UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

SEDE VIÑA DEL MAR – JOSÉ MIGUEL CARRERA

CONTROL DE ACCESO MEDIANTE RECONOCIMIENTO FACIAL

Trabajo de Titulación para optar al Título de Ingeniero de Ejecución en CONTROL E INSTRUMENTACIÓN INDUSTRIAL

Alumno:

Sebastián Alejandro Pérez Latorre

Profesor Guía:

Mag. Guelis Montenegro Zamora

2024

RESUMEN

KEYWORDS: Raspberry Pi, Python, Reconocimiento facial, Redes neuronales convolucionales.

A continuación, se presentará el proceso de implementación de un control de acceso mediante reconocimiento fácil utilizando inteligencia artificial y la visión de computadora. Para lograr este propósito será necesario usar hardware y software, dentro del primero mencionado se usará: chapa eléctrica, cámara web, Módulo de relés y Raspberry Pi, y en el último mencionado se usará raspbian y lenguaje de programación Python con sus respectivas librerías.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

SIGLAS Y SIMBOLOGÍA

A. SIGLAS:

1D	: Una dimensión
2D	: Dos dimensiones
ANN	: Red neuronal artificial
CNN	: Red neuronal convolucional
CNNDO	: Red neuronal convolucional con dropout
COM	: Común
DP	: Deep Learning
GPIO	: Puerto general de entrada o salida
HSV	: Modelo representativo del color de manera no lineal
IA	: Inteligencia artificial
ML	: Machine Learning
NO	: Normalmente abierto
NC	: Normalmente cerrado
N°	: Numero
RGB	: Modelo presentativo del color
RNN	: Red neuronal recurrente
SNN	: Red neuronal simulada
TV	: Televisión
RAM	: Memoria de acceso aleatorio
CPU	: Unidad central de procesamiento
Rpi3b+	: Raspberry pi 3 model b+
Rpi5	: Raspberry pi 5

B. SIMBOLOGÍA:

%	: Porcentaje
°	: grados de una circunferencia
ϴ	: Theta
W	: Ponderación o peso
X	: Entradas
VCC	: Señal voltaje continuo
VAC	: Señal voltaje alterno
nm	: Nanómetro
mA	: Miliamperes
mm	: Milímetro
GHz	: GigaHertz

Gb : Gigabits

INTRODUCCIÓN

En conjunto con el desarrollo industrial, se ha desarrollado el control de procesos industriales para que estos sean más eficientes y seguros. Los cuales, gracias a sus avances, se pueden aplicar actualmente a la vida cotidiana solucionando diversos problemas del día a día.

En base en lo antes mencionado, el presente trabajo de título, se expondrá el proyecto que tiene por nombre “Control de acceso mediante reconocimiento facial”, donde se desarrollará la temática mencionada anteriormente, utilizando inteligencia artificial y visión de computadora, donde se expondrán los hardware y software utilizados, como también se analizará la forma en que el ser humano percibe el mundo mediante los ojos, en base en esto último, se describirá cómo es que un computador, independiente de su tamaño, puede lograr identificar y encontrar similitudes en imágenes, en otras palabras, reconocer. Además, se observará cómo la base de datos para entrenamiento-validación, las épocas de entrenamiento y el modelo de red neuronal pueden afectar con el objetivo.

CAPÍTULO 1: ANTECEDENTES GENERALES

1. ANTECEDENTES GENERALES

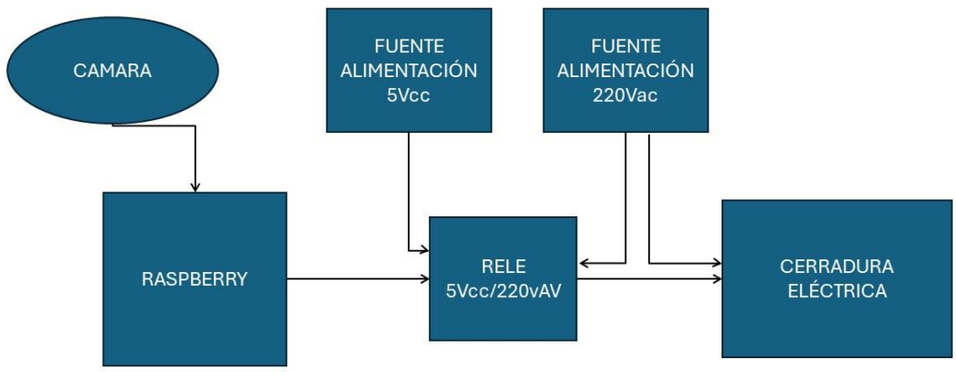
En el capítulo que se observará a continuación, se detallarán los elementos necesarios para llevar a cabo el proyecto que tiene como título “Control de acceso mediante reconocimiento facial”.

1.1. INFORMACIÓN DEL PROYECTO

En el siguiente capítulo se abordarán las generalidades del tema del proyecto solicitado como los hardware, software utilizado y el diagrama de bloques, donde se observará cómo los componentes interactúan entre sí para lograr que el proceso funcione.

1.2. DIAGRAMA DE BLOQUES

A continuación, en la Figura 1-1, se observará el diagrama de bloques del sistema de proyecto solicitado, donde se podrá apreciar los componentes que se utilizaran para llevar a cabo el control de acceso mediante reconocimiento facial, en este diagrama se encontrará con una cámara web, una tarjeta Raspberry, un relé, una Cerradura eléctrica, y dos fuentes de alimentación, de las cuales una es de 5Vcc que alimentara el relé antes mencionado, mientras que la otra es de 220Vac, entregara suministro eléctrico a la cerradura para que actúe.



Fuente: Elaboración propia mediante PowerPoint

Figura 1-1. Diagrama de bloques

1.3. COMPONENTES

A continuación, se mencionará y describirán los componentes utilizados separados en dos grupos, los cuales son hardware y software.

1.3.1. Hardware

Elementos físicos donde el software operará, el cual compondrán la maqueta del trabajo de título donde se desarrollará.

1.3.1.1. Cámara web

Este elemento está destinado para capturar las imágenes del rostro de las personas y enviar dicha información en forma de señal digital para que el controlador, mediante una programación, procese la imagen y determine la acción a ejecutar, es decir, permitirá o denegara el acceso.

Datos técnicos:

- Resolución máxima: 1080p/30 fps-720p/30fps
- Campo visual diagonal: 78°
- USB-A

A continuación, en la Figura 1-2, se podrá apreciar la cámara web que se utilizará en el desarrollo del proyecto de la marca Logitech, la cual, como antes se mencionó, cumplirá la función de capturar imágenes y enviar dicha captura al controlador.

Fuente: https://www.logitech.com/es-es/products/webcams/c920-pro-hd-webcam.960-001055.html

Figura 1-2. Cámara Web marca Logitech

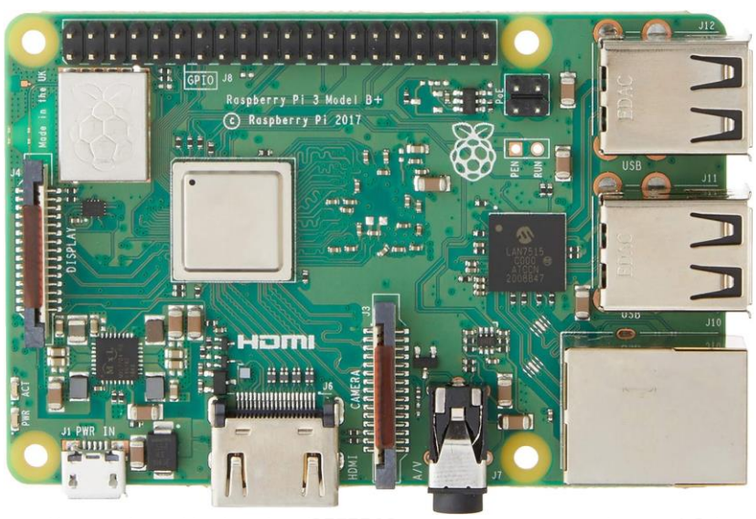
1.3.1.2. Raspberry pi

Computador de tamaño reducido, donde, gracias a su sistema operativo, se podrá ejecutar el programa para realizar las acciones de control deseadas, a este llegará la imagen capturada por la cámara web en forma de señales digitales, ésta se procesará y se determinará que acción tomar de acuerdo con el script realizado.

Datos técnicos:

- CPU de 4 núcleos de 1,4 GHz de 64bit
- 40 pines GPIO
- 1 GB de RAM
- 1 puerto Ethernet 10/100
- Corriente máxima por GPIO es de 16mA
- Corriente total máxima de todos los GPIO es de 50mA
- Voltaje GPIO 3,3Vcc

A continuación, se podrá apreciar la Figura 1-3, se muestra la tarjeta de desarrollo raspberry pi 3 model B+.



Fuente: https://www.ubuy.cl/sp/product/35F556O-rs-components-raspberry-pi-3-b-motherboard

Figura 1-3. Raspberry 3 model B+

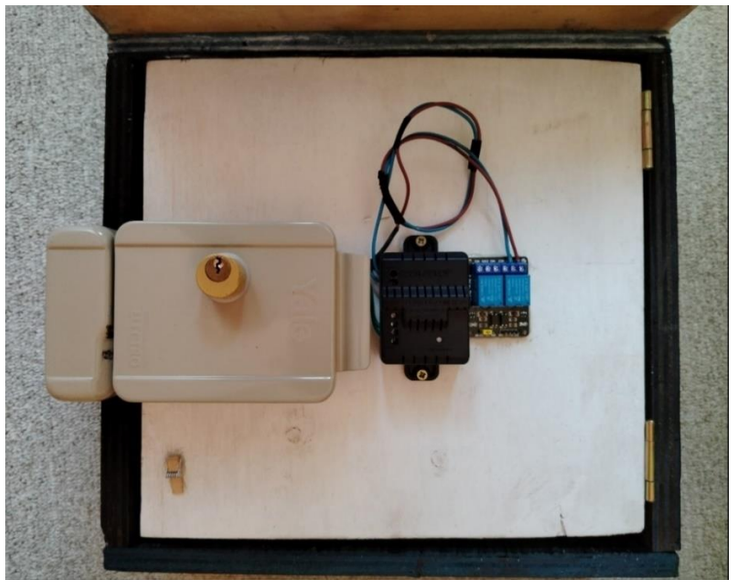
1.3.1.3. Chapa eléctrica

Elemento actuador del proyecto, que está conformado por el módulo de relés y chapa eléctrica, el cual recibe la señal para actuar desde el controlador, es decir, la tarjeta raspberry pi 3 model b+.

Es necesario considerar los diferentes voltajes en los cuales los diferentes elementos funcionan, ya que la chapa eléctrica se activa cuando hay 220VAC en sus

entradas, mientras que el módulo de relé actúa cuando tiene 5VCC continuos en sus entradas, por lo tanto, se trabajaran con diferentes fuentes de alimentación.

La disposición de los elementos antes mencionados se puede verificar en la siguiente Figura 1-4, en la cual se podrá apreciar el módulo de relés que está conectada eléctricamente con la cerradura eléctrica.



Fuente: elaboración propia tomada desde cámara de teléfono

Figura 1-4. Chapa eléctrica

1.3.1.4. Módulo de relé

Dispositivo que se encontrará conectado con el controlador mediante contactos y a su vez con su bobinado, éste tiene como finalidad cerrar su contacto cuando reciba la señal del controlador, permitiendo alimentar la chapa eléctrica con 220VAC, en otras palabras, permite controlar circuitos eléctricos de mayor potencia utilizando señales bastante menores, cabe destacar que la bobina antes mencionada actúa con 5VCC.

Como se puede apreciar en la Figura 1-5, este módulo cuenta con dos relés, cada uno de ellos cuenta con 3 pines que corresponden a los terminales de los contactores, es decir, un terminal COM el cual es el punto de conexión común, en este pin se puede conectar un positivo o negativo, se debe tener la precaución de que en los demás pines que se mencionarán a continuación se conecte la misma señal del COM, por ejemplo, si al común se conectó un positivo, en los otros debe considerar una señal positiva para no generar un cortocircuito en estos terminales. Mientras que NO y NC corresponden a los pines Normalmente Abierto y Normalmente cerrado respectivamente. El pin COM recibirá la señal alterna de 220VAC, y saldrá por NO o NC según corresponda, mientras que el otro externo del módulo, se podrán encontrar 4 entradas, las cuales son GND, VCC, INT1 y INT2. Donde INT1 corresponde al control del relé N° 1, y el restante al relé N°2.

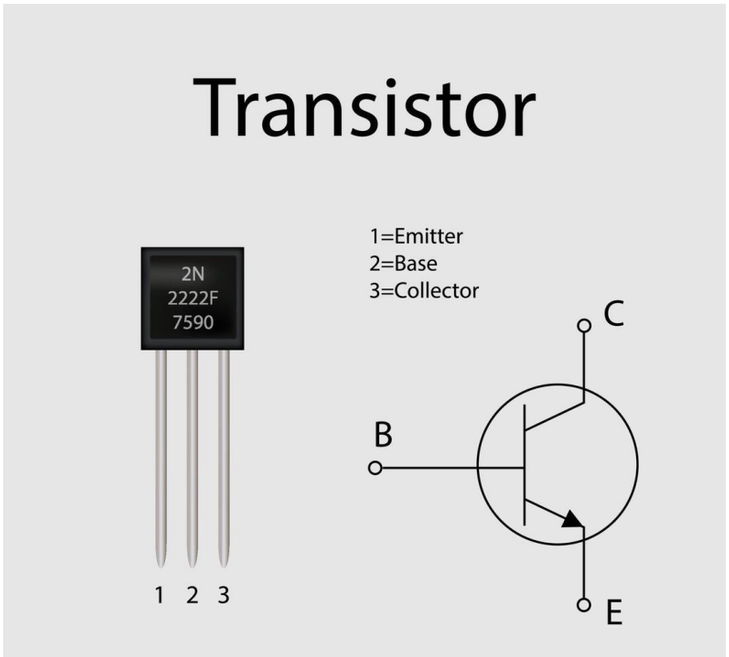
En la siguiente pagina se puede apreciar la Figura 1-15 que se menciono anteriormente.

Fuente: https://ardumotica.cl/producto/modulo-rele-5v-2-canales-optocoplados/

Figura 1-5. Módulo de relé de 2 canales

1.3.1.5. Transistor NPN

Dispositivo electrónico con tres terminales que corresponde al Emisor, Base y colector, donde para poder conducir entre el colector y emisor es necesario aplicar una diferencia de potencial en la Base, a diferencia del transistor PNP que al realizar esta acción éste deja de conducir entre los terminales inicialmente mencionados. Este elemento facilitará controlar el módulo de relés que funciona con 5VCC, como se pudo observar en el Punto 1.3.1.4. mientras que en el Punto 1.3.1.2. se puede observar que los GPIO funcionan con 3,3Vcc. A continuación, en la Figura 1-6 se podrá observar el transistor antes descrito y su simbología.



Fuente: https://es.vecteezy.com/arte-vectorial/10043747-transistor-componente-electronico-con-su-simbolo-diagrama-vector-ilustracion-eps-10

Figura 1-6. Transistor NPN

1.3.2. Software

Software es un conjunto de instrucciones que permiten a una computadora, para el caso de este trabajo de título, la raspberry, realizar tareas específicas, en otras palabras, es la sección intangible del sistema, la cual determina como deben funcionar los hardware, además de qué acciones deben tomar estos. Esto se obtiene mediante instrucciones escritas, denominado script, donde se puede establecer que decisiones tomar hasta la interfaz.

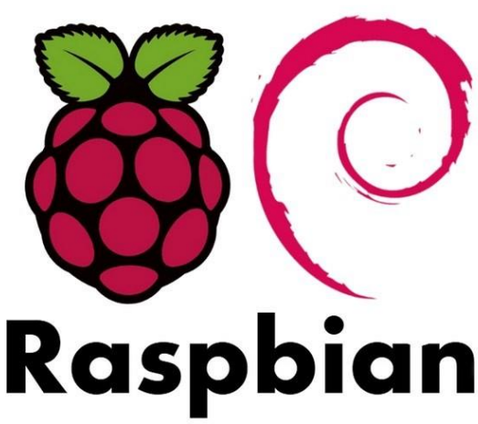
1.3.2.1. Raspbian

Sistema operativo Linux optimizado para ser utilizado en el hardware Raspberry Pi. Está basado en Debian, el cual es un sistema operativo de código abierto, gratuito que a su vez se basa en GNU/Linux. Debian se caracteriza además por soportar múltiples arquitecturas, gestión de paquetes instalados como administrador, ser estable como sistema operativo, buen soporte de hardware para las diferentes arquitecturas.

Gracias a sus características, es recomendado para ser usado en educación para enseñar programación, robótica y conceptos de computación, del mismo modo puede ser usado para desarrollar proyectos de domótica, entre otros proyectos de electrónica, además con sus lenguajes de programación permite desarrollar software.

Este contiene herramientas y aplicaciones útiles preinstaladas, como por ejemplo Python y Scratch, que se utilizan para programación y desarrollo de software educativos, así mismo como el Minecraft Pi, además contiene herramientas avanzadas de matemáticas como lo es Wolfram Alpha y Mathematica, como también el LibreOffice, que corresponde a un paquete de oficina, en el cual se puede encontrar hojas de cálculo, procesador de textos, programas de presentación, herramientas de dibujo, entre otros.

A continuación, en la Figura 1-7, se observará el logo del sistema operativo antes mencionado.



Fuente: https://www.xatakahome.com/trucos-y-bricolaje-smart/quieres-una-distribucion-raspbian-ligera-con-este-tutorial-no-puede-ser-mas-sencillo

Figura 1-7. Raspbian

1.3.2.2. Lenguaje de Programación Python.

Python es un lenguaje de programación con tres grandes características, las cuales son: lenguaje de alto nivel; orientado a objetos y de código abierto. Estas características antes mencionadas resultan en un lenguaje con sintaxis cercana a la utilizada por los humanos para comunicarse, por otra parte, está centrado en la organización de software respecto a los datos u objetos, además de ser de uso público. Del mismo modo como cuenta con otras características no menos importantes, ya que es de tipado dinámico, es decir, no es necesario anunciar los tipos de variables cuando se escribe el código. Es de carácter multiplataforma, ya que el código se puede ejecutar en diferentes sistemas operativos. Soporta varios paradigmas de programación, como por ejemplo ser orientado a objetos, estructurada, imperativa y funcional.

Este lenguaje es comúnmente utilizado en desarrollo web, ya que permite la elaboración rápida de aplicaciones web robustas y estables, como también en la ciencia de datos y análisis, debido a sus diferentes librerías que facilitan esta labor, como lo son Numpy, Matplotlib, entre otras. Además, mediante librerías como Tensorflow, Keras permite desarrollar modelos de inteligencia artificial y ML, y a su vez, las bibliotecas como Tkinter o Pyqt permiten la elaboración de aplicaciones de escritorio con interfaces gráficas de usuario.

Cabe destacar que, en este lenguaje de programación, casi no hay diferencias entre librerías y bibliotecas, ya que ambas contienen módulos y funciones que facilitan la realización de una tarea específica, pero la primera mencionada suele contener conjuntos específicos de funciones, mientras que la segunda mencionada, es de carácter más amplio, ya que en ocasiones puede contener una agrupación de librerías.

A continuación, en la Figura 1-8, se observará el logo del lenguaje antes mencionado.

Fuente: https://1000logos.net/python-logo/

Figura 1-8. Python

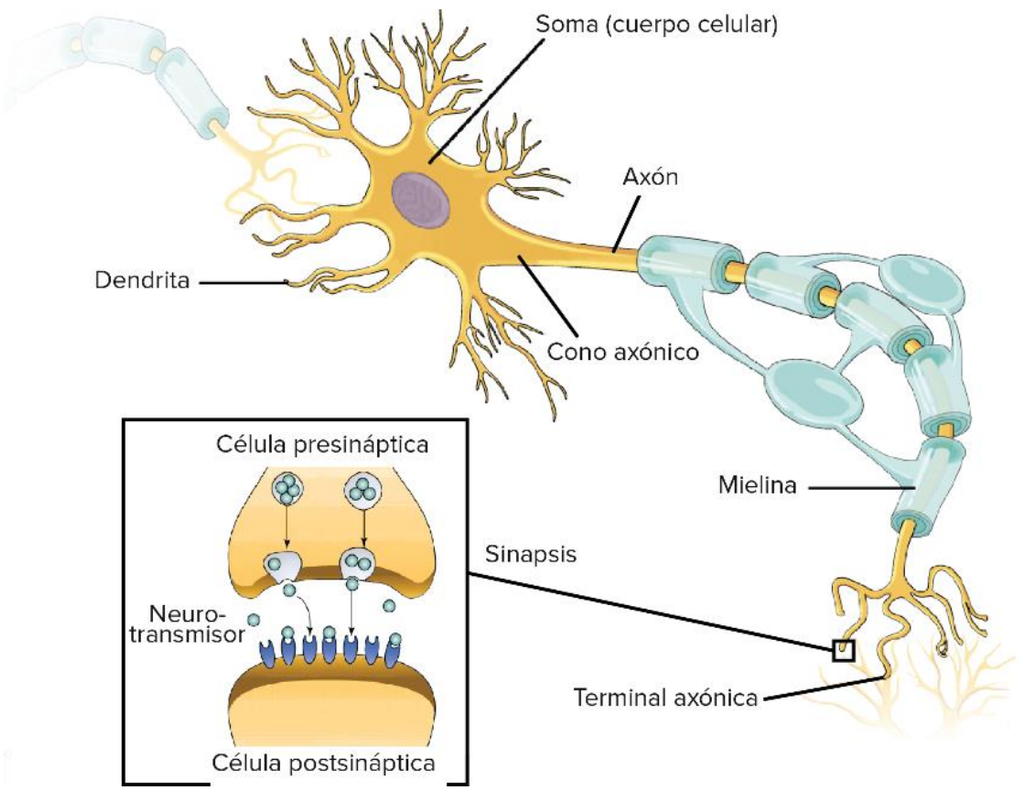
1.4. REDES NEURONALES

Las redes neuronales artificiales o redes neuronales simuladas constituyen la base de la inteligencia artificial, siendo un subconjunto de la ML y el eje central del algoritmo de la DP. Su nombre se debe a sus homónimas presentes en el cerebro humano, la cuales son encargadas de recibir y enviar información en forma de impulsos eléctricos. Este funcionamiento, al igual que el nombre, es copiado por las ANN.

1.4.1. Neurona humana

Las neuronas forman parte del sistema nervioso central, son las unidades funcionales y estructurales del antes mencionado. Básicamente su función es recibir la información desde los sentidos del mundo exterior, procesarla y enviar dicha información en forma de impulsos eléctricos hacia otras neuronas para ejecutar alguna respuesta asociada al estímulo.

A continuación, en la Figura 1-9, se puede apreciar la estructura de una neurona humana, donde se puede observar desde sus dendritas hasta sus terminales axónicas, pasando por el Soma, Cono axónico, Axón, Mielina.



Fuente: https://es.khanacademy.org/science/biology/human-biology/neuron-nervous-system/a/overview-of-neuron-structure-and-function

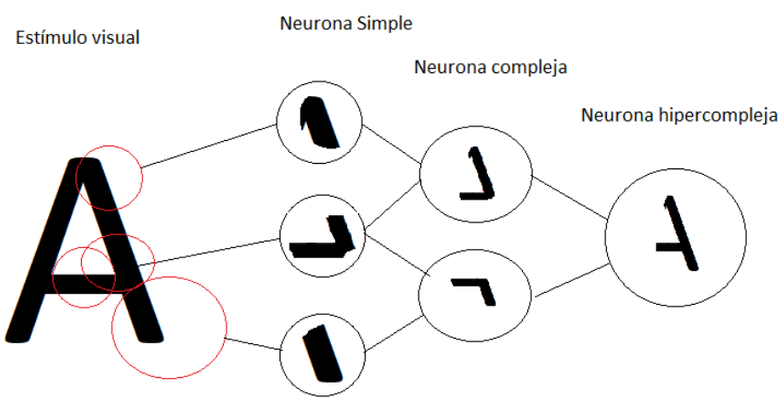
Figura 1-9. Neurona humana

1.4.2. Descubrimiento de David Hubel y Torsten Wiesel

En el año 1981, los profesores David Hubel y Torsten Weisel recibieron el premio Nobel de medicina por sus descubrimientos en relación con el procesamiento de la información en el sistema visual. Los cuales se obtuvieron tras realizar experimentos en la década de los 60 del siglo pasado en la facultad de medicina de la Universidad de Harvard, donde utilizaron a un gato para investigar cómo es que el sistema visual procesa una imagen. La primera observación fue determinante, ya que descubrieron que la corteza visual no responde a estímulos luminosos como puntos de luz, pero si a figuras complejas, como líneas y figuras geométricas. Posterior a esto, descubrieron que cada neurona responde a estímulos que están ubicados en un determinado lugar del campo visual, además de que la intensidad del impulso emitido por la neurona dependía de la orientación de la figura y a su vez, algunas neuronas solo se activaban cuando la figura se movía en una sola dirección. Las neuronas que respondían a lo antes descrito las llamadas “neuronas simples”. Del mismo modo, descubrieron que algunas neuronas reaccionaban a estímulos más complejos como bordes de figuras y los movimientos de estos, a estas neuronas las llamaron “neuronas complejas”, así mismo descubrieron las “neuronas hipercomplejas” que respondían a estímulos compuestos más complejos, por ejemplos, pares de figuras formando ángulos entre ellas en campos visuales más amplios que las anteriores mencionadas.

Estos descubrimientos demostraron que el sistema visual basa su funcionamiento en la identificación de detalles simples, los cuales se hacen complejos a medida que avanzan en el sistema jerárquico de neuronas simples, complejas e hipercomplejas.

En la Figura 1-10, se puede observar lo antes mencionado, ya que la letra “A”, se puede descomponer entre sus partes más características.



Fuente: Elaboración propia mediante software para crear imágenes

Figura 1-10. Neurona simple, compleja e hipercompleja

1.4.3. Neurona artificial

Las ANN o SNN están compuestas por capas de entrada, salida y ocultas. Estas están compuestas por varias neuronas o nodos artificiales que, a su vez, están interconectadas con otros nodos de su misma o diferentes capas. Dichos nodos tienen entradas, un peso (ponderación) asociado a cada entrada, un umbral (sesgo) y una salida.

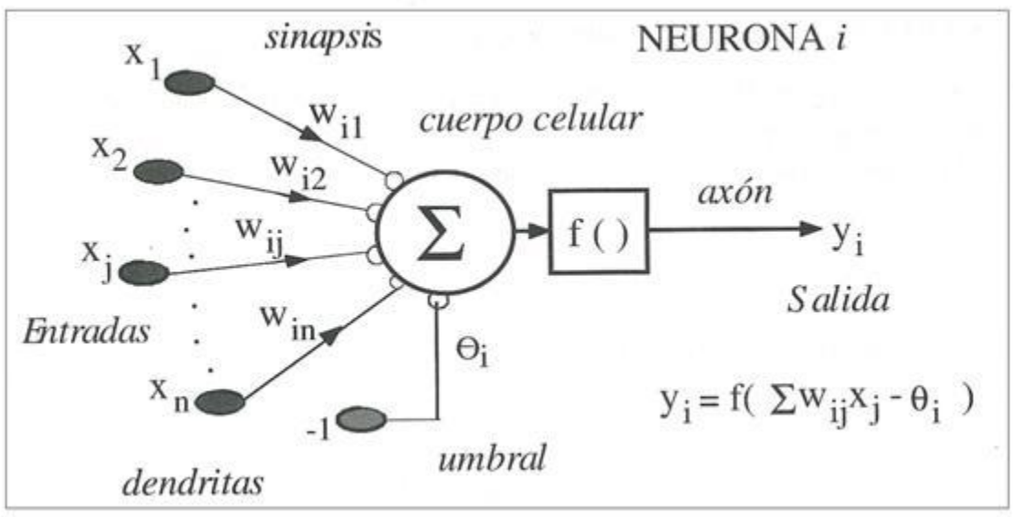
El peso o ponderación, representado por w, hace referencia a la importancia que se le asigna a cada entrada por separado, donde las que tienen mayor valor aportarán significativamente más a la salida versus las que tienen menos valor.

Las entradas, representadas por la letra X, hace referencia al valor que ingresa a cada nodo o neurona de una determinada capa, ya sea la de entrada, salida u oculta.

El umbral o sesgo, representado por Θ, hace referencia a una cantidad asignada, la cual debe ser superada para que la neurona pase la información a la siguiente capa o nodo.

En otros términos, la multiplicación de las entradas con sus respectivos pesos debe superar el valor del umbral para que la neurona o nodo se active y transmita la información al siguiente lugar. Esta acción de enviar información hacia adelante tiene como nombre propagación directa.

A continuación, en la Figura 1-11 se puede apreciar la estructura de una neurona artificial, donde se puede visualizar sus entradas o dendritas, umbral, su cuerpo celular y el lugar donde se ejecutará la función de activación, el cual se encuentra entre el cuerpo celular y la salida, donde se determinará si la neurona tendrá o no una salida para pasar a la siguiente estructura neuronal.



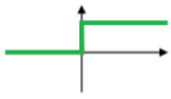
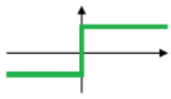
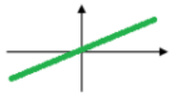
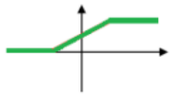
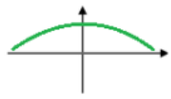
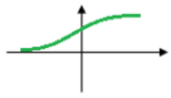
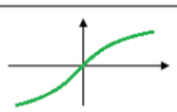
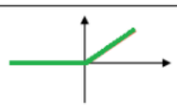
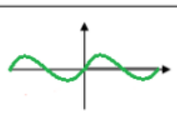
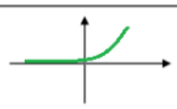
Fuente: https://grupo.us.es/gtocoma/pid/pid10/RedesNeuronales.htm

Figura 1-11. Neurona artificial y su fórmula de salida

1.4.4. Función de activación

La función de activación entrega un valor de salida en una neurona artificial de acuerdo con un valor de entrada, usualmente ésta está dentro de un rango de valores determinados de acuerdo con la función utilizada, en otras palabras, es una función matemática que se aplica a la salida de una neurona artificial, en la cual se decide si ésta se activa o no, convirtiendo la entrada ponderada en una salida que será utilizada en la neurona siguiente de las capas de la ANN, la función antes descrita es de vital importancia para la red, ya que le agregan la no linealidad lo que permite el aprendizaje y representar relaciones compleja en los datos procesados, mientras que esto no sucedería si su comportamiento fuera lineal.

A continuación, en la Figura 1-12 se observará un listado de funciones con su respectivo rango de valores, como el tipo de neurona en la cual se aplica.

TIPO DE FUNCIÓN DE ACTIVACIÓN	ECUACIÓN	EJEMPLO O MODELO	GRÁFICO
Función "Step" o Heaviside	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	PERCEPTRÓN	
Función Signo	$\phi(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	PERCEPTRÓN	
Función Lineal	$\phi(z) = z$	ADALINE	
Función lineal definida a trozos	$= \begin{cases} 1 & z \geq \frac{1}{2} \\ z + \frac{1}{2} & -\frac{1}{2} < z < \frac{1}{2} \\ 0 & z \leq -\frac{1}{2} \end{cases}$	MÁQUINAS DE VECCTOR SOPORTE	
Función Gaussiana	$\phi(z) = Ae^{-Bz^2}$	REDES NEURONALES RBF	
Función Sigmoide o Logística (Curva "S")	$\phi(z) = \frac{1}{1 + e^{-z}}$	REDES NEURONALES MULTICAPA	
Tangente hiperbólica	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	REDES NEURONALES MULTICAPA	
Función rectificadora o función ReLU Unidad Lineal Rectificada	$\phi(z) = max(0, z)$	REDES NEURONALES MULTICAPA	
Función Sinusoidal	$\phi(z) = A\,\mathrm{sen}(\omega z + \varphi)$	CLASIFICACIÓN DE PATRONES	
Función rectificadora suavizada (softplus)	$\phi(z) = \ln(1 + e^{z})$	REDES NEURONALES MULTICAPA	

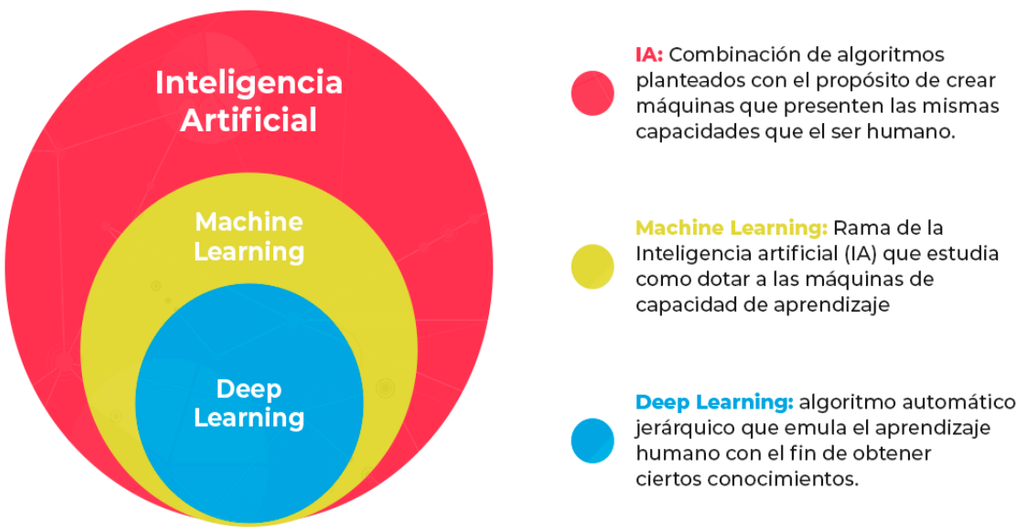
Fuente: https://www.joober.eu/introduccion-a-la-inteligencia-artificial-ia/#google_vignette

Figura 1-12. Funciones de activación

1.4.5. Inteligencia artificial

Cuando se habla de inteligencia artificial se puede hacer referencia a un sistema creado, el cual se estudia dentro de la informática, éste es capaz de realizar tareas que, hasta este entonces, se pensaba que se requería de inteligencia humana para llevarlas a cabo, ya que el aprendizaje, razonamiento, percepción y la comprensión del lenguaje necesitan de un determinado grado de nivel cognitivo para poder ejecutarlas de manera adecuada. En concreto, la IA es la simulación de los procesos de la inteligencia humana por parte de sistemas informáticos, el cual abarca cierta cantidad de subdisciplinas y técnicas, desde el aprendizaje automática hasta redes neuronales profundas, como por ejemplo la visión por computadora. Donde en el primero mencionado se hace uso de algoritmos que identifica patrones y tendencias en grandes volúmenes de datos para poder mejorar el rendimiento de una tarea, además destaca la adaptabilidad del sistema, ya qué este puede recepcionar de buena manera nuevos datos para mejorar el rendimiento, mientras que el último mencionado utiliza un conjunto ordenado y finito de operaciones que permite identificar y clasificar objetos, personas en imágenes y videos.

Para continuar es necesario definir la manera en que la IA, ML, DP y redes neuronales se relacionan entre sí. Al observar la Figura 1-13 se puede apreciar como la IA es el sistema global que, dentro de sí, contiene como subconjunto la ML y a su vez la DL es un subcampo de la ML, mientras que las redes neuronales son la columna vertebral del algoritmo de la DL.



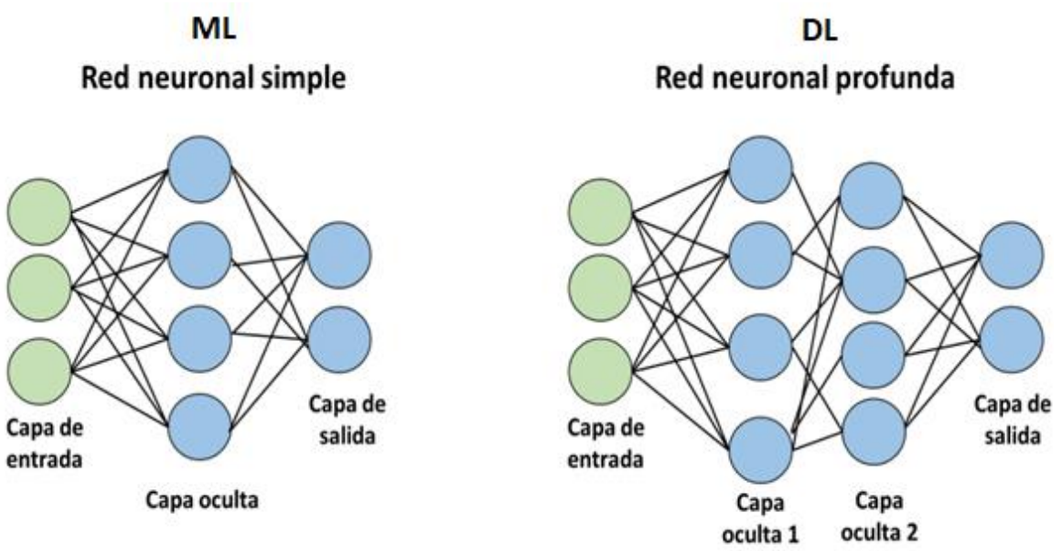
Fuente: https://www.masterdatascienceucm.com/que-es-machine-learning/

Figura 1-13. Diagrama IA

1.4.5.1. Deep Learning

Como se mencionó en el párrafo anterior, este subcampo de la ML usa redes neuronales de múltiples capas para poder simular el funcionamiento del cerebro humano cuando realiza la toma de decisiones complejas.

En la Figura 1-14 se puede apreciar la diferencia entre ML y DL respecto a sus capas, ya que ML solo tiene una capa oculta, mientras que DL es de multicapas como se mencionó anteriormente. Es necesario aclarar que al hablar de capas se hace referencia al conjunto de nodos que forman la red, las que pueden ser capa de entrada y de salida que son las capas visibles, y la capa oculta que es donde se procesa la información mediante cálculos.



Fuente: https://www.masterdatascienceucm.com/que-es-machine-learning/

Figura 1-14. Multicapas de redes neuronales

La DL para poder imitar el funcionamiento del cerebro humano respecto a la toma de decisiones realiza la combinación de entrada de datos, ponderaciones y sesgos. Y de este modo, poder reconocer, clasificar y describir con precisión los objetos dentro de los datos de entrada.

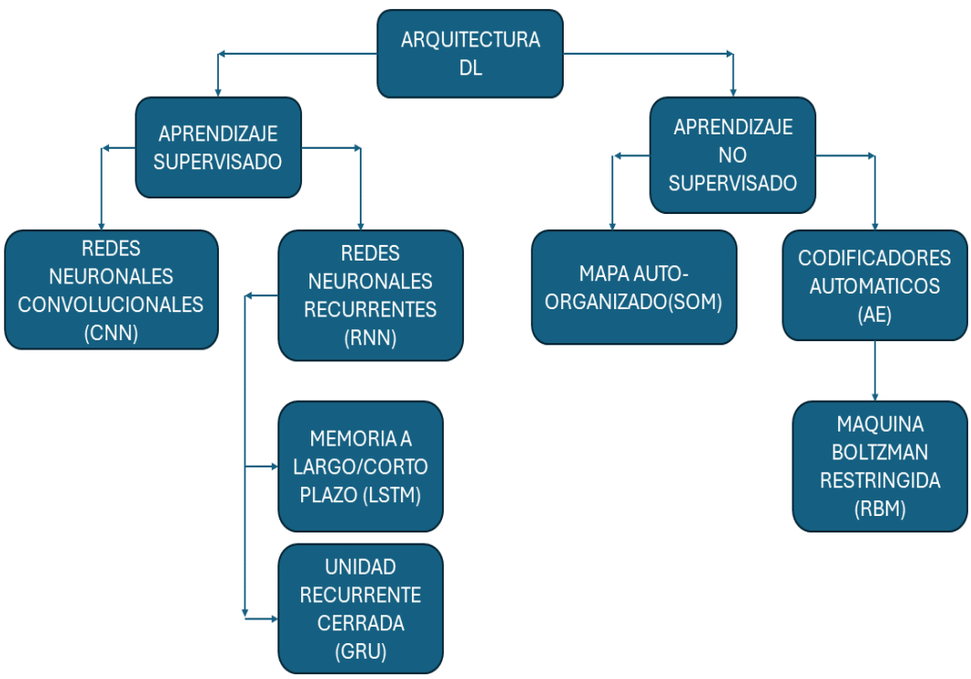
La interconexión de los nodos de las diferentes capas permite contar con propagación directa y retroprogramación. la primera hace referencia a la información del nodo anterior que se utiliza para refinar y optimizar la predicción o categorización, mientras que la segunda opción utiliza algoritmos como el descenso de gradiente para calcular el error en la predicción, y así poder entrenar el modelo moviéndose hacia los nodos anteriores para cambiar los pesos y sesgos de las capas. Cuando trabajan en conjunto logran que en el tiempo el algoritmo se vuelva más preciso, es decir, su porcentaje de error disminuye y la precisión de las predicciones aumenta, pero no siempre una cantidad de tiempo exagerada de entrenamiento logrará resultados positivos, en ocasiones se producen sobreajustes, o subajustes si se cuenta con un tiempo acotado de entrenamiento, para evitar esto se deben tomar estrategias para mitigar los problemas antes mencionados.

1.4.6. Arquitectura DL

Cuando se habla de DL no se hace referencia a solo un algoritmo y topología, sino a diferentes clases que se pueden modelar y además de aprender complejas representaciones de datos, las cuales se pueden aplicar a diferentes problemáticas según el requerimiento, estos se pueden separar en aprendizaje no supervisado y supervisado, como se puede observar en la Figura 1-15.

El primer aprendizaje antes mencionado se caracteriza por no contar con datos etiquetados en el destino en su etapa de entrenamiento, donde se destacan las redes neuronales convolucionales y las redes neuronales recurrentes, mientras que el aprendizaje supervisado si utiliza en su etapa de entrenamiento datos claramente etiquetados para hacer las predicciones, donde podemos encontrar el mapa auto-organizado y los codificadores automáticos.

Este conjunto de modelos neuronales permite solucionar una basta variedad de problemas complejos mediante el aprendizaje de características abstractas de un gran conjunto de datos y representaciones jerárquicas de estos. Además, la selección de la arquitectura adecuada dependerá del tipo de dato y la tarea especifica a realizar.



Fuente: Elaboración propia mediante PowerPoint

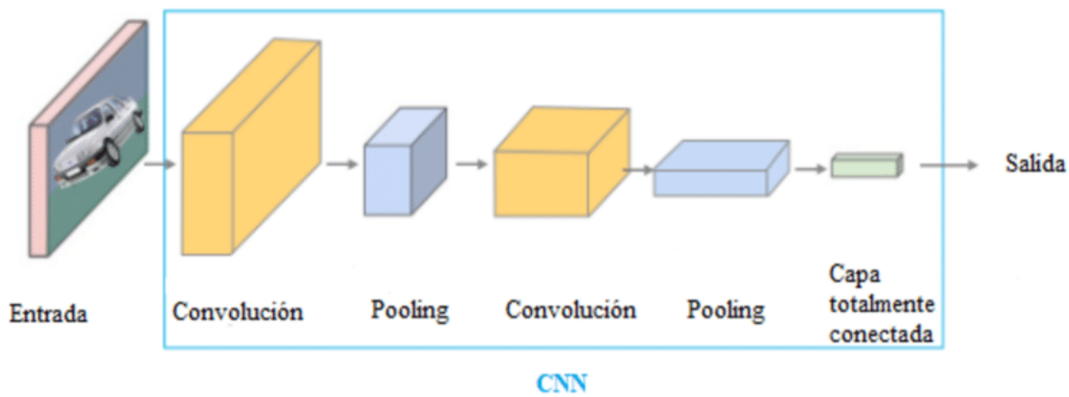
Figura 1-15. Arquitectura Deep Learning

1.4.6.1. Red neuronal convolucional

Mediante esta red, la cual es un tipo de red neuronal perteneciente al DL, se puede realizar la tarea de reconocer imágenes u objetos utilizando datos tridimensionales, ya que está diseñada especialmente para procesar y analizar datos en una estructura de matriz o de cuadricula, como los son las imágenes, además de capturar características especiales y patrones locales a través de las operaciones convolucionales. Esta información de entrada será procesada por sus tres capas características, las cuales son:

- Capa convolucional
- Cada de agrupación o pooling
- Capa completamente conectada

En la Figura 1-16 se puede apreciar el orden de las capas que generalmente se presenta en la red neuronal descrita anteriormente. Cabe destacar que la primera y última capa es la convolucional y completamente conectada respectivamente, mientras que las otras dos pueden ser adicionadas para realizar análisis más complejos de la imagen de entrada. En otras palabras, el primer conjunto de capa convolucional y de agrupación se centrará en procesar características simples de la imagen, como los colores y contorno, mientras que, con las capas adicionales de las capas antes descritas, se podrá reconocer características más complejas, además de poder analizar mayor área de la imagen en cuestión. Esto permitirá que el reconocimiento y clasificación de imágenes se vuelva más preciso y eficiente, ya que irá mejorando la capacidad de extraer los atributos de los datos de entrada.



Fuente: https://www.researchgate.net/figure/Figura-1-Descripcion-del-funcionamiento-de-una-red-neuronal-convolucional-CNN-10_fig1_348825166

Figura 1-16. Estructura red neuronal convolucional

1.4.6.1.1. Capa convolucional

Capa donde se realizan la mayor cantidad de los cálculos. Se considera como el componente central de las CNN y que para su funcionamiento requiere datos de entrada, filtro, núcleo o detector de características y un mapa de características, mapa de activación o característica convolucional.

Para comprender cómo operan sus componentes, se supondrá como dato de entrada una imagen formada por matriz de pixeles, la cual cuenta con 3 dimensiones (alto, ancho y profundidad). Donde el filtro, que es una matriz de peso de dos dimensiones (generalmente de 3x3, al igual que el campo receptivo), se aplicará a un área de la imagen, donde se procederá a aplicar el producto escalar del píxel de entrada y el núcleo, para luego introducir el resultado en una matriz de salida. Una vez finalizado este proceso en toda la imagen se obtendrá como desenlace el mapa de características.

A medida que el detector de características va avanzando por la imagen mantiene constante su peso. El cual, como parámetro, solo se ajusta en la etapa de entrenamiento mediante la retropropagación y el descenso de gradiente. Mientras que existes tres hiperparámetros que afectan el tamaño del volumen de salida y que se deben establecer antes del entrenamiento de la red neuronal.

- Cantidad de filtro: determina la profundidad de la salida, es decir, con tres filtros diferentes producirán 3 mapas de características diferentes, a su vez, creando una profundidad de 3.
- Zancada: número de pixeles, o distancia que el filtro se mueve por la matriz de entrada, normalmente se usan valores inferiores a 3 en la zancada, ya que una zancada mayor entrega un resultado menor.
- Relleno cero: Se usa cuando el núcleo no se ajusta de manera adecuada a la imagen de entrada, igualando a cero todos los elementos que queden fuera de la matriz de entrada.

Existen tres tipos de relleno:

- Relleno válido: Conocido como sin relleno, esto produce que la última convolución se descarte si los valores no se alinean.
- Mismo relleno: Cuando la capa de salida tiene el mismo tamaño que la capa de entrada.
- Relleno completo: Cuando se aumenta el tamaño de la salida agregando ceros a la entrada.

1.4.6.1.2. Capa convolucional adicional

La capa convolucional adicional es la que se agrega a continuación de otra capa convolucional. Con esto se puede obtener que la CNN sea de carácter jerárquico, ya que la capa adicional verá los pixeles del campo receptivo de la capa anterior, como se puede observar en la Figura 1-17. Es decir, las partes separadas de la bicicleta constituyen una parte inferior de la red neuronal, mientras que la combinación de sus partes representa un patrón de nivel superior.

Fuente: https://www.ibm.com/topics/convolutional-neural-networks
Figura 1-17. Ejemplo bicicleta

1.4.6.1.3. Capa de agrupación o pooling

Esta capa realiza la reducción de la dimensionalidad de los parámetros de la entrada, haciendo un barrido de un filtro por toda la entrada, al igual que la capa antes mencionada, pero con la diferencia que este núcleo no tiene peso. Éste aplica una función de agregación dentro del campo receptivo, para completar la matriz de salida.
Destacando dos tipos de agrupación:

- Agrupación máxima: Se selecciona el píxel con mayor valor para ser enviado a la matriz de salida cuando el filtro se mueve a través de la entrada, esta agrupación se utiliza con más frecuencia.
- Agrupación promedio: Se calcula el promedio de los valores del campo receptivo para ser enviado a la matriz de salida.

Esta capa aporta beneficios como reducir la complejidad, mejorar la eficiencia y limitar el riesgo de sobreajuste, a pesar de la perdida de información.

1.4.6.1.4. Capa completamente conectada

Esta capa, que también es conocida como capa densa, tiene cada nodo conectado directamente con la capa de salida, a diferencia de las capas anteriores. Ésta cumple la función de clasificar las características extraídas a través de las capas anteriores mediante sus diferentes filtros, las cuales utilizan funciones lineales rectificadas, mientras que la capa descrita utiliza la función de activación softmax para clasificar la entrada de manera adecuada. El propósito de la capa densa se cumple gracias a su función de transición de 2D a 1D, es decir, la matriz que entrega la capa convolucional y de agrupación, la pasa a un vector unidimensional.

1.5. OBJETIVOS

En esta sección, se describirán el objetivo general y los específicos, los cuales se deberán alcanzar durante el desarrollo del presente documento, y así demostrar el proceso de aplicación de una red neuronal al reconocimiento facial.

1.5.1. Objetivo general

Elaborar un dispositivo de control de acceso mediante reconocimiento facial, utilizando hardware y software adecuados.

1.5.2. Objetivos específicos

- Seleccionar hardware y lenguaje de programación adecuado
- Elegir el modelo neuronal y cantidad de neuronas adecuadas
- Recopilar una buena base de datos para realizar entrenamiento
- Verificar sobre/subajustes durante o después del entrenamiento
- Una vez con la ANN entrenada, realizar programación para reconocimiento facial
- Aplicar reconocimiento facial a un control de acceso e interfaz para interactuar con la programación

CAPÍTULO 2: DESARROLLO DEL POYECTO

2. DESARROLLO DEL PROYECTO

En el capítulo que se observará a continuación, se abordarán los temas relacionados al progreso del proyecto en cuanto a las definiciones, significados y analogías, que ayudará a comprender el proyecto, pasando desde cómo se pueden representar los colores, viendo además el entrenamiento de una red neuronal y la interfaz de la programación realizada para interactuar con el script en Python.

2.1. COLORES DEL ESPECTRO

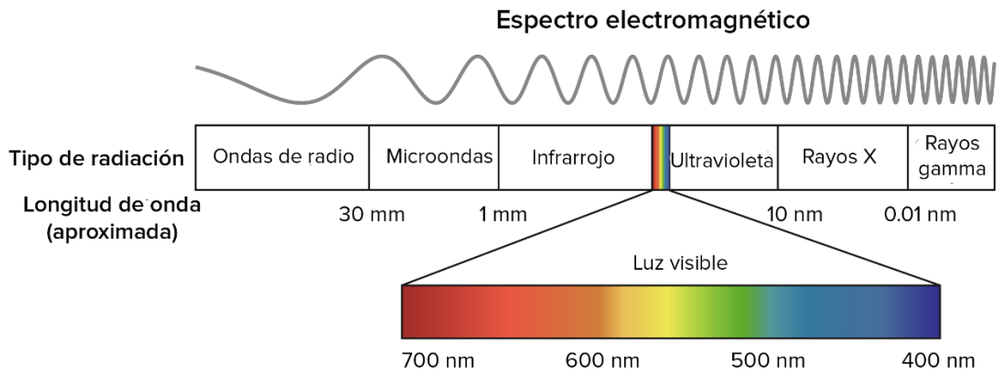
Para iniciar el desarrollo del proyecto anteriormente mencionado, es necesario comenzar describiendo como es que se logra percibir el mundo mediante el órgano sensorial llamado ojo.

En primer lugar, se debe conocer que la luz visible se define como aquel fragmento del espectro electromagnético que puede ser percibido por el ojo humano, esta fracción comprende las longitudes de onda entre los 700 nm y los 400 nm, lo que corresponde al color rojo y violeta respectivamente, además los colores que abarcan los colores que se pueden observar durante la aparición de los arco iris, cabe mencionar que existen más colores, como por ejemplo el ultravioleta que corresponde a valores menores a 400nm de longitud de onda y el infrarrojo que son valores superiores a los 700nm, pero solo se puede apreciar los que están dentro de la longitud de onda que en un inicio se definió.
Ejemplo de longitudes de ondas con sus respectivos colores:

- Rojo: 620-750 nm
- Naranja: 590-620 nm
- Amarillo: 570 - 590 nm
- Verde: 495 - 570 nm
- Azul: 450 - 495 nm
- Índigo: 425 - 450 nm
- Violeta: 380 - 425 nm

A continuación, en la Figura 2-1, se podrá observar lo que abarca el espectro electromagnético, el cual dependiendo de su longitud de onda, será el tipo de radiación, por ejemplo, según la Figura antes mencionada, ondas que estén por sobre los 30 mm corresponde a las señales de radio, mientras que las que estén dentro de los valores de 30 mm y 1 mm corresponden a las microondas, las que estén dentro de 1 mm y los 700 nm son las señales infrarrojas, después se tiene el segmento de la luz visible por el ser humano, posterior están las señales de ultravioleta las cuales tienen medidas de 400 nm hasta los 10

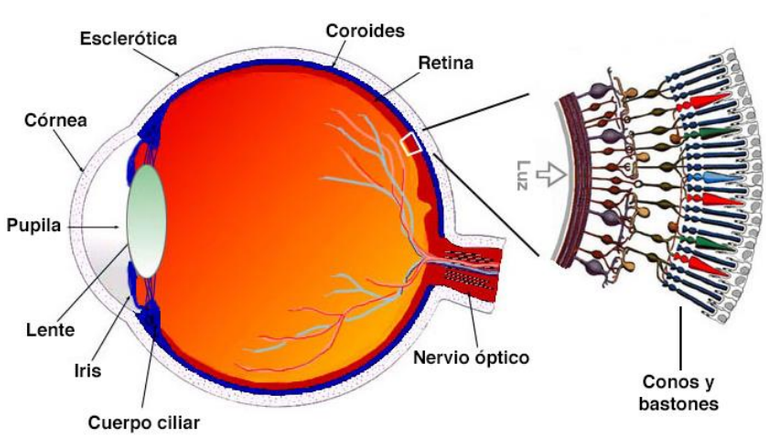
nm, los rayos X que son señales de 10 nm hasta los 0.01 nm de longitud de onda y los rayos gamma que son señales con menor de 0.01 nm de longitud de la onda.



Fuente: https://cdn.kastatic.org/ka-perseus-images/7ee5c536263ee5898fba95eb1e464f7f083d742c.png

Figura 2-1. Espectro electromagnético

La luz visible entra por la pupila impactando la retina, donde se alojan celulas especializadas para convertir la señal lumínica en señal eléctrica, éstas se denominan conos y bastones. Además, como se puede ver en la Figura 2-2, el ojo se compone de diferentes partes, las cuales puede interferir en la vision si es que algo llega a suceder, dando como resultado alteraciones en cómo percibimos el entorno, por ejemplo, cataratas que afectan el lente, mientras que la cornea puede desarrollar astigmatismo, queratitis, miopia, por otra parte, el nervio óptico normalmente sufre de glaucoma, el cual va dañando dicho nervio de manera irreversible produciendo la pérdida de la visión, por otra parte, en la retina, que es la capa de tejido sensible a la luz, donde se encuentran los conos y bastones mencionados anteriormente, se produce problemas en la pigmentación en los conos, produciendo cegera a ciertos colores.



Fuente: https://www.blueconemonochromacy.org/wp-content/uploads/2011/02/1_ES.jpg

Figura 2-2. Globo ocular

2.1.1. RGB

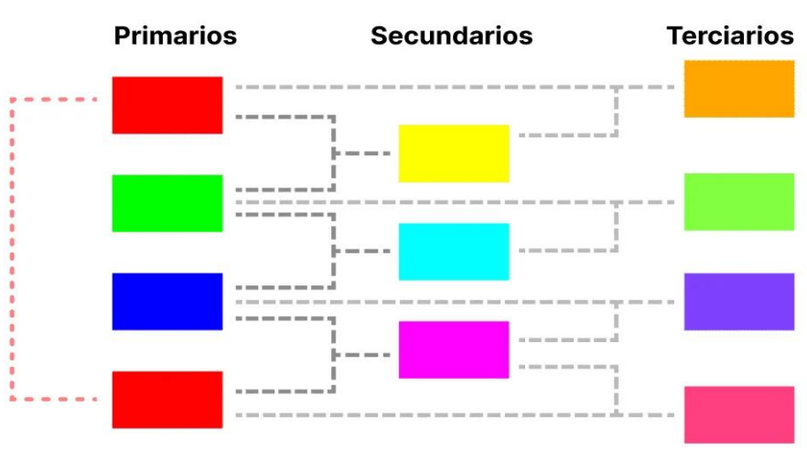
Este modelo representativo del color es de carácter aditivo, es decir, para conseguir un color diferente a sus tres primarios es necesario sumar estos últimos, siendo el rojo, verde y azul los colores mencionados anteriormente. Y dependiendo de la concentración de sus colores primarios se obtendrán diferentes resultados, los que tendrán como nombre colores secundarios, y a su vez, al mezclar los primarios y secundarios, dará como resultado los colores terciarios, como se pudo apreciar, este modelo se basa en la teoría de que la mayoría de los colores visibles se pueden obtener mezclando los tres colores que se consideran como primarios en diferentes proporciones

Este modelo de color recibe su nombre por las siglas de sus colores primarios en inglés (Red, Green, Blue), además es ampliamente usado en pantallas de TV, computadoras, entre otras tecnologías que impliquen la representación de imágenes en pantallas, ya que se aprovecha la capacidad del ojo humano para distinguir una amplia variedad de colores a través de la combinación de los tres colores primarios mencionados con anterioridad.

Para comprender cómo se consiguen la gran variedad de colores y sus diferentes concentraciones, se debe considerar que para cada concentración de los colores primarios se utilizarán 8 bits, por lo tanto, se podrán obtener 256 valores que van desde el 0 hasta el 255, el cual indica su concentración, donde:

- (0,0,0) es el color negro
- (255,255,255) es el color blanco
- (255,0,0) es el color rojo puro
- (0,255,0) es el color verde puro
- (0,0,255) es el color azul puro

En la Figura 2-3, se podrá apreciar que combinación de colores primarios dan como resultado un determinado color secundario, del mismo modo, como la mezcla de los dos antes mencionados se obtienen los terciarios.



Fuente: https://aprenderuxui.com/modelos-de-color-rgb-y-hex/aprender/uidesign/

Figura 2-3. Modelo de color RGB

2.1.2. HSV

Este modelo se caracteriza por ser la transformación no lineal del modelo del color antes mencionado, en otras palabras, en éste se puede describir y manipular los colores que se representan en un cono donde la saturación, el matiz y el valor se usan para mantener de manera ordenada cada una de las posibles variantes de colores.

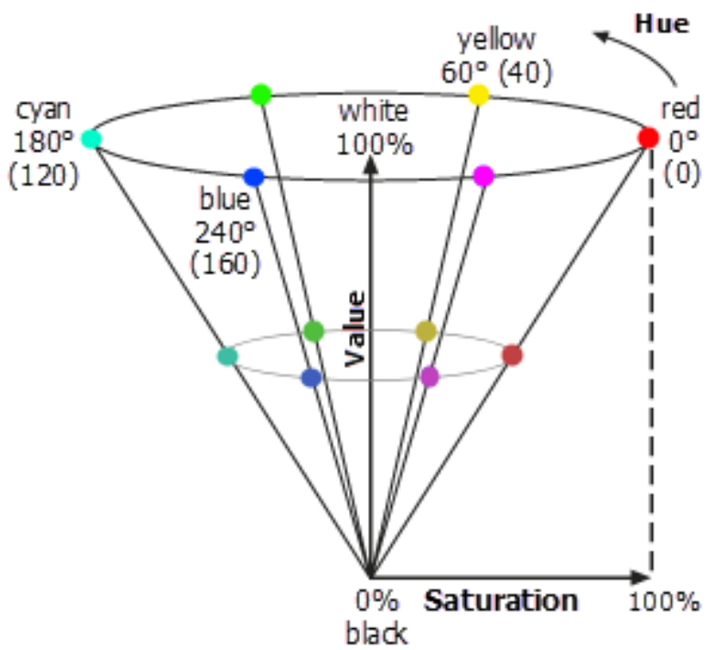
A continuación, se definirá los conceptos antes mencionados para poder entender la manera en que se relacionan entre sí.

- Matiz (HUE): Hace referencia al tono de cada color, ordenados del 0 al 240 de manera antihoraria, utilizando los 360° de la circunferencia en cuestión.

Los colores primarios están ubicados a 120° entre sí, partiendo con el rojo a los 0° en sentido antihorario, por lo tanto, el verde estará a los 120° y el azul a los 240°. Sucede lo mismo con los colores secundarios pero el amarillo está ubicado a los 60°

- Saturación (SATURATION): Se expresa con números enteros que abarcan desde el 0 hasta el 255, incluyendo ambas cifras. Lo que a su vez representa el 0% y 100% de la intensidad de la saturación presente en el color, es decir, para este modelo que se grafica en forma de cono, se puede expresar la saturación como la cercanía que tendrá un color en específico de un color neutro como lo es el gris, en otras palabras, si un color está en condición no saturado (0% de saturación), este será igual al color gris, mientras que si el mismo color está saturado (100% de saturación), este se puede presentar como un color puro.

A continuación, en la Figura 2-4, se podrá apreciar cómo los términos antes mencionados se ubican para representar la gama de colores.

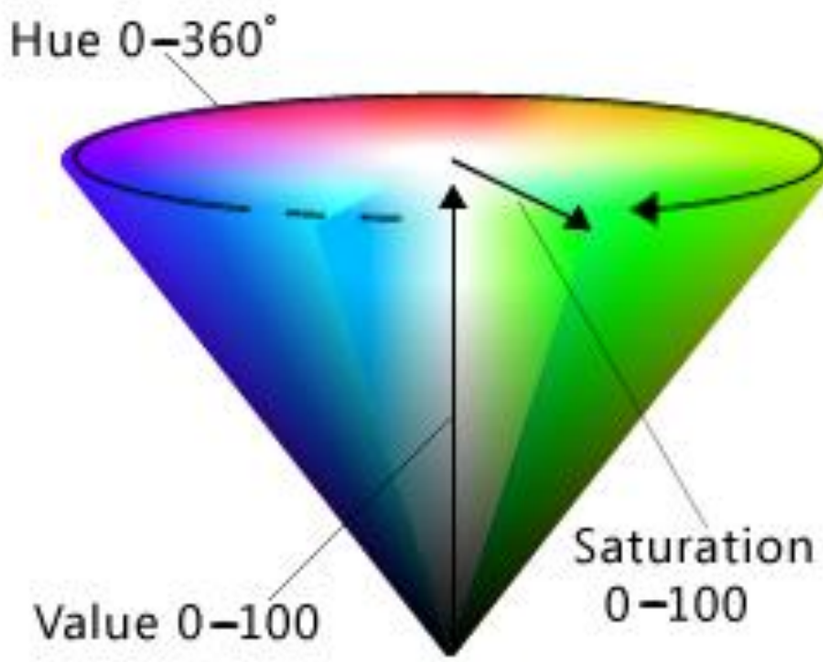


Fuente: https://pro.arcgis.com/es/pro-app/latest/help/analysis/raster-functions/color-model-conversion-function.htm

Figura 2-4. Modelo de color HSV ángulos

- Valor (VALUE): Se expresa con números enteros que abarcan desde el 0 hasta el 255, incluyendo ambas cifras. Lo que a su vez representa el 0% y 100% de la intensidad del blanco en el color, es decir, con un 100% de valor se conseguirá que el color sea lo más brillante posible, además si la saturación es del 0% el color pasará a ser blanco, mientras que si el valor es de 0% aparecerá la versión más oscura del color, y si se mantiene la misma condición antes mencionada, el color será negro.

En la Figura 2-5, se podrá observar cómo se representan los colores de acuerdo con el valor de estos.



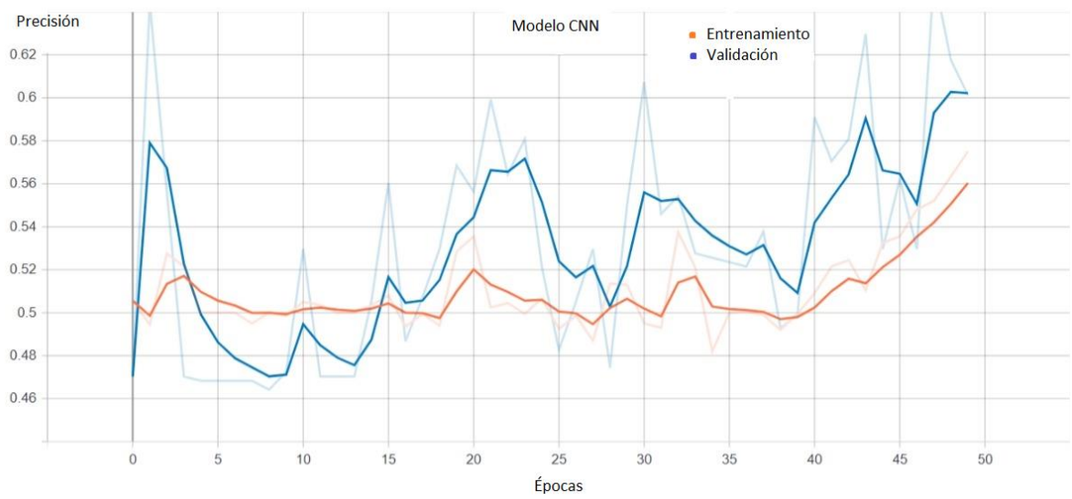
Fuente: https://learn.microsoft.com/es-es/windows/win32/uxguide/vis-color
Figura 2-5. Modelo de color HSV colorido

El modelo antes descrito se caracteriza por ser especialmente útil donde se requiera aplicar la manipulación, edición y procesamiento de imágenes mediante aplicaciones, como también el diseño gráfico, ya que se refleja de mejor manera cómo el humano percibe y procesa los colores, pero no es directamente compatible con la forma en que los dispositivos electrónicos con pantalla general el color que es observado, mientras que su contra parte, tiene como ventaja ser intuitivo para dichos dispositivos que emiten luz.

El modelo RGB basa su método de representación de colores en la combinación de colores primario que dan como resultado los secundarios y la mezcla de los dos antes mencionados dan los terciarios, además variado la concentración de los colores se pueden obtener mayor variedad de estos.

2.2. ENTRENAMIENTO RED NEURONAL

El objetivo del entrenamiento de la ANN es que ésta aprenda a realizar una predicción en base a los datos de entrada que ésta reciba, este fin se puede conseguir llevando a la red a analizar un conjunto de datos con o sin etiquetas. A lo largo de este proceso los datos conocidos con las predicciones se van comparando para realizar ajustes a las ponderaciones de manera gradual, con la finalidad de que la red se vuelva más precisa a medida que avanzan las épocas de entrenamiento y validación, generalmente este proceso se evalúa de acuerdo con los valores de pérdida y precisión de validación de las predicciones realizadas durante cada época antes mencionada. Al finalizar este proceso se pueden graficar los valores antes mencionados, de manera de verificar un posible sobreajuste o subajustes. Respecto al primer concepto, se da cuando un modelo de redes neuronales es muy complejo para la aplicación deseada, o en su defecto, no está optimizado para analizar los datos de entrenamiento y validación, como también, esto ocurre cuando se realizan pocas épocas del proceso en cuestión. En concreto, la red memoriza los datos mencionados con anterioridad en lugar de aprender a detectar características específicas que le ayudarán a identificar patrones en datos no incluidos durante el proceso antes descrito, por otra parte, en el gráfico de precisión esta condición sobreajuste se puede identificar ya que se obtiene una precisión alta durante el entrenamiento, pero baja durante la validación; además, las curvas de los datos antes mencionados convergen a lo largo del proceso en cuestión. Para la Figura 2-6 se puede observar solo 50 épocas de entrenamiento en un modelo de red adecuado para el procesamiento de imágenes, como lo es la red CNN, en este se puede apreciar la aparición de varias convergencias y grandes diferencias de precisión entre validación y entrenamiento.



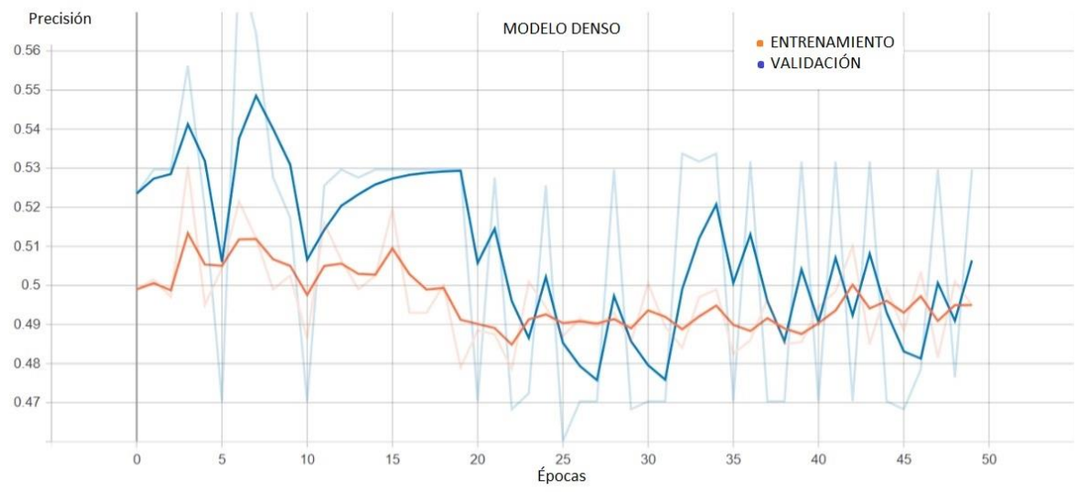
Fuente: Tensorboard

Figura 2-6. Gráfico precisión/épocas Modelo CNN

Respecto a la Figura 2-6, cabe destacar que el proceso que muestra la figura antes mencionada, existen dos curvas diferentes que se representan, las cuales sirven para evaluar y mejorar el rendimiento del modelo, estas son el entrenamiento y la validación. Respecto a la primera mencionada, corresponde al proceso en el cual el modelo aprende los datos analizados, además ajusta sus parámetros, como el peso de una red neuronal, para que la función de perdida de entrenamiento sea lo menor posible. Mientras que la segunda mencionada, corresponde a la fase en la cual se evalúa el rendimiento del modelo, es decir, la capacidad de que tiene el modelo para realizar predicciones precisas en datos que no se analizaron durante el entrenamiento.

Del mismo modo, su contraparte, el subajuste se produce por lo simple o inadecuado que es el modelo para analizar los datos deseados, que para el caso del presente documento los datos corresponden a imágenes de rostros de hombres y mujeres, también por disponer de bajos datos para el entrenamiento, del mismo modo, bajas épocas de entrenamiento también que de manera conjunta producen el efecto antes mencionado. Esta condición se caracteriza por tener bajos valores de precisión en el entrenamiento y validación, los cuales irán con pendiente negativa o mantendrán sus valores a lo largo de las épocas de entrenamiento de la red neuronal, como también contar con convergencias en la gráfica antes mencionada.

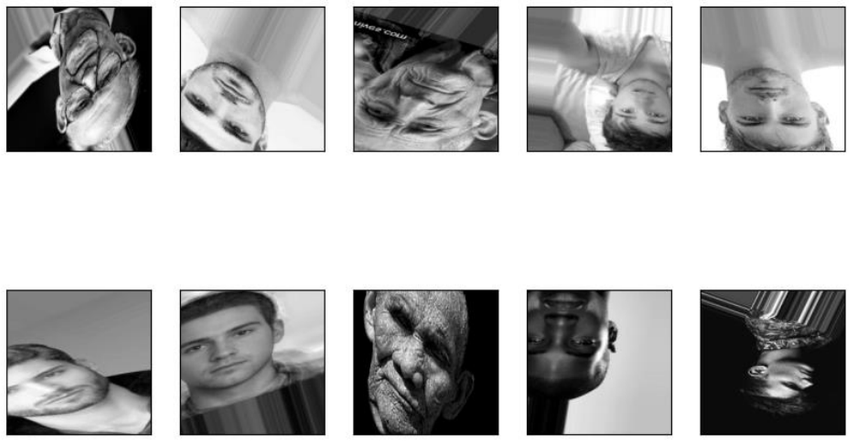
En la Figura 2-7 se puede observar un bajo nivel de precisión y convergencias durante todo el proceso, además de usar un modelo inadecuado para el procesamiento de imágenes como lo es el modelo denso.



Fuente: Tensorboard

Figura 2-7. Gráfico precisión/épocas Modelo DENSO

Existen diferentes maneras para contrarrestar los dos fenómenos mencionados, la primera de ellas es aplicar regulaciones como la L1 (Lasso); L2 (Ridge) y dropout o regulación por abandono, además, como se puede observar en la Figura 2-8, se puede diversificar los datos de entrenamiento en base a los existentes, haciendo rotaciones, recortes y traslaciones, como también aplicar funciones optimizadoras. Mientras que para el segundo fenómeno descrito anteriormente se deben reducir las regulaciones y aumentar la cantidad de datos de entrenamiento.



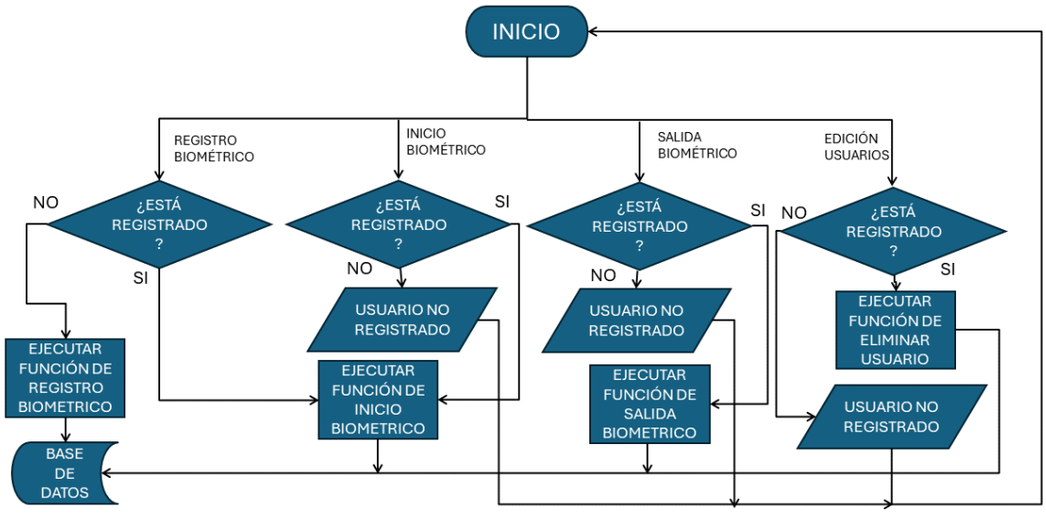
Fuente: Elaboración propia mediante sotfware para crear imágenes
Figura 2-8. Imágenes alteradas aleatoriamente

Por otra parte, una ANN sin los dos sucesos antes descritos se puede identificar que la gráfica de entrenamiento y validación irán aumentando gradualmente manteniendo su brecha hasta llegar a un valor estacionario alto y semejante entre ambos procesos, lo cual se verificará en el capítulo 3.

2.3. DIAGRAMA DE FLUJO DE LA PROGRAMACIÓN

De acuerdo con la elaboración del programa, éste tendrá 4 bonotes en la pantalla principal, los cuáles serán las entradas en el diagrama de flujo, donde al elegir una opción, las cuales son registro biométrico, inicio biométrico, salida biométrica y edición usuario, posterior, se abrirá una pantalla secundaria donde dependiendo cual sea la opción elegida, y se ejecutará una función que variará entre el registro biométrico; inicio biométrico, salida biométrica, la edición de los usuarios registrados anteriormente.

En la Figura 2-9 se puede observar cómo será la ejecución del programa en funcionamiento, donde al ingresar la opción deseada, dentro de las que se mencionó anteriormente, se ejecutará la función deseada.



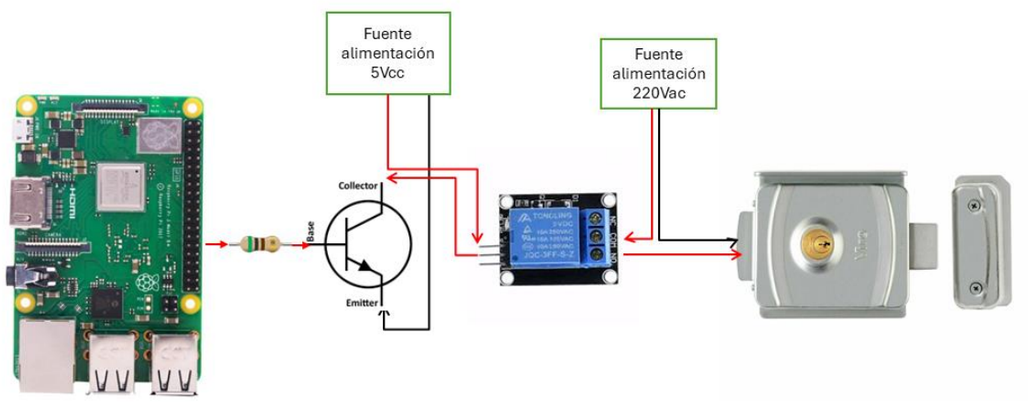
Fuente: Elaboración propia mediante PowerPoint

Figura 2-9. Diagrama de flujo de la programación

2.4. CONEXIONADO

De acuerdo con la Figura 2-10 se observa ver el diagrama de conexionado de los diferentes componentes que se mencionaron en el capítulo 1 del presente documento, los cuales son:

- Raspberry pi 3 model b+
- Transistor NPN y su respectiva resistencia de base para limitar la corriente
- Modulo relé
- Chapa eléctrica
- Alimentación de 220Vac y 5Vcc



Fuente: Elaboración propia mediante software para crear imágenes

Figura 2-10. Diagrama conexionado

2.5. MODIFICAR PERSONAS EN INTERFAZ

A continuación, se observarán diferentes interfaces con las cuales se interactuará con la programación en Python.

2.5.1. Registro biométrico

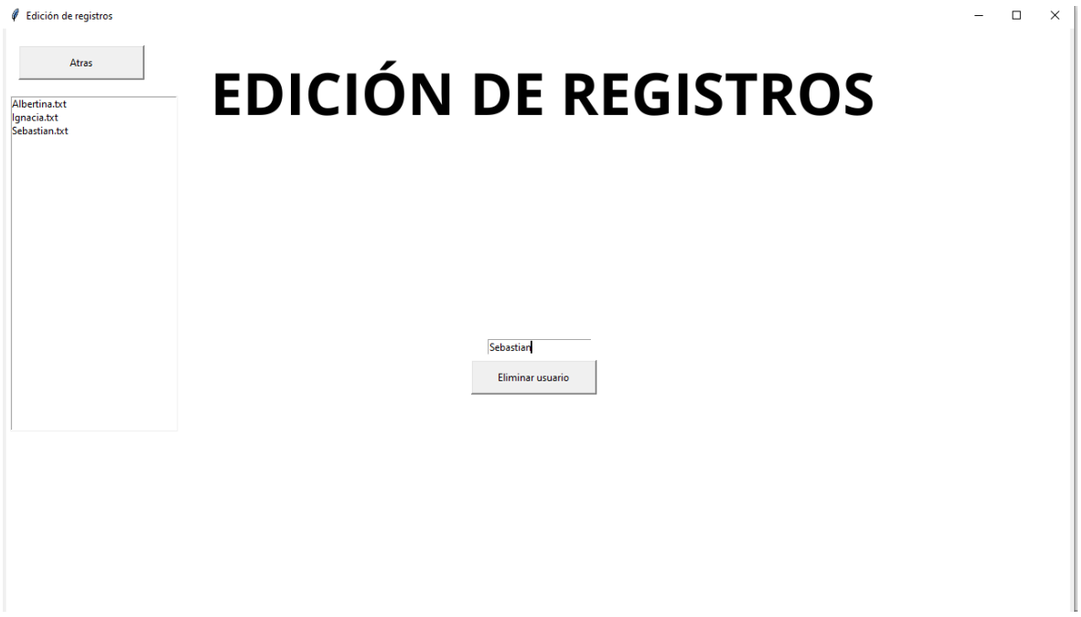
En la Figura 2-11 corresponde a la pantalla de inicio, donde se podrá realizar el registro de los diferentes usuarios. Para realizar esta acción se debe escribir el nombre que se desee registrar para que quede almacenado en la base de datos como el usuario, posterior a seleccionar la opción correspondiente, se abrirá una pantalla secundaria donde se tomará la imagen para ser guardada en la base de datos asociada al nombre del usuario, además, se podrá acceder a las distintas opciones que aparece en el interfaz.

Fuente: Elaboración propia, captura pantalla interfaz

Figura 2-11. Pantalla principal interfaz

2.5.2. Edición de usuarios

En la Figura 2-12 se puede apreciar la pantalla de edición de usuarios, la que corresponde a un toplevel de la pantalla principal. En ésta se debe ingresar el usuario y presionar en “eliminar usuario”, si el nombre se escribió adecuadamente, éste será eliminado, mientras que si se ingresa uno que no está en el listado, aparecerá un mensaje de error, los datos registrados se pueden visualizar en la parte superior izquierda donde están los usuarios registrados anteriormente.



Fuente: Elaboración propia, captura pantalla interfaz

Figura 2-12. Pantalla Edición usuarios

2.5.3. Hora de ingreso

A continuación, en la Figura 2-13 se puede observar la pantalla que muestra cuando se realiza el inicio biométrico, tras seleccionar la opción adecuada se procede a ejecutar la función inicio biométrico en el script en Python, donde se activará la cámara y se tomará una imagen para compararla con los rostros de la base de datos, cuando se encuentre una concordancia, se mostrará la imagen que se guardó del usuario en la etapa de registro biométrico, además del nombre de éste y la hora en que realizó el ingreso

Fuente: Elaboración propia, captura pantalla interfaz

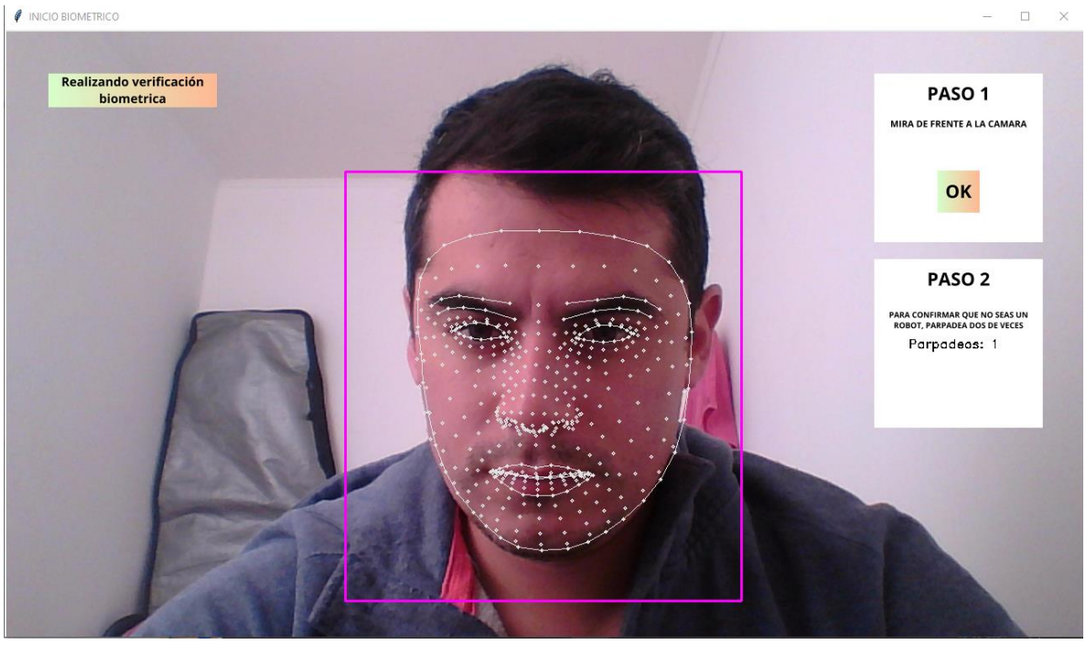
Figura 2-13. Pantalla inicio biométrico

2.5.4. Hora de salida

A continuación, en la Figura 2-14 se puede observar la pantalla que muestra cuando se realiza la salida biométrica, es decir, cuando seleccione la opción antes mencionada, se ejecutaré la función que tiene como nombre “Salida biométrica” en el script escrito en Python, dicha función tiene como resultado lo que se muestra en la Figura en cuestión, es decir, se mostrará la imagen capturada durante la ejecución de la función antes descrita, además del nombre en el cual se realizó el registro, como también la hora en la cual el usuario realizó la salida biométrica, mientras que la Figura 2-15 se puede observar la pantalla donde se activa la cámara para tomar la imagen para comprar con la base de datos.

Fuente: Elaboración propia, captura pantalla interfaz

Figura 2-14. Pantalla salida biométrica



Fuente: Elaboración propia, captura pantalla interfaz

Figura 2-15. Reconocimiento para inicio/salida biométrica

CAPÍTULO 3. PUESTA EN MARCHA DEL PROYECTO

3. PUESTA EN MARCHA DEL PROYECTO

En este capítulo, se darán a conocer los resultados del entrenamiento de los diferentes modelos, la ejecución de las neuronas entrenadas, como también de los resultados de las pruebas prácticas del programa funcional.

3.1. PRUEBAS PRÁCTICAS

A continuación, se analizarán los resultados del entrenamiento con 1000 imágenes para entrenar y 240 imágenes para validar de 3 redes neuronales, las cuales son: red neuronal densa, red neuronal convolucional y red neuronal convolucional con dropout, a los tres modelos se le aplicó un optimizador "ADAM" que es un optimizador adaptativo que se encarga de ajustar la tasa de aprendizaje para cada parámetro de manera individual, la cual acelera el aprendizaje en direcciones relevantes y haciendo lo contrario en direcciones irrelevantes; función de pérdida o de error es muy útil para evaluar la ANN, ya que indica la diferencia entre la predicción realizada con el valor real esperado, al realizar este proceso se permite ajustar los pesos de las neuronas para hacer que el modelo sea más preciso en las futuras predicciones, y como se podrá ver qué medida que las épocas de entrenamiento avanzan la función de pérdida irá disminuyendo en su valor, mientras que la precisión de la predicción irá aumentando de manera progresiva. Para este entrenamiento se utilizó "Binary cross-entropy" (entropía cruzada binaria) como función de pérdida, la cual es la medida cuantitativa de predicciones correctas respecto a los valores reales, es decir mide el error entre la predicción del modelo y la salida esperada, y métrica de evaluación se utilizará "Accuracy" (función de precisión), la cual se encarga de indicar cuántas predicciones correctas se hicieron en el total de predicciones realizadas.

3.1.1. Resultados del entrenamiento

En la Tabla 3-1, se puede observar que a medida que fueron aumentando las épocas de entrenamiento, los modelos CNN y CNNDO fueron disminuyendo los valores de pérdida, mientras que la precisión de la predicción fue aumentando, mientras que el modelo denso, siempre se mantuvo igual, es decir, no mejoró al aumentar las iteraciones en el periodo de aprendizaje.

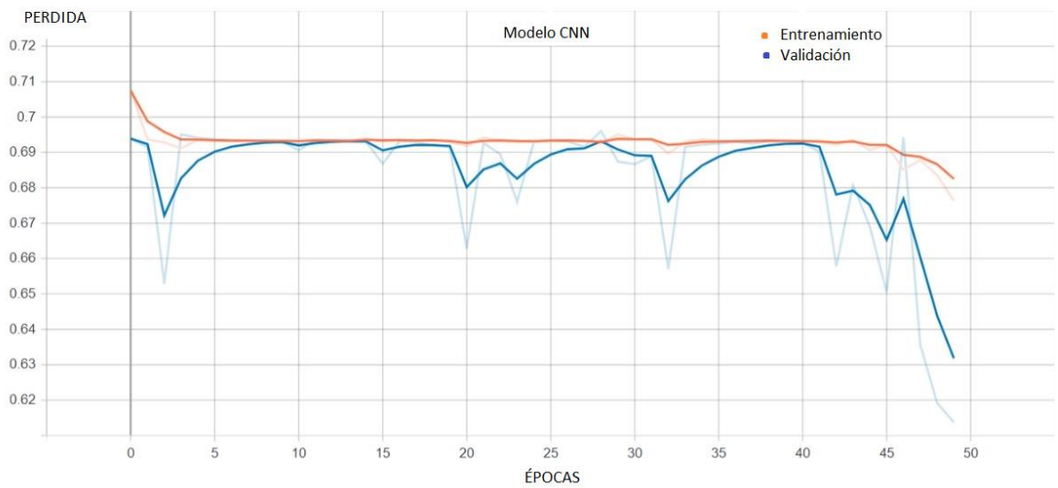
Tabla 3-1. Resultado entrenamiento

Modelos	DENSO		CNN		CNNDO	
Épocas	Entrenamiento	Validación	Entrenamiento	Validación	Entrenamiento	Validación
	Pérdida-Precisión	Pérdida-Precisión	Pérdida-Precisión	Pérdida-Precisión	Pérdida-Precisión	Pérdida-Precisión
50	0,69 - 0,49	0,69 - 0,52	0,67 - 0,57	0,61 - 0,60	0,60 - 0,66	0,54 - 0,72
100	0,69 - 0,49	0,69 - 0,52	0,55 - 0,71	0,44 - 0,82	0,56 - 0,7	0,59 - 0,7
200	0,69 - 0,47	0,69 - 0,47	0,48 - 0,75	0,36 - 0,84	0,48 - 0,76	0,35 - 0,82
300	0,69 - 0,49	0,69 - 0,47	0,35 - 0,82	0,29 - 0,86	0,42 - 0,79	0,34 - 0,85

Fuente: Elaboración propia

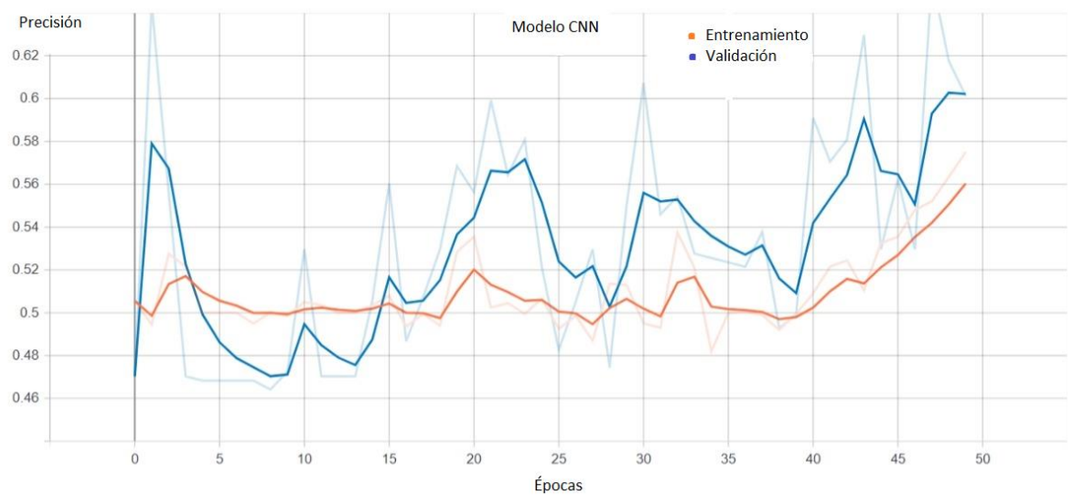
3.1.1.1 Gráficos 50 épocas CNN y CNNDO

En el siguiente grupo de gráficos, que abarca desde la Figura 3-1 hasta la 3-4, se puede apreciar como el modelo entrenado está subajustado por bajas iteraciones de entrenamiento como, por ejemplo, bajos valores de precisión y altos valores de pérdida



Fuente: Elaboración propia desde tensorboard

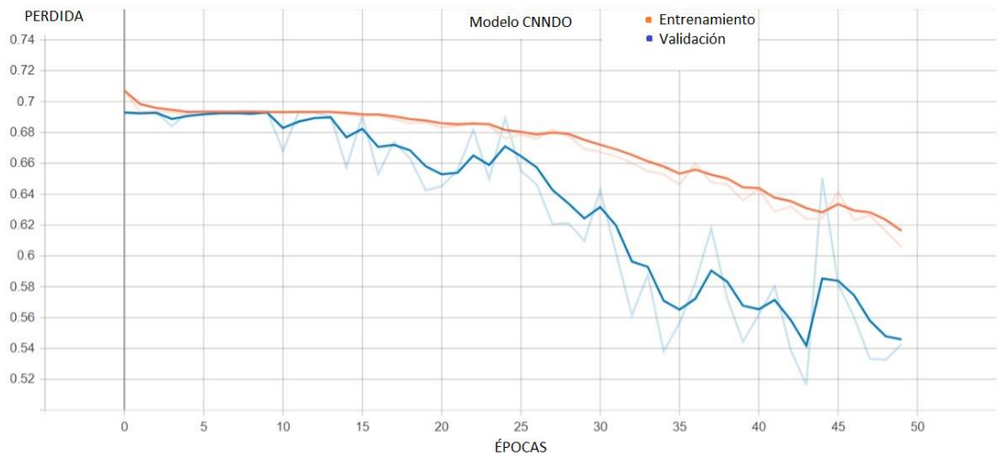
Figura 3-1. Gráfico Modelo CNN Función Perdida 50 épocas



Fuente: Elaboración propia desde tensorboard

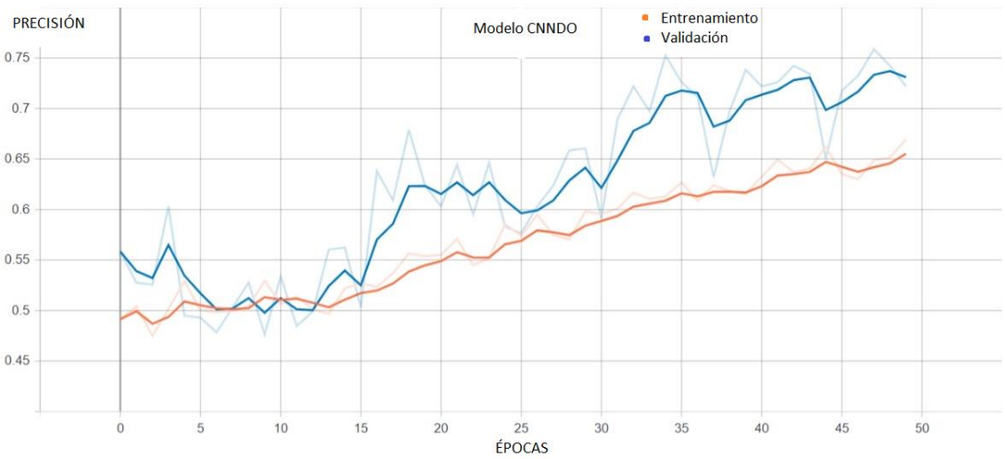
Figura 3-2. Gráfico Modelo CNN Función Precisión 50 épocas

En las Figura 3-1 y 3-2, se puede apreciar cómo la pérdida y la precisión se mantiene casi constantes hasta el periodo de entrenamiento número 40, lo cual da indicios que, al momento de aumentar las épocas, los valores irán acordes a los usos de las redes neuronales convolucionales.



Fuente: Elaboración propia desde tensorboard

Figura 3-3. Gráfico Modelo CNNDO Función Perdida 50 épocas



Fuente: Elaboración propia desde tensorboard

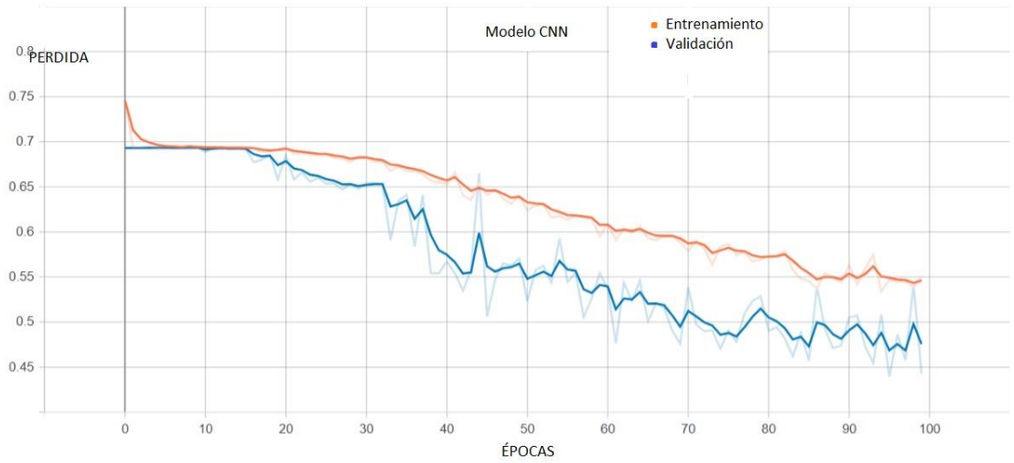
Figura 3-4. Gráfico Modelo CNNDO Función Precisión 50 épocas

En las Figuras 3-3 y 3-4 se puede apreciar como el modelo con dropout fue disminuyendo los valores de la función de pérdida de manera temprana, es decir, después de la época 15 aproximadamente, lo mismo se puede apreciar en la gráfica que muestra la función de precisión, ya que ésta va aumentando su valor a contar del periodo número 15.

Por lo tanto, en las Figuras 3-1 hasta la 3-4 se pudo apreciar que éstas tenían pocas épocas de entrenamiento, es decir, solo 50, pero a pesar de esto las gráficas tenían pendiente positiva de la función de precisión en el modelo, es decir, sus valores iban aumentando a medida que avanzaba el entrenamiento de la red neuronal, mientras que en la función de pérdida las gráficas tenían pendiente negativa, lo que se traduce en la disminución de los valores presentados, en el modelo CNNDO, mientras que el modelo CNN estuvo constante pero cerca del final del periodo comenzó a tomar valores esperados en una red de este tipo, esto da muestras que ambas redes neuronales convolucionales con y sin dropout son las adecuadas.

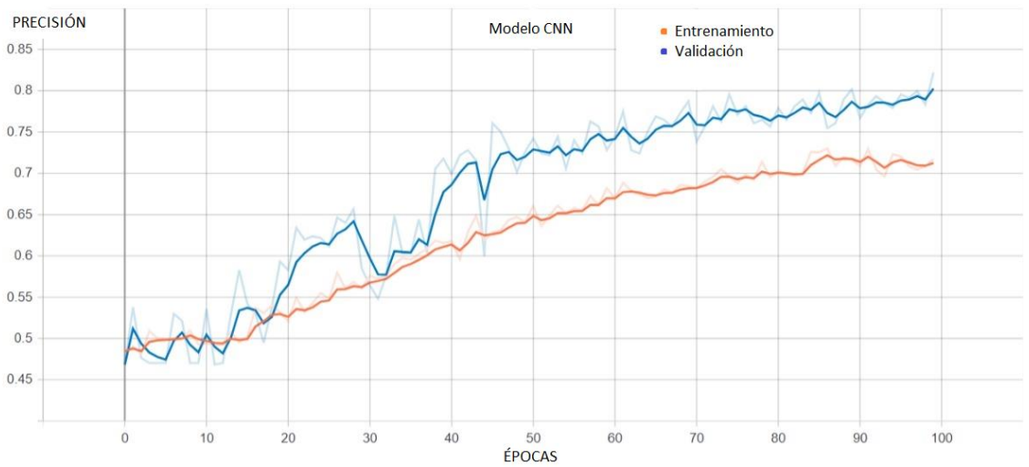
3.1.1.2. Gráficos 100 épocas CNN y CNNDO

En el siguiente grupo de gráficos, que comprende desde la Figura 3-5 hasta la Figura 3-8, se puede verificar como al aumentar las iteraciones se mejoran las curvas, pero aún siguen subajustadas.



Fuente: Elaboración propia desde tensorboard

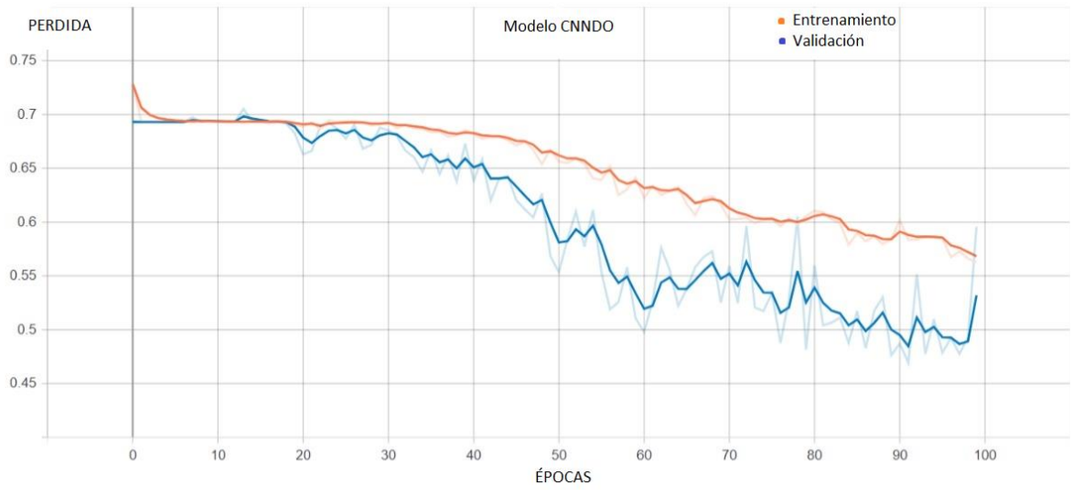
Figura 3-5. Gráfico Modelo CNN Función Perdida 100 épocas



Fuente: Elaboración propia desde tensorboard

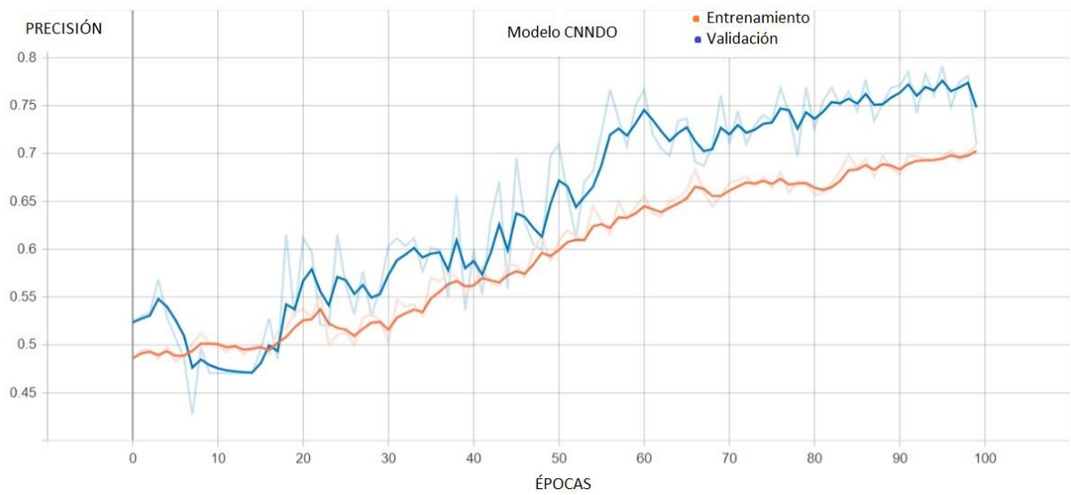
Figura 3-6. Gráfico Modelo CNN Función Precisión 100 épocas

En las Figuras 3-5 y 3-6, los cuales muestran las gráficas de pérdida y precisión respectivamente respecto a las 100 épocas de entrenamiento, se puede apreciar como éstas van tomando valores esperados respecto a una red CNN, es decir, la pérdida va disminuyendo y la precisión aumentando.



Fuente: Elaboración propia desde tensorboard

Figura 3-7. Gráfico Modelo CNNDO Función Pérdida 100 épocas



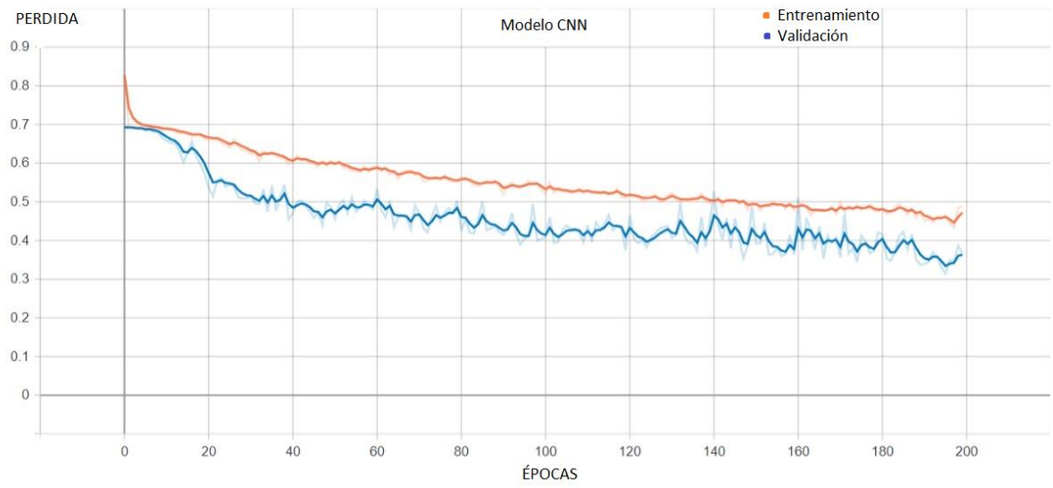
Fuente: Elaboración propia desde tensorboard

Figura 3-8. Gráfico Modelo CNNDO Función Precisión 100 épocas

En las Figuras que comprenden desde la 3-5 hasta la 3-8, se puede apreciar como el aumento de las épocas de entrenamiento permite que estos modelos se comiencen a comportar de acuerdo con sus características y según lo esperado, es decir, curvas de pérdidas con pendientes negativas mientras que las curvas de precisión con pendiente positiva, pero en las cuatro gráficas presentadas se puede apreciar que existe subajuste producto de que siguen siendo insuficientes los periodos de entrenamientos aplicados, esto se puede apreciar con las convergencias que se presentan durante los primeros 20 ciclos entre las curvas de los procesos de entrenamiento, de color naranjo, y el proceso de validación, de color azul.

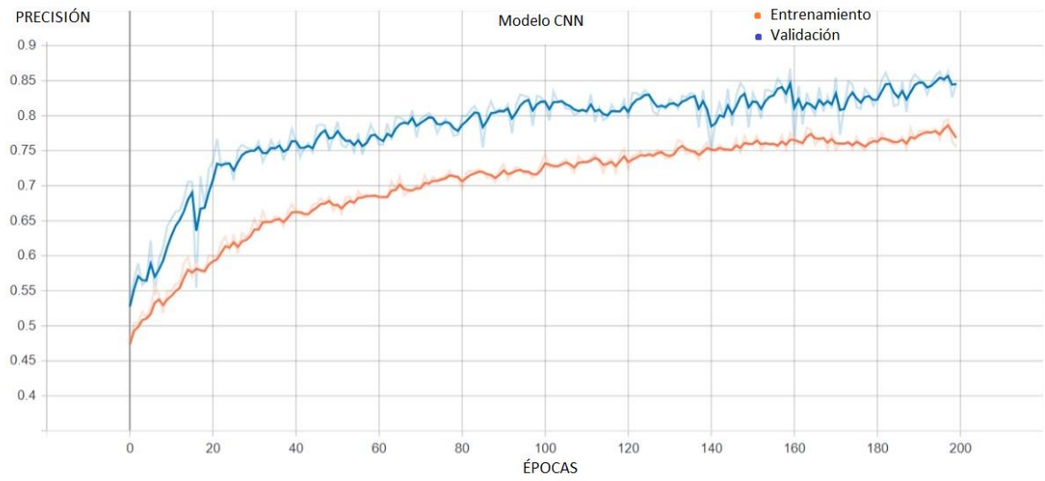
3.1.1.3. Gráficos 200 épocas CNN y CNNDO

En el siguiente grupo de gráficos, que abarcan desde la Figura 3-9 hasta la 3-12, se puede identificar como ya no se tiene convergencias que den luces de un sobreajuste o subajuste.



Fuente: Elaboración propia desde tensorboard

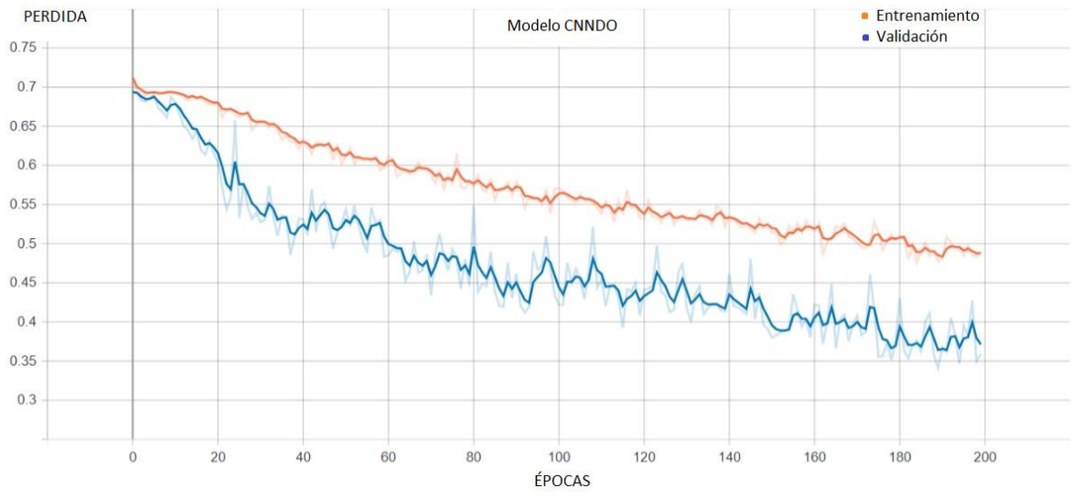
Figura 3-9. Gráfico Modelo CNN Función Pérdida 200 épocas



Fuente: Elaboración propia desde tensorboard

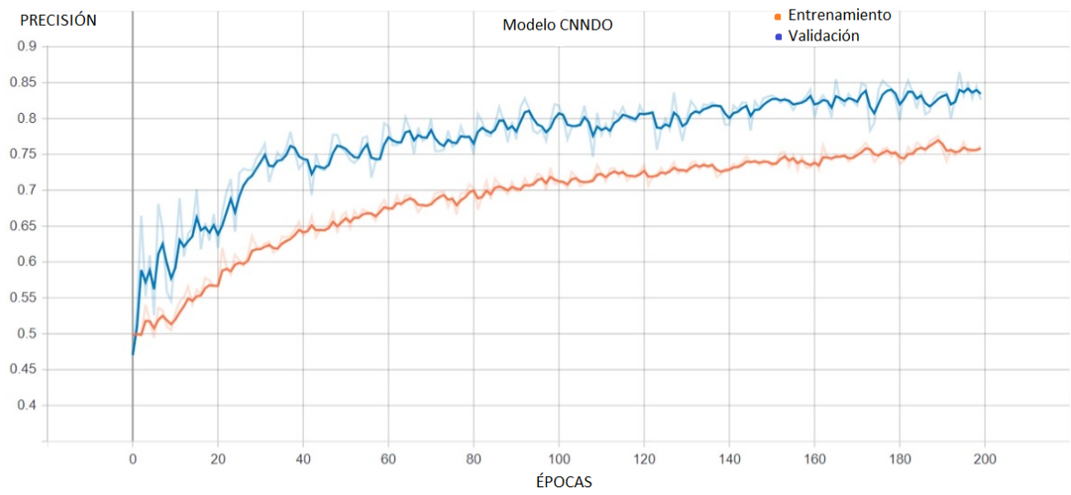
Figura 3-10. Gráfico Modelo CNN Función Precisión 200 épocas

En las Figuras 3-9 y 3-10 se puede apreciar el proceso de entrenamiento con 200 épocas de la red neuronal CNN, donde en las curvas se puede verificar que ya no existen las convergencias que se presentaron en los casos anteriores.



Fuente: Elaboración propia desde tensorboard

Figura 3-11. Gráfico Modelo CNNDO Función Pérdida 200 épocas



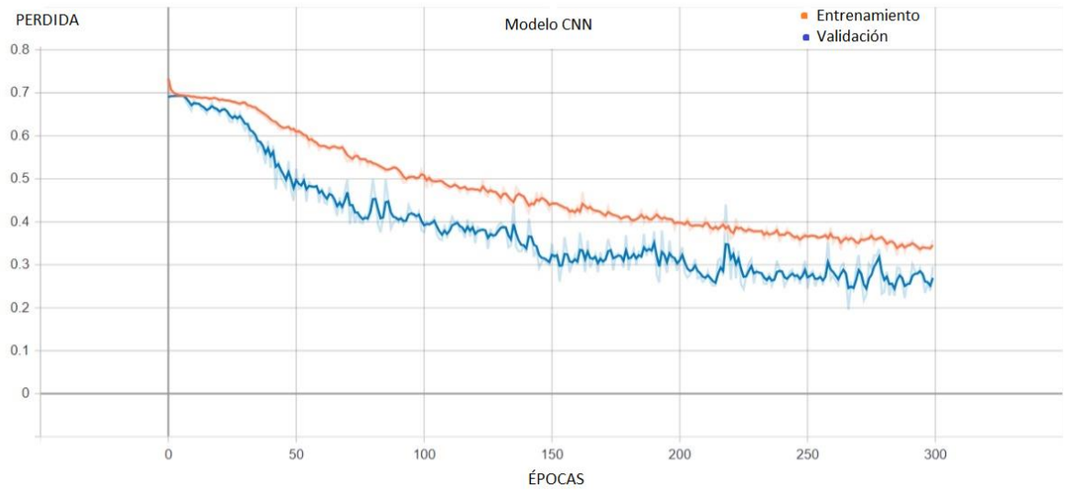
Fuente: Elaboración propia desde tensorboard

Figura 3-12. Gráfico Modelo CNNDO Función Precisión 200 épocas

En las Figuras que comprenden desde los números 3-9 hasta la 3-12 se puede apreciar como 200 ciclos de entrenamiento lograron que en ambos modelos se consiguieran resultados acordes a sus características, ya que el modelo CNN está diseñado para el reconocimiento de imágenes, al igual que el CNNDO, que solo lleva dropout como técnica de regulación, en ambos casos se puede verificar que en las funciones de pérdidas la curva de la validación obtiene valores menores a la gráfica de entrenamiento, mientras que en la función de precisión se obtienen valores inversos, es decir, la curva de entrenamiento tiene valores menores que la primera mencionada.

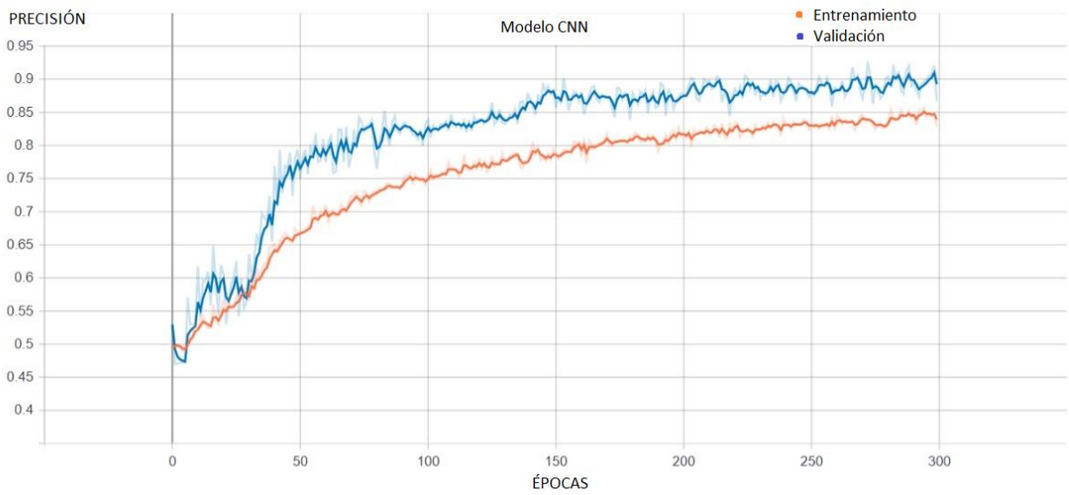
3.1.1.4 Gráficos 300 épocas CNN y CNNDO

En el siguiente grupo de gráficos, que comprenden entre la Figura 3-13 hasta la Figura 3-16, ya se puede observar que las curvas obtienen valores aceptables de pérdidas y de precisión sin tener convergencias ni grandes diferencias entre los valores de entrenamiento y validación.



Fuente: Elaboración propia desde tensorboard

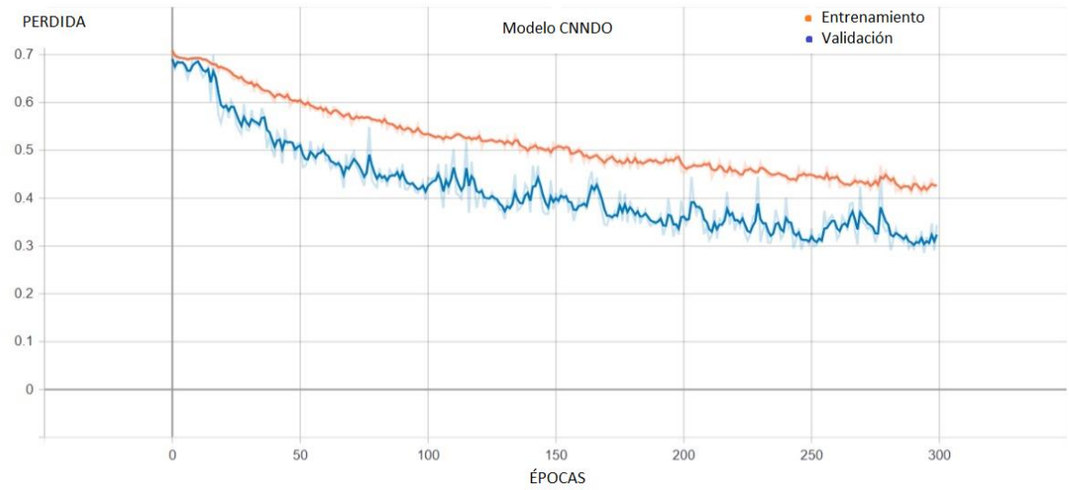
Figura 3-13. Gráfico Modelo CNN Función Pérdida 300 épocas



Fuente: Elaboración propia desde tensorboard

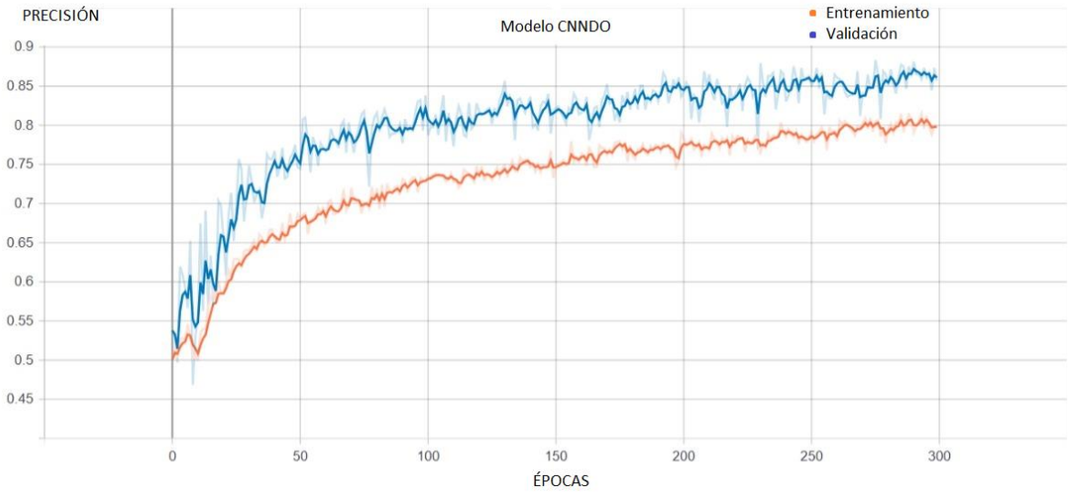
Figura 3-14. Gráfico Modelo CNN Función Precisión 300 épocas

En las Figuras 3-13 y 3-14 se puede observar la gráfica de validación y entrenamiento de una red CNN en ambas se puede apreciar que antes de las 50 épocas se produce un fenómeno que puede indicar un subajuste, como se mencionó anteriormente, pero después de dichas épocas se puede apreciar como las curvas comienzan a comportarse de manera esperada, es decir, con distancias constantes y casi paralelas, con lo que se puede establecer que el modelo mejoró y se puede considerar como un buen ajuste.



Fuente: Elaboración propia desde tensorboard

Figura 3-15. Gráfico Modelo CNNDO Función Pérdida 300 épocas



Fuente: Elaboración propia desde tensorboard

Figura 3-16. Gráfico Modelo CNNDO Función Precisión 300 épocas

En las Figuras que comprenden desde la 3-15 hasta la 3-16 se puede apreciar que antes de los 50 ciclos de entrenamiento, la red CNNDO también presentó señales de subajuste, como las gráficas anteriores, pero éstas mejoraron a lo largo de las épocas, mostrando señales de un buen ajuste. Cabe destacar que no existe una distancia específica entre curvas de validación y entrenamiento para determinar el proceso de entrenamiento como bueno, pero si se debe guiar en los indicios antes mencionados desde la Figura 3-1 hasta la 3-16, ya que curvas convergentes son una clara señal de subajuste, además como que las distancias a lo largo del entrenamiento no mantengan sus valores constantes imposibilitando que sus curvas estén paralelas, como se pudo observar en las gráficas que muestran los resultados con 50 épocas, mientras que en las últimas gráficas presentadas, se puede observar cómo se presentaron señales de subajuste en ciclos tempranos en las diferentes curvas, pero éstas fueron mejorando a lo largo del proceso manteniendo sus distancias y que por lo tanto obteniendo curvas casi paralelas.

3.1.1.5. Aplicación de modelos al reconocimiento facial

A continuación, como se puede observar en la Figura 3-17, se aplicó el modelo denso entrenado a un programa desarrollado en Python para el reconocimiento facial, donde se debe discriminar sí el rostro que se está reconociendo es mujer u hombre, según este modelo, el rostro expuesto tiene 49% de similitud con una mujer.

Fuente: Elaboración propia desde script para reconocimiento facial

Figura 3-17. Modelo denso aplicado al reconocimiento facial

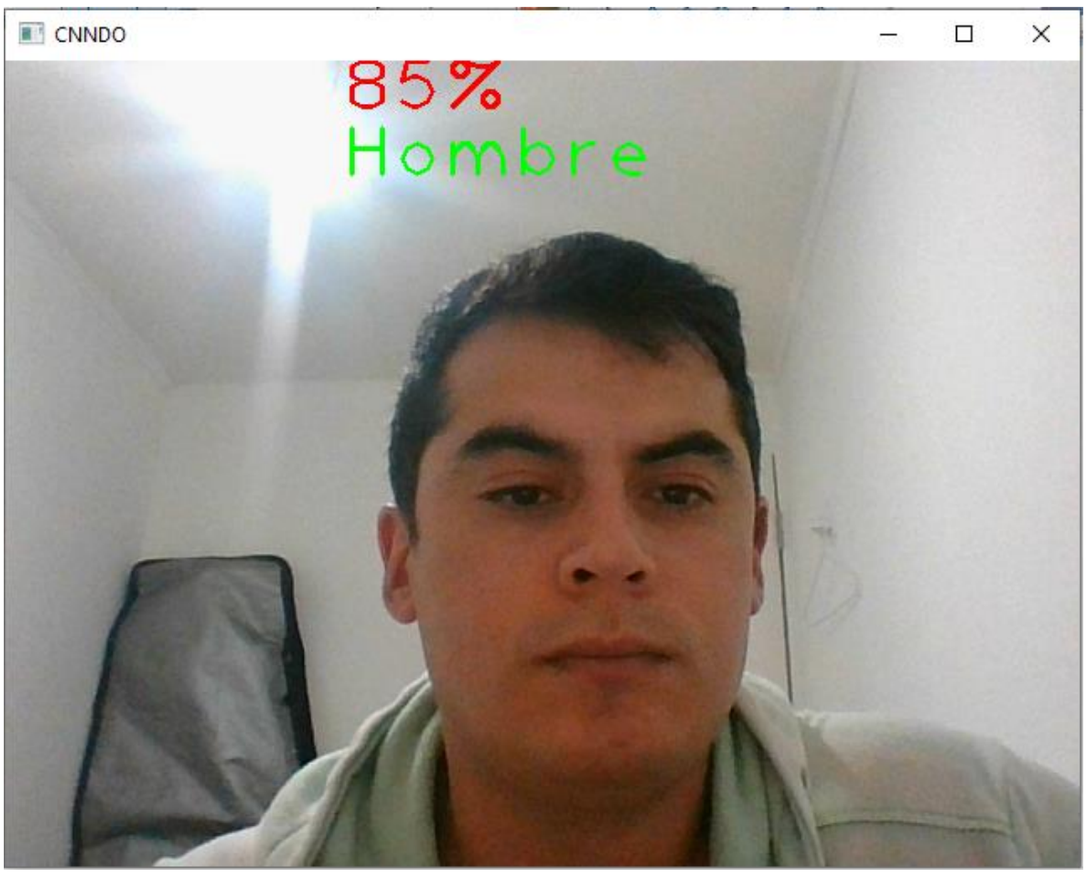
En la Figura 3-17, se muestra el rostro de un hombre, pero el modelo denso entrenado indica que se obtuvo un 49% de similitud con el rostro de una mujer, como se mencionó anteriormente, este porcentaje no variaba si se alteraba de alguna manera la dirección en la cual se mira hacia la cámara o si se atenuaba la intensidad de la iluminación, lo que indica que este modelo no responde de manera satisfactoria, a pesar que durante el entrenamiento se agregaron métodos para diversificar los datos para dicho propósito, por lo tanto, este modelo no cumple con lo deseado, es decir, no reconoce de manera satisfactoria el rostro de un hombre.

En la Figura 3-18, se puede apreciar el resultado de aplicar una red CNN entrenada al mismo programa mencionado anteriormente, es decir se debe discriminar sí el rostro reconocido pertenece a una mujer o a un hombre, donde, como resultado se obtuvo una similitud del 50% con un hombre ante el mismo sujeto que se analizó anteriormente. Pero con la salvedad que en este caso el porcentaje variaba según el movimiento del rostro de manera radial. Con lo antes mencionado se puede comprobar que al realizar la diversificación de los datos de entrenamiento haciendo que las imágenes cambiaran de posición de diferentes maneras y cambiando su tonalidad generó que el modelo realizara mejores predicciones.

Fuente: Elaboración propia desde script para reconocimiento facial

Figura 3-18. Modelo CNN aplicado al reconocimiento facial

En la Figura 3-19, se aplicó al mismo programa mencionado anteriormente para reconocer el rostro de una mujer o un hombre y diferenciarlos, además de indicar el porcentaje de similitud con un de los dos géneros antes mencionado, de acuerdo a la red neuronal aplicada fue la CNN con una técnica de regulación la cual es dropout, siendo por lo tanto CNNDO, además se puede apreciar que con este modelo se obtuvo el mayor porcentaje de similitud con un hombre versus los otros dos modelos mencionados anteriormente, cabe destacar, que, en este caso, al igual que el anterior, al momento de mover el rostro en diferentes ángulos el porcentaje mencionado iba variando, pero nunca bajando del 50%. Esto se debe a que, al igual que en los casos anteriores, durante el entrenamiento se realizó la diversificación de las imágenes de hombres y mujeres haciendo que éstas se roten en su propio eje en diferentes ángulos, cambiando la iluminación de las imágenes, algunas es escala de grises, entre otras alteraciones, y del mismo como que en el caso del modelo CNN se puede observar que este modelo un buen porcentaje de similitud.



Fuente: Elaboración propia desde script para reconocimiento facial

Figura 3-19. Modelo CNNDO aplicado al reconocimiento facial

3.1.2. Reconocimiento facial

A continuación, se observará en dos casos, como el programa realiza la captura de una imagen para posteriormente realizar el reconocimiento facial en las etapas de registro de usuario, inicio y salida biométrica.

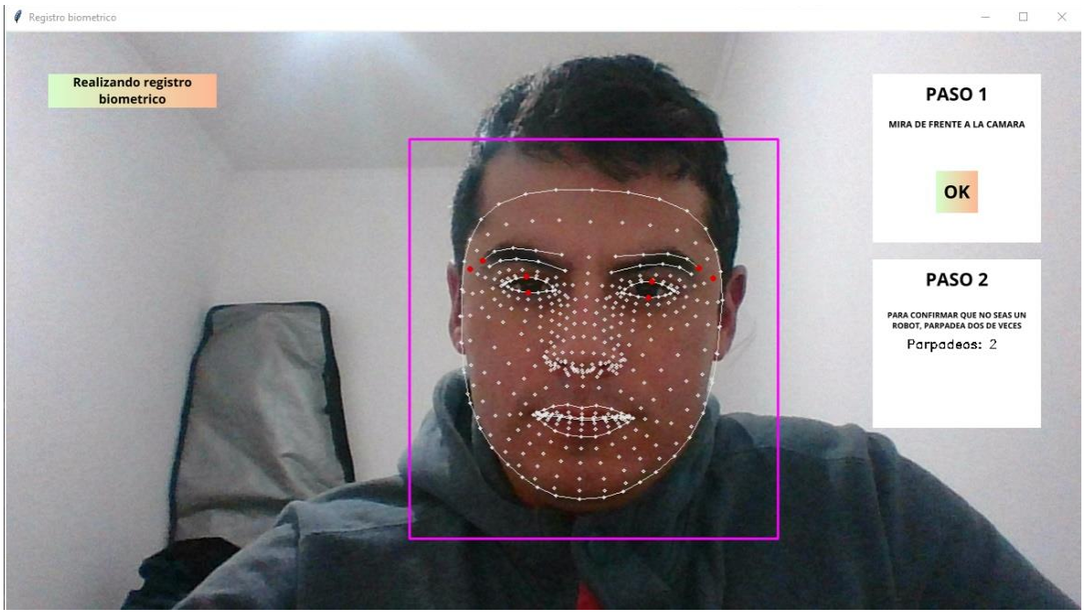
Cabe destacar que la forma en que se realiza el reconocimiento facial es la misma para las diferentes etapas mencionadas anteriormente.

3.1.3. Caso 1

Para el primer caso se analizará la Figura 3-20, en ésta se puede verificar la existencia de 8 puntos dentro de la malla facial. Con estas referencias se pueden realizar los pasos 1 y 2 de la programación, el primero de estos se trata de que se debe asegurar que el usuario esté mirando hacia el frente cuando realice el registro, inicio o salida biométrica, para conseguir esto en la programación se basó en los puntos de la malla ubicados en los parietales y los extremos de las cejas, estos tienen un valor de 139, 368 para los parietales derechos e izquierdo respectivamente y 70, 300 para las cejas derechas e izquierdas respectivamente. Mientras que, para el segundo paso, que corresponde al parpadeo, se basará en la distancia que existe entre los puntos ubicados en el centro de los parpados de la malla facial, los cuales son 145, 159 para el ojo derecho, mientras que para el ojo izquierdo son 374, 386.

Los puntos antes mencionados hacen referencia a las marcas de la malla facial que se obtiene con la librería mediapipe que contiene redes neuronales pre entrenadas, de Google. Ésta funciona con dos redes neuronales que trabajan en forma conjunta, una está entrenada para detectar rostros y ubicarlos dentro de un recuadro, la antes descrita es la “face detection model” (Modelo Detección de rostros), mientras que el modelo de puntos de referencias del rostro o “face landmark model”, opera sobre el rostro detectado aplicando una malla sobre el rostro con 468 puntos clave de referencia, para los cuales cada valor corresponde a un punto de la malla antes mencionada, por ejemplo, se puede apreciar que en la Figura 3-20, hay dos puntos rojos, los cuales en el script tienen un valor de 159 para el centro del parpado superior del ojo derecho y 145 para el parpado inferior del mismo ojo, con estos valores, podremos determinar si se parpadea o no.

Los puntos antes mencionados no son valores fijos, estos van variando a medida que nuestro rostro detectado se mueve de posición.



Fuente: Elaboración propia desde script para reconocimiento facial

Figura 3-20. Captura imagen para registro

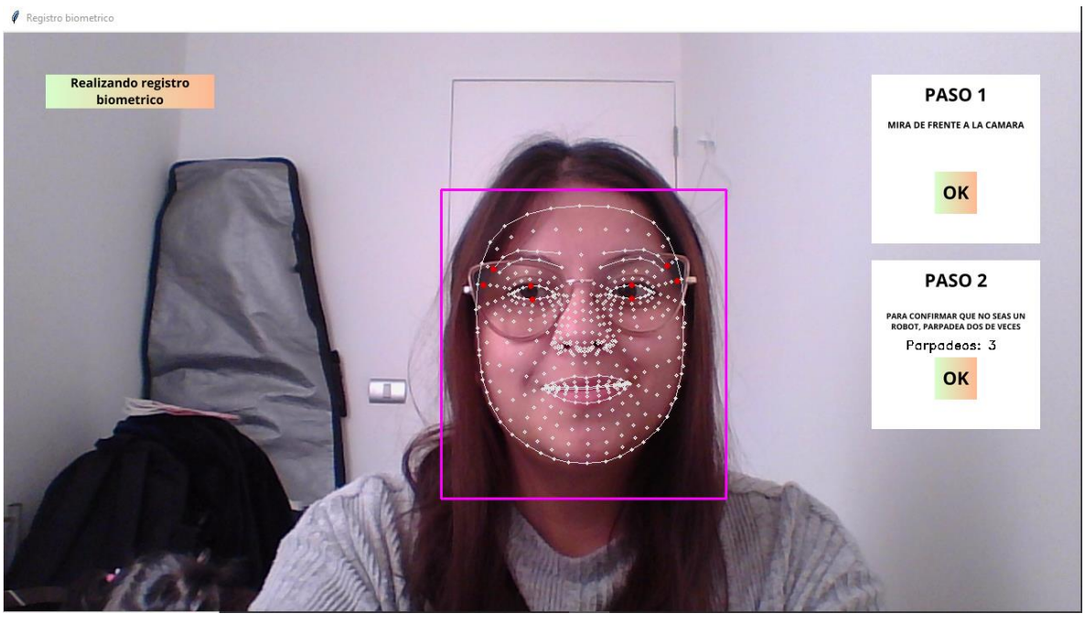
En la Figura 3-21, se puede apreciar la imagen capturada al momento de realizar los dos pasos antes mencionados, ésta se guardará en la base de datos, donde se almacenarán los datos ingresados, como las imágenes capturadas, nombres de usuarios, horas de ingreso y salida. Dentro de este conjunto de información, se utilizarán las imágenes guardadas para realizar una comparación entre el archivo antes mencionado de la base de datos con la imagen que será tomada cuando se realice el inicio o salida biométrica. Si se obtiene una similitud del 50% (Establecido por programación) se permitirá el acceso por reconocimiento facial.

Fuente: Elaboración propia desde script para reconocimiento facial

Figura 3-21. Captura imagen para base de datos

3.1.4. Caso 2

En la Figura 3-22, se realizó el registro de una usuaria con lentes, para verificar el correcto funcionamiento de la programación en cuestión, se puede observar que los puntos de referencia mencionados anteriormente están ubicados de manera adecuada a pesar de la existencia de un elemento adicional al rostro detectado.



Fuente: Elaboración propia desde script para reconocimiento facial

Figura 3-22. Captura imagen para registro con lentes ópticos puestos

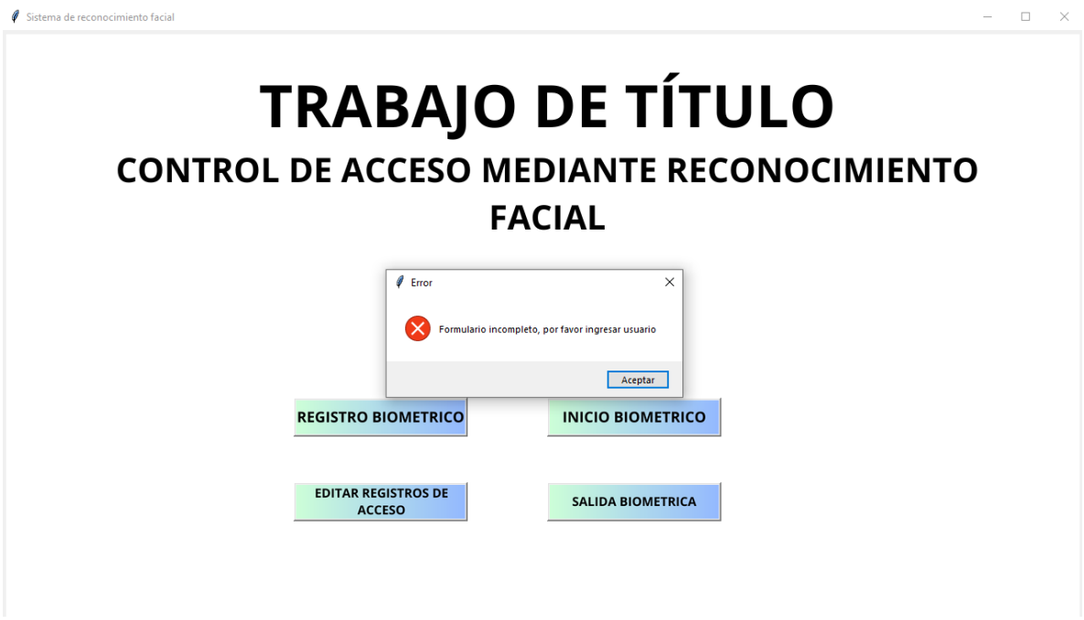
En la Figura 3-23, se puede apreciar la imagen que se capturó tras cumplir con los pasos establecidos en el programa para guardar está en la base de datos, para realizar una futura comparación cuando esta usuaria quiera realizar el inicio o salida biométrica.

Fuente: Elaboración propia desde script para reconocimiento facial

Figura 3-23. Captura imagen para base de datos con uso de lentes

3.1.5. Alarmas

En el siguiente grupo de Figuras se podrá observar los diferentes tipos de alarmas que presentará el sistema en determinados casos, por ejemplo, en la Figura 3-24 se puede apreciar el mensaje que se dará cuando se desee registrar un usuario sin antes escribir en la entrada de texto

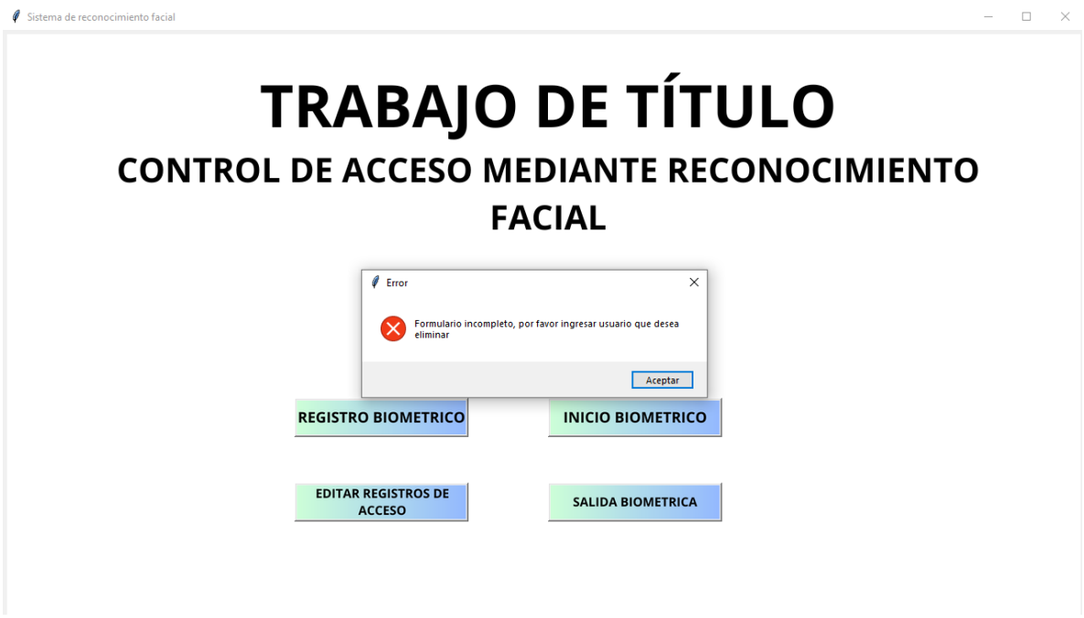


Fuente: Elaboración propia desde script para reconocimiento facial
Figura 3-24. Alarma “Formulario incompleto”

Mientras que en la Figura 3-25 se podrá apreciar el mensaje de error cuando el Nombre de usuario se encuentra registrado anteriormente.

Fuente: Elaboración propia desde script para reconocimiento facial
Figura 3-25. Alarma “Usuario registrado anteriormente”

Además, cuando se desee eliminar un usuario, pero no se ingresa el nombre en la entrada de texto, se indicará un mensaje de error, el cual se muestra en la Figura 3-26.



Fuente: Elaboración propia desde script para reconocimiento facial

Figura 3-26. Alarma “Formulario incompleto para eliminar usuario”

Por otra parte, la Figura 3-27 muestra el mensaje de error que aparecerá cuando se desee eliminar un usuario no registrado y cuando un usuario no registrado quiera ingresar o salir del lugar.

Fuente: Elaboración propia desde script para reconocimiento facial

Figura 3-27. Alarma “Usuario no registrado”

3.2. FUTURAS MEJORAS

En la siguiente sección, se sugerirán dos posibles mejoras al proyecto presentado, y del mismo modo, los beneficios que traerá la mejora propuesta.

3.2.1. Implementar interfaz web

Para aplicar una interfaz web en el proyecto es necesario seguir los pasos que se mencionarán a continuación:

- Diseñar la interfaz de usuario/administrador, si se desea ésta puede ser similar a la pantalla principal del programa, o usar los siguientes criterios de diseño.
  - Diseño Visual: Define estética de la interfaz, eligiendo colores, tipografía, iconos, entre otros elementos visuales.
  - Diseño de la experiencia de usuario: Crea un flujo de usuario que sea intuitivo y fácil de navegar
  - Diseño responsivo: Procurar que la interfaz funcione bien en diferentes dispositivos, como teléfonos móviles, tablets y computadores de escritorios.
- Desarrollar el backend para interactuar con la base de datos.
- Integrar el frontend y backend para así crear la aplicación web funcional.
- Testear la aplicación en diferentes navegadores y dispositivos para verificar el correcto funcionamiento, donde se realizarán pruebas:
  - Pruebas unitarias: Verificar que las partes individuales del código funcionen bien
  - Pruebas de integración: Verificar que los diferentes componentes funcionen correctamente de manera conjunta.
  - Pruebas de compatibilidad: Verificar que la interfaz funcione en diferentes navegadores y dispositivos.
- Implementar la aplicación en una nube para que este accesible en línea.

Los beneficios de aplicar la mejora propuesta, es que se tendrá acceso de manera remota en el caso que solo el administrador tenga la opción de ingresar al interfaz, por otra parte, si un usuario se puede agregar de manera remota, también adiciona beneficios, ya que no será necesario estar presencialmente para realizar el registro.

3.2.2. Implementar una adecuada placa de desarrollo

En el “ANEXO A: SCRIPTS RECONOCIMIENTO FACIAL”, se podrá verificar que en la primera parte del script se declararon diversas librerías y bibliotecas, las cuales son necesarias para llevar a cabo el objetivo del proyecto anteriormente descrito, del mismo modo, las antes mencionadas necesitan de otras librerías que no fueron declaras, pero si están instaladas y éstas funcionan de manera conjunta con las declaradas en el código inicialmente nombrado. Como es el caso de la biblioteca “open-cv-python”, la cual permite el procesamiento de videos, imágenes, entre varias funciones que permiten la visión de computadora. Además, a modo de complemento se utilizó la biblioteca “Face_recognition” para poder detectar, reconocer y manipular los rostros en una imagen, siendo construida sobre la biblioteca “Dlib”. Las tres bibliotecas mencionadas anteriormente requieren de un compilador llamado “Cmake” para gestionar las dependencias complejas de las bibliotecas antes mencionadas, de igual modo, estas requieren de requisitos de hardware para su correcto funcionamiento e instalación.

A continuación, en la Figura 3-2 se podrá observar los diferentes requerimientos de CPU, RAM de las diferentes bibliotecas para que puedas funcionar correctamente, además de las características de la raspberry pi 3 model b+ que se utilizara en el proyecto, como también la raspberry pi 5, la cual, como se verificara, tiene mejores características.

Tabla 3-2. Requerimiento bibliotecas y placa de desarrollo

Requisito/Biblioteca	C-make	Opencv	Dlib	Face_recognition	Rpi3b+	Rpi5
CPU(Tipo-nucleos)	i5-4	i5-4	i5-4	i5-4	BCM(1.4)-4	BCM(2.4)-4
RAM Min-Recom. (Gb)	4 u 8	4 u 8	4 u 8	4 u 8	1	4 u 8

Fuente: Elaboración propia

CONCLUSIONES

Aplicar de manera correcta los avances tecnológicos puede facilitar el día a día, pero es importante destacar el concepto antes mencionado, es decir, la aplicación correcta de la tecnología, ya que como se pudo observar en el capítulo 3, del presente documento, aplicar una ANN no destinada para el reconocimiento de imágenes dará resultado predicciones de manera deficiente, presentando subajustes que no pueden ser corregidos con las técnicas antes expuestas, y del mismo modo, no irá aprendiendo a pesar de aumentar el número de épocas de entrenamiento. Por otra parte, la aplicación de una ANN adecuada para el procesamiento de imágenes irá mejorando y siendo más eficiente cuando la cantidad de épocas va aumentando, la aplicación de optimizadores y regulaciones se obtendrá un modelo ajustado y óptimo para su funcionamiento. Siguiendo en la misma línea, la utilización de manera óptima de la tecnología traerá grandes beneficios, por ejemplo, usar la librería openCV, permitirá procesar el video que está siendo utilizado por la interfaz. Esto permite establecer condiciones para el registro e inicio y salida biométrica, y de esta manera, asegurar que el inicio se está realizando con la misma persona que realizó el registro, y no un tercero que está realizando el inicio biométrico con una imagen de otra persona, lo que es importante para la seguridad. Complementando lo anterior, para poder llevar a cabo el objetivo de este proyecto, es necesario contar con un controlador o un computador que cumpla con los requerimientos mínimos de CPU y RAM de las bibliotecas necesarias para ejecutar el programa descrito en el Anexo A.

De acuerdo con lo presentado en el capítulo anterior, se concluye que el modelo de redes neuronales óptimo para el desarrollo del proyecto antes expuesto corresponde al modelo CNNDO, en otras palabras, modelo de redes neuronales convolucionales con dropout, con la cual se consiguió un 85% de semejanza al detectar el rostro de un hombre, por otra parte, de acuerdo con las características de la raspberry pi 3 model b+ y los requerimientos de las bibliotecas descritas en el punto 3.2.2. se determina que esta placa de desarrollo no es la optima para poder ejecutar la programación, mientras que, de acuerdo con los datos de características técnicas de la rapsberry pi 5 el script si se podrá ejecutar.

BIBLIOGRAFÍA

IBM. ¿Qué son las redes neuronales? [En línea].

< https://www.ibm.com/es-es/topics/neural-networks > [Visitado: Marzo 2024]

IBM. ¿Qué es aprendizaje profundo? [En Línea].

<https://www.ibm.com/topics/deep-learning> [Visitado: Marzo 2024]

IBM. ¿Qué son las redes neuronales convolucionales? [En Línea].

<https://www.ibm.com/topics/convolutional-neural-networks> [Visitado: Marzo 2024]

DLIB. Dlib C++ Library [En Línea].

< http://dlib.net/> [Visitado: Septiembre 2024]

OpenCV. OpenCV – Open Computer Vision Library [En Línea].

< https://opencv.org/ > [Visitado: Septiembre 2024]

Face_recognition. Face-recognition [En Línea].

< https://pypi.org/project/face-recognition/ > [Visitado: Septiembre 2024]

Cmake. Cmake Python [En Línea].

< https://cmake.org/cmake/help/latest/module/FindPython.html > [Visitado: Septiembre 2024]

ANEXOS

ANEXO A: SCRIPTS RECONOCIMIENTO FACIAL

```
# librerias
import tkinter as tk
from tkinter import *
from tkinter import messagebox # para mensajes en la pantalla
from tkinter import ttk
import cv2
import face_recognition as fr
import numpy as np
import mediapipe as mp
import os
from PIL import Image, ImageTk
import imutils
import math
import time
from datetime import datetime
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler, FileSystemEvent
from gpiozero import LED



#funciones


#funcion perfil, la que puede ser para abrir la puerta


def HoraSalida():
    global pasos, conteo, usuarios, usuario, registroUsuario, Acceso, HoraAccesoSalida, captura, pantalla5
    pin_chapa = 11  # gpio 17
    chapa = LED(pin_chapa)

    # reseo variables
    conteo = 0
    pasos = 0

    # crear nueva pantalla pero es opcional, solo debe abrir la puerta
```

```
pantalla5 = Toplevel(pantalla)
pantalla5.title("Salida")
pantalla5.geometry("1280x720")

background5 = Label(pantalla5, image=imagenFondoSalida, text='Salida')
background5.place(x=0, y=0, relheight=1, relwidth=1)

# Archivo
UserFile = open(f"{usuarios}/{registroUsuario}.txt", 'r')
InfoUser = UserFile.read().split(',')
# Name = InfoUser[0]
usuario = InfoUser[0]
# Pass = InfoUser[2]
UserFile.close()

# verificar si usuario esta
if usuario in clases:
    # interface
    texto1 = Label(pantalla5, text=f"Hasta pronto {registroUsuario}")
    texto1.place(x=580, y=30)
    # escribir mensaje con hora de acceso
    ahora = datetime.now()
    texto2 = Label(pantalla5, text=f"Hora de salida  {ahora}")
    texto2.place(x=480, y=50)

    lblimage = Label(pantalla5)
    lblimage.place(x=390, y=200)

    # imagen
    imgUsuario = cv2.imread(f"{Crostros}/{registroUsuario}.png")
    imgUsuario = cv2.cvtColor(imgUsuario, cv2.COLOR_BGR2RGB)
    imgUsuario = Image.fromarray((imgUsuario))

    IMG = ImageTk.PhotoImage(image=imgUsuario)

    lblimage.configure(image=IMG)
```

```
        lblimage.image = IMG
        # activacion pin
        chapa.on()
        time.sleep(5) # se puede cambiar
        chapa.off()

        # guardar hora de acceso en el usuario Rh=RegistroHora
        Fecha = ahora.strftime('%Y:%m:%d')
        Hora = ahora.strftime('%H:%M:%S')
        Rh = open(f"{HoraAccesoSalida}/Salida.txt", 'a')
        Rh.writelines(f'\n{registroUsuario} Salida,  {Fecha}, {Hora}')
        Rh.close()
    # Boton atras
    BotonAtrasSal = tk.Button(pantalla5, text='Atras', command=BotonAtrasSalida)
    BotonAtrasSal.place(x=20, y=20)  # arreglar
    BotonAtrasSal.config(width=20, height=2)



# funcion para retroceder
def BotonAtrasSalida():
    global pantalla5
    pantalla5.destroy()



# ingreso
def perfil():
    global pasos, conteo, usuarios, usuario, registroUsuario, Acceso, HoraAccesoSalida,
pantalla4
    #pin_chapa = 11 #gpio 17
    #led = LED(pin_chapa)
    #reseo variables
    conteo = 0
    pasos = 0

    # crear nueva pantalla pero es opcional, solo debe abrir la puerta
    pantalla4 = Toplevel(pantalla)
    pantalla4.title("Inicio de sesion")
```

```
pantalla4.geometry("1280x720")



background4 = Label(pantalla4, image=imagenFondo4, text='Inicio')
background4.place(x=0, y=0, relheight=1, relwidth=1)

# Archivo
UserFile = open(f"{usuarios}/{registroUsuario}.txt", 'r')
InfoUser = UserFile.read().split(',')
#Name = InfoUser[0]
usuario = InfoUser[0]
#Pass = InfoUser[2]
UserFile.close()

# verificar si usuario esta

if usuario in clases:
    # interface
    texto1 = Label(pantalla4, text= f"Bienvenido {registroUsuario}")
    texto1.place(x=580, y=30)
    # escribir mensaje con hora de acceso
    ahora = datetime.now()
    texto2 = Label(pantalla4, text=f"Hora de acceso  {ahora}")
    texto2.place(x=480, y=50)
    # activacion pin

    #led.on()
    #print(led)
    #time.sleep(5) # se puede cambiar
    #led.off()



    lblimage = Label(pantalla4)
    lblimage.place(x=390, y=200)

    #imagen
```

```
        imgUsuario = cv2.imread(f"{Crostros}/{registroUsuario}.png")
        imgUsuario = cv2.cvtColor(imgUsuario, cv2.COLOR_BGR2RGB)
        imgUsuario = Image.fromarray((imgUsuario))

        IMG = ImageTk.PhotoImage(image = imgUsuario)

        lblimage.configure(image=IMG)
        lblimage.image = IMG

        # guardar hora de acceso en el usuario Rh=RegistroHora
        Fecha = ahora.strftime('%Y:%m:%d')
        Hora = ahora.strftime('%H:%M:%S')
        Rh = open(f"{HoraAccesoSalida}/Acceso.txt", 'a')
        Rh.writelines(f'\n{registroUsuario} Ingresa,  {Fecha}, {Hora}')
        Rh.close()

    # Boton atras
    BotonAtrasAcc = tk.Button(pantalla4, text='Atras', command=BotonAtrasAcceso)
    BotonAtrasAcc.place(x=20, y=20)  # arreglar
    BotonAtrasAcc.config(width=20, height=2)

    # funcion para retroceder
def BotonAtrasAcceso():
    global pantalla4
    pantalla4.destroy()



# funcion para codificar imagenes
def Code_Face(images):
    # lista
    listacod = []
    #iteramos
    for img in images:
        #color
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

```
        # codificar imagen
        cod = fr.face_encodings(img)[0]
        # guardar lista
        listacod.append(cod)

    return listacod

# cerrar ventana

def Close_Window2():
    global pasos, conteo

    conteo = 0
    pasos = 0
    pantalla3.destroy()

# funcion para inicio de sesion

def InicioBiometrico():
    global Crostros, cap, pantalla3, FaceCode, clases, images, captura, pasos, parpadeo,
conteo, pantalla, lblvideo, usuarios, registroUsuario, Acceso

    # verificar captura
    if captura is not None:
        ret, frame = captura.read()

        frameSave = frame.copy()

        # redimencionar frame
        #frame = imutils.resize(frame, width=1280)

        # RGB para ir cambiando
        frameRGB = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frameSave = frame.copy()

        # para que la imagen no se vea azul
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
    if ret == True:
        # interferencia face mesh
        res = FaceMesh.process(frameRGB)
        # entregar resultado en lista
        px = []
        py = []
        lista = []

        if res.multi_face_landmarks:
            # extraer detecciones de rostros
            for rostros in res.multi_face_landmarks:
                # dibujar
                mpDraw.draw_landmarks(frame, rostros,
FacemeshObject.FACEMESH_CONTOURS, configDraw, configDraw)

                # extraer informacion de los puntos de la malla
                for id, puntos in enumerate(rostros.landmark):

                    # informacion img
                    al, an, c = frame.shape
                    x, y = int(puntos.x * an), int(puntos.y * al)
                    px.append(x)
                    py.append(y)
                    lista.append([id, x, y])

                    # 468 puntos de la malla
                    if len(lista) == 468:
                        # ojo derecho
                        x1, y1 = lista[145][1:]
                        x2, y2 = lista[159][1:]
                        longitud1 = math.hypot(x2 - x1, y2 - y1)
                        #print(int(longitud1))

                        # ojo izqueirdo
                        x3, y3 = lista[374][1:]
                        x4, y4 = lista[386][1:]
```

```
longitud2 = math.hypot(x4 - x3, y4 - y3)
# print(int(longitud2))

# parietal derecho
x5, y5 = lista[139][1:]
# parietal izquierdo
x6, y6 = lista[368][1:]
# ceja derecha
x7, y7 = lista[70][1:]
# print(y7)
# print(x7)
# cv2.circle(frame, (x7, y7), 2, 225, 2, 2, )

# ceja izquierda
x8, y8 = lista[300][1:]

# detector de rostros
faces = detector.process(frameRGB)

if faces.detections is not None:
   for face in faces.detections:

      # recuerdo del rostro
      score = face.score
      score = score[0]
      bbox = face.location_data.relative_bounding_box

      # umbral
      if score > confThreshold:
         # pasar a pixeles
         xi, yi, anc, alt = bbox.xmin, bbox.ymin, bbox.width, bbox.height
         xi, yi, anc, alt = int(xi * an), int(yi * al), int(anc * an), int(alt * al)

         # offset x
         offsetan = (offsetx / 100) * anc
         xi = int(xi - int(offsetan / 2))
```

```
                    anc = int(anc + offsetan)
                    xf = xi + anc

                    # offset y
                    offsetal = (offsety / 100) * alt
                    yi = int(yi - int(
                        offsetal / 2))  # si no se divide en dos el margen inferior queda al
menton
                    alt = int(alt + offsetal)
                    yf = yi + alt

                    # error
                    if xi < 0: xi = 0
                    if yi < 0: yi = 0
                    if anc < 0: anc = 0
                    if alt < 0: alt = 0

                    # steps o pasos

                    if pasos == 0:
                        # dibujar
                        cv2.rectangle(frame, (xi, yi, anc, alt), (255, 0, 255), 2)

                        # imagen realizando verificacion biometrica
                        als0, ans0, c = (imagen_RVB.shape)  # poner mensaje inicial, por
ej "hola, verificaremos tu rostro"
                        frame[50:50 + als0, 50:50 + ans0] = imagen_RVB  # esta posicion
es arriba a la izquierda

                        # imagen paso 1
                        als1, ans1, c = imagen_paso1.shape  # poner mensaje inicial, por
ej "hola, verificaremos tu rostro"
                        frame[50:50 + als1, 1030:1030 + ans1] = imagen_paso1  # alto y
ancho

                        # imagen paso 2
                        als2, ans2, c = imagen_paso2.shape  # poner mensaje inicial, por
ej "hola, verificaremos tu rostro"
```

```
                frame[270:270 + als2,
                1030:1030 + ans2] = imagen_paso2  # alto y ancho, separados con
coma

                # mirada centrada

                if x7 > x5 and x8 < x6:
                    # ingrasar imagen ok
                    alok, anok, c = imagen_ok.shape  # poner mensaje inicial, por ej
"hola, verificaremos tu rostro"
                    frame[165:165 + alok, 1105:1105 + anok] = imagen_ok

                    # conteo parpadeo
                    if longitud1 <= 16 and longitud2 <= 16 and parpadeo == False:
                        conteo = conteo + 1
                        parpadeo = True

                    elif longitud1 > 16 and longitud2 > 16 and parpadeo == True:
                        parpadeo = False

                    # COMANDO PARA PONER TEXTO
                    cv2.putText(frame, f'Parpadeos: {int(conteo)}', (1070, 375),
                                cv2.FONT_HERSHEY_DUPLEX, 0.5, (0, 0, 0), 1)

                    # condicion de conteo
                    if conteo >= 3:
                        # ingrasar imagen ok
                        alok, anok, c = imagen_ok.shape  # poner mensaje inicial, por
ej "hola, verificaremos tu rostro"
                        frame[385:385 + alok, 1105:1105 + anok] = imagen_ok  # 385
hasta 385

                        if longitud1 > 16 and longitud2 > 16:



                            # paso 1


                            pasos = 1
```

```
                else:
                    conteo = 0

            if pasos == 1:
                # dibujar
                cv2.rectangle(frame, (xi, yi, anc, alt), (0, 255, 0), 2)
                # imagen realizando verificacion biometrica
                als0, ans0, c = (imagen_RVB.shape)  # poner mensaje inicial, por ej "hola, verificaremos tu rostro"
                frame[50:50 + als0, 50:50 + ans0] = imagen_RVB  # esta posicion es arriba a la izquierda
                # imagen registro biometrico finalizado
                #alich, anlich, c = imagen_RBF.shape
                #frame[50:50 + allich, 50:50 + anlich] = imagen_RBF


                # encontrando los rostros
                facess = fr.face_locations(frameRGB)
                facescod = fr.face_encodings(frameRGB, facess)

                # iteramos
                for facecod, facesloc in zip(facescod, facess):

                    # emparejamiento o comparacion
                    Match = fr.compare_faces(FaceCode, facecod)

                    # extraer similitud

                    similitud = fr.face_distance(FaceCode, facecod)

                    # minimo error
                    min = np.argmin(similitud)

                    if int(min) == 1:
                        print("usuario no registrado")
                        messagebox.showerror("Error", "Usuario no se encuentra registrado")
```

```
                                captura.release()
                                close = pantalla3.protocol("WM_DELETE_WINDOW",
Close_Window2)
                                pantalla3.destroy()
                            if Match[min]:
                                #usuario
                                registroUsuario = clases[min].upper()
                                pantalla3.destroy() # opcional


                                perfil()


                        # cerrar ventana
                        close = pantalla3.protocol("WM_DELETE_WINDOW", Close_Window2)


                    # circulos para verificar puntos
                    # cv2.circle(frame, (x1,y1), 2, 225,2,2, )
                    # cv2.circle(frame, (x2, y2), 2, 225, 2, 2, )
                    # cv2.circle(frame, (x3, y3), 2, 225, 2, 2, )
                    # cv2.circle(frame, (x4, y4), 2, 225, 2, 2, )
                    # cv2.circle(frame, (x5,y5), 2, 225,2,2, )
                    # cv2.circle(frame, (x6, y6), 2, 225, 2, 2, )
                    # cv2.circle(frame, (x8, y8), 2, 225, 2, 2, )
        # redimensionar imagenes

            frame = imutils.resize(frame, width=1280)


        # convertir video
            im = Image.fromarray(frame)
            img = ImageTk.PhotoImage(image=im)


        # mostrar video
            lblvideo.configure(image=img)
            lblvideo.image = img
            lblvideo.after(10, InicioBiometrico)


    else:
        captura.release()
```

```
#Funcion para cerrar ventana

def Close_Window():
    global pasos, conteo

    conteo = 0
    pasos = 0
    pantalla2.destroy()



# Registro biométrico
def RegistroBiometrico():
    global pantalla2, conteo, parpadeo, img_info, pasos, captura, lblvideo, pantalla,
registroUsuario
    #boton hacia atras

    #verificar captura
    if captura is not None:

        ret, frame = captura.read()

        frameSave = frame.copy()

        #redimencionar frame
        frame = imutils.resize(frame, width=1280)

        #RGB para ir cambiando
        frameRGB = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frameSave = frame.copy()

        #para que la imagen no se vea azul
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        if ret == True:
            #interferencia face mesh
```

```
        res = FaceMesh.process(frameRGB)
        #entregar resultado en lista
        px = []
        py = []
        lista = []

        if res.multi_face_landmarks:
           #extraer detecciones de rostros
           for rostros in res.multi_face_landmarks:
              #dibujar
              mpDraw.draw_landmarks(frame, rostros,
FacemeshObject.FACEMESH_CONTOURS, configDraw, configDraw)

              # extraer informacion de los puntos de la malla
              for id, puntos in enumerate(rostros.landmark):

                 #informacion img
                 al, an, c = frame.shape
                 x, y = int(puntos.x * an), int(puntos.y * al)
                 px.append(x)
                 py.append(y)
                 lista.append([id, x, y])

                 # 468 puntos de la malla
                 if len(lista) == 468:

                    #ojo derecho
                    x1, y1 = lista[145][1:]
                    x2, y2 = lista[159][1:]
                    longitud1 = math.hypot(x2-x1, y2-y1)
                    #print(int(longitud1))
                    cv2.circle(frame, (x1,y1), 2, 225,2,2, )
                    cv2.circle(frame, (x2, y2), 2, 225, 2, 2, )

                    #ojo izqueirdo
                    x3, y3 = lista[374][1:]
                    x4, y4 = lista[386][1:]
```

```
longitud2 = math.hypot(x4 - x3, y4 - y3)
#print(int(longitud2))
cv2.circle(frame, (x3, y3), 2, 225, 2, 2, )
cv2.circle(frame, (x4, y4), 2, 225, 2, 2, )

# parietal derecho
x5, y5 = lista[139][1:]
#parietal izquierdo
x6, y6 = lista[368][1:]
#ceja derecha
x7, y7 = lista[70][1:]
#print(y7)
#print(x7)
cv2.circle(frame, (x7, y7), 2, 225, 2, 2, )
cv2.circle(frame, (x5, y5), 2, 225, 2, 2, )
cv2.circle(frame, (x6, y6), 2, 225, 2, 2, )

#ceja izquierda
x8, y8 = lista[300][1:]
cv2.circle(frame, (x8, y8), 2, 225, 2, 2, )



#detector de rostros
faces = detector.process(frameRGB)

if faces.detections is not None:
   for face in faces.detections:

      # recuerdo del rostro
      score = face.score
      score = score[0]
      bbox = face.location_data.relative_bounding_box



      #umbral
      if score > confThreshold:
```

```
                    # pasar a pixeles
                    xi, yi, anc, alt = bbox.xmin, bbox.ymin, bbox.width, bbox.height
                    xi, yi, anc, alt = int(xi * an), int(yi * al), int(anc * an), int(alt * al)

                    # offset x
                    offsetan = (offsetx / 100) * anc
                    xi = int(xi - int(offsetan/2))
                    anc = int(anc + offsetan)
                    xf = xi + anc

                    # offset y
                    offsetal =  (offsety / 100) * alt
                    yi = int(yi - int(offsetal/2))  # si no se divide en dos el margen inferior
queda al menton
                    alt = int(alt + offsetal)
                    yf = yi + alt

                    #error
                    if xi < 0: xi = 0
                    if yi < 0: yi = 0
                    if anc < 0: anc = 0
                    if alt < 0: alt = 0

                    #steps o pasos

                    if pasos == 0:
                       #dibujar
                       cv2.rectangle(frame, (xi, yi, anc, alt), (255, 0, 255), 2)

                       #imagen paso 0
                       als0, ans0, c = (imagen_RRB.shape) #poner mensaje inicial, por ej
"hola, verificaremos tu rostro"
                       frame[50:50 + als0, 50:50 + ans0] = imagen_RRB # esta posicion
es arriba a la izquierda

                       # imagen paso 1
                       als1, ans1, c = imagen_paso1.shape  # poner mensaje inicial, por
ej "hola, verificaremos tu rostro"
```

```
                frame[50:50 + als1, 1030:1030 + ans1] = imagen_paso1 #alto y
ancho


                # imagen paso 2
                als2, ans2, c = imagen_paso2.shape  # poner mensaje inicial, por
ej "hola, verificaremos tu rostro"
                frame[270:270 + als2, 1030:1030 + ans2] = imagen_paso2  # alto y
ancho, separados con coma



                # mirada centrada


                if x7 > x5 and x8 < x6:
                    # ingrasar imagen ok
                    alok, anok, c = imagen_ok.shape  # poner mensaje inicial, por ej
"hola, verificaremos tu rostro"
                    frame[165:165 + alok, 1105:1105 + anok] = imagen_ok


                    # conteo parpadeo
                    if longitud1 <=16 and longitud2 <= 16 and parpadeo == False:
                        conteo = conteo + 1
                        parpadeo = True


                    elif longitud1 > 16 and longitud2 > 16 and parpadeo == True:
                        parpadeo = False


                    #COMANDO PARA PONER TEXTO
                    cv2.putText(frame, f'Parpadeos: {int(conteo)}', (1070,375),
cv2.FONT_HERSHEY_DUPLEX, 0.5, (0,0,0),1 )


                    # condicion de conteo
                    if conteo >= 3:
                        # ingrasar imagen ok
                        alok, anok, c = imagen_ok.shape  # poner mensaje inicial, por
ej "hola, verificaremos tu rostro"
                        frame[385:385 + alok, 1105:1105 + anok] = imagen_ok # 385
hasta 385


                        if longitud1 > 16 and longitud2 > 16:
```

```
                    # cortar
                    cut = frameSave[yi:yf, xi:xf]

                    #guardar rostro
                    cv2.imwrite(f"{Crostros}/{registroUsuario}.png", cut)

                    # paso 1

                    pasos = 1
                else:
                    conteo = 0

            if pasos == 1:
                # dibujar
                cv2.rectangle(frame, (xi, yi, anc, alt), (0, 255, 0), 2)
                # imagen registro biometrico finalizado
                allich, anlich, c = imagen_RBF.shape
                frame[50:50 + allich, 50:50 + anlich] = imagen_RBF
                time.sleep(2)
                pantalla2.destroy() # borrar

        # cerrar ventana
        close = pantalla2.protocol("WM_DELETE_WINDOW",Close_Window)


            #circulos para verificar puntos


    #cv2.circle(frame, (x5,y5), 2, 225,2,2, )
    #cv2.circle(frame, (x6, y6), 2, 225, 2, 2, )
    #cv2.circle(frame, (x8, y8), 2, 225, 2, 2, )


#convertir video
im = Image.fromarray(frame)
img = ImageTk.PhotoImage(image=im)
```

```
        #mostrar video
        lblvideo.configure(image = img)
        lblvideo.image = img
        lblvideo.after(10, RegistroBiometrico)

    else:
        captura.release()



#inicio sesion biometrico
def Inicio():
    global Crostros, cap, lblvideo, pantalla3, FaceCode, clases, images, captura,
registroUsuario

    # base de datos de rostros
    images = []
    clases = []
    lista = os.listdir(Crostros)

    # leer rostros guardados
    for lis in lista:
        # leer imagenes en base de datos
        imgdb = cv2.imread(f"{Crostros}/{lis}")
        #guardar imagen de base de datos
        images.append(imgdb)
        # guardamos nombre de imagenes
        clases.append(os.path.splitext(lis)[0])

    # codigo de rostros
    FaceCode = Code_Face(images)

    # ventana 3
    pantalla3 = Toplevel(pantalla)
    pantalla3.title("INICIO BIOMETRICO")
    pantalla3.geometry("1280x720")
```

```
    # labelVideo
    lblvideo = Label(pantalla3)
    lblvideo.place(x=0, y=0)

    # captura video
    captura = cv2.VideoCapture(0, cv2.CAP_DSHOW)
    captura.set(3, 1280)
    captura.set(4, 720)
    InicioBiometrico()



#Registro facial
def Registro():
    global registroUsuario, ingresoUsuario, captura, lblvideo, pantalla2
    # extraer nombre del usuario para guardarlo
    registroUsuario = ingresoUsuario.get()
    #formulario incompleto
    if len(registroUsuario) == 0:
        print("Formulario incompleto")
        messagebox.showerror("Error", "Formulario incompleto, por favor ingresar usuario")

    else:
        #formulario completo
        ListaUsuario = os.listdir(RevUsuarios)
        NombreUsuario = []

        #verificar usuario
        for lis in ListaUsuario:
            usuario = lis
            usuario = usuario.split('.')
            NombreUsuario.append(usuario[0])

        if registroUsuario in NombreUsuario:
            print('USUARIO REGISTRADO ANTERIORMENTE')
            messagebox.showerror("Error", "Nombre de usuario registrado anteriormente, por
favor intente con otro nombre")
        else:
```

```
        info.append(registroUsuario)
        #guardar informacion
        f = open(f"{usuarios}/{registroUsuario}.txt", 'w')
        f.write(registroUsuario)
        f.close()
        # borrar celdas
        ingresoUsuario.delete(0, END)

        #Pantalla registro biometrico
        pantalla2 = Toplevel(pantalla)
        pantalla2.title("Registro biometrico")
        pantalla2.geometry("1280x720")

        #labelVideo
        lblvideo = Label(pantalla2)
        lblvideo.place(x=0, y=0)

        #captura video
        captura = cv2.VideoCapture(0, cv2.CAP_DSHOW)
        captura.set(3, 1280)
        captura.set(4, 720)
        RegistroBiometrico()


# clase para actualizar los cambios en la carpeta de usuarios
class CarpetaControlEvento(FileSystemEventHandler):
    def __init__(self, listado):
        super().__init__()
        self.listado = listado

    def on_any_event(self, event):
        #ante cualquier cambio llamara a la funcion que muestra el contenido
        ContenidoCarpeta(RevUsuarios)

# funcion editar usuario
def Edicion():
```

```
    global ingresoUsuarioeliminar, pantallaR
    #print("Hola")
    #Toplevel(pantalla)
    pantallaR = tk.Toplevel(pantalla)
    pantallaR.title('Edición de registros')
    pantallaR.geometry('1280x720')
    backgroundR = tk.Label(pantallaR, image=imagenEdReg, text='Edición Registros')
    backgroundR.place(x=0, y=0, relheight=1, relwidth=1)
    # boton para eliminar usuarios
    BotonEliminar = tk.Button(pantallaR, text='Eliminar usuario',
command=Eliminarusuario)
    #Button.pack_configure()
    BotonEliminar.place(x=560, y=395) # arreglar
    BotonEliminar.config(width=20, height=2)
    #BotonEliminar.pack()
    # Boton atras
    BotonAtrasE = tk.Button(pantallaR, text='Atras', command=BotonAtras)
    BotonAtrasE.place(x=20, y=20) # arreglar
    BotonAtrasE.config(width=20, height=2)

    #ingreso nombre del usuario para borrar
    ingresoUsuarioeliminar = Entry(pantallaR)
    ingresoUsuarioeliminar.place(x=580, y=370)

    # funcion para mostrar contenido
    global listado
    listado = tk.Listbox(pantallaR)
    listado.place(x=10, y=80, width=200, height=400)
    ContenidoCarpeta(RevUsuarios)

    # observador para la carpeta donde borraran los usuarios
    observador = Observer()
    observador.schedule(CarpetaControlEvento(listado), path=f"{RevUsuarios}",
recursive=False)
    observador.start()

#mostrar contenido carpeta
```

```
def ContenidoCarpeta(RevUsuarios):
    global listado
    #eliminar datos de listado
    listado.delete(0, tk.END)
    #guardar en contenido la lista de usuarios
    Contenido = os.listdir(RevUsuarios)
    #agregar cada nombre de usuario a archivo
    for archivo in Contenido:
        listado.insert(END, archivo)


# funcion para retroceder
def BotonAtras():
    global pantallaR
    pantallaR.destroy()


# funcion para eliminar usuario
def Eliminarusuario():
    global borrar, NombreUsuarioelimiar, ingresoUsuarioeliminar, listado


    NombreUsuarioelimiar = ingresoUsuarioeliminar.get()
    #print(NombreUsuarioelimiar)
    ListaUsuarios = os.listdir(RevUsuarios)
    #print(ListaUsuarios)
    NombreUsuarioE = []


    if len(NombreUsuarioelimiar) == 0:
        print("Formulario incompleto")
        messagebox.showerror("Error", "Formulario incompleto, por favor ingresar usuario
que desea eliminar")
        pantallaR.destroy()
    else:


        # verificar usuario
        for lis in ListaUsuarios:
            usuario = lis
            usuario = usuario.split('.')
            NombreUsuarioE.append(usuario[0])
```

```
    #borrar usuario
        if NombreUsuarioelimiar in NombreUsuarioE:
            print("Usuario en listado")
            os.remove(f"{usuarios}/{NombreUsuarioelimiar}.txt")
            os.remove(f"{Crostros}/{NombreUsuarioelimiar}.png")
            ingresoUsuarioeliminar.delete(0, END)
            #messagebox.showinfo("Atención", "Usuario eliminado satisfactoriamente")
            #pantallaR.destroy()


        else:
            print("usuario no esta en el listado")
            messagebox.showerror("Error", "Usuario no se encuentra registrado")
            pantallaR.destroy()




# funcion para marcar salida biometrica
def SalidaBiometrica():
    #print ("Hora de salida")
    global Crostros, cap, pantalla6, FaceCode, clases, images, captura, pasos, parpadeo,
conteo, pantalla, lblvideo, usuarios, registroUsuario, Acceso


    # verificar captura
    if captura is not None:
        ret, frame = captura.read()


        frameSave = frame.copy()


        # redimencionar frame
        #frame = imutils.resize(frame, width=1280)


        # RGB para ir cambiando
        frameRGB = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frameSave = frame.copy()


        # para que la imagen no se vea azul
```

```
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    if ret == True:
        # interferencia face mesh
        res = FaceMesh.process(frameRGB)
        # entregar resultado en lista
        px = []
        py = []
        lista = []

        if res.multi_face_landmarks:
            # extraer detecciones de rostros
            for rostros in res.multi_face_landmarks:
                # dibujar
                mpDraw.draw_landmarks(frame, rostros,
FacemeshObject.FACEMESH_CONTOURS, configDraw, configDraw)

                # extraer informacion de los puntos de la malla
                for id, puntos in enumerate(rostros.landmark):

                    # informacion img
                    al, an, c = frame.shape
                    x, y = int(puntos.x * an), int(puntos.y * al)
                    px.append(x)
                    py.append(y)
                    lista.append([id, x, y])

                    # 468 puntos de la malla
                    if len(lista) == 468:
                        # ojo derecho
                        x1, y1 = lista[145][1:]
                        x2, y2 = lista[159][1:]
                        longitud1 = math.hypot(x2 - x1, y2 - y1)
                        # print(int(longitud1))

                        # ojo izqueirdo
                        x3, y3 = lista[374][1:]
```

```
x4, y4 = lista[386][1:]
longitud2 = math.hypot(x4 - x3, y4 - y3)
# print(int(longitud2))

# parietal derecho
x5, y5 = lista[139][1:]
# parietal izquierdo
x6, y6 = lista[368][1:]
# ceja derecha
x7, y7 = lista[70][1:]
# print(y7)
# print(x7)
# cv2.circle(frame, (x7, y7), 2, 225, 2, 2, )

# ceja izquierda
x8, y8 = lista[300][1:]

# detector de rostros
faces = detector.process(frameRGB)

if faces.detections is not None:
   for face in faces.detections:

      # recuerdo del rostro
      score = face.score
      score = score[0]
      bbox = face.location_data.relative_bounding_box

      # umbral
      if score > confThreshold:
         # pasar a pixeles
         xi, yi, anc, alt = bbox.xmin, bbox.ymin, bbox.width, bbox.height
         xi, yi, anc, alt = int(xi * an), int(yi * al), int(anc * an), int(alt * al)

         # offset x
         offsetan = (offsetx / 100) * anc
```

```
                    xi = int(xi - int(offsetan / 2))
                    anc = int(anc + offsetan)
                    xf = xi + anc

                    # offset y
                    offsetal = (offsety / 100) * alt
                    yi = int(yi - int(
                        offsetal / 2))  # si no se divide en dos el margen inferior queda al
menton
                    alt = int(alt + offsetal)
                    yf = yi + alt

                    # error
                    if xi < 0: xi = 0
                    if yi < 0: yi = 0
                    if anc < 0: anc = 0
                    if alt < 0: alt = 0

                    # steps o pasos

                    if pasos == 0:
                        # dibujar
                        cv2.rectangle(frame, (xi, yi, anc, alt), (255, 0, 255), 2)

                        # imagen realizando verificacion biometrica
                        als0, ans0, c = (imagen_RVB.shape)  # poner mensaje inicial, por
ej "hola, verificaremos tu rostro"
                        frame[50:50 + als0, 50:50 + ans0] = imagen_RVB  # esta posicion
es arriba a la izquierda

                        # imagen paso 1
                        als1, ans1, c = imagen_paso1.shape  # poner mensaje inicial, por
ej "hola, verificaremos tu rostro"
                        frame[50:50 + als1, 1030:1030 + ans1] = imagen_paso1  # alto y
ancho

                        # imagen paso 2
```

```
                als2, ans2, c = imagen_paso2.shape  # poner mensaje inicial, por ej "hola, verificaremos tu rostro"
                frame[270:270 + als2,
                1030:1030 + ans2] = imagen_paso2  # alto y ancho, separados con coma

                # mirada centrada

                if x7 > x5 and x8 < x6:
                    # ingrasar imagen ok
                    alok, anok, c = imagen_ok.shape  # poner mensaje inicial, por ej "hola, verificaremos tu rostro"
                    frame[165:165 + alok, 1105:1105 + anok] = imagen_ok

                    # conteo parpadeo
                    if longitud1 <= 16 and longitud2 <= 16 and parpadeo == False:
                        conteo = conteo + 1
                        parpadeo = True

                    elif longitud1 > 16 and longitud2 > 16 and parpadeo == True:
                        parpadeo = False

                    # COMANDO PARA PONER TEXTO
                    cv2.putText(frame, f'Parpadeos: {int(conteo)}', (1070, 375), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0, 0, 0), 1)

                    # condicion de conteo
                    if conteo >= 3:
                        # ingrasar imagen ok
                        alok, anok, c = imagen_ok.shape  # poner mensaje inicial, por ej "hola, verificaremos tu rostro"
                        frame[385:385 + alok, 1105:1105 + anok] = imagen_ok  # 385 hasta 385

                        if longitud1 > 16 and longitud2 > 16:


                            # paso 1
```

```
                    pasos = 1
                else:
                    conteo = 0

            if pasos == 1:
                # dibujar
                cv2.rectangle(frame, (xi, yi, anc, alt), (0, 255, 0), 2)
                # imagen realizando verificacion biometrica
                als0, ans0, c = (imagen_RVB.shape)  # poner mensaje inicial, por ej "hola, verificaremos tu rostro"
                frame[50:50 + als0, 50:50 + ans0] = imagen_RVB  # esta posicion es arriba a la izquierda
                # imagen registro biometrico finalizado
                #alich, anlich, c = imagen_RBF.shape
                #frame[50:50 + allich, 50:50 + anlich] = imagen_RBF


                # encontrando los rostros
                facess = fr.face_locations(frameRGB)
                facescod = fr.face_encodings(frameRGB, facess)

                # iteramos
                for facecod, facesloc in zip(facescod, facess):

                    # emparejamiento o comparacion
                    Match = fr.compare_faces(FaceCode, facecod)
                    print(Match)
                    # extraer similitud
                    similitud = fr.face_distance(FaceCode, facecod)
                    print(similitud)
                    # minimo error
                    min = np.argmin(similitud)
                    print(min)
                    if int(min) == 1:
                        print("usuario no registrado")
```

```
                                messagebox.showerror("Error", "Usuario no se encuentra
registrado")
                                captura.release()
                                #close = pantalla6.protocol("WM_DELETE_WINDOW",
Close_Window3)
                                pantalla6.destroy()
                            if Match[min]:
                                #usuario
                                print(Match[min])
                                registroUsuario = clases[min].upper()
                                pantalla6.destroy() # opcional
                                HoraSalida()


                    # cerrar ventana
                    close = pantalla6.protocol("WM_DELETE_WINDOW", Close_Window3)


                # circulos para verificar puntos
                # cv2.circle(frame, (x1,y1), 2, 225,2,2, )
                # cv2.circle(frame, (x2, y2), 2, 225, 2, 2, )
                # cv2.circle(frame, (x3, y3), 2, 225, 2, 2, )
                # cv2.circle(frame, (x4, y4), 2, 225, 2, 2, )
                # cv2.circle(frame, (x5,y5), 2, 225,2,2, )
                # cv2.circle(frame, (x6, y6), 2, 225, 2, 2, )
                # cv2.circle(frame, (x8, y8), 2, 225, 2, 2, )
    # redimensionar imagenes

        frame = imutils.resize(frame, width=1280)


    # convertir video
        im = Image.fromarray(frame)
        img = ImageTk.PhotoImage(image=im)


    # mostrar video
        lblvideo.configure(image=img)
        lblvideo.image = img
        lblvideo.after(10, SalidaBiometrica)
```

```
    else:
        captura.release()



def Close_Window3():
    global pasos, conteo


    conteo = 0
    pasos = 0
    pantalla6.destroy()


def Salida():
    global Crostros, cap, lblvideo, pantalla6, FaceCode, clases, images, captura,
registroUsuario


    # base de datos de rostros
    images = []
    clases = []
    lista = os.listdir(Crostros)


    # leer rostros guardados
    for lis in lista:
        # leer imagenes en base de datos
        imgdb = cv2.imread(f"{Crostros}/{lis}")
        # guardar imagen de base de datos
        images.append(imgdb)
        # guardamos nombre de imagenes
        clases.append(os.path.splitext(lis)[0])


    # codigo de rostros
    FaceCode = Code_Face(images)


    # ventana 3
    pantalla6 = Toplevel(pantalla)
    pantalla6.title("SALIDA BIOMETRICA")
    pantalla6.geometry("1280x720")
```

```
    # labelVideo
    lblvideo = Label(pantalla6)
    lblvideo.place(x=0, y=0)

    # captura video
    captura = cv2.VideoCapture(0, cv2.CAP_DSHOW)
    captura.set(3, 1280)
    captura.set(4, 720)
    SalidaBiometrica()

#carpetas
usuarios = 'C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Basedatos/Usuarios'
RevUsuarios= 'C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Basedatos/Usuarios/'
Crostros = 'C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Basedatos/Rostros'
RevRostros = 'C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Basedatos/Rostros/'
HoraAccesoSalida = 'C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Basedatos/Hora acceso y salida'

# lectura imagenes
imagen_FI = cv2.imread("C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/FI.png")
imagen_SB = cv2.imread("C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/SB.png")
imagen_ERA = cv2.imread("C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/ERA.png")
imagen_UNR = cv2.imread("C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/UNR.png")
imagen_RVB = cv2.imread("C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/RVB.png")
imagen_RBF = cv2.imread("C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/RBF.png")
imagen_RRB = cv2.imread("C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/RRB.png")
imagen_info = cv2.imread("C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/info.png")
imagen_ok = cv2.imread("C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/ok.png")
```

```
imagen_paso1 = cv2.imread("C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/Paso1.png")

imagen_paso2 = cv2.imread("C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/Paso2.png")

imagen_pantregistro = cv2.imread("C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/PantRegistro.png")

#imagen_iniciobio = cv2.imread("C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/InicioBio.png")


#varibles

parpadeo = False

conteo = 0

muestra = 0

pasos = 0

FRAME = 0


#margenes

offsety = 30

offsetx = 20


# umbral presicion de deteccion

confThreshold = 0.5


# herramienta dibujo de la malla facial

mpDraw = mp.solutions.drawing_utils

configDraw = mpDraw.DrawingSpec(thickness=1, circle_radius=1)


#configuracion malla facial

FacemeshObject = mp.solutions.face_mesh

FaceMesh = FacemeshObject.FaceMesh(max_num_faces=1)


#declarar detector de rostros

FaceObject = mp.solutions.face_detection

detector = FaceObject.FaceDetection(min_detection_confidence=0.5, model_selection=1)


# lista de informacion
```

```
info = []


# interface ventana principal


pantalla = Tk()

pantalla.title('Sistema de reconocimiento facial')

pantalla.geometry('1280x720')


#imagen de fondo

imagenFondo = PhotoImage(file="C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y
titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de
titulo/Plantillas/Pantalla1.png")

background = Label(pantalla, image = imagenFondo, text= 'Inicio')

background.place(x=0, y=0, relheight=1, relwidth=1)


# imagen fondo inicio


# imagen de fondo

imagenFondo4 = PhotoImage(file="C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y
titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de
titulo/Plantillas/FondoBio.png")

#imagen fondo de salida

imagenFondoSalida =
PhotoImage(file="C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT
Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/FondoSalida.png")

# imagen de fondo

imagenEdReg = PhotoImage(file="C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y
titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/EdReg.png")


# ingreso usuario

ingresoUsuario = Entry(pantalla)

ingresoUsuario.place(x=580, y=370)


#boton registro


imagenRegistro = PhotoImage(file="C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y
titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/RegBio.png")

BotonReg = Button(pantalla, text='Registro', image=imagenRegistro, height=40,
width=200, command=Registro)
```

```
BotonReg.place(x=345, y=435)


# boton inicio


imagenInicio = PhotoImage(file="C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/InicioBio.png")

BotonInicio = Button(pantalla, text='InicioSesion', image=imagenInicio, height=40, width=200, command=Inicio)

BotonInicio.place(x=645, y=435)


# Boton edicion de datos de usuarios registrados

imagenEdicion = PhotoImage(file="C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/ERA.png")

BotonEdicion = Button(pantalla, text='InicioEdicion', image=imagenEdicion, height=40, width=200, command=Edicion)

BotonEdicion.place(x=345, y=535)


# Bonton para salida biometrica

imagenSalida = PhotoImage(file="C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Reconocimiento facial/Trabajo de titulo/Plantillas/SB.png")

BotonSalida = Button(pantalla, text='InicioSalida', image=imagenSalida, height=40, width=200, command=Salida)

BotonSalida.place(x=645, y=535)



#e = PANTALLA(pantalla)

#e.pack(fill=BOTH, expand=YES)


pantalla.mainloop()
```

ANEXO B: SCRIPTS ENTRENAMIENTO REDES NEURONALES

```
import os

import tensorflow as tf

import cv2

import matplotlib.pyplot as plt

import numpy as np

from tensorflow.keras.callbacks import TensorBoard

from tensorflow.keras.preprocessing.image import ImageDataGenerator # generador de imagenes

# direccion imagenes

entrenamiento = 'C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Red neuronal convolucional/RedNeuronalConvolucional/BaseDatos/Entrenamiento'

validacion = 'C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Red neuronal convolucional/RedNeuronalConvolucional/BaseDatos/Validacion'

listaEntrenamiento = os.listdir(entrenamiento)

listavalidacion = os.listdir(validacion)

# parecemetros de fotos entre otros

ancho, alto = 200, 200

# Listas entrenamiento

etiquetas = []

fotos = []

datos_entenamiento = []  #train es entrenamiento

contador = 0 # esta como con

# lista validacion

etiquetas2 = []

fotos2 = []

datos_validacion = []

contador2 = 0

# extraer en una lista las fotos para entrenamiento

for nameDir in listaEntrenamiento: # se extraen las fotos de entrenamiento y se almacenan en una lista y le asignan etiquetas

    nombre = entrenamiento + '/' + nameDir # extrae las fotos de la direccion de entrenamiento

    for fileName in os.listdir(nombre):

        etiquetas.append(contador) # contador aparece como con; valor de la etiqueta segun el contador, 0 para la 1era y 1 para la 2da

        img = cv2.imread(nombre + '/' + fileName, 0) # se lee la imagen
```

```
        img = cv2.resize(img,(ancho, alto), interpolation=cv2.INTER_CUBIC) # se
redimencionan las imagenes a 200x200
        img = img.reshape(ancho, alto, 1) # dejamos solo 1 canal
        datos_entenamiento.append([img, contador])
        fotos.append(img) # añadimos las imagenes a EDG
    contador = contador + 1
# extraer fotos para validacion
for nameDir2 in listavalidacion: # se extraeran las fotos para almacenar en una lista
    nombre2 = validacion + '/' + nameDir2 # se leen las fotos
    for fileName2 in os.listdir(nombre2): #se asignaran las etiquetas a cada foto
        etiquetas2.append(contador2) # valor de las etiquetas, 0 para la 1era, 1 para la 2da
        img2 = cv2.imread(nombre2 + '/' + fileName2, 0)
        img2 = cv2.resize(img2, (ancho, alto), interpolation=cv2.INTER_CUBIC)
        img2 = img2.reshape(ancho, alto, 1)
        datos_validacion.append([img2, contador2])
        fotos2.append(img2) # añadimos las imagenes en EDG
    contador2 = contador2 + 1
# normalizar las imagenes en escala de grises, para tener valores de 0 a 1
fotos = np.array(fotos).astype(float) / 255
print(fotos.shape)
fotos2 = np.array(fotos2).astype(float) / 255
print(fotos2.shape)
# se pasa la lista a Array
etiquetas = np.array(etiquetas)
etiquetas2 = np.array(etiquetas2)
# para evitar el sobreajuste se cambiara la forma de las imagenes haciendolas rotar
usando el generador de imagenes de tensorflow


imgTrainGen = ImageDataGenerator(
    rotation_range = 50, # rotacion aleatoria de las imagenes
    width_shift_range = 0.3, #mover las imagenes a los lados
    height_shift_range = 0.3, #mover la imagen hacia arriba
    shear_range = 15, # inclinar la imagen
    zoom_range = [0.5, 1.5], # hacemos zoom a la imagen
    vertical_flip = True, # giros verticales aleatorios
    horizontal_flip = True, #giros horizontal aleatorio
)
```

```
# para ver las imagenes que se generaran
imgTrainGen.fit(fotos)
plt.figure(figsize=(20,8))
for imagen, etiqueta in imgTrainGen.flow(fotos, etiquetas, batch_size=10, shuffle=False):
    for i in range(10):
        plt.subplot(2, 5, i + 1)
        plt.xticks([])
        plt.yticks([])
        plt.imshow(imagen[i], cmap='gray')
    plt.show()
    break
# imagenes para entrenamiento
imgtrain = imgTrainGen.flow(fotos, etiquetas, batch_size=32)
# red neuronal con modelo de capas densas
ModeloDenso = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape = (200,200,1)), # capa de entrada con 40.000 neuronas
   tf.keras.layers.Dense(150, activation='relu'), # capa densa de 150 neuronas
    tf.keras.layers.Dense(150, activation='relu'), # capa densa de 150 neuronas
    tf.keras.layers.Dense(1, activation='sigmoid'), # capa densa de 1 neurona
])
# Modelo de capas convolucionales
ModeloCNN = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation = 'relu', input_shape = (200,200, 1)), # capa entrada convolucional 32 kernel
    tf.keras.layers.MaxPooling2D(2,2), # capa de max Pooling
    tf.keras.layers.Conv2D(64, (3,3), activation = 'relu'), # capa convolucional 64 kernel
    tf.keras.layers.MaxPooling2D(2,2), # capa de max Pooling
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),  # capa convolucional 64 kernel
    tf.keras.layers.MaxPooling2D(2, 2),  # capa de max Pooling o agrupacion
    # capa densa de clasificacion
    tf.keras.layers.Flatten(),  # capa de entrada con
    tf.keras.layers.Dense(256, activation='relu'),  # capa densa de 256 neuronas
    tf.keras.layers.Dense(1, activation='sigmoid'),  # capa densa de 1 neuronas
])
# modelo con capa convolucional y Drop out, probar con dropout en la entrada
ModeloCNNDO = tf.keras.models.Sequential([
```

```
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(200, 200, 1)),
    # capa entrada convolucional 32 kernel
    tf.keras.layers.MaxPooling2D(2, 2),  # capa de max Pooling
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),  # capa convolucional 64 kernel
    tf.keras.layers.MaxPooling2D(2, 2),  # capa de max Pooling
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),  # capa convolucional 64 kernel
    tf.keras.layers.MaxPooling2D(2, 2),  # capa de max Pooling o agrupacion
    # capa densa de clasificacion
    tf.keras.layers.Dropout(0.5), # en cada ciclo de entrenamiento 50% de neuronas estaran apagadas
    tf.keras.layers.Flatten(),  # capa de entrada con
    tf.keras.layers.Dense(256, activation='relu'),  # capa densa de 256 neuronas
    tf.keras.layers.Dense(1, activation='sigmoid'),  # capa densa de 1 neuronas
])
# compilamos los modelos, se agregan los optimizadores y las funciones de perdida
ModeloDenso.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
ModeloCNN.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
ModeloCNNDO.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
# entrenamiento de las red neuronal
#entrenar modelo Denso
BoardDenso = TensorBoard(log_dir='C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Redes entrenadas/100E1000DoInicio/Denso')
ModeloDenso.fit(imgtrain,batch_size = 32, validation_data = (fotos2, etiquetas2), epochs = 100, callbacks = [BoardDenso], steps_per_epoch = int(np.ceil(len(fotos)/float(32))), validation_steps = int(np.ceil(len(fotos2)/ float(32))))
# guardamos el modelo
ModeloDenso.save('clasidicadorDenso.h5')
ModeloDenso.save_weights('pesosDenso.h5')
print('Modelo Denso terminado')
#entrenar modelo CNN
BoardCNN = TensorBoard(log_dir='C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Redes entrenadas/300ETodaslasfotos/CNN')
ModeloCNN.fit(imgtrain,batch_size = 32, validation_data = (fotos2, etiquetas2), epochs = 400, callbacks = [BoardCNN], steps_per_epoch = int(np.ceil(len(fotos)/float(32))), validation_steps = int(np.ceil(len(fotos2)/ float(32))))
# guardamos el modelo
ModeloCNN.save('clasidicadorCNN.h5')
```

```
ModeloCNN.save_weights('pesosCNN.h5')

print('Modelo CNN terminado')

# modelo CNNDO es decir, con dropout

#entrenar modelo CNN con dropout

BoardCNNDO = TensorBoard(log_dir='C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Redes entrenadas/300ETodaslasfotos/CNNDO')

ModeloCNNDO.fit(imgtrain,batch_size = 32, validation_data = (fotos2, etiquetas2), epochs = 174, callbacks = [BoardCNNDO], steps_per_epoch = int(np.ceil(len(fotos)/float(32))), validation_steps = int(np.ceil(len(fotos2)/ float(32))))

# guardamos el modelo

ModeloCNNDO.save('clasidicadorCNNDO.h5')

ModeloCNNDO.save_weights('pesosCNNDO.h5')

print('Modelo CNNDO terminado')
```

ANEXO C: SCRIPTS RECONOCIMIENTO CON MODELOS ENTRENADOS

```
# librerias
import tensorflow as tf
import cv2
import matplotlib.pyplot as plt
import numpy as np
from keras_preprocessing.image import img_to_array
# direcciones de los modelos
ModeloDenso = 'C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Red neuronal convolucional/RedNeuronalConvolucional/venv/Scripts/clasidicadorDenso.h5'
ModeloCNN = 'C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Red neuronal convolucional/RedNeuronalConvolucional/venv/Scripts/clasidicadorCNN.h5'
ModeloCNNDO = 'C:/Users/spere/OneDrive/Escritorio/USM/Proyecto y titulacion 2023/TT Nuevo/Red neuronal convolucional/RedNeuronalConvolucional/venv/Scripts/clasidicadorCNNDO.h5'
# lectura redes neuronales
#Denso
Denso = tf.keras.models.load_model(ModeloDenso)
pesosDenso = Denso.get_weights()
Denso.set_weights(pesosDenso)
#CNN
CNN = tf.keras.models.load_model(ModeloCNN)
pesosCNN = CNN.get_weights()
CNN.set_weights(pesosCNN)
#CNNDO
CNNDO = tf.keras.models.load_model(ModeloCNNDO)
pesosCNNDO = CNNDO.get_weights()
CNNDO.set_weights(pesosCNNDO)
# captura de imagen
cap = cv2.VideoCapture(0)
#Empieza nuestro while true
while True:
    #lectura videocaptura
    ret, frame = cap.read()
    #pasar a escala de grises
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
    #Redimensionar la imagen
    gray = cv2.resize(gray,(200,200), interpolation=cv2.INTER_CUBIC)
    # normalizar la imagen
    gray = np.array(gray).astype(float) / 255
    #convertimos la imagen a matriz
    img = img_to_array(gray)
    img = np.expand_dims(img, axis=0)
    #realizar la prediccion
    prediccion = CNNDO.predict(img)
    prediccion = prediccion[0]
    prediccion = prediccion[0]
    Prediccion = int(prediccion * 100)
    print(prediccion)
    #Realizar clasificacion
    if prediccion <= 0.5:
        cv2.putText(frame, "Mujer", (200,70), cv2.FONT_HERSHEY_PLAIN, 3,(0,0,255),2)
        cv2.putText(frame, str(Prediccion) + '%', (200, 30), cv2.FONT_HERSHEY_PLAIN, 3, (0, 0,
255), 2)
    else:
        cv2.putText(frame, "Hombre", (200, 70), cv2.FONT_HERSHEY_PLAIN, 3, (0, 255, 0), 2)
        cv2.putText(frame, str(Prediccion) + '%', (200, 30), cv2.FONT_HERSHEY_PLAIN, 3, (0, 0,
255), 2)
    # mostrar fotograma
    cv2.imshow("CNNDO", frame)
    t=cv2.waitKey(1)
    if t == 27:
        break

cv2.destroyAllWindows()
cap.release()
```