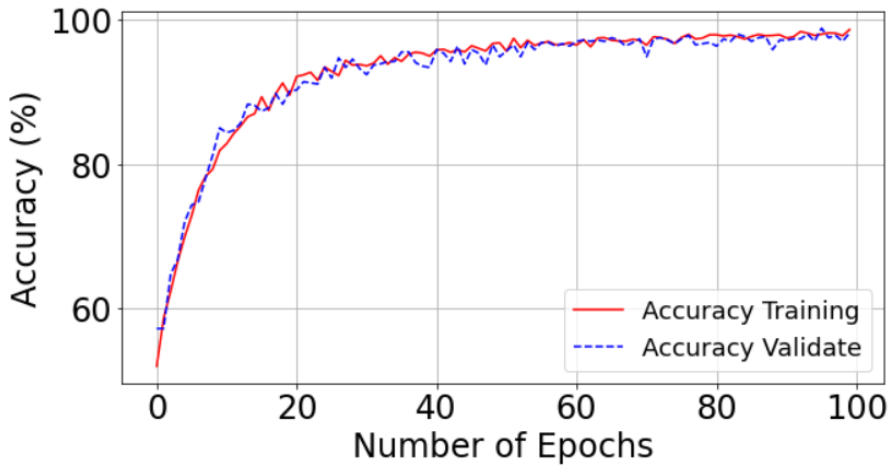


Figura 41

Evolución de loss durante 100 épocas de entrenamiento

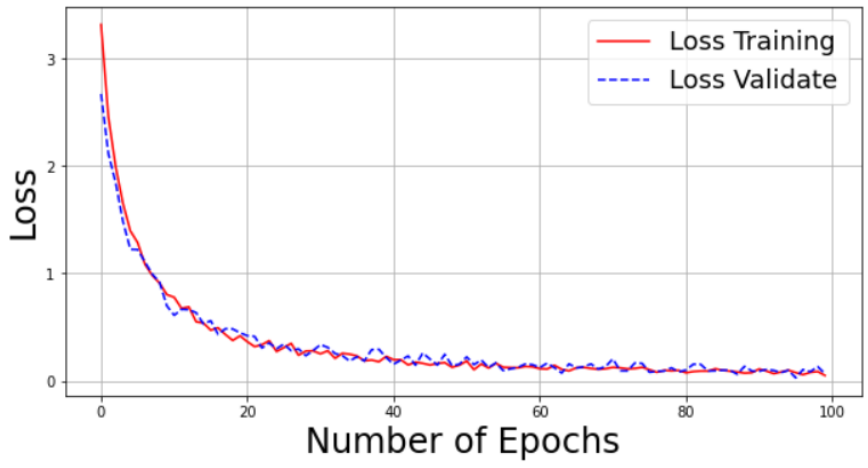


Figura 42

Corroboración de la evolución del acurracy durante 100 épocas

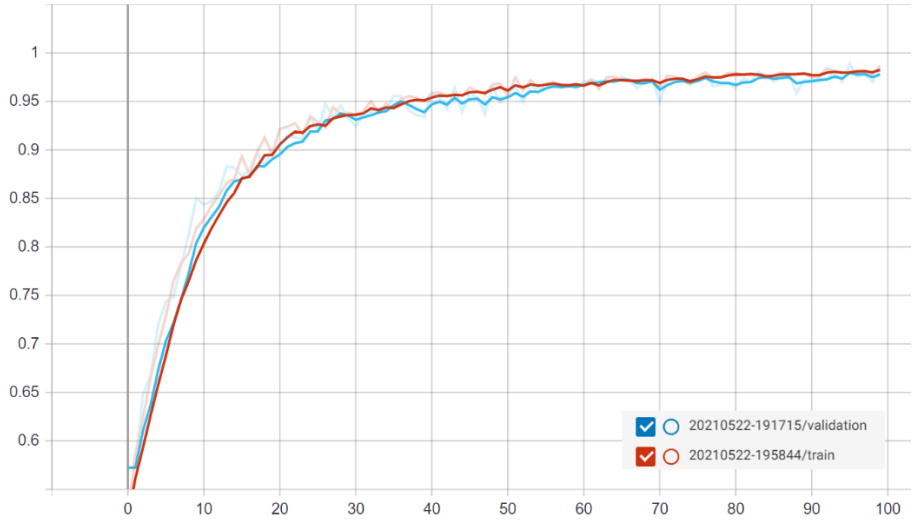
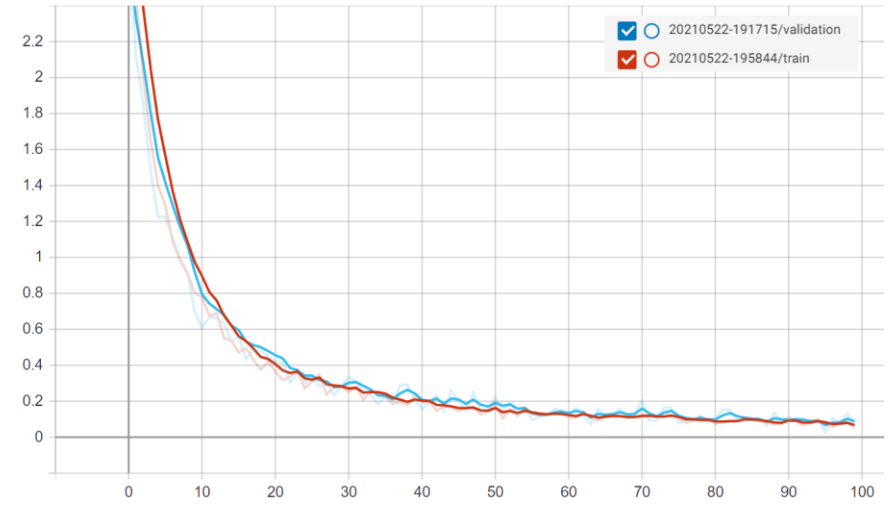


Figura 43

Corroboración de la evolución de loss durante 100 épocas



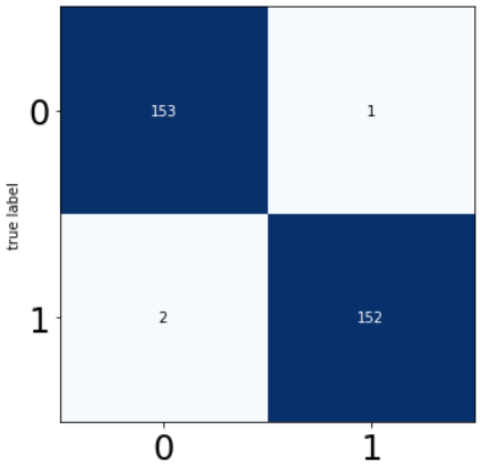
Como se puede observar desde la Figura 40 hasta la Figura 43 la red neuronal evoluciona alcanzando una precisión del 0.9903 y una pérdida menor al 0.1 pero estos valores se pueden mejorar de dos formas, la primera aumentando la base de datos para su entrenamiento o la segunda forma es aumentando el número de épocas de entrenamiento, en

esta segunda opción se corre el riesgo que en una determina etapa el sistema y sus pesos dejan de ajustarse debido al sobreajuste, lo que ocasiona que los resultados sean singulares y con la imposibilidad de comprender nuevos datos de entrada.

Posterior al entrenamiento se descarga el modelo (**modelJ1.h5**) y se procede a realizar pruebas de predicciones en el entorno de Google Colab con la finalidad de obtener la matriz de confusión y sus parámetros. De acuerdo con el vector de las clases declarado en la programación, el valor de 0 es equivalente a ***Mask*** mientras que la etiqueta 1 es equivalente a ***NoMask,*** en base a lo mencionado podemos observar el resultado de las predicciones en la Figura 44.

Figura 44

Matriz de Confusión con imágenes de prueba.



Como se observa en la Figura 44 para una base de test total de 308 imágenes y 154 imágenes por clase se determina lo siguiente: en la categoría 0 (Mask) el sistema presenta 2 falsos positivos mientras que en la categoría 1 (NoMask) tiene 1 falso positivo, mediante estos datos y con la ayuda de la librería ***sklearn*** se calcula los parámetros descritos anteriormente de la matriz de confusión y se detalla en la Tabla 2.

Tabla 2

Resultados de las predicciones con imágenes de prueba.

	Precisión	Recall	F1-score	Support
0	0.9871	0.9935	0.9903	154
1	0.9935	0.9870	0.9902	154
Accurracy			0.9903	308
Macro avg	0.9903	0.9903	0.9903	308
Weighted avg	0.9903	0.9903	0.9903	308

En base a lo presentado en la Tabla 2, se demuestra que el sistema presenta una buena respuesta para la detección de uso de mascarilla con un porcentaje de predicción del 99.03%.

Pruebas realizadas en el sistema de detección de mascarilla

Para realizar las diferentes pruebas en el sistema de detección de mascarillas, es necesario mencionar el comportamiento del algoritmo. Primero, la programación limita el espacio a ser analizado para lo cual en cada frame se determina la ubicación de cada rostro a lo cual lo denominaremos como región de interés, estas regiones de interés al ser ubicadas se extraen de cada fotograma para posteriormente analizarlo mediante la red neuronal y determinar de cada una de ella si utiliza o no mascarilla, es decir que mediante el desarrollo del algoritmo este determinará si las personas captadas por la cámara están usando mascarilla encerrándolo en un rectángulo de color verde, caso contrario el indicador será de color rojo en base a un previo entrenamiento de la red neuronal.

Para observar el comportamiento del sistema y la evolución en la predicción del sistema se analiza para diversos modelos entrenados en Google Colab empezando desde las 25, 50 y finalizando en las 100 épocas, se procede a mostrar desde la Figura 46 hasta la Figura 54, las cuales indicaran los resultados del sistema hasta llegar a una mayor exactitud de predicción a imágenes expuestas en tiempo real a una cámara web.

Primeramente, se analizan las predicciones del sistema con un modelo de 25 épocas de entrenamiento a continuación podemos observar en la Figura 45 la matriz de confusión y en la Tabla 3 los valores de las métricas de la matriz.

Figura 45

Matriz de confusión para 25 épocas de entrenamiento

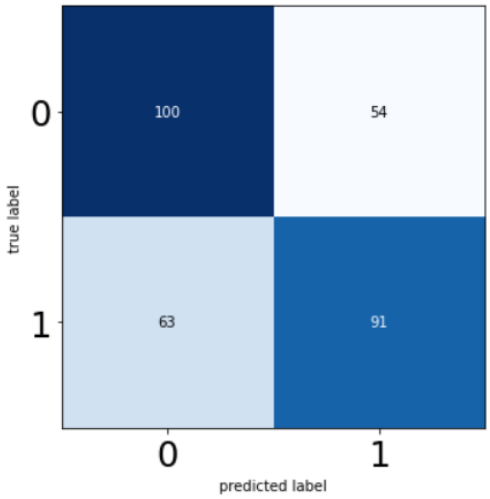


Tabla 3

Resultado de los parámetros para una red entrenada con 25 épocas

	Precisión	**Recall**	**F1-score**	**Support**
0	0.6135	0.6494	0.6309	154
1	0.6276	0.5909	0.6087	154
Accurracy			0.6201	308
Macro avg	0.6205	0.6201	0.6198	308
Weighted avg	0.6205	0.6201	0.6198	308

Como se observa en la Tabla 3 el valor de la exactitud de predicción para una red entrenada con 25 épocas de entrenamiento es del 0.6201. A continuación, en la secuencia de imágenes de la Figura 46 hasta la Figura 48 se observa la predicción del sistema en tiempo real, cabe mencionar que en las siguientes figuras las imágenes que son analizadas son descargadas de internet y ajenas a la base de datos o la base de pruebas.

Figura 46

Predicción del sistema a tres rostros sin mascarilla

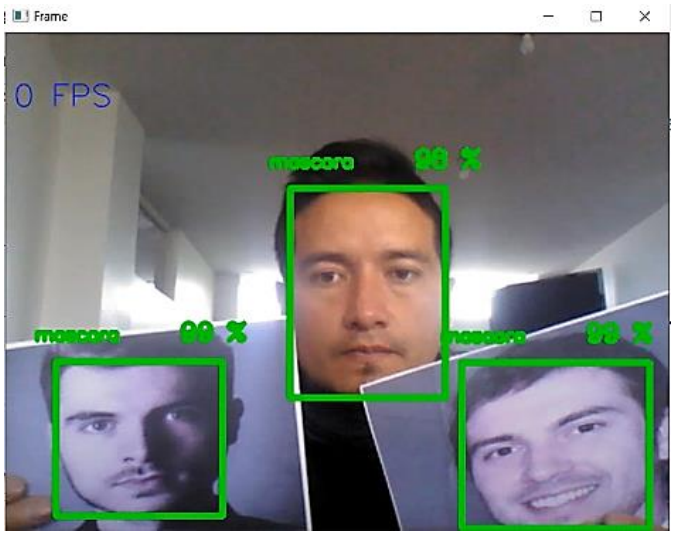


Tabla 4

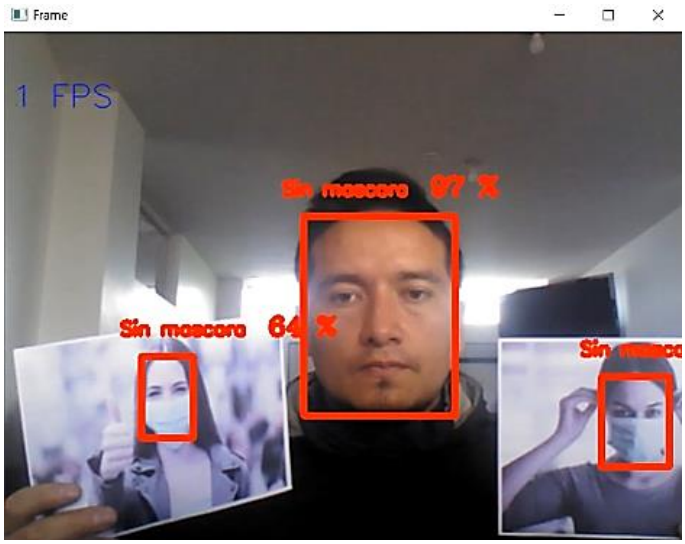
Descripción del sistema de la Figura 46

	Detección correcta	Detección incorrecta	Observaciones
Rostro 1		x	Error
Rostro 2		x	Error
Rostro 3		x	Error

En la Tabla 4 indica que los resultados son incorrectos ya que de acuerdo con la Figura 46 los tres rostros detectados por la cámara se marcan con la etiqueta de ***mascara*** en color verde pero la imagen muestra que no se usa mascarilla por lo cual las etiquetas deberían mencionar ***Sin mascara*** encerrados en un rectángulo de color rojo.

Figura 47

Predicción del sistema a dos rostros con y uno sin mascarilla



Nota. Sistema de detección de mascarilla.

Tabla 5

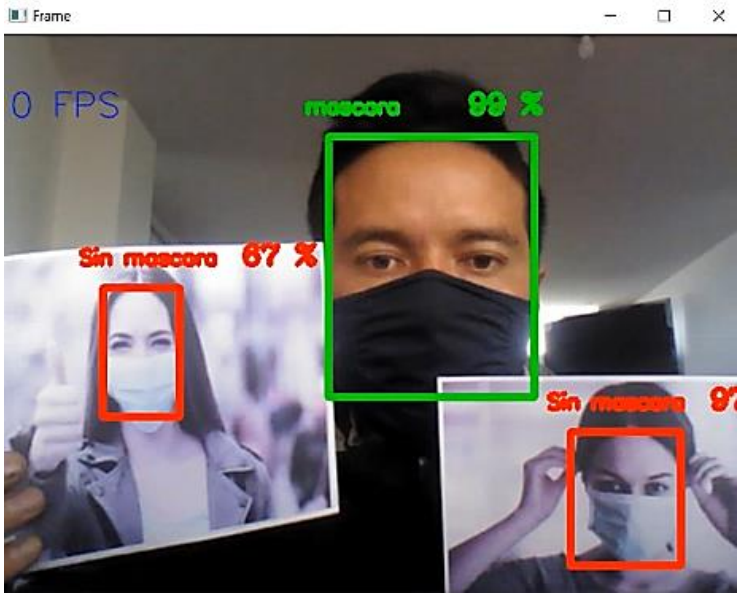
Descripción del sistema de la Figura 47

	Detección correcta	Detección incorrecta	Observaciones
Rostro 1		x	Error
Rostro 2	x		No existe error
Rostro 3		x	Error

En la Tabla 5 se observa que el rostro 1 y 3 poseen mascarillas en la cual el sistema brinda una respuesta equivocada ya que de acuerdo con la Figura 47 deben marcarse con la etiqueta ***mascara*** encerradas en color verde mientras que el rostro 2 no presenta error.

Figura 48

Predicción del sistema a tres rostros con mascarilla



Nota. Sistema de detección de mascarilla.

Tabla 6

Descripción del sistema de la Figura 48

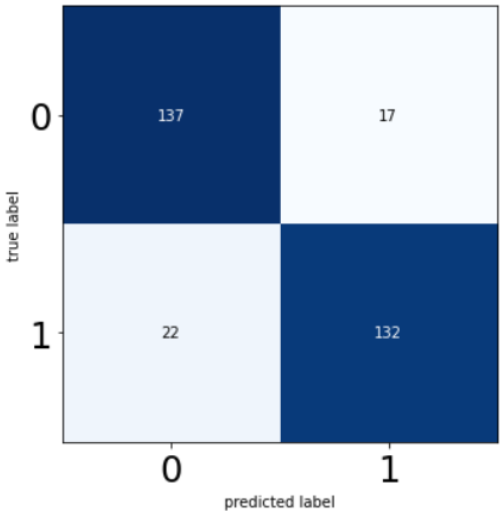
	Detección correcta	Detección incorrecta	Observaciones
Rostro 1		x	Error
Rostro 2	x		No existe error
Rostro 3		x	Error

En la Tabla 6 se puede observar que los resultados son incorrectos para los rostros 1 y 3 ya que en la Figura 48 muestra que estos rostros están utilizando mascarilla, pero en el rostro 2 no presenta errores.

Una vez puesta en prueba el sistema para la detección de mascarilla y con la necesidad de reducir los falsos positivos en la predicción del uso de mascarillas se procede a entrenar a la red neuronal para 50 épocas, produciendo la matriz de confusión que se puede observar en la Figura 49 y los valores de las métricas de la matriz en la Tabla 7.

Figura 49

Matriz de confusión para 50 épocas de entrenamiento



Como se observa en la Figura 49 se nota la reducción de falsos positivos en relación a la Figura 45, para este caso el número de falsos positivos para la etiqueta 0 (***Mask***) es de 22 elementos mientras que para la etiqueta 1 (***NoMask***) el número de falsos positivos es 17, lo cual nos nuestra una mejora en la predicción de las imágenes.

Tabla 7

Resultado de los parámetros para una red entrenada con 50 épocas

	Precisión	**Recall**	**F1-score**	**Support**
0	0.8616	0.8896	0.8754	154
1	0.8859	0.8571	0.8713	154
Accurracy			0.8734	308
Macro avg	0.8738	0.8738	0.8733	308
Weighted avg	0.8738	0.8738	0.8733	308

Como se observa en la Tabla 7 el valor de la exactitud de predicción para una red entrenada con 50 épocas de entrenamiento aumento el valor de 0.6201 al 0.8734. A continuación, en la secuencia de las Figuras 50 hasta la Figura 52 se mostrará los resultados del sistema para un modelo entrenado de 50 épocas, la misma que presenta una gran mejora en la predicción de la respuesta con respecto a su antecesor de 25 épocas.

Figura 50

Predicción del sistema a tres rostros sin mascarilla, 50 épocas

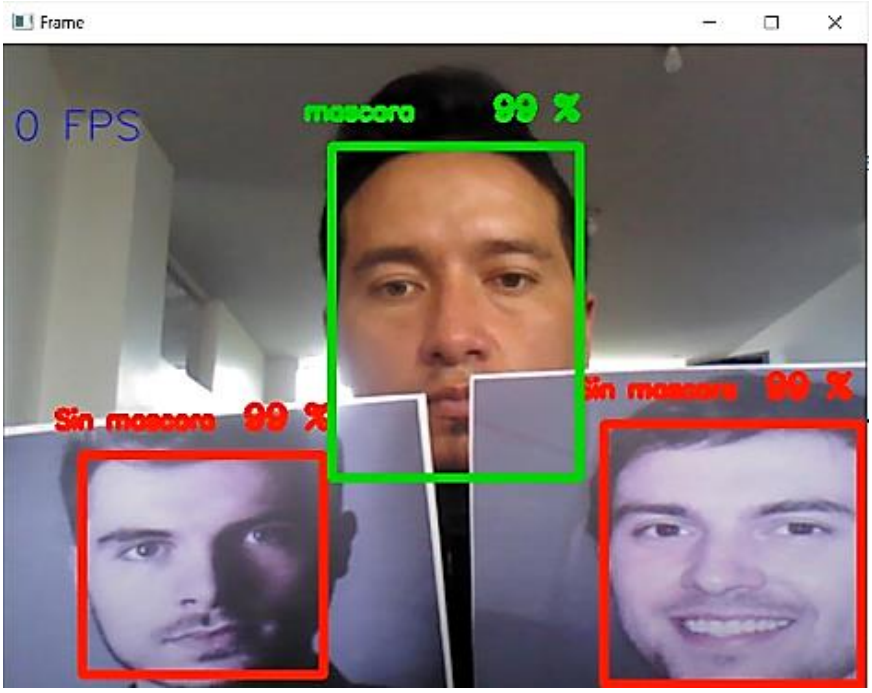


Tabla 8

Descripción del sistema de la Figura 50

	Detección correcta	Detección incorrecta	Observaciones
Rostro 1	x		No existe error
Rostro 2		x	Error
Rostro 3	x		No existe error

En la Tabla 8 se observa que el sistema responde de mejor manera antes las imágenes expuestas de las cuales una de ellas su respuesta fue incorrecta marcando el rostro 2 con la etiqueta ***mascara*** ya que de acuerdo a la Figura 50, este rostro no posee ninguna mascarilla.

Figura 51

Predicción del sistema para tres rostros, 50 épocas.

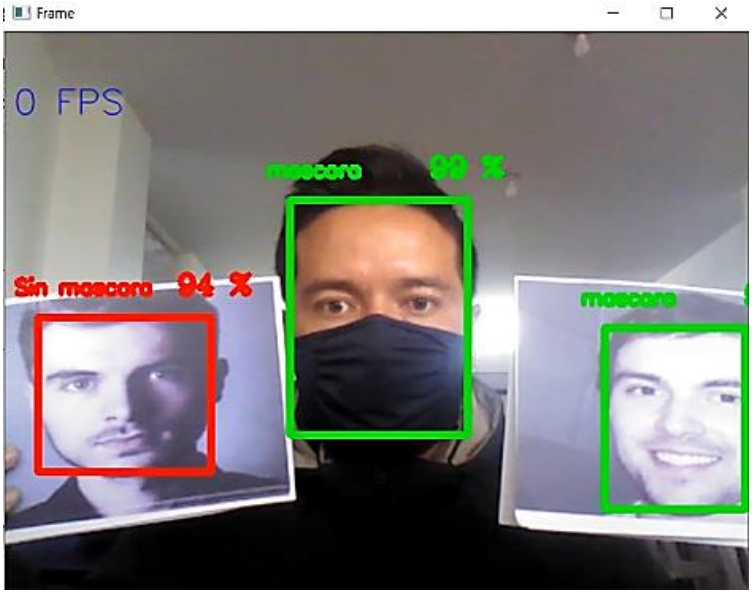


Tabla 9

Descripción del sistema de la Figura 51

	Detección correcta	Detección incorrecta	Observaciones
Rostro 1	x		No existe error
Rostro 2	x		No existe error
Rostro 3		x	Error

En la Tabla 9 se observa que los rostros 1 y 2 la respuesta del sistema a la predicción es correcta, mientras que en el rostro 3 la etiqueta es incorrecta, según se muestra en la Figura 51.

Figura 52

Predicción del sistema para tres rostros con mascarilla, 50 épocas.

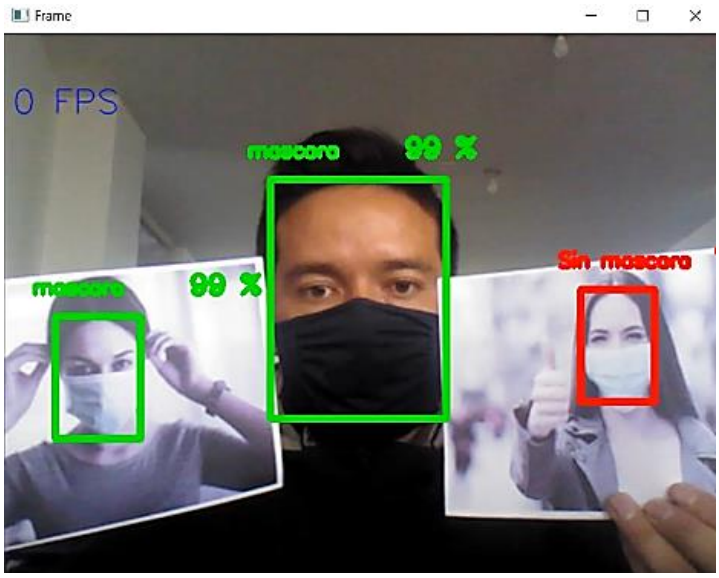


Tabla 10

Descripción del sistema de la Figura 52

	Detección correcta	Detección incorrecta	Observaciones
Rostro 1	x		No existe error
Rostro 2	x		No existe error
Rostro 3		x	Error

Como se observa en la Tabla 10 en el rostro 1 y 2 no presenta ninguna falla en la predicción, mientras que en el rostro 3 la etiqueta es incorrecta debido a que en la Figura 52 este rostro si utiliza mascarilla.

Finalmente se pone a prueba la respuesta del sistema para una red entrenada con 100 épocas la matriz de confusión y los valores de las métricas se puede observar en la Figura 44 y Tabla 2 respectivamente. A continuación, en la secuencia de las Figuras 53 hasta la Figura 55 se mostrarán los resultados del sistema para un modelo entrenado de 100 épocas con una precisión de predicción del 0.9903 detallado en la sección anterior. Podemos notar que al exponer las imágenes en la cámara la respuesta del sistema es el deseado.

Figura 53

Predicción del sistema a tres rostros sin mascarilla

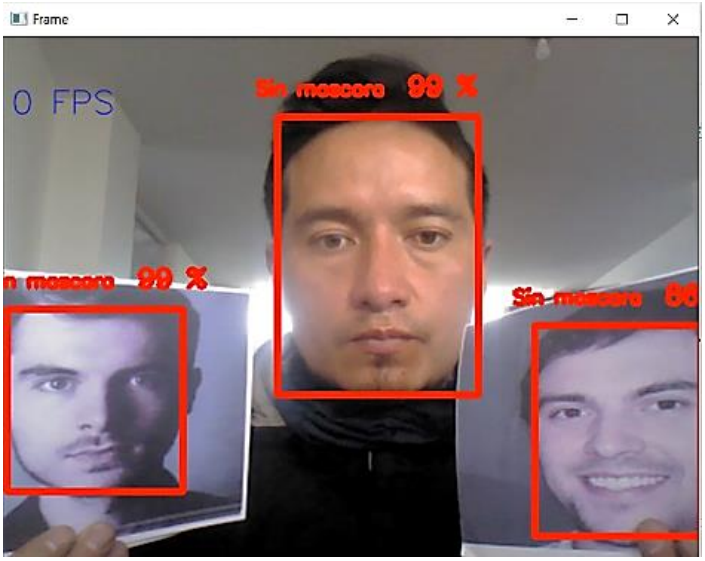


Tabla 11

Descripción del sistema de la Figura 53

	Detección correcta	Detección incorrecta	Observaciones
Rostro 1	x		No existe error
Rostro 2	x		No existe error
Rostro 3	x		No existe error

Figura 54

Predicción del sistema para tres rostros con mascarilla

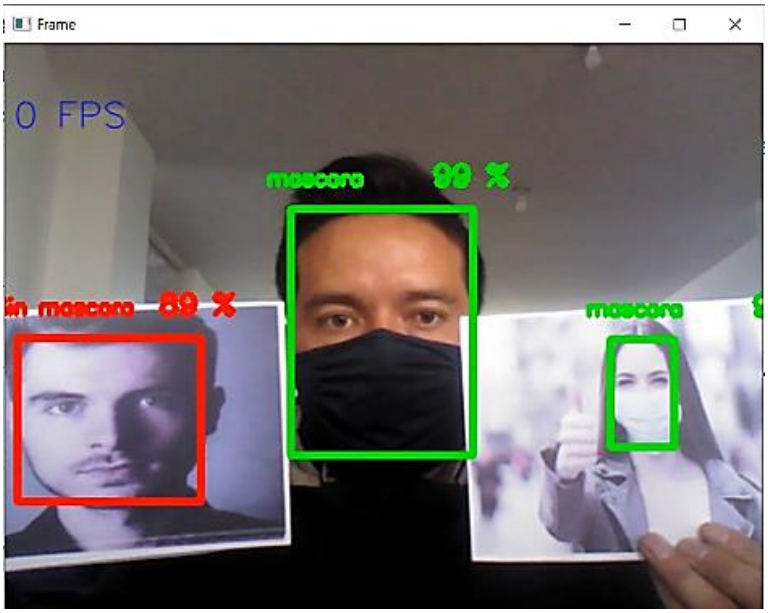


Tabla 12

Descripción del sistema de la Figura 54

	Detección correcta	Detección incorrecta	Observaciones
Rostro 1	x		No existe error
Rostro 2	x		No existe error
Rostro 3	x		No existe error

Figura 55

Predicción del sistema para tres rostros con y sin mascarilla

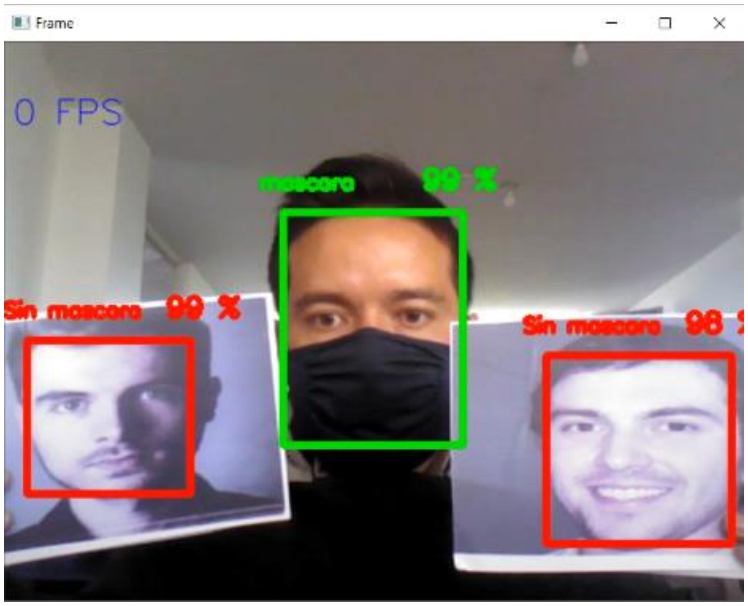


Tabla 13

Descripción del sistema de la Figura 55

	Detección correcta	Detección incorrecta	Observaciones
Rostro 1	x		No existe error
Rostro 2	x		No existe error
Rostro 3	x		No existe error

Como se observó en las diferentes tablas la respuesta del sistema evoluciona y mejora conforme al número de épocas (ciclos de entrenamiento). Para finalizar, el sistema presenta un buen porcentaje de reconocimiento, por lo cual queda demostrado que el sistema para la detección de mascarillas es el adecuado con una buena respuesta de predicción independientemente de las imágenes que sea expuesta en tiempo real.

Verificación de la Hipótesis

Para verificar el comportamiento de la hipótesis propuesta se realiza el siguiente análisis en dos partes.

Primero, se evalúa el modelo entrenado de la red neuronal convolucional, este análisis se lo realizó en el apartado 4.1 en el cual se realizan predicciones de uso de mascarilla con 308 imágenes divididas en dos categorías con 154 imágenes cada una, obteniendo un resultado de predicción del 0.9903 o el 99.03%.

Segundo, para evaluar el sistema de detección de mascarillas se lo integró en Jupyter Notebook con una cámara web y precargando los diferentes modelos entrenados de 25, 50 y

100 ciclos de entrenamiento, esto se analizó en la sección 4.1, de esta forma se concluye que el modelo entrenado tiene un elevado nivel de precisión al clasificar imágenes nuevas.

Capítulo V

Conclusiones y recomendaciones

Conclusiones

- El sistema se desarrolló en dos etapas, la etapa de entrenamiento la cual se realizó totalmente en la plataforma Google Colab con una base de datos de 3194 imágenes divididas en *train, valid* y *test* con el 70, 20 y 10% respectivamente y la segunda etapa se desarrolló en software libre Anaconda Navigator la cual permite aplicaciones, entornos y paqueterías dedicados al desarrollo en lenguaje Python, como editor de código se utiliza Jupyter Notebook y sus diferentes paqueterías para el desarrollo de la programación.
- Una de los beneficios primordiales al instante de ejecutar un modelo de Machine Learning y Deep Learning con Google Colab es que posibilita la implementación de una GPU que es muchísimo más veloz que en una CPU. La GPU resulta muchísimo más potente en hacer cálculos matemáticos por ejemplo descenso de gradiente, multiplicaciones matriciales o el Backpropagation de una red neuronal, es decir, algoritmos que anteriormente tomaban días en entrenar, pero ahora se resuelven en horas.
- La implementación del algoritmo para la detección de uso de mascarilla en tiempo real utilizando procesamiento de imágenes y redes neuronales convolucionales opera de manera efectiva debido a que permite extraer y categorizar gracias al entrenamiento realizado en Google Colab, para esto la imagen de entrada debe estar en dimensiones de 224 x 224.
- La arquitectura de una red neuronal está basada en la red escogida para entrenar, para este proyecto la red es la Vgg-16 que contiene 16 capas (3 capas densas y 13 capas convolucionales) las mismas que son preentrenadas con la base de datos de ImageNet

para una clasificación de 1000 clases, debido a su estructura implica menos memoria, menos requerimientos computacional y por lo tanto menos tiempo de entrenamiento.

- La red neuronal convolucional implementada consta de 16 capas plenamente conectadas entre sí, para la cual se ocupa el procedimiento de aprendizaje supervisado. Los resultados se evaluaron por medio de la matriz de confusión, en la cual se obtuvo un porcentaje de predicción del 99.03%, un porcentaje elevado para la categorización de uso de mascarilla.
- Para la predicción y categorización del sistema utilizando IA (inteligencia artificial) se debe tener en cuenta diversos componentes como la iluminación, distancia y resolución de la imagen debido a que puede perjudicar en la elección del reconocimiento.
- Los errores en la determinación de la categorización del uso de mascarilla se presentan con mayor frecuencia cuando la red es entrenada para pocas épocas debido a que el sistema no reconoce puntos característicos de la imagen, pero conforme se aumenta la cantidad de épocas o a su vez la base de datos se reduce dichas predicciones erróneas.
- Se puede entrenar el sistema para la detección y clasificación de otros elementos pues la programación se lo realizo en una interfaz de software libre lo que implica que el código fuente sea abierto y se puede escalar otras actividades dependiendo de la necesidad y el uso de la inteligencia artificial.

Recomendaciones

- Se requiere de una base de datos la cual contenga un gran número de imágenes y las dos categorías mascarilla y sin mascarilla. Además, estas imágenes deben ser claras e indicar la región de interés a predecir, para este caso las imágenes recolectadas son imágenes que contienen solo el rostro, esto optimiza los puntos de interés que debe considerar la red neuronal al entrenar.
- Es necesario el uso de un equipo que contenga una tarjeta gráfica, de ser posible una NVIDIA CUDA, debido que una GPU resume el tiempo de entrenamiento y los procesos computacionales matemáticos de la red neuronal.
- Para lograr un mejor entrenamiento de la red neuronal se sugiere descartar imágenes con objetos ajenos al análisis, además, balancear el número de imágenes en cada categoría con el fin de eludir el sobreajuste del modelo
- En la situación de usar cámaras web de más grande resolución para el análisis de imágenes, se debe considerar que la cámara no presente configuraciones de alta resolución, debido que al exponerlo a exteriores las imágenes transmitidas están saturadas en color imposibilitando la ubicación del rostro y la predicción.

Bibliografía

- Artola Moreno, Á. (2019). Clasificación de imágenes usando redes neuronales convolucionales en Python.

- Caballero, J., & Grossmann, I. (2007). Una revisión del estado del arte en optimización. Revista Iberoamericana de Informática y Automática Industrial, 5-23.

- Cano-Ramırez, F. A., & Cruz-Roa, A. Entrenamiento de una Red Neuronal Convolucional con Jupyter y TensorFlow para la clasificación de tejido invasivo de cáncer de seno.

- Chanampe, H., Aciar, S., Vega, M. D. L., Molinari Sotomayor, J. L., Carrascosa, G., & Lorefice, A. (2019, June). Modelo de redes neuronales convolucionales profundas para la clasificación de lesiones en ecografías mamarias. In *XXI Workshop de Investigadores en Ciencias de la Computación (WICC 2019, Universidad Nacional de San Juan).*

- Cireşan, D. C., Giusti, A., Gambardella, L. M., & Schmidhuber, J. (2013, September). Mitosis detection in breast cancer histology images with deep neural networks. In *International conference on medical image computing and computer-assisted intervention* (pp. 411-418). Springer, Berlin, Heidelberg.

- Cobo Fernández, G. (2017). Desarrollo de una aplicación móvil de realidad virtual para el aprendizaje en las aulas.

- Esqueda, J., & Palafox, L. (2005). Fundamentos de procesamiento de imágenes. *Baja California, México: Universidad Autónoma de Baja California*.

- Freeman, J. A., & Skapura, D. M. (1993). Redes neuronales. Algoritmos, aplicaciones

- Guevara, M. L., Echeverry, J. D., & Uruena, W. A. (2008). Detección de rostros en imágenes digitales usando clasificadores en cascada. *Scientia et technica*, *1*(38).

- Huaman, K., Bonilla, C., Huaroto, F., Curisinche, M., Reyes, N., Gutierrez, E., & Caballero, P. (2020). Uso de mascarillas y respiradores para la prevención y control de infecciones por virus respiratorios.

- Jain, B., Singh, P. y Goel, R. (2019, septiembre). Reconocimiento de imágenes de caracteres en marathi manuscritas utilizando una red neuronal convolucional. En *2019 International Conference on Computing, Power and Communication Technologies (GUCON)* (págs. 326-329). IEEE.

- Krizhevsky, A., Sutskever, I. y Hinton, GE (2012). Clasificación de Imagenet con redes neuronales convolucionales profundas. En *Avances en sistemas de procesamiento de información neuronal* (págs. 1097-1105).

- Kuo, CCJ (2016). Comprensión de las redes neuronales convolucionales con un modelo matemático. *Revista de comunicación visual y representación de imágenes*, *41*, 406-413.

- LeCun, Y., Bengio, Y. y Hinton, G. (2015). Aprendizaje profundo. *naturaleza*, *521* (7553), 436-444.

- López, F. J. A., Avi, J. R., & Fernández, M. V. A. (2018). Control estricto de matrices de confusión por medio de distribuciones multinomiales. Geofocus: Revista Internacional de Ciencia y Tecnología de la Información Geográfica, (21), 6.

- López-Saca, F. (2019). Clasificación de imágenes usando redes neuronales convolucionales (Master's thesis, Universidad Autónoma Metropolitana (México). Unidad Azcapotzalco. Coordinación de Servicios de Información.).

- Montoya, Ó. P. (2018). Redes Neuronales Convolucionales Profundas para el reconocimiento de emociones en imágenes. *Universidad Politécnica de Madrid*.

- Morocho Jiménez, J. I. (2019). *Detección de tumores cutáneos malignos y benignos usando una red neuronal convolucional* (Bachelor's thesis, Quito, 2019.).

- Pacheco, M. A. (2017). Identificación de sistemas no lineales con redes neuronales convolucionales.

- Palomino, N. L. S., & Concha, U. N. R. (2009). Técnicas de segmentación en procesamiento digital de imágenes. *Revista de investigación de Sistemas e Informática*, *6*(2), 9-16.

- Peralta, D., Herrera-Poyatos, A., & Herrera, F. Un Estudio sobre el Preprocesamiento para Redes Neuronales Profundas y Aplicación sobre Reconocimiento de Dıgitos Manuscritos.
- Pérez-Aguilar, D., Risco-Ramos, R., & Casaverde-Pacherrez, L. Transfer learning en la clasificación binaria de imágenes térmicas Transfer Learning for Binary Classification of Thermal Images. Machine learning (ML), 550, 4.
- Rodríguez Álvarez, J. J. J. (2019). Desarrollo de un sistema inteligente basado en visión computacional para detectar bacterias escherichia coli en verduras frescas.
- Sánchez Anzola, N. (2015). Máquinas de soporte vectorial y redes neuronales artificiales en la predicción del movimiento USD/COP spot intradiario. ODEON-Observatorio de Economía y Operaciones Numéricas, (9).
- Trujillo Quevedo, A. (2017). Sistema de reconocimiento de caracteres manuscritos usando redes neuronales convolucionales implementado en Python.

Anexos