Universidad Nacional de Ingeniería

Área de Conocimiento de Tecnología de la Información y Comunicación

Prototipo de control de acceso estudiantil por reconocimiento facial y reconocimiento de placas para entrada vehicular empleando base de datos y servidor en la tarjeta de desarrollo Raspberry Pi

Trabajo Monográfico para optar al título de Ingeniero electrónico

Elaborado por:

Br. Brookling Osmanny Moncada Sequeira
Carnet: 2016-0333U

Br. Larry Uriel Ruiz Cano
Carnet: 2014-0634U

Tutor:

MSc. Fernando Antonio Flores Guido

17 de febrero de 2026
Managua, Nicaragua

Managua, 10 de febrero de 2026

MSc. Claudia Lucía Benavidez Rugama
Directora DAC-TIC
Sus manos.

Estimada Maestra Benavidez:

Reciba cordiales saludos y deseos de éxitos en el desempeño de sus funciones.

El motivo de la presente es para solicitarle sus atentas gestiones para llevar a cabo el proceso de defensa para la monografía que actualmente estoy como tutor con el tema: **"Prototipo de control de acceso estudiantil por reconocimiento facial y reconocimiento de placas para entrada vehicular empleando base de datos y servidor en la tarjeta de desarrollo Raspberry Pi",** presentado por los Bachilleres:

- Br. Brookling Osmanny Moncada Sequeira Carné: 2016 – 0333U
- Br. Larry Uriel Ruiz Cano Carné: 2014 – 0634U

Según la revisión del informe final, los bachilleres se encuentran listos para la presentación de su trabajo en el proceso de defensa.

Agradezco de antemano su atención a la presente comunicación y me despido enviándole un cordial saludo.

Atentamente,

MSc. Fernando Flores Guido
Tutor del trabajo monográfico
Cc.: Archivo

Universidad Nacional de Ingeniería
Recinto Universitario "Simón Bolívar"
Facultad de Electrotecnia y Computación
Decanatura
DF-03-2022-12

09 de marzo del 2022.

Bachilleres.
Larry Uriel Ruiz Cano 2014-0634U
Brookling Osmanni Moncada Sequeira 2016-0333U
Egresados de la Carrera de Ingeniería Electrónica.

Estimados Bachilleres:

El suscrito Decano de la Facultad de Electrotecnia y Computación, a través de la presente autoriza de manera formal la inscripción de la Monografía Titulada "**Prototipo de control de acceso estudiantil por reconocimiento facial y reconocimiento de placas para entrada vehicular empleando base de dato y servidor en la tarjeta de desarrollo raspberry pi.**" Para optar al Título de Ingeniero Electrónico, para tal efecto se nombra como Tutor de la Monografía al **Ing. Fernando Flores.**

Así mismo le solicito proceda a la **Inscripción de dicho Tema Monográfico** en Secretaria Académica de la facultad, con la finalidad de darle control y seguimiento, de acuerdo a los reglamentos establecidos.

Se les recuerda que, según la normativa para los trabajos monográficos, a partir de la fecha de inscripción tiene 12 meses para defender dicho trabajo.

Sin más a que referirme y deseándoles mucho éxito en la culminación de esta etapa, me despido.

Atentamente,

Ing. Ronald Torres
Decano FEC

Cc: Ing. María Lourdes Montes. Secretaria Académica.
Ing. Marlon Robleto. Jefe de Dpto. de Sistemas Digitales y Telecom.
Ing. Fernando Flores. Tutor.

Archivo.

Secretaria Académica
DACTIC

SECRETARIA DE ÁREA ACADÉMICA

F-8:CARTA DE FINALIZADO PLAN DE ASIGNATURA

El Suscrito Secretario del **ÁREA DE CONOCIMIENTO DE TECNOLOGÍA DE LA INFORMACIÓN Y COMUNICACIÓN** hace constar que:

RUIZ CANO LARRY URIEL

Carné: **2014-0634U** Turno: **Nocturno** Plan de Asignatura: **2015** de conformidad con el Reglamento Académico vigente en la Universidad, ha aprobado todas las asignaturas correspondientes a la carrera de **INGENIERÍA ELECTRÓNICA**, en el año 2021 y solo tiene pendiente la realización de una de las formas de culminación de estudio.

Se extiende la presente **CARTA DE FINALIZADO PLAN DE ASIGNATURA**, a solicitud del interesado en la ciudad de Managua, a los veinte y nueve días del mes de enero del año dos mil veinte y seis.

Atentamente,

MSc. Luisa Massiel Mercado Gutiérrez
SECRETARIO DE ÁREA ACADÉMICA

Móvil: (505) 83803517

Recinto Universitario Simón Bolívar
Avenida Universitaria.
Managua, Nicaragua.
Apdo: 5595

hazzely.orozco@dactic.uni.edu.ni
www.uni.edu.ni

Secretaria Académica
DACTIC

SECRETARIA DE ÁREA ACADÉMICA

F-8:CARTA DE FINALIZADO PLAN DE ASIGNATURA

El Suscrito Secretario del **ÁREA DE CONOCIMIENTO DE TECNOLOGÍA DE LA INFORMACIÓN Y COMUNICACIÓN** hace constar que:

MONCADA SEQUEIRA BROOKLING OSMANNI

Carné: **2016-0333U** Turno: **Diurno** Plan de Asignatura: **2015** de conformidad con el Reglamento Académico vigente en la Universidad, ha aprobado todas las asignaturas correspondientes a la carrera de **INGENIERÍA ELECTRÓNICA**, en el año 2021 y solo tiene pendiente la realización de una de las formas de culminación de estudio.

Se extiende la presente **CARTA DE FINALIZADO PLAN DE ASIGNATURA**, a solicitud del interesado en la ciudad de Managua, a los veinte y nueve días del mes de enero del año dos mil veinte y seis.

Atentamente,

MSc. Luisa Massiel Mercado Gutiérrez
SECRETARIO DE ÁREA ACADÉMICA

Móvil: (505) 83803517

Recinto Universitario Simón Bolívar
Avenida Universitaria.
Managua, Nicaragua.
Apdo: 5595

hazzely.orozco@dactic.uni.edu.ni
www.uni.edu.ni

Dedicatoria

Primeramente, dedico este trabajo monográfico a mi Señor Jesucristo por haberme permitido llegar a esta etapa de mi vida. Sin ÉL nada de esto fuera posible y no hubiera logrado todo lo que he hecho.

También, dedico este trabajo a mis padres Roger Rafael Ruiz Mercado y Perla Danelia Cano Baltodano. Les agradezco enormemente por su incondicional apoyo y ánimos que me han brindado en todo tiempo a pesar de mis múltiples errores. Por darme la oportunidad de estudiar la carrera que he querido, por todo el apoyo económico que recibí en todo momento y por todo el amor que se me fue dado desde que fui niño. Es de mi total agrado y con todo mi amor honrarlos dedicándoles este trabajo.

Y finalmente, dedico este trabajo a todas aquellas personas que me han brindado su apoyo y que me han extendido la mano sin dudarlo ni un instante.

Br. Larry Uriel Ruiz Cano

Le dedico este trabajo primeramente a Dios quien ha sido el que me ha brindado apoyo y consuelo en los momentos más turbulentos de mi vida al igual que él me ha permitido darme la sabiduría y fuerza cuando estuve en esta casa de estudios.

Agradezco a mis tutores a mi Abuela Marina y a Mi Tío Omar, quienes se han hecho cargo de mí, desde los primeros años de mi vida, al igual que le dedico este trabajo a mi prima Nataly, quien me apoyo durante los primeros años de universidad, al igual que a mi hermano Brandon, el cual me demuestra, que siempre se debe de luchar, al igual que está luchando con sus adicciones.

Br. Brookling Osmanny Moncada Sequeira

Resumen del tema

Este trabajo monográfico está basado en un prototipo de sistema de control de acceso mediante reconocimiento facial y reconocimiento de placas vehiculares los cuales se han trabajado en un lenguaje de programación que facilite los métodos a utilizar, en este caso, se ha realizado el trabajo en Python mediante el uso de varias librerías para la creación de redes neuronales las cuales se van a encargar de realizar el proceso de aprendizaje para su posterior reconocimiento de las personas y también para la detección de imágenes en las placas vehiculares.

El sistema de detección se puede explicar desde la lógica de programación basado en una entrada, luego una Blackbox (caja negra) que se encarga de realizar los procesos y posteriormente una salida.

Basado en lo anteriormente mencionado, la entrada será una imagen del rostro captada mediante una cámara la cual será capturada usando “métodos” de las distintas librerías de Python tales como open CV, Tensorflow. Estos nos ayudarán a la detección de las imágenes y la captura de las mismas.

Luego estas imágenes serán procesadas mediante DSP (digital signal processing) para hacer más liviana la imagen, enmarcarla, detectar bordes para así recopilar características del rostro a reconocer. Todo esto es logrado por la red que es la que está actuando por detrás de todo el sistema la cual está elaborado mediante la librería de Keras. Este sistema de redes convolucionales es el más óptimo para trabajar imágenes y videos. Como salida saldría la persona detectada, en este caso es el reconocimiento facial.

Lo mismo ocurre para las placas. Estas tienen una imagen de una placa como entrada, se establece un parámetro similar de detección de imágenes las cuales tendrán que reconocer números y letras dejando así por un lado todo el resto de información de la imagen la cual es inútil y podría considerarse como basura puesto que cargaría la red convolucional de información que no nos interesa. Como resultado tenemos la detección de letras y números los cuales son los datos de importancia al analizar la placa. Esto es montado en una tarjeta de desarrollo para pruebas Raspberry pi.

Índice

Introducción

Para el siguiente proyecto se desarrolló una arquitectura de red neuronal llamada red siamés o red gemela. Esto fue con el objetivo de realizar una comparación con una imagen de entrada mediante una cámara, más una imagen guardada como referencia, para encontrar similitudes.

Toda la programación se implementó en un prototipo montado en una Raspberry Pi 5 que tiene por objetivos el realizar la validación o reconocimiento de características biométricas, en este caso el rostro. De igual modo, para la detección de placas vehiculares se usó un código el cual detecta el rectángulo de la placa y posteriormente extrae el número de la placa. Como punto adicional, se envían los datos a una base de datos que emplea NoSQL; es decir, una base de datos no estructural la cual se eligió que fuera MONGODB, ésta mantendrá la información almacenada en un servidor, información la cual se puede acceder remotamente solo con la validación de la dirección ip a la cual apunta el servidor, mediante el programa especial visual para BD.

Se abordaron las definiciones y las técnicas usadas para este proyecto, al igual que las observaciones, entre ellas los resultados. También se hizo mención de técnicas, algunas librerías y algunos métodos de programación. La información presentada pretende ser lo más óptima. En algún punto las definiciones serán llamadas y se adicionó mayor profundidad a alguna de ellas para explicar algunos fenómenos, el tema de redes neuronales es muy amplio y el estudio de estos mismos conlleva al conocimiento de muchos conceptos, conceptos que muchas veces al aprender pueden llegar a ser frustrantes.

Antecedentes

El uso de herramientas que permiten un análisis y registro de rostros, es un nuevo campo de desarrollo que permite el facilitar al acceso o confirmación de individuos, esta herramienta se ha desarrollado a medida que la tecnología crece, los principales antecedentes que se encontraron a nivel tanto nacional e internacional son:

Nacional

Se desarrolló una aplicación web la cual permitía reconocer por medio de la webcam los rostros de los estudiantes, para facilitar el reconocimiento sobre los estudiantes presentes, para apoyar a los docentes. De la universidad UNAN-León (2020) [Cerros C, Gabriel F, Reyes Dávila, Norvin E, Munguía, Carlos G. Aplicación web para evaluaciones en línea con reconocimiento facial utilizando la librería face-api.js en apoyo a la docencia.

Esta monografía realizar guías de laboratorio enfocadas en el procesamiento de imágenes, brindando los algoritmos necesarios para comprender la estructura y filtros para una imagen, implementa raspberry pi, como computador. De la universidad UNI-Central (2016) Pavel F. A A. se tiene Desarrollo de un sistema de reconocimiento de imágenes utilizando Raspberry Pi y MATLAB® que contribuya al aprendizaje del tratamiento digital de señales).

Internacional

Esta monografía estudia el comportamiento de la tecnología que se utiliza en el Kinect, tanto reconocimiento de voz, reconocimiento de gestos y reconocimiento facial. De la universidad San Carlos Guatemala se tiene: Guía del funcionamiento de tecnologías de reconocimiento de Kinect (Martínez V. Pedro P. 2017) ,

Esta monografía se tiene que se desarrolló un sistema de reconocimiento facial el cual verifica en tiempo real si las personas que entran a las instalaciones, forman parte de la base de datos, de las personas que laboran ahí. De la universidad Tecnológica de Panamá se tiene: Implementación de un sistema de control de acceso basado en reconocimiento facial (Poveda. Merchán, F. 2016

Repositorio Internacional (Latinoamericano)

Lo que se realiza en esta tesis es un sistema de control de acceso usando el reconocimiento facial de manera experimental para mejorar el control de acceso a los estudiantes para los laboratorios de la universidad. De la universidad de Perú se tiene: Sistema de reconocimiento facial para el control de acceso de estudiantes a los laboratorios de la FIIS-UNAC, (Yañez N, Margarita L.2019).

Esta tesis muestra las técnicas para mejorar y modificar imágenes, filtrando, y de este modo obteniendo los datos de las placas vehiculares, que se desean analizar. Del instituto politécnico nacional de México se tiene: Reconocimiento de placas vehiculares, (Jose,L,D,M 2010).

Esta tesis tiene como objetivo apoyar a empleados de tiendas a reconocer ciertos artículos, herramientas mediante el procesamiento de imágenes de open cv, la tlapalería es un tipo de tienda en donde venden artículos de ferretería, pinturas o artículos de albañilería. De la universidad autónoma de México se tiene Reconocimiento de objetos utilizando OpenCV y Python en una Raspberry pi 2 en una tlapalería (2017)

Se utiliza el procesamiento de imágenes, para el reconocimiento facial de personas para el control de acceso con una efectividad del 85% a una distancia de 1.5 metros, fue implementado en la facultad de la universidad para evitar la inseguridad, fue ejecutado en raspberry pi. De la universidad de Venezuela se tiene SISTEMA DE SEGURIDAD BASADO EN RECONOCIMIENTO FACIAL UTILIZANDO UNA RASPBERRY PI (2017), Sousa Karina Mora.

Esta tesis abarca sobre la generación de alertas, para un auto móvil cuando se acerca un usuario que no está registrado en el sistema, hace uso de gps, datos, y de una aplicación de mensajería, en este caso telegram, para enviar las alertas y de una base datos, al igual que Google maps, para registrar los recorridos. De la escuela politécnica de Ecuador se tiene “DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE SEGURIDAD Y ALERTA PARA VEHICULOS, BASADO EN RECONOCIMIENTO FACIAL Y LOCALIZACIÓN GPS, EN UNA RASPBERRY PI B PLUS. ”(HENRY P,E,P, 2016),

Justificación

La creación del prototipo facilita el acceso más fluido de estudiantes en el Recinto Universitario Simón Bolívar de la UNI. El fácil y rápido acceso permite evitar la aglomeración de personas en la entrada del recinto, esto debido a que se desea complementar el sistema de acceso por carnet, y brindar una alternativa que permita el ingreso de los estudiantes en el caso de no portar su carnet. El costo del desarrollo del dispositivo será bastante reducido en comparación con el precio que existe en el mercado de dispositivos embebidos en reconocimiento facial que ofertan muchas empresas enfocadas en el ámbito de seguridad. Del mismo modo, al no haber un sistema de seguridad ni cámaras en el campus, se pretende contar con esta herramienta la cual permite tener un registro de las personas que ingresan y ayudar a la disminución de pérdidas materiales que se pueda dar en los diferentes espacios de la universidad, provocado por el acceso a personas con intenciones de hurtar lo ajeno.

Existen diversas bibliotecas en el lenguaje de programación de Python que nos permiten y a la vez, facilitan trabajar herramientas para programar y desarrollar una inteligencia artificial capaz de detectar imágenes y detectar a las personas mediante facial recognition. Estas librerías pueden ser Tensorflow-, Open-cv, Pytorch…etc.

Estas herramientas mencionadas facilitan la manipulación de arreglos de datos manipulación de grandes valores numéricos, obtención de imágenes, captura de imágenes, procesamiento digital de señales (DSP) para reducir el tamaño de las imágenes mediante la eliminación de ruido y valores innecesarios de la captura mediante la cámara; también, facilitan la creación de redes neuronales capaz de realizar el aprendizaje de la IA.

Como estudiantes de ingeniería electrónica de esta Alma Mater se pretende realizar este estudio mediante los principios de programación y de detección de imágenes y rasgo biométricos, un área que puede ser desarrollada en la carrera. El uso de técnicas para el procesamiento de señales digitales nos permite tener la posibilidad de mejorar la calidad de las imágenes mediante el filtrado de la señal y eliminar el ruido provocado por la digitalización.

Objetivos

Objetivo general

- Elaborar un prototipo de control de acceso estudiantil mediante reconocimiento facial y control de acceso vehicular mediante reconocimiento de placas.

Objetivo específico

- Desarrollar un prototipo de reconocimiento facial y detección de imágenes implementando la tarjeta de desarrollo Raspberry pi y cámara.
- Usar un código que será usado para reconocimiento facial y el reconocimiento de placas vehiculares.
- Configurar una base de datos para generar un registro de cada una de las actividades de entrada estudiantil y de entrada vehicular.
- Estructurar un servidor para el registro de los datos que se obtienen y así tener acceso a estos.

Marco teórico

Inteligencia artificial

La inteligencia artificial o (IA) por sus siglas en español o (AI) por sus siglas en inglés (Artificial Intelligence), es una tecnología que permite que las computadoras simulen la inteligencia humana (IBM, s.f.) y las capacidades humanas para resolver problemas, por si sola o empleando otras tecnologías, la IA puede realizar tareas que de otro modo solo podrían ser resueltas por humanos (AWS, 2023), por ejemplo, al adicionarle sensores se logran aplicaciones como:

Asistentes digitales, Asistentes de Voz, Guías Por GPS, Vehículos autónomos, herramientas de inteligencia artificial Generativa, estas al adicionarles prompts, tales como ChatGPT de open Ai.

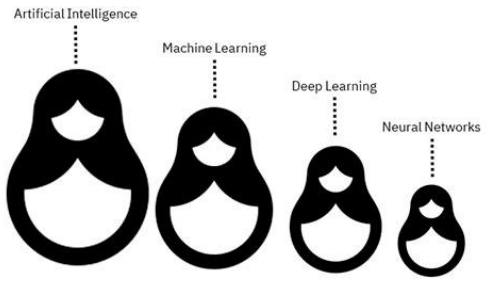


Ilustración 1 Representación de los diferentes modos conceptos de Ia representados en matriohkas. (Francisco Alonso, Sf) recuperado de : https://www.futurespace.es/redes-neuronales-y-deep-learning-capitulo-1-preludio/

Machine learning

Machine learning o (Aprendizaje automático) es una sub rama de la inteligencia artificial (IA) (Data Science, s.f.) y la informática, se centra en el uso de datos y algoritmos para imitar la forma en la que aprenden los seres humanos, (IBM, s.f.) el Machine Learning o abreviado (ML) consiste en dejar que los algoritmos aprendan patrones (Patterns) en conjuntos de datos (Inputs) los cuales pueden ser (Textos, Números, Imágenes o datos de estadística).

El Machine Learning o “non-Deep” depende mucho de la intervención humana para aprender, esto debido a que los seres humanos deben de declarar las jerarquías de los datos, por lo cual se traduce como datos más estructurados y mayor dependencia para el entrenamiento de los datos (AWS, 2023).

Deep Learning

El Deep Learning o (DL) es un subcampo de la inteligencia artificial, a su vez el Deeplearning es un es subcampo del Machine Learning, el aprendizaje profundo o DL se refiere a una red neuronal compuesta por más de tres capas, que incluyen capa de entradas y salida y capas ocultas, se le considera un aprendizaje profundo.

El DeepLearning se diferencia con el machine learnig además del tamaño de la red, con respecto a las capas que usa en la automatización de características, lo que permite una mayor escalabilidad y menor intervención humana.

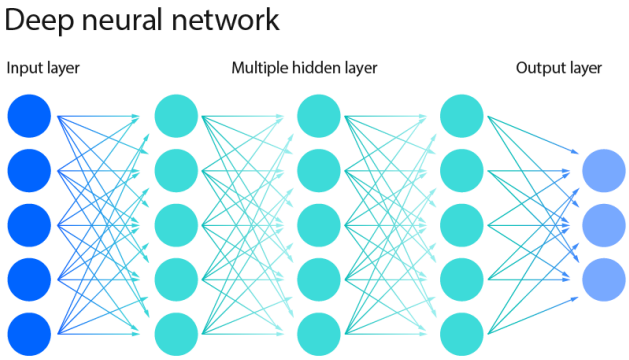


Ilustración 2 Modelo de DeepLearning (IBM, SF) recuperado de : https://www.ibm.com/mx-es/topics/artificial-intelligenc

Frameworks

Un Framework o marco de trabajo es una plantilla para desarrollar softwares de manera rápida y eficiente (ebac, 2023), se podría decir que un Framework es un conjunto de herramientas, bibliotecas y normas que facilitan la creación de un Software por ende para cualquier desarrollo a futuro no se tendría que partir desde cero, Titulo.sf. (DIGIZONES LAB) . **El Framework Utilizado en este trabajo monográfico es TensorFlow**

Dataset

Un **Data-set** es un conjunto de datos que “habitualmente” están estructurados, se podría decir que una tabla de SQL sería un dataset, teniendo las columnas como una variable las filas representando diferentes registros, esta unión de filas y columnas junto con los valores conforman el dataset (The Data School , 2024).

Un **Dataset**, puede estar compuesto por datos como: texto, números, imágenes, videos, registros de eventos, etc. Estos datos se almacenan y se recopilan con el propósito de ser analizados, tomar decisiones o entrenar modelos de machine Learning (The Data School , 2024).

Son **fundamentales**, para el entrenamiento de datos en modelos de inteligencia artificial, cuanto más grande sea el data set será mejor el rendimiento en el cumplimiento del modelo de inteligencia artificial (Cañadas, 22) (Navarro, KEEPCODING, 2024).

Imagen

Una imagen no es mas que una matriz de valores matemáticos los cuales reciben el nombre de Pixeles, los pixeles (Picture Element) es la unidad más pequeña de una imagen, los pixeles contienen dentro de su estructura los conocidos llamados como sub-Pixeles, los cuales permiten que una imagen de manera informática se pueda representar los canales son: RGB (RED; GREEN; BLUE) estos son los colores primarios de los cuales se pueden formar todos los colores del espectro.

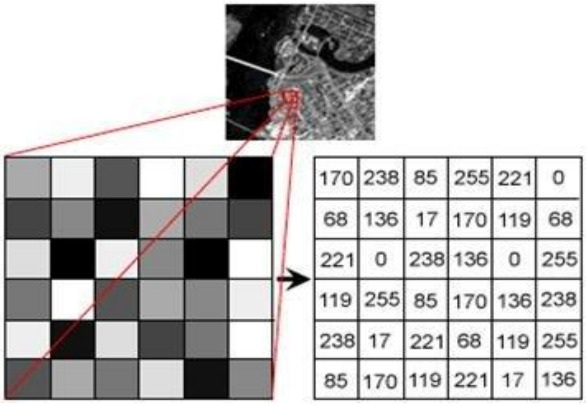


Ilustración 3 Vision computarizada (Negi, 2022) recuperado de: https://medium.com/@chawthirisan/spatial-vs-frequency-domain-a-guide-to-image-interpretation-d9c16b129b3f el 26 de junio del 2024

RGB

RGB (Red, Green, Blue) son los colores primarios estos permiten representar una imagen a color también conocida como representación **cromática**. (Porto, 2022), en programación para el uso de imágenes los modelos mas usados son RGB, para imágenes a color, declarando 3 canales y Escala de grises.

Por ejemplo:

En programación, se pueden mandar a llamar cada uno de los canales y de este modo trabajarlos como pixeles individuales, o directamente se define el canal con el cual se quiere trabajar como los siguientes: Rojo = (255, 0, 0), Verde= (0,255,0), Azul(0,0,255).

Ilustración 4 computarizada (Negi, 2022) recuperado de: https://medium.com/@chawthirisan/spatial-vs-frequency-domain-a-guide-to-image-interpretation-d9c16b129b3f el 26 de junio del 2024

Escala de grises

La escala de grises es la representación de un solo canal en ese canal se tiene blanco y negro, con la característica que cada pixel contiene un byte de información, de tal modo que puede haber colores negros (0), colores obscuros claros (100) y el color blanco al llegar al final de byte (255).

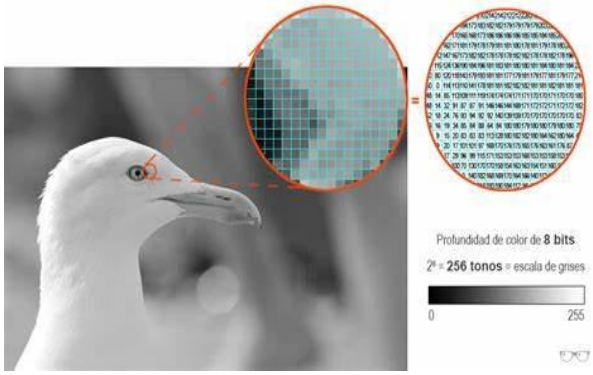


Ilustración 5 Vision computarizada (Pérez, 2022) recuperado de: https://www.elvisualista.com/2016/05/05/que-es-un-mapa-de-bits-1/ el 15 de Julio del 2024

Escala de blanco y negro

La escala en blanco y negro a diferencia de la escala de Grises, lo que hace es tomar solamente dos posibles valores (Bit) blanco para los más altos (255) y negro para los más bajos (0), se podrá ver en la siguiente Ilustración 6, es por ello que los modelos para visión computarizada usualmente se trabajan en escala de grises, dado a las

ventajas que la escala provee. También es válido destacar que una forma de ver la escala de blanco y negro es como un bit encendido y un bit apagado.

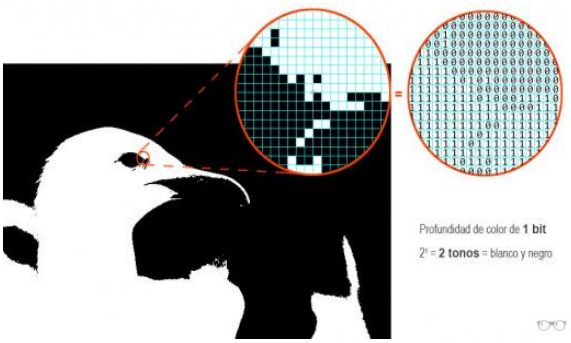


Ilustración 6 Vision computarizada (Pérez, 2022) recuperado de: https://www.elvisualista.com/2016/05/05/que-es-un-mapa-de-bits-1/ el 15 de Julio del 2024.

Open CV

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel, openCV significa Open Computer Vision (Visión Artificial Abierta). Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en una gran cantidad de aplicaciones, y hasta 2020 se la sigue mencionando como la biblioteca más popular de visión artificial. Detección de movimiento, reconocimiento de objetos, reconstrucción 3D a partir de imágenes, son sólo algunos ejemplos de aplicaciones de OpenCV.

Redes neuronales

Las redes neuronales también conocidas como redes artificiales (**ANN**), o redes simuladas (**SNN**), son un subconjunto de machine learning y constituyen el eje de los algoritmos del Deep-Learning sf (IBM, n.d.), son un modelo computacional el cual tiene el propósito de asemejar o emular los procesos que realiza el cerebro humano, específicamente las neuronas orgánicas, para reconocer patrones, clasificar datos y predecir eventos a futuro sf (Mathworks (Matlab(, n.d.).

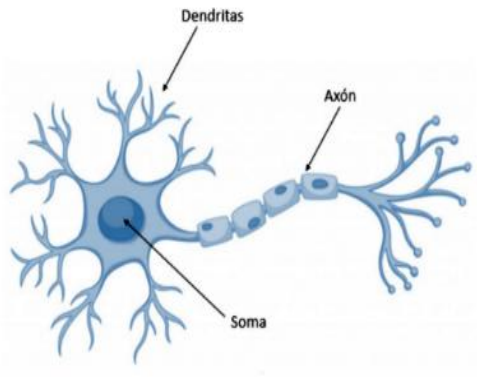


Ilustración 7 Imagen de una Neurona orgánica, (Universidad de Guanajuato, 2022) recuperado de : https://blogs.ugto.mx/rea/clase-digital-2-bases-biologicas-de-la-conducta/

El perceptrón

Las redes neuronales están formadas por nodos de capas, cada nodo tiene la capacidad de interconectarse entre si mismos, teniendo que pueden interconectarse entre cada neurona.

El perceptrón es una neurona artificial o unidad de red neuronal, es el que realiza ciertos cálculos al igual que aprendizaje (ICHIPRO, 2024), es el componente básico de las redes artificiales, el perceptrón es un modelo simplificado de la neurona real que se intenta imitar matemáticamente.

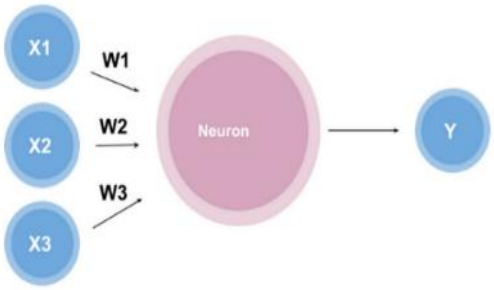


Ilustración 8 Imagen de Un perceptrón, (Ichy.pr, sf) recuperado de: https://ichi.pro/es/que-es-un-perceptron-conceptos-basicos-de-las-redes-neuronales-216397265290640

Un perceptrón se puede representar matemáticamente como una sumatoria de la multiplicación de los pesos por las entradas más una función de activación.

$$z = \sum_{i=1}^{n} x_i w_i$$

$$\phi(z) = \begin{cases} -1 & z \leq w_0 \\ 1 & z > w_0 \end{cases}$$

Ilustración 9 Función matemática de Perceptrón (Ichi.pro, 2024) recuperado de: https://ichi.pro/es/perceptron-explicacion-implementacion-y-un-ejemplo-visual-66582574588625

El perceptrón consta de tres componentes, La entrada, los pesos y la función de activación (Redes Neuronales Artificiales , 2023).

Como se puede apreciar en la ilustración 8, se tiene al perceptrón al cual tienen unas entradas llamadas $X1, X2, X3$**,** estas entradas se les conoce como capa de entrada, se conectan al perceptrón o neurona artificial, y este lo que hará será multiplicar los pesos por las entradas y realizar una sumatoria de los demás datos que ingresan a la neurona.

Como se puede apreciar en la fórmula:

Z= Salida, Output. $Z = \sum_{i=0}^{n} Xi * Wi$ en donde se tiene que Xi representa las entradas y que Wi representan los pesos, esto se puede apreciar mejor en la siguiente imagen:

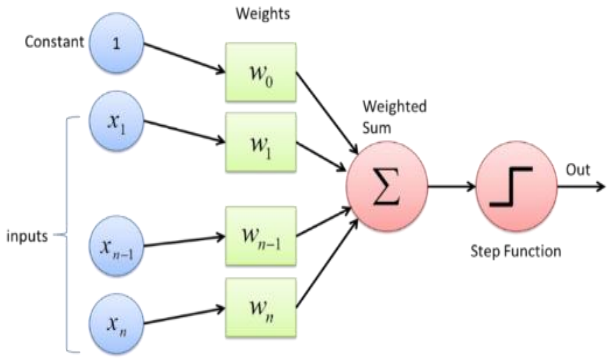


Ilustración 10 Imagen de perceptrón (Data Science Team,202) recuperado de: ¿Perceptrón? — Aprendizaje automático — DATA SCIENCE

Dentro de esta **Ilustración** se puede apreciar los datos de entrada, la sumatoria que se realiza, dentro de la sumatoria está implícito la multiplicación de los pesos por las entradas y antes de pasar a la salida está una función o condición de activación

llamada **Función de activación** que es la que permite que bajo ciertas condiciones se active o no se active la neurona en dependencia del modelo al cual se haya entrenado esta misma.

Peso

Peso también conocido como parámetro, son valores que se asocian con las conexiones de neuronas, cada conexión de neurona tiene un peso, los pesos son los valores que se añaden a la entrada de la neurona, por ejemplo, en la **ilustración 10** se puede ver que primero están los datos de entrada y después están los pesos o **Weight.**

Los pesos representan las fuerzas de la relación de entradas y salida de una red, al ajustar los pesos durante el entrenamiento, el modelo aprende a asignar la importancia adecuada a cada característica, mejorando su capacidad para hacer predicciones.

Ejemplo: Si se requiere detectar un correo de **Spam**, se puede entrenar un modelo con ejemplos y este modelo será supervisado por lo cual, el peso podría verse ajustado cada que se encuentre la palabra "**Gratis",** dado a que este encontrara la mayor cantidad de características de manera automática, por lo cual, en base a los ejemplos y la supervisión de estos mismos las neuronas a entrenar, podrán corregir, sus valores, hasta ajustarse a una salida parecida a la cual se le entreno, y poder realizar predicciones de hasta un 99.98 % de efectividad.

Segundo Ejemplo:

Imaginemos que tenemos que crear un modelo de Machine Learning, entrenando una red neuronal, que sea capaz de valorar los precios de los autos, para este caso solamente se le darán dos ejemplos:

- Año del equipo.
- Modelo del Equipo.

El modelo del equipo en correlación con el año del equipo, podrían dar un valor al auto, a menos que sea un auto coleccionable o único, se podría estimar que mientras más viejo el auto, menos valor tenga y mientras más reciente mayor valor tenga, por lo cual, apenas se den los datos para la detección, los pesos del año se dispararan.

Esto pasara también con el modelo, si es un modelo de Colección al ser más antiguo, el valor aumentara.

Las redes neuronales (ANN) están compuestas de los siguientes elementos, capas de entradas, capas ocultas y capas de salida. Se tiene que cada elemento de una capa sea de entrada, oculta o de salida es una neurona.

Función de activación

La función de activación (Ver Ilustración 10) es un elemento esencial para diseñar una red Neuronal la elección de la función de activación proveerá el control completo sobre el proceso de entrenamiento del modelo de Red (Team, 2021).

La función de activación es un filtro que funge como limitador o umbral, para los resultados de salida, de la neurona, modifica el valor de salida o limita este mismo, antes de ser enviado a otra neurona, las funciones de activaciones harán que las neuronas sean no lineales.

La función de activación recibe un valor de entrada y se encarga de devolver una salida que corresponde a la decisión o el peso determinado a partir de la entrada, la función de activación es una combinación de información que transporta, esta misma que proviene de los sesgos y pesos las funciones de activación pueden ser Lineales o no Lineales (Team, 2021). Se ejecutan por cada red y por cada Neurona.

Una red neuronal sin una función de activación es un modelo de regresión lineal clásico, la función de activación realiza la transformación no lineal de la entrada lo que le hace capaz de aprender y realizar tareas más complejas.

Para elegir la función de activación dependerá de la tarea que se esté ejecutando, como se puede ver en la siguiente Ilustración, esta ayuda en resumir las funciones más comunes usadas para problemas de: Regresión y Clasificación. Cabe destacar que este modelo, no toma en cuenta múltiples funciones de activación que existen a día de hoy.

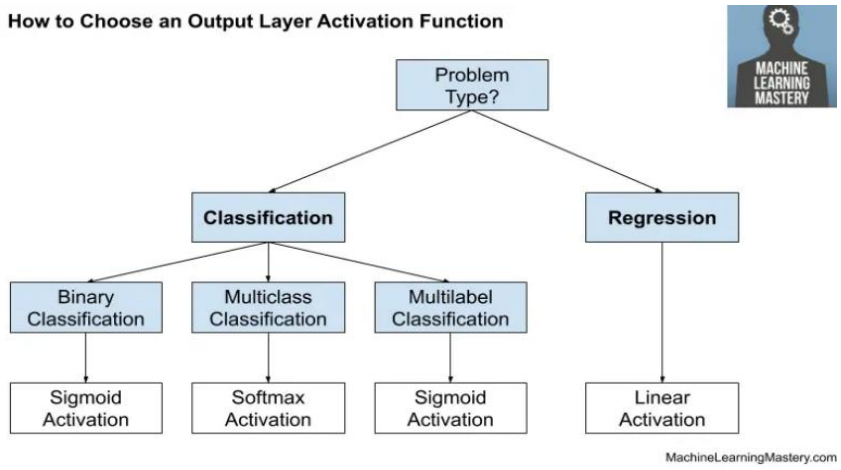


Ilustración 11 Que tipo de modelo usar según la necesidad, (Machine Learning, 2021) recuperado de: https://topbigdata.es/como-elegir-una-funcion-de-activacion-para-el-aprendizaje-profundo/

Tipos de funciones de activación

Se tienen la siguiente lista de funciones de activación, todas y cada una de estas tienen funciones básicas y útiles para diferentes esquemas y tipos de redes neuronales, para este trabajo se estará abordando solamente 3, Lineal, Sigmoidal, Relu, dado a que estas son de las más importantes, cabe destacar que existen diferentes y nuevas funciones de activación, la mayoría son derivaciones de las antes descritas, por lo cual, no se ahondara en este documento sobre ellas. Se mencionan a continuación algunas:

Función Lineal, Función Sigmoidal, Función Relu.

Función lineal

La función lineal es la que da el mismo resultado de la entrada a la salida.

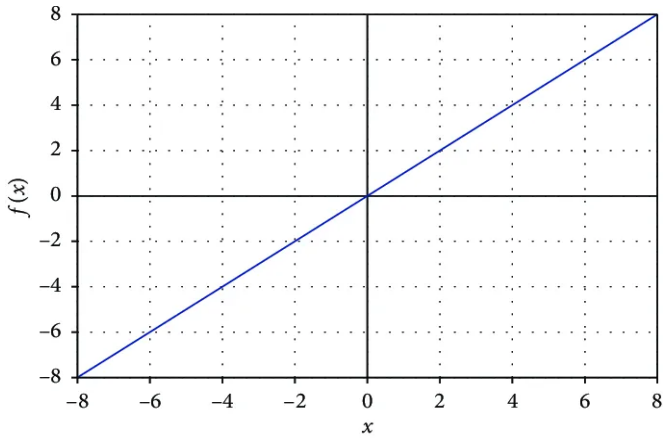


Ilustración 12 Función de Activación Lineal (Cristina Ortega, s.f.) recuperado de: https://www.questionpro.com/blog/es/modelos-de-machine-learning/

Se puede apreciar en la Ilustración 12 que la entrada va directo conectado a la salida no pasa por ningún filtro, este tipo de modelos para activación de Perceptrones es útil para tareas como las conversiones de grados Celsius a Fahrenheit, al igual que las estimaciones de predicciones para el valor de un hogar como posibles cambios en la bolsa de valores.

Se utiliza cuando se requiere mayormente, prever una salida numérica constante. Otra manera gráfica de comprender la función lineal es la siguiente, se tienen dos grupos a procesar por la red neuronal, ambos grupos según la forma que se tienen, pueden ser intervenidos por una función lineal, teniendo el siguiente resultado:

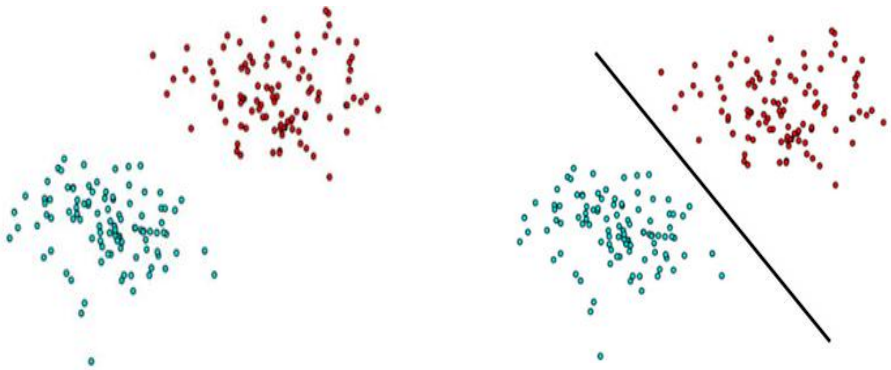


Ilustración 13 Clasificación de Datos con una función de activación Lineal (Francisco Alonso, sf) recuperado de : https://www.futurespace.es/redes-neuronales-y-deep-learning-capitulo-2-la-neurona/

Como se puede observar en la **figura 13**, la función lineal es muy buena, para datos de entrada y salida, como conversiones numéricas, de temperatura o el cálculo de valor de propiedad.

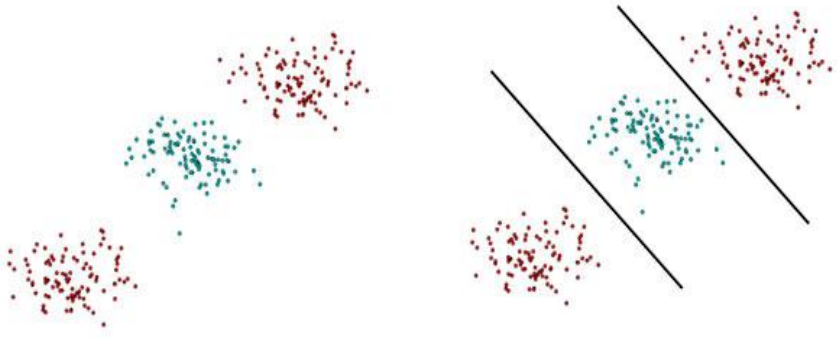


Ilustración 14 Conjuntos de Datos No Lineales (Francisco Alonso, sf) recuperado de: https://www.futurespace.es/redes-neuronales-y-deep-learning-capitulo-2-la-neurona/

En dependencia de la cantidad de datos, que se tengan y se requieran tratar se pueden utilizar más neuronas, y el resultado será el que se puede observar en la **Ilustración 14**.

¿Pero qué pasa si se tiene la siguiente distribución de datos?

Ilustración 15 Conjuntos de Datos No Lineales (Francisco Alonso, sf) recuperado de: https://www.futurespace.es/redes-neuronales-y-deep-learning-capitulo-2-la-neurona/

Para este tipo de escenarios a la neurona, se le haría prácticamente imposible solucionar este esquema no lineal, se pueden incluir múltiples neuronas, para obtener un resultado aproximado, pero no es lo más adecuado computacionalmente hablando y no se llegará al resultado esperado, es debido a esto, que existen otros modelos de funciones de activaciones para estas funciones de activaciones se usan las no lineales.

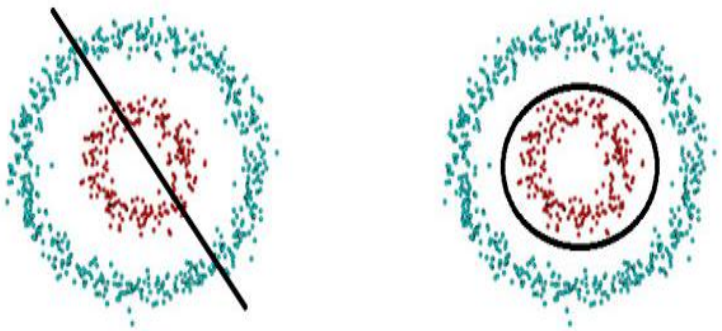


Ilustración 16 Distribución de datos con Neuronas lineales y Distribución de datos con Neuronas no lineales (Francisco Alonso, sf) recuperado de: https://www.futurespace.es/redes-neuronales-y-deep-learning-capitulo-2-la-neurona/

Por ejemplo, en la siguiente **ilustración 16**, se tiene como se comportaría una red con función de activación Lineal vs el resultado que tomaría una red no lineal.

Función sigmoide

La función sigmoide lleva todos nuestros valores entre (0, 1). Por lo tanto, se usa especialmente para modelos en los que tenemos que predecir la probabilidad de algo, ya que utilizamos 0 como 0% y 1 como 100%, esto obedece a procesos de Normalización de datos.

La función es diferenciable, es decir, podemos encontrar la pendiente de la curva sigmoidea en dos puntos cualesquiera. Además, la función es monótona, pero la derivada de la función no lo es.

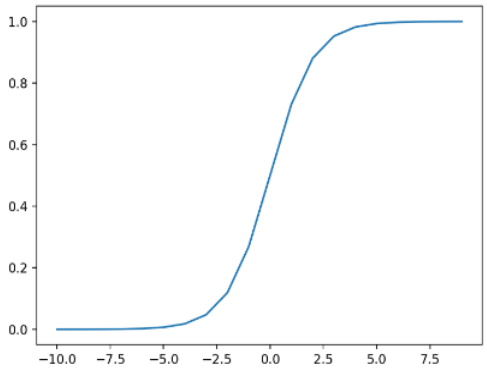


Ilustración 17 Función Sigmoide, (Machine Learning 2024) recuperado de: https://topbigdata.es/como-elegir-una-funcion-de-activacion-para-el-aprendizaje-profundo/

Función Relu

La función ReLu o unidad lineal rectificadora, es la función de activación más utilizada en el mundo en estos momentos, ya que se utiliza en casi todas las redes neuronales convolucionales o de Deep Learning, la ventaja más significante de Relu, es su función de velocidad de convergencia acelerada estocástica o **Stochastic Gradient Descent** (SGD), que indica que tan rápido está aprendiendo la neurona, en comparación con las funciones Sigmoide y tangente Hiperbólica,

En la función Relu todos los valores Negativos se vuelven ceros, mientras que los demás valores se vuelven 1, esto computacionalmente hablando es de gran ayuda, dado a que descarta la mayoría de los valores y solo toma los que son positivos o llegan a cierto grado de accuracy

$$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$

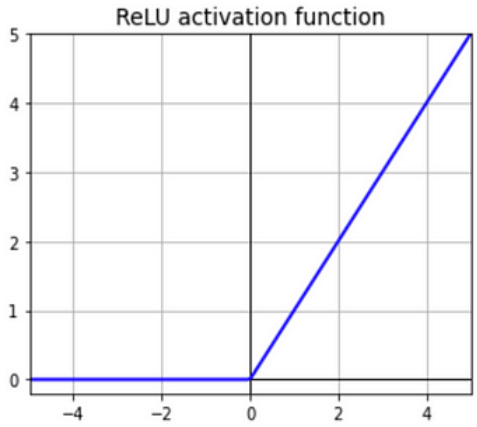


Ilustración 18 Función de Activación Relu (Ichi.pro, sf) recuperado de: https://ichi.pro/es/7-funciones-de-activacion-populares-que-debes-conocer-en-deep-learning-y-como-usarlas-con-keras-y-tensorflow-2-43657675202534

La gran fortaleza de Relu se convierte en parte en su gran debilidad, esta velocidad de aprendizaje **Stochastic Gradient Descent** (SGD), puede hacer que los pesos de las neuronas, estén oscilando entre sus valores más óptimos, por lo cual esto puede causar que algunas de las neuronas que no alcancen estos valores (Óptimos), simplemente jamás serán usadas y quedaran "Muertas", lo cual es una gran debilidad el tener neuronas muertas.

Backpropagation

BackPropagation o (Propagación hacia atrás) es un método que utilizan las redes neuronales durante el entrenamiento con el objetivo de minimizar los errores (Tecnologia, 2023) y entender las pérdidas (Fernández, 2022) son capaces de identificar patrones de datos incompletos o arbitrarios y encontrar una solución más adecuada.

El algoritmo de Backpropagation se usa para entrenar una red neuronal artificial a través de un método llamado regla de la cadena (Tecnología, 2023) cada paso hacia adelante que hacen las neuronas durante el entrenamiento después de cada paso, backpropagation hace un paso hacia atrás, mientras ajusta los parámetros de los modelos (Pesos y sesgos).

Se utiliza usualmente para entrenar modelos multicapa, Back Propagación se compone de 6 fases principales:

1. Elección de entrada y salida.
2. Configuración.
3. Cálculo de error.
4. Minimización de errores.
5. Actualización de parámetros.
6. Modelado para la predicción.

Es un método de Gradientes, que implementa la regla Delta que compara el resultado deseado con el resultado real (δ = ai(deseado) - ai(observado)). Se suele calcular el error cuadrático medio y se utiliza para su ponderación más precisa cuando se retroalimenta a la red.

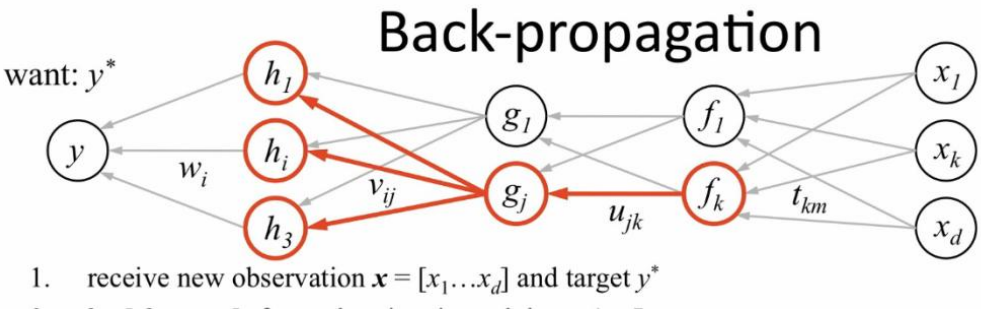


Ilustración 19 (Sushma 2023) https://www.linkedin.com/pulse/neural-network-back-propagation-sushma-krishnamurthy

Overfitting y Underfitting

Overfitting y Underfitting, son los principales y comunes problemas en Machine Learning (Dev, 2024), se tienen como la principal causa de obtener malos resultados, se pueden traducir del inglés **Overfitting** y **Underfitting** como “Sobre-Ajuste” y “Sub-Ajuste”. (NA8, 2017), el Overfitting y Underfitting ocurren durante el entrenamiento de Machine y Deep-Learning, esto debido a un bajo rendimiento del modelo, (THE 365 TEAM, 2023).

Overfiftting

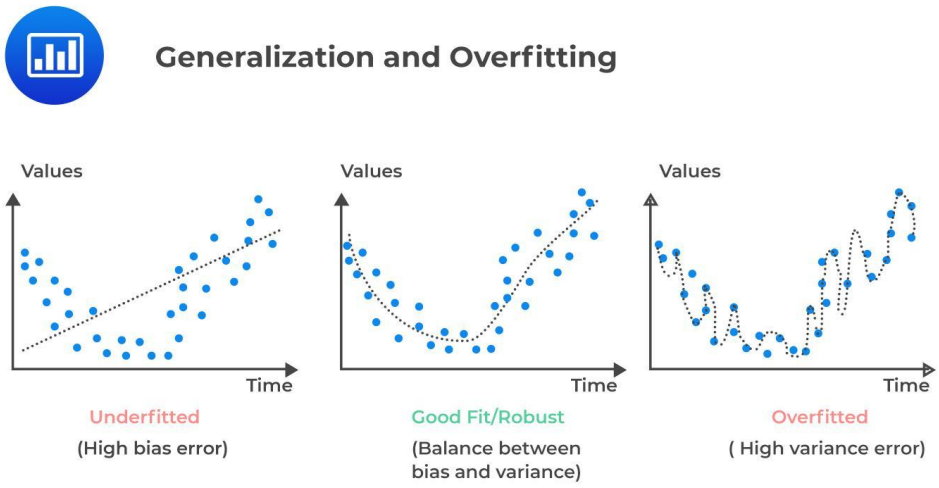
El Overfitting o **Sobre ajuste**, se produce cuando el modelo es complejo y se ajusta demasiado a los modelos de entrenamiento, se podría entender como la manera que nuestro modelo se enfoca tanto en los datos de entrenamiento, que no permite adaptarse a un nuevo aprendizaje (THE 365 TEAM, 2023), de tal modo que el modelo sobre analiza los datos y tiende a memorizar los patrones de entrenamiento, para comparar los datos de entrada (Navarro, 2024)

Underfitting

Underfitting o sub entrenamiento, se tiene que este error ocurre, cuando no se tienen los suficientes datos, para el entrenamiento del modelo.

Solución a la problemática de Underfitting y Overfitting.

Se debe de encontrar un equilibrio en donde el modelo de Machine Learning DeepLearning, no incurra en overfitting o underfitting, para evitar esta problemática se encontraron diversas soluciones entre ellas, es dividir el modelo donde una parte del Dataset se envía a entrenar y la otra parte que es desconocida, se envía para test o train (entrenamiento), del modelo y poder ajustar los valores, para optimizar el modelo de machine learning.



Usualmente el modelo utilizado es 80/20 en donde 80% es el porcentaje de datos a entrenar y 20% es el porcentaje de datos a probar (Test).

Ilustración 20 (Fuente Trail, 2025) recuperado de : https://es.sourcetrail.com/pit%C3%B3n/pi%C3%B1%C3%B3n/overfitting-vs-underfitting-gu%C3%ADa-completa-con-se%C3%B1ales-causas-y-soluciones/

Como se puede apreciar en la figura 20 se tienen 3 ejemplos Underfitting, el modelo es muy simple por lo cual, no se logra el objetivo, para evitarlo se requiere aumentar mas el dataset, para el Overfitting, es un sobre entrenamiento, memoriza cada punto y este modelo predice resultados deficientes, con muy alta varianza, para el modelo robusto es el de en medio mantiene un buen sesgo y varianza, de igual manera existen muchos ejemplos aconsejables para evitar el Overfitting.

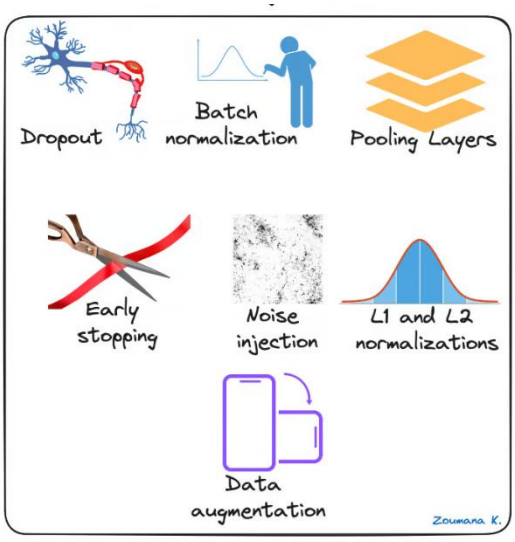


Ilustración 21 Consejos para evitar el Overfitting (Zoumana, 2024) recuperado : https://www.datacamp.com/es/tutorial/introduction-to-convolutional-neural-networks-cnns

Red neuronal convolucional

Las redes convolucionales, conocidas como Convolutional Neural Networks (CNNs), son una clase de redes neuronales profundas que han mostrado un rendimiento sobresaliente en tareas de reconocimiento de imágenes y procesamiento de señales. (Goodfellow, Bengio, & Courville, 2016). Para la red neuronal convolucional existen ciertos componentes los cuales son esenciales en este diseño de IA.

La red convolucional está inspirada en cómo funciona el córtex visual del cerebro de los organismos biológicos, las redes convolucionales permiten extraer patrones y características de las imágenes (KEITA, 2024)..

Como se puede apreciar en la **Ilustración** el cerebro humano existen diferentes áreas las cuales se encargan del registro de datos visuales, de tal manera que existe un córtex dedicado a patrones, líneas y todas estas funciones se unen a un final córtex para producir una imagen final.

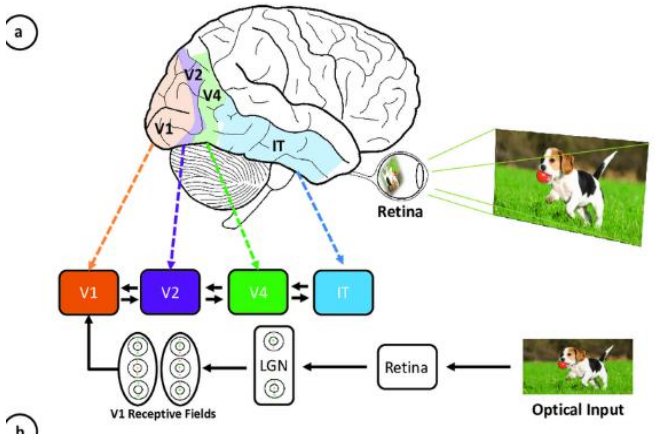


Ilustración 22 Vision computarizada (García, 2023) recuperado de: https://www.researchgate.net/publication/382882639/figure/fig3/AS:11431281269742886@1722905715790/below-illustrates-the-layered-architecture-of-the-human-visual-cortex-which-served-as.ppm el 15 de Julio

Neuronas complejas

La capa más compleja o capa V2, se encarga de extraer los patrones de las capas sencillas, estas se encargan de la detección de diferentes formas, y estas se van especializando en base a las conexiones de las capas anteriores (Simples), estas pueden agrupar mejor la información para encontrar formas (Sotaquirá, 2019).

Agrupación

El cerebro agrupa primero características como Líneas tanto verticales y horizontales y después las neuronas complejas agrupan formas y figuras, para después unirlas y poder dar un resultado de lo que se está viendo.

En la capa V4, las neuronas, ya tiene la capacidad de detectar figuras, en el caso de los rostros, tales como narices, ojos, cejas, bocas, (Sotaquirá, 2019). hasta llegar a una combinación en donde se tiene la reconstrucción de todos estos filtros teniendo como resultado una imagen (Ringatech, 2021).

El sistema de visión biológico para poder comprender lo que se está viendo, adiciona más neuronas simples y complejas hasta formar una imagen.

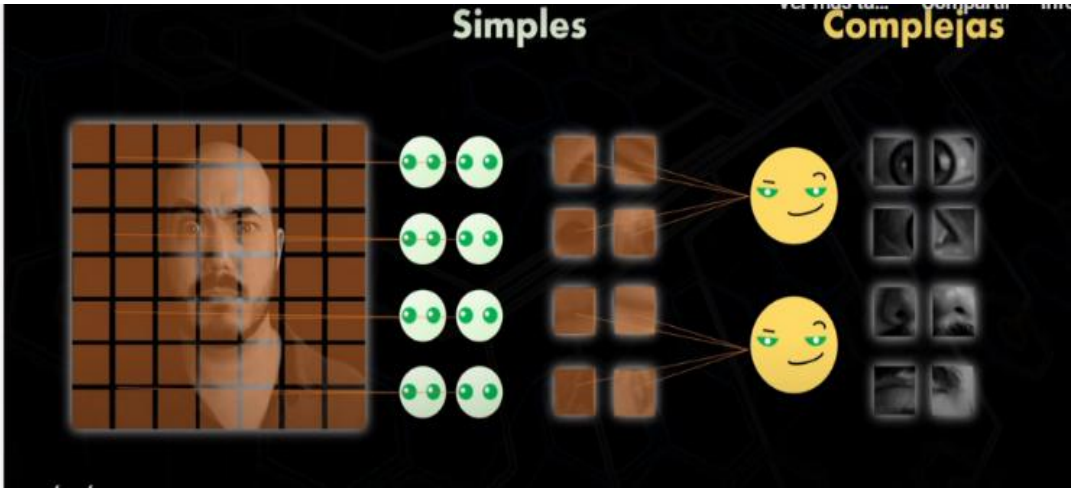


Ilustración 23 Ringatech Redes convolucionales https://www.youtube.com/watch?v=4sWhhQwHqug

¿Cómo logran las redes convolucionales extraer las características?

Las redes convolucionales, pese a tener estos principios biológicos estas a diferencia de los humanos, lo que ven son pixeles, por lo cual para poder ellas extraer las características de una imagen y poder comprender si esta se trata de un borde, líneas, curvas ETC, se aplica una técnica matemática llamada **convolución**.

La **convolución** es una técnica matemática, que se utiliza usualmente para el tratamiento de señales, se trata de una operación matemática que permite la combinación de dos señales, en otras materias como tratamiento de sistemas digitales de señales se emplea para conocer que le pasara a una señal al pasar por un dispositivo o filtro.

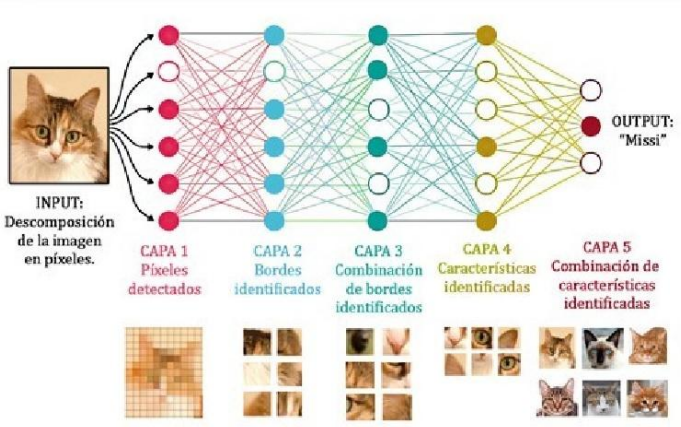


Ilustración 24 Convolucion completa https://digitalia.home.blog/2025/01/13/redes-neuronales-convolucionales-el-poder-detras-del-reconocimiento-de-imagenes/

Capas convolucionales

Las capas convolucionales aplican filtros (kernels) sobre la entrada para generar mapas de características. La operación de convolución permite a la red extraer características locales y aprender patrones importantes en los datos (LeCun, Bengio, & Hinton, 2015).

- Filtros/Kernels: Son matrices de pesos que se entrenan para detectar características específicas en la entrada, como bordes y texturas.
- Stride: Define el paso con el que se mueve el filtro sobre la entrada.
- Padding: Añade bordes a la entrada para controlar el tamaño del mapa de características resultante (O'Shea & Nash, 2015).

El proceso de convolución se encarga de revisar una imagen pixel por pixel, pero tomando en cuenta los pixeles de alrededor, de esta manera es que al pasar un filtro de convolución por una imagen, puede hacer una comparativa según el núcleo del filtro de convolución y se pueden tener diferentes resultados.

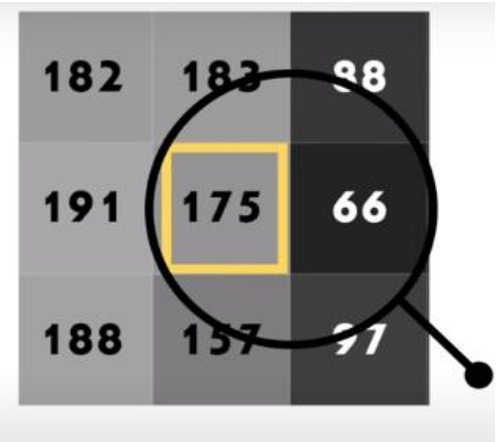


Ilustración 25 RINGATECH 2021 RECUPERADO DE :https://www.youtube.com/watch?v=4sWhhQwHqug

Núcleo o kernel de convolución

Para poder extraer las características de una imagen se emplea lo que se le conoce como núcleo o filtro, los valores del filtro son valores definidos por el programador, al igual que cuanto espacio se podrá mover, el núcleo puede ser de un tamaño de 3x3

como se puede apreciar en la figura de arriba, o directamente de valores mayores, se utiliza en base a la información que se requiere obtener o filtrar.

El filtro Multiplica los valores que contiene en su interior con los valores de la imagen así logrando obtener un resultado en donde se pueden extraer características y se puede reducir una imagen hasta su estado más eficiente.

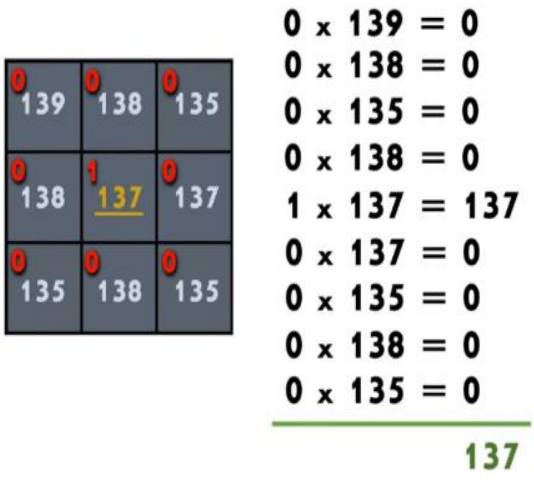


Ilustración 26 Ringatech 2021 recuperado de : Capas convolucionales

Como se puede apreciar en la figura 26, el núcleo contiene valores con los cuales ira multiplicando la imagen, y sumando los resultados, al realizar esta acción el kernel o núcleo, se puede desplazar por toda la imagen realizando operaciones matemáticas pixel por pixel, para poder extraer los datos más significativos o transformar los datos una vez pasado el filtro.

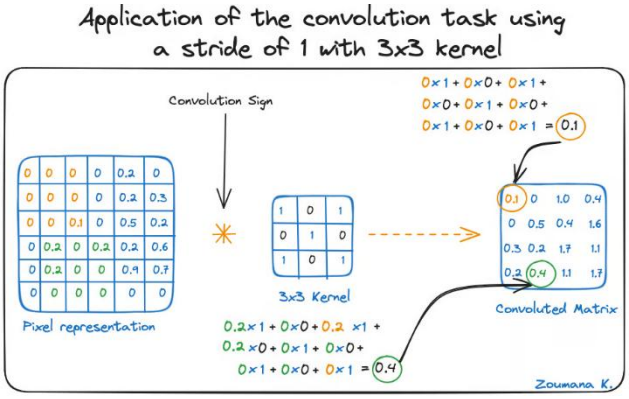


Ilustración 27 Convolucional Filtros: (Udemy 2025) recuperado de: https://www.datacamp.com/es/tutorial/introduction-to-convolutional-neural-networks-cnns

En la figura 27 se puede apreciar el movimiento del kernel para filtrado, y el resultado de este mismo en una imagen reducida.

Filtros de convolución

Al variar el contenido en el kernel se pueden obtener diversos resultados, resultados como la misma imagen, enfoque, un degradado, sobreexposición, entre otros efectos.

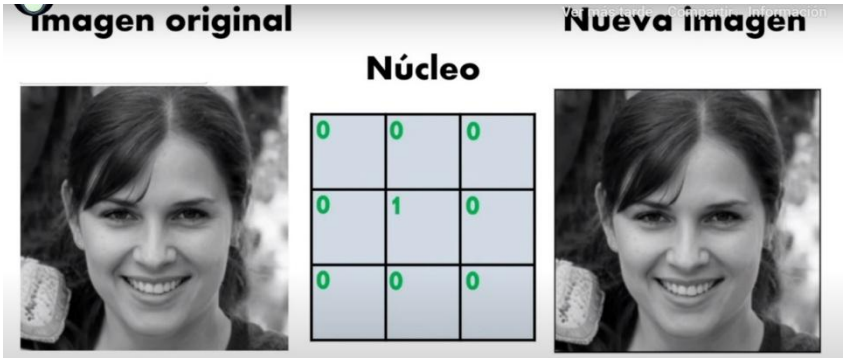


Ilustración 28 FILTROS DE CONVOLUCION RINGATECH 2021 recuperado de: https://www.youtube.com/watch?v=4sWhhQwHqug

Como se puede apreciar en la ilustración se tiene que al pasar un filtro con valor en el centro 1, el resultado es el mismo, esto dado a que, si se mueve el filtro, pixel por pixel se logra obtener el mismo resultado.

Para la red neuronal lo que más interesa es encontrar, los bordes, líneas y figuras como se puede apreciar en la siguiente ilustración, todo esto se logra mediante los filtros, también se pueden obtener otros resultados.

Ilustración 29 Filtro de convolución Bordes Ringatech 2021 recuperado de: https://www.youtube.com/watch?v=4sWhhQwHqug

Ejemplo de filtro

A continuación, se muestran el anteriormente mencionado con una prueba hecha por nosotros. A la izquierda se observa la imagen de entrada en escala de grises y a la derecha la imagen filtrada.

Ilustración 30 filtro de convolución prueba local

Ahora se muestra otra imagen filtrada y en escala de blancos y negros para la obtención de líneas verticales y horizontales las cuales son las que permiten determinar la forma del objeto o persona a detectar en cuestión.

Ilustración 31 Filtro de convolución

Elementos para filtraje de redes convolucionales

Padding

El padding es una propiedad para adicionar 0 en los espacios en donde no existe información (espacios vacíos por los filtros de convolución) , esto con el objetivo de

mantener el mismo tamaño de la imagen y que esta no se vea reducida(Sotaquira, 2019) .

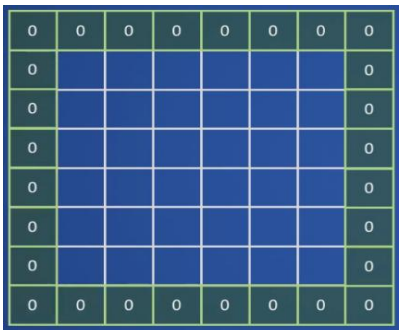


Ilustración 32 Padding funcionamiento de una red convolucional

Como se puede apreciar en la imagen este es el resultado de pasar por un filtro de red convolucional, la imagen se reduce, pero aumentan los filtros, si se desea mantener el mismo tamaño de imagen de entrada con salida entonces se debe de usar padding.

Strides

Los strides son los movimientos que realiza el kernel entre cada iteración, movimientos que puede ser laterales o de arriba abajo, el valor de stride (salto) determina cuantas posiciones se puede mover el filtro de convolución, por ende puede determinar el tamaño final de la imagen resultante (Rodríguez, s.f.).

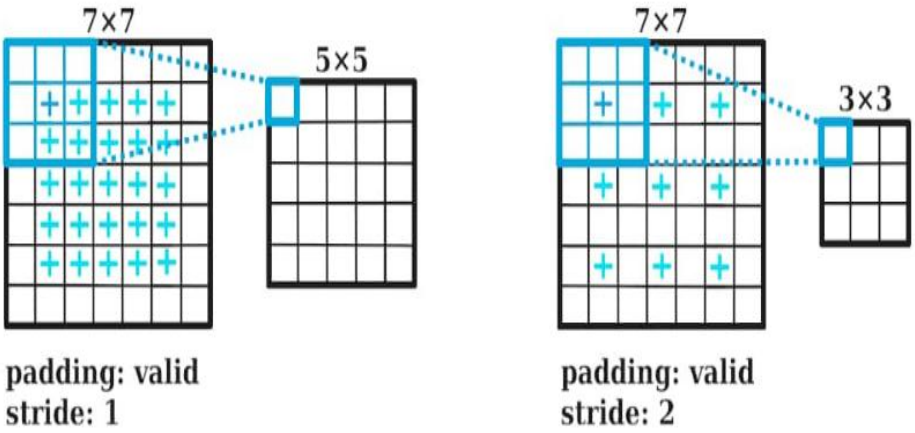


Ilustración 33 Strides padding

Capas de pooling

Las capas de pooling reducen la dimensionalidad de los mapas de características, disminuyendo la complejidad computacional y destacando las características más importantes. Los tipos más comunes son:

- Max pooling: selecciona el valor máximo en cada ventana de la entrada
- Average pooling: calcula el valor promedio en cada ventana de la entrada.

Max pooling

Es una técnica que se aplica de submuestreo el cual su tarea es reducir la resolución de los datos de entrada por ejemplo, si se tiene una imagen a tratar una imagen de 4x4, si se aplica un maxpooling de 2x2, se tendrá una imagen resultante de 2x2.

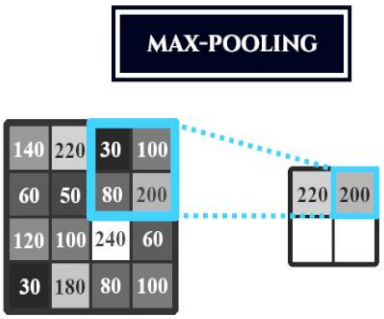


Ilustración 34 Max-Poooling Ejemplo grafico

Por otro lado, si se tuviera el mismo tamaño de Maxpooling de 2x2 en una imagen 6x6, el resultado seria 9 ventanas o filtros obtenidos de 2x2.

Ilustración 35 Maxpooling con tamaño 2x2

Stacking

Es el apilamiento de imágenes resultantes proveniente de los filtros aplicados de convolución. Por ejemplo:

Si se tiene una imagen de entrada de 6x6, y se usan 10 Kernels de 3x3, con strides de 1, sin padding, se obtendrá el siguiente resultado:

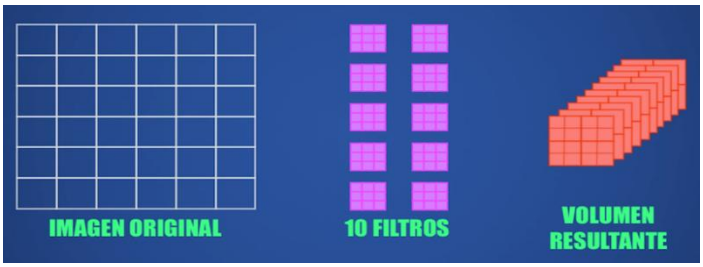


Ilustración 36 Explicación del Stacking

Raspberry pi

La Raspberry Pi es una serie de microcomputadoras de bajo costo y tamaño reducido desarrolladas por la Fundación Raspberry Pi en el Reino Unido. Estas computadoras están diseñadas para promover la enseñanza de ciencias de la computación y la programación básica en las escuelas y en los países en desarrollo. Desde su lanzamiento inicial en 2012, las Raspberry Pi han ganado popularidad entre los entusiastas de la electrónica y la programación debido a su versatilidad, bajo costo y la gran comunidad de usuarios y desarrolladores que las respaldan.

Características Clave de la Raspberry Pi

Hardware

- Procesador: Utiliza procesadores ARM; la potencia varía según el modelo, siendo la Raspberry Pi 4 uno de los modelos más recientes que tiene un procesador de cuatro núcleos.
- Memoria RAM: Varía según el modelo: las versiones más tempranas tenían 512 MB, y las más recientes tienen 8 GB.
- Almacenamiento: Utiliza una tarjeta microSD para almacenamiento principal; esa elección ofrece flexibilidad y facilita la actualización.

- Puertos: Incluyen puertos USB (2.0 para modelos más tempranos, GBN en las versiones posteriores), HDMI para video, GPIO para conectar con dispositivos electrónicos, Ethernet, puerto de afonía de 3.5 mm.

Tensorflow

Tensorflow es una librería basada en Python para el desarrollo de aprendizaje de maquina o Machine Learning y aplicaciones de redes neuronales. Tensorflow es un sistema creado por el equipo de Google Brain liberado en el año 2015. Es una librería open source para computación numérica y larga escala de aprendizaje automático o Machine learning. Tensorflow agrupa una gran cantidad de modelos y algoritmos de aprendizaje automático y aprendizaje profundo o **Deep learning**.

Tensorflow es un **Framework** y este a su vez compite con otros Frameworks de programación de machine Learning como: Pytorch, Caffe y Apache MXNet los cuales pueden entrenar y correr redes neuronales para:

- Clasificación de Escritura a mano.
- Reconocimiento de imágenes
- Lectura de palabras.
- Redes neuronales recurrentes
- Modelos de ML para traducción secuencia a secuencia.

Actualmente Tensorflow es la librería de aprendizaje profundo más famosa del mundo, los productos de Google usan Machine learning como mejoras para búsquedas, traducción, recomendaciones y subtitulación de imágenes.

Tensorflow recibe su nombre debido a que su principal función es el procesamiento y adecuación de datos Multidimensionales, los cuales también son conocidos como **tensores,** por lo cual cuando se trata de modelos de Machine Learning, la sangre o la vida de estos se encuentran en estos mismos datos multidireccionales que se les conoce como sets de datos (Navarro, 2024).

Tensorflow 2.0

Al pasar el tiempo, los modelos de Tensorflow se hacían cada vez más complicados de crear y entender, por ello Tensorflow 2.0 se diseñó con el objetivo de facilitar la

creación de redes neuronales para el aprendizaje automático, por ello Tensorflow 2.0 implementa una Api, llamada Keras.

Bases de datos

Una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica, en un sistema informático *(Oracle),* normalmente una base de datos está controlada por un sistema de gestión de bases de datos (DBMS) (Data Base Managment System) o sistemas de gestión de bases de datos.

.

Ilustración 37 Imagen alusiva a base de datos (UNKNOW,2018) recuperado de: https://basesdedatosextructuraymodelos.blogspot.com.

Las bases de datos han sido uno de los avances de almacenamiento de información y recopilación más grande que se ha tenido con el avance de la computación, dado a que las empresas ya no deben de realizar todo a mano como se realizaba antes, ahora pueden ingresar, editar, eliminar la información de manera inmediata y precisa usando los administradores de bases de datos.

Dentro de los datos existen múltiples tipos de datos de diferente naturaleza, por ejemplo: Datos Numéricos, Imágenes, Fechas, Textos, Monedas, Booleanos, en base a este tipo de datos es que se dividen las estructuras y los métodos a trabajar con ellos.

Los 3 Tipos de Datos

Datos Estructurados:

Los datos Estructurados son todos aquellos que tienen un formato definido como tabla, filas, columnas, son los datos de mayor organización, se ajustan a las tablas e incluyen datos discretos, como números, fechas, texto.

Datos Semi Estructurados

Los datos semi estructurados son todos aquellos que no se ajustan a la estructura formal de los modelos de base de datos relacionales(No encajan en una tabla de filas y columnas) (telefonia tech, s.f.), sin embargo contienen etiquetas u otros marcadores para separar a los elementos y poder identificarlos. Los datos no estructurados se producen cada vez más desde la llegada de internet, dado a que ya no solamente se tienen datos como textos y las bases de datos ya no son la única forma de datos (writer, 2022).

Datos No estructurados

Los datos no estructurados son todos aquellos que se definen como presentes, pero están sin estructurar, son todos aquellos que no se han organizado en un formato que facilite el acceso y procesamiento, ejemplo de ellos, las publicaciones en redes sociales, los videos de vigilancias, los documentos de las empresas (AWS, 2023).

Bases de Datos Relacionales y No Relacionales

Como se pudo observar las bases de datos tienen múltiples datos y en base a ellos es que se crean estrategias o métodos, para trabajar dichos datos, de ellos nacen dos metodologías, las BD Relacionales y las BD No Relacionales:

BD Estructuradas o BD MSQL

Se tiene que el modelo de Bases de Datos Relacionales o MYSQL, almacena los datos en forma tabular de Filas y Columnas, las columnas contienen atributos de datos, mientras que en las filas hay valores de datos. Las tablas se pueden vincular a otras para obtener o maximizar la información acerca de un tema, dentro de sus características se tiene que cada tabla se le asigna una clave principal, esta clave es única y exclusiva estas claves sirven para relacionar estas tablas con otras.

Bases de Datos No relacionales

Las bases de datos no relacionales, son bases de datos diseñadas específicamente para modelos de datos, almacenados en esquemas flexibles, que tienden a la escalabilidad. Para aplicaciones modernas, las bases de datos no relacionales o NoSQL son reconocidas porque son fáciles de desarrollar (AWS AMAZON, 2023).

Las bases de datos NoSQL son ideales para todos aquellos modelos que requieran flexibilidad para datos no estructurados y semi estructurados, poseen una gran escalabilidad dado a que están diseñados para tener un crecimiento horizontal, las bases de datos no relacionales proveen o proporcionan Apis altamente funcionales y tipos de datos diseñados para sus respectivos modelos de datos.

MongoDB

MongoDB es un sistema de gestión de bases de datos (DBMS) no relacional de código abierto que utiliza documentos flexibles en lugar de tablas y filas para procesar y almacenar diversas formas de datos. (IBM, s.f.)

MongoDB soporta el tipo de replicación primario-secundario. Cada grupo de primario y sus secundarios se denomina replica set. El primario puede ejecutar comandos de lectura y escritura. Los secundarios replican los datos del primario y sólo se pueden usar para lectura o para copia de seguridad, pero no se pueden realizar escrituras. Los secundarios tienen la habilidad de poder elegir un nuevo primario en caso de que el primario actual deje de responder.

Ilustración 38 Logo Mongo DB https://www.mongodb.com/company/newsroom/press-releases/mongodb-launches-five-new-capabilities-for-mongodb-atlas-to-build-new-classes-of-applications

Ventajas de MongoDB

MongoDB ha sido una opción bastante útil para muchas empresas que han optado por una base de datos no relacional que sea fácil de manejar, pero sobre todo que buscan un sistema el cual sea completamente escalable.

Se mencionan algunas de las características que hacen de MongoDB un database confiable:

- Equilibrio de carga: El proceso de uso compartido del equilibrio de carga de MongoDB distribuye grandes conjuntos de datos a través de varias máquinas virtuales a la vez manteniendo rendimientos aceptables de lectura y grabación. Esta escalada horizontal se llama *sharding* (fragmentación) y ayuda a las organizaciones a evitar el coste de la escalada vertical de hardware al mismo tiempo que expande la capacidad de los despliegues basados en cloud.
- Soporte a múltiples lenguajes: una de las principales ventajas es que MongoDB es bastante amigable para ser usado por muchos lenguajes de programación. Esto incluye C++, C, JS, Python, etc.

Redes siamesas

Una red neuronal siamesa (a veces llamada red neuronal gemela) es una red neuronal artificial que utiliza los mismos pesos mientras trabaja en conjunto en dos vectores de entrada diferentes para calcular vectores de salida comparables (Chicco, 2020).

Básicamente, estas redes neuronales convolucionales contienen dos entradas las cuales ambas son redes neuronales simples; es decir, cada una es una entrada y el objetivo por el cual se usan es para evaluar la “distancia” entre ambas. Esto significa que evalúa la diferencia, es decir, la resta entre ambas redes y en base al valor obtenido se determina la similitud de las dos entradas.

A continuación, se muestra una imagen que detalla específicamente que es lo que hace, las entradas, el diferencial y la salida.

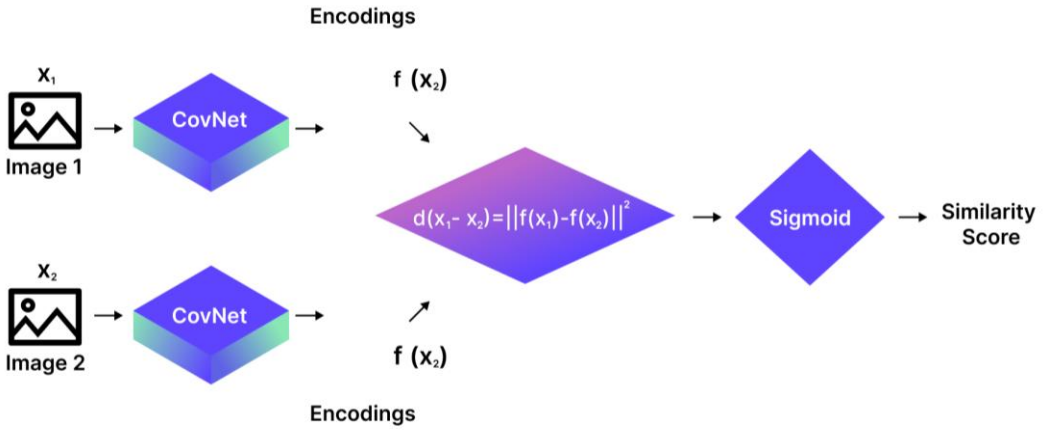


Ilustración 39 Red siamesa (Encord, 2025) recuperado de: [One-Shot Learning in AI - Definition and Examples | Encord]

Los usos de las medidas de similitud en los que se puede utilizar una red gemela son cosas como el reconocimiento de cheques escritos a mano, la detección automática de rostros en las imágenes de la cámara y la coincidencia de consultas con documentos indexados.

La aplicación quizás más conocida de las redes gemelas es el reconocimiento facial, en el que las imágenes conocidas de las personas se calculan previamente y se comparan con una imagen de un torniquete o similar.

El aprendizaje en redes gemelas se puede realizar con pérdida de triplete o pérdida contrastiva. Para el aprendizaje por pérdida de tripletes se compara un vector de referencia (imagen de anclaje) con un vector positivo (imagen verdadera) y un vector negativo (imagen falsa). El vector negativo forzará el aprendizaje en la red, mientras que el vector positivo actuará como un regularizador. Para el aprendizaje por pérdida contrastiva debe haber una caída de peso para regularizar los pesos, o alguna operación similar como una normalización.

Ventajas

- Esta red neuronal permite mejor velocidad y precisión en comparación a otras redes neuronales cuando se trata de grandes volúmenes de datos.
- No es necesario entrenar intensamente la red nuevamente para detectar la presencia de nuevas clases. Caso contrario ocurre con las otras redes neuronales las cuales necesitan un nuevo entrenamiento cada que hay nuevas clases.

- Mejor obtención de resultados a nivel general puesto que los datos de entrada son similares, pero no necesariamente idénticas.

Existen dos modelos importantes de redes siamesas (Educative, 2025)

- One shot learning
- Zero shot learning

One-Shot learning

El One-Shot learning es un modelo de red siamesa la cual se basa en la evaluación de similitud y diferencia entre dos imágenes mediante un algoritmo de clasificación. El objetivo del aprendizaje de una sola vez es enseñar al modelo a establecer sus propias suposiciones sobre sus similitudes en función de una cantidad mínima de elementos visuales. Sólo puede haber una imagen (o un número muy limitado de ellas, en cuyo caso se suele denominar aprendizaje de pocas tomas) para cada clase.

Estos ejemplos se utilizan para construir un modelo que luego puede hacer predicciones sobre otros elementos visuales desconocidos. (Logunova, 2022)

No se puede subestimar la importancia del aprendizaje único en el ámbito de la IA. Este enfoque transforma fundamentalmente el panorama del aprendizaje automático al mitigar la dependencia de conjuntos de datos voluminosos, acelerando así el proceso de capacitación y mejorando la adaptabilidad de los modelos de IA en diversos escenarios. Además, el aprendizaje de una sola vez tiene un inmenso potencial para permitir que los sistemas de inteligencia artificial obtengan conocimientos y hagan predicciones precisas, incluso en casos en los que prevalece la escasez de datos.

Aplicaciones

Se han utilizado algoritmos de aprendizaje de una sola vez para tareas como clasificación de imágenes, detección y localización de objetos, reconocimiento de voz y más. (Logunova, 2022)

Entre las aplicaciones más importantes destaca el uso para el reconocimiento facial usado en distintos ámbitos tanto a nivel empresarial, como a nivel público, también en aeropuertos para la detección de personas mediante sus pasaportes y así validar.

El aprendizaje de una sola vez es esencial para la visión por computadora, en particular para que los drones y los automóviles autónomos reconozcan objetos en el entorno.

Otra área es el reconocimiento de palabras en varios idiomas, donde se aplica el aprendizaje de una sola vez para identificar palabras desconocidas en el idioma de traducción. (Logunova, 2022).

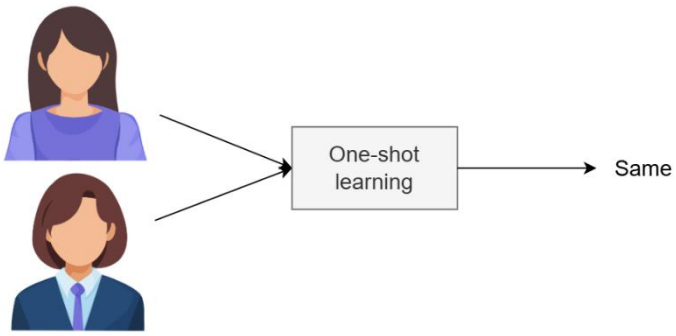


Ilustración 40 (Educative 2025) recuperado de: https://www.educative.io/answers/what-is-one-shot-learning

Zero Shot Learning

El Zero Shot Learning es un modelo de Machine Learning en el que se entrena al modelo de Inteligencia Artificial, para reconocer y categorizar objetos o conceptos sin haber visto ningún ejemplo, este tipo de modelo es practico donde no se requieren modelos supervisados (Bergmann, 2025).

Análisis y presentación de resultados

Diseño metodológico

Enfoque

El análisis de este trabajo monográfico fue de índole mixta; es decir, se hizo mediante un proceso analítico cuantitativo y cualitativo.

Cuantitativo puesto que se hacen muchos análisis numéricos, sobre todo en los datos obtenidos a nivel de similitudes para determinar exactitud y/o diferencias al momento de comparar las imágenes. Esto es realizado mediante la red convolucional siamés la

cual compara 2 imágenes de entrada, realiza la convolución para obtener la diferencia y así determinar el nivel de similitud.

Fue cualitativo puesto que también se pueden obtener datos alfabéticos durante el proceso de reconocimiento facial y en la detección de placas se pueden obtener datos tales como: true, false, nombres de las personas, etc. Esto es posible debido al análisis mediante la base de datos donde está toda la información. Al momento de realizar la convolución y determinar el nivel de similitud es posible que se logre hacer un print donde determine la persona que ha detectado.

Este enfoque mixto está también relacionado con los objetivos que se han planteado por alcanzar.

Para la codificación no es simplemente la escritura de líneas de código mediante la implementación de librerías, métodos y submetódos. En este caso se está implementando Python el cual es un lenguaje de programación orientada a objetos. También es necesario comprender a nivel matemático lo que es una convolución, la suma de los valores en cada neurona, funciones diferenciales para la determinación de niveles de similitud entre las personas a comparar mediante imágenes y más funciones realizadas por la Convolutional neural Network (CNN).

Diseño de investigación

Para este trabajo se empleó el **estudio de caso** como el diseño de investigación.

El motivo por el cual se ha trabajado de esta manera es debido a que se trata de la propuesta de un sistema de control de acceso de personas y vehículos. Esto se logra mediante la implementación de un prototipo de sistema de reconocimiento usando la tarjeta de desarrollo embebido Raspberry Pi. Este diseño va enfocado en un proceso analítico exhaustivo de un caso complejo de analizar con simples datos, se requiere de gran tiempo para la recolección de información, recolección de datos, análisis matemático mediante la utilización de datos para analizar en tiempo real y con imágenes en la base de datos implementada.

Población

La población se basa en los estudiantes de la universidad nacional de ingeniería, la cantidad de estudiantes los cuales estén ingresados en nuestra base de datos y con

los cuales se realice el entrenamiento va a ser determinante al momento de realizar la predicción y el entrenamiento de la misma red

Muestra

El modelo de la red Siamés cuenta con una muestra de 2 a 6 personas, de las cuales en la base de datos se cuenta con únicamente dos estudiantes de la UNI.

El tamaño de la muestra seleccionada es debido a que al probar los códigos en dos computadoras se determinó que para este prototipo de pruebas es más óptimo realizar el análisis con una muestra inicial no tan grande, pero tampoco tan pequeña. Al tener más muestras es necesario darle más ciclos de aprendizaje al modelo (epochs), eso significa mayor consumo de recursos, mayor tiempo de aprendizaje del modelo y finalmente mayor tiempo para la detección de las personas. Para un modelo muy pequeño que requiera de manera extrema pocos datos, se puede implementar el modelo de 10 imágenes, pero para un modelo normal, lo recomendables de 50 a 250 imágenes para entrenar.

Procedimientos y técnicas de análisis de datos

Durante este proceso de elaboración de este trabajo monográfico se ha recurrido a múltiples plataformas, base de datos, servidor, lenguaje de programación Python y la codificación de los scripts los cuales se han utilizado para el análisis de los datos y la obtención de los resultados.

A continuación, se inicia el proceso detallado de todo lo realizado, así como los softwares usados:

Tensorboard

Tensorboard es una herramienta visual de Tensorflow el cual permite poder observar en tiempo real el comportamiento vía Graficas de los entrenamientos, permite tomar registro de numero de épocas, permite predecir si hubo errores al entrenar al igual que llevar control de la perdida. Es una herramienta profesional utilizada por grandes desarrolladoras y preferida por todas sus bondades al igual que por escalabilidad y predicción para diferentes modelos de Machine Learning y Deep Learning.

Servidor

Para poder acceder al servidor se tiene que hacer uso de la herramienta de Mongo-Compass.

Machine learning

Para completar la tarea de reconocimiento de placas y reconocimiento de rostros la programación habitual no podría acaparar este reto, no se lograría mediante las técnicas habituales de múltiples lenguajes de programación. Las técnicas de programación secuenciales, condicionales y cíclicas no permiten, por ejemplo, extraer la información de un rostro y decir si se trata de una persona, es por esto que se implementó un modelo de machine learning para poder completar esta tarea.

Modelo de Red Neuronal

El modelo implementado son las redes neuronales, estos modelos permiten un aprendizaje parecido al cerebro humano; El modelo de machine learning que se eligió fue la red neuronal convolucional gemela o siamesa que permite extraer características y datos biométricos sin necesidad de indicar qué o cuáles, simplemente mediante ejemplos. La red convolucional es la red por excelencia y la piedra angular de la visión computarizada, por ende, se usó este modelo para este proyecto.

Para poder comparar un rostro con otro y hacer un correcto Face Recognition se validó, evaluó y se tomó el modelo de One-shot Learning, el cual es uno de los modelos que permite validar información con solo una imagen, dentro del tipo de modelo de One-shot Learning.

Capítulo I: Desarrollo de prototipo de Reconocimiento de Rostros y Placas Vehiculares.

1.1 Diagrama de bloque Face y Plate Recognition

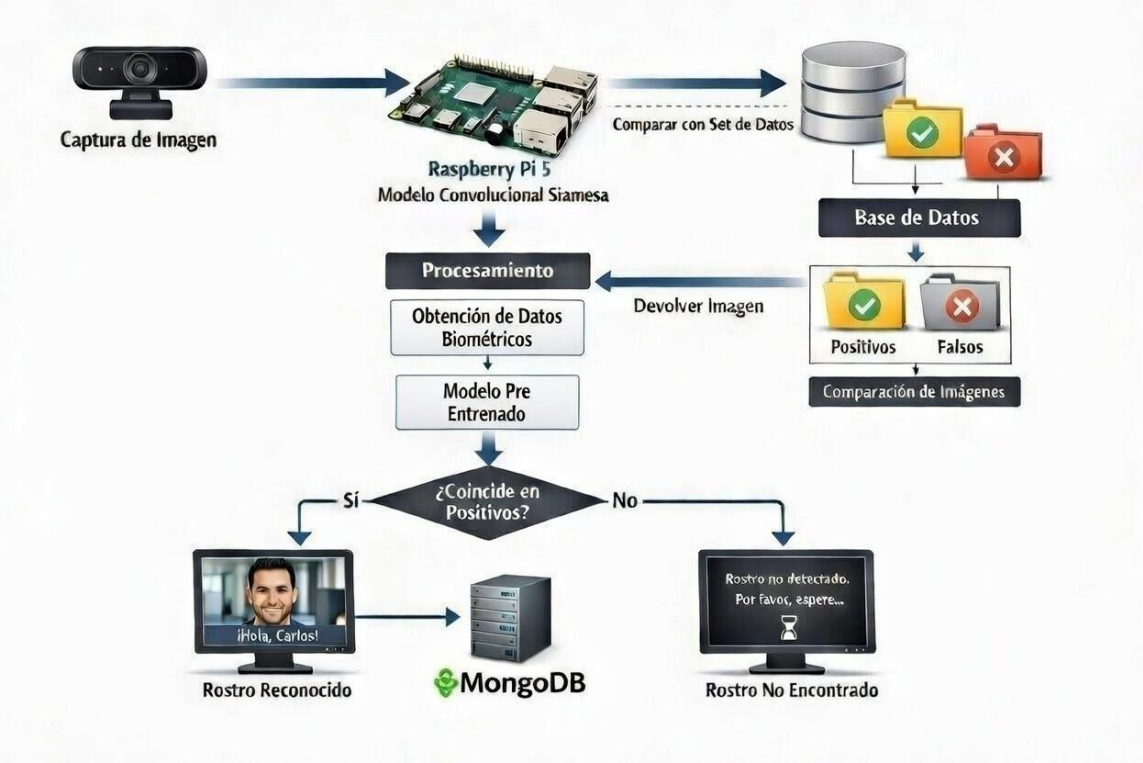


Ilustración 41 Diagrama de bloques del proceso de reconocimiento de rostros y Placas

Como se puede ver en la **ilustración 41** se tiene el diagrama de bloques del proyecto donde primero se tiene la cámara la cual captura tanto los rasgos biométricos como las placas vehiculares, para posterior ingresar en la Raspberry la cual ya con un programa (modelo preentrenado), compara con una base de datos local y toma la decisión sobre si coincide la imagen de entrada con la imagen guardada, en caso de coincidir envía la información al servidor Mongo DB, quien también guarda los registros en su BD de los casos positivos, en caso de no coincidir se envía una alerta de Rostro o placa no encontrado y se genera un reporte de Rostro no encontrado o Placa no encontrado en Mongo DB.

Se estará abordando en los siguientes capítulos, la programación para crear el modelo de entrenamiento de inteligencia artificial (Machine Learning) para detección y reconocimiento de Rostros y Placas al Igual que las BD y Servidor implementado.

1.2 Camara

La cámara que se utilizó para este proyecto es la Joy-DocCam la cual es un modelo de cámara articulada que permite ajustar manualmente la distancia, orientación y luz, lo cual permitió una enorme facilidad para el desarrollo de este proyecto, dado a que una cámara fija, u otra cámara no cuenta con todas estas prestaciones.

Modelo: JoyDocCam V500S Voltaje: 5V Intensidad = 2A.

Ilustración 42 Camara Joy-DocCam Imagen (Imagen Extraída de la Red E imagen tomada localmente)

Esta cámara cuenta con una resolución de 3256 x2440p a 30 fps, con zoom digital x100, esta cámara es útil para el proyecto por todas las prestaciones previamente mencionadas, pese a ello, la resolución no es determinante ni detonante para un proyecto de inteligencia artificial, pues la red de convolución solamente usa (105x105) pixeles, los cuales deben ser cargados desde la cámara. Cabe destacar que si bien lo mencionado previamente es cierto, también una cámara con mayor sensibilidad y mayor entrada de luz mejora de sobre manera los resultados tanto de Input como de comparación en la Red Neuronal.

1.3 Raspberry Pi

Para el proyecto de Detección, Reconocimiento, Base de datos y Servidor se decidió hacer uso de la tarjeta de desarrollo Raspberry pi 5 dado a que este modelo es uno de los más potentes que ha lanzado Raspberry al igual que su amplia compatibilidad con Tensor Flow el Principal Framework usado para esta programación, al igual que el gran alcance que tiene en su Software Debian 12 BookWorn (Software Actual instalado).

Ilustración 43 Raspberry Pi 5 Foto tomada localmente

Como se puede apreciar en la **Ilustración 43**, uno de los motivos por el cual se escogió este modelo, además de su capacidad, es la compatibilidad con módulos. Para este proyecto se le adicionaron los siguientes módulos:

- Módulo de enfriamiento,
- Modulo adaptador SSD NVME M.2,
- Case como se puede ver en la ilustración

Dichos módulos se adicionaron para proteger el equipo ante daños físicos en el caso del case al igual que para enfriar el dispositivo mientras trabaje procesando el proyecto, posterior se adiciono el módulo de NVME el cual permite instalar una tarjeta de memoria SSD; para el proyecto se usó una SSD de 256GB para almacenar tanto las bases de datos, los modelos Pre-entrenados al igual que los programas y dependencias del proyecto.

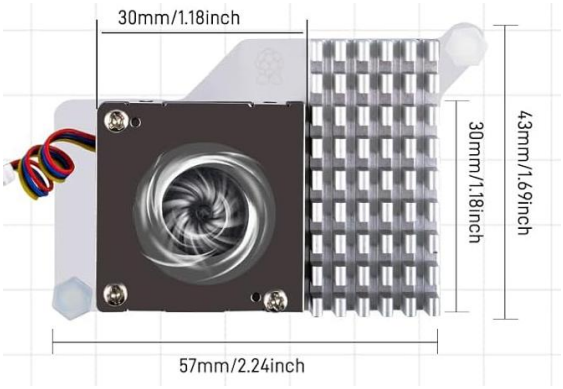


Ilustración 44 Enfriador de Raspberry pi 5 Recuperada de: https://www.amazon.com/-/es/dp/B0CY49Z6PN?ref_=ppx_hzod_title_mob_b_fed_asin_title_0_0

El equipo Raspberry Pi 5 funciona a 5V al igual que 5A cuenta con 2 puertos 3.0 y 2 puertos 2.0 al igual que un puerto de Red (RJ-45), 2 Salidas de Video Micro HDMI (Tipo D) al igual que alimentación vía USB Tipo C, al igual que un Pinout de 40 pines para Alimentación, entrada y salida, conexiones inalámbricas Wifi y Bluetooth, al igual que una de las funciones más importantes de esta tarjeta de desarrollo es el botón de Apagado, el cual permite encender y apagar el dispositivo, permitiendo evitar que se corrompan los datos ante un mal apagado.

Periféricos Conectados a Raspberry Pi

Para este prototipo las conexiones Físicas son las siguientes:

- Teclado y Mouse (Para programar el Equipo)
- Camara
- Dispositivo De Respaldo USB.
- Salida de Video Monitor.

Como se indica en la lista, para este proyecto se utilizaron los Módulos USB para la conexión del teclado y Mouse, la cámara la cual conecta la visión del mundo con el hardware del equipo, a su vez el puerto USB libre se implementando para el respaldo vía almacenamiento USB, esto permite en el peor escenario que la programación no sufriera algún daño.

Ilustración 45 Memoria Kingston Respaldo de proyecto

Para la salida de Video la Raspberry pi 5, se usó un adaptador Micro HDMI Tipo D a VGA, conectado a un monitor con el cual se programó el proyecto.

Ilustración 46 Adaptador Micro HDMI a VGA

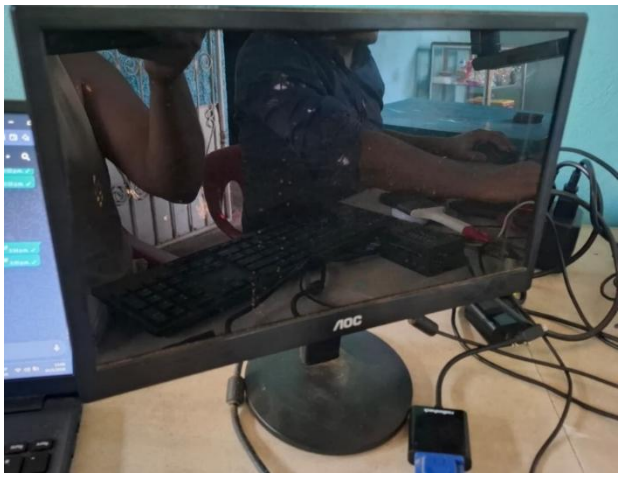


Ilustración 47 Monitor usado para salida de Video Raspberry pi Programación

Capítulo II Programación (Entrenamiento de Red Neuronal Siamés para Rostros y Placas):

2.1 Júpiter-Notebook Laptop

La programación de este proyecto se realizó en un computador con tarjeta gráfica dedicada RTX-3050 TI usando Jupyter-Notebook el cual se implementó mediante 2 vías:

- Anaconda
- WSL2

2.2 Anaconda (Jupyter-Notebook).

Para el método de Anaconda se necesitó solamente descargar el Software desde la página web, con esta herramienta se realizaron las pruebas, las instalaciones y el código en su totalidad.

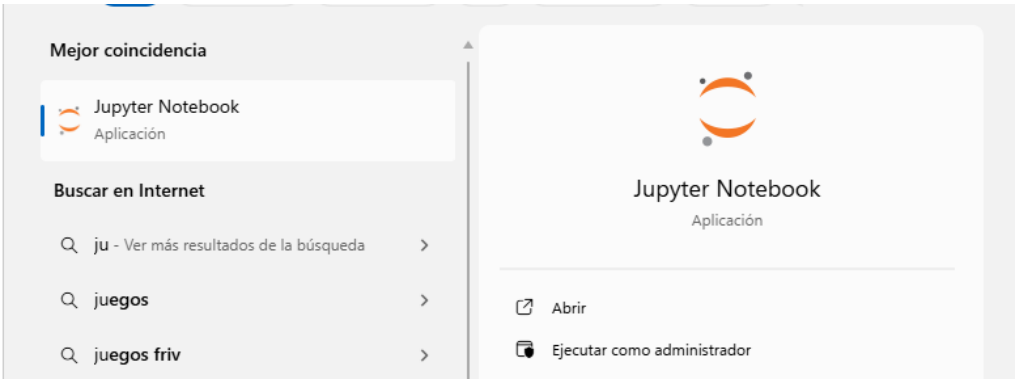


Ilustración 48 Jupyter Notebook ya instalado

Una vez instalado Anaconda se ejecutó la herramienta que apertura una página web con un servidor el cual muestra un espacio de trabajo donde se crean los Notebooks. Una vez creado los Notebooks se instalaron librerías y dependencias necesarias para el proyecto.

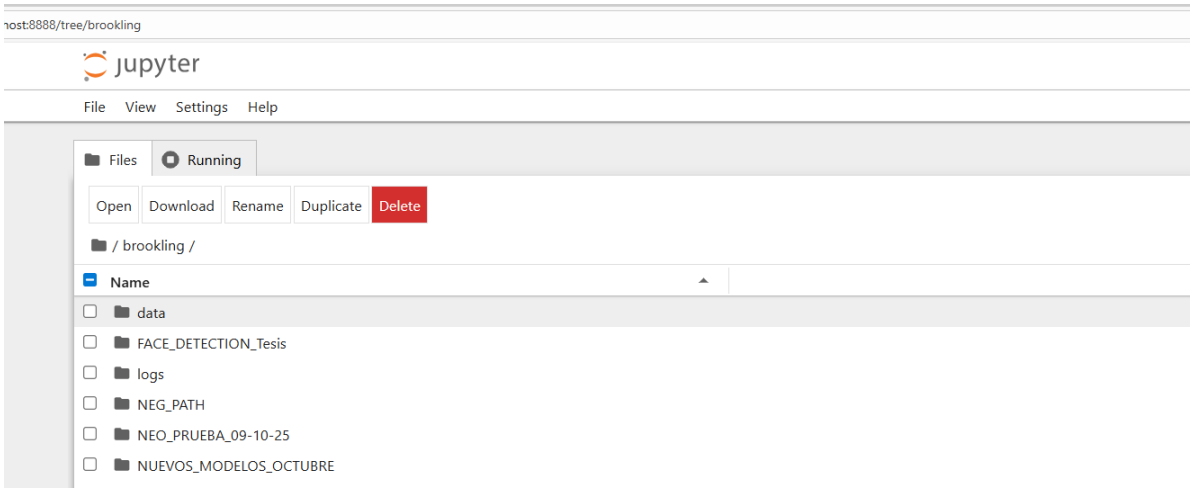


Ilustración 49 Notebook Jupiter

Como se puede observar en la ilustración 49 se apertura la pestaña en el navegador y posterior se puede iniciar con la programación en un nuevo Notebook.

2.3 WSL2 (Windows Subsystem for Linux)

WSL2 es una capa de compatibilidad de Linux para ejecutar en Windows, este método es necesario para posterior vía Linux (ejecutado en Windows) se pueda ejecutar Jupyter, con la ventaja de poder usar los recursos para tarjetas gráficas con el objetivo de mejorar el procesamiento de datos en el entrenamiento ahorrando tiempo, haciendo uso de CUDA y CDNN.

Para ello se deberá seguir los pasos que indica el desarrollador de Tensorflow:

1. Instalación Vía PowerShell con el siguiente comando “wsl –install”
2. WSL ya instalado.

Ilustración 50 WSL Instalado

Como se puede observar en la Ilustración 51 una vez instalado WSL, se puede usar como interfaz hibrida a Windows.

2.4 Cuda y CDNN.

Para este proyecto se utilizaron las Apis brindadas por el Fabricante NVIDIA llamadas Cuda y CDNN. **Cuda** (Compute Unified Device Architecture) Plataforma de computación paralela permite utilizar la potencia de la GPU (Luca, 2025). CDNN (**CUDA Deep Neural Network Library)**, es una librería de Nvidia que permite mejorar el rendimiento de las operaciones de Deep-Learning.

2.4.1 Instalación de Cuda y CDNN

utilizando los comandos: “Python3 -m pip install tensorflow[and-cuda]”

Verificación de detección de tarjeta gráfica.

Una vez finalizado el proceso de Instalación de Dependencias para Cuda y Cdnn, el cual toma algunas horas y descarga un aproximado entre 50gb a 70GB de Datos (dependencias) dado a que se descargan dependencias como Tensorflow, Open CV, entre otras librerías para Machine Learning, se usa el siguiente comando para comprobar que la computadora pueda detectar las tarjetas gráficas “**nvidia-smi**” para corroborar las tarjetas gráficas instaladas en el computador y se tiene lo siguiente:

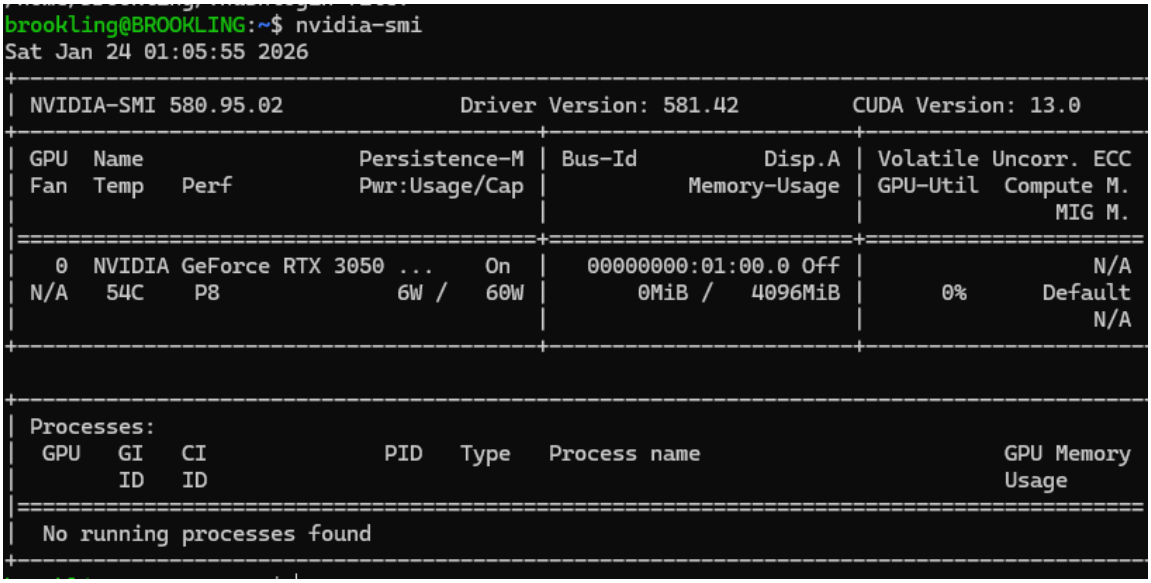
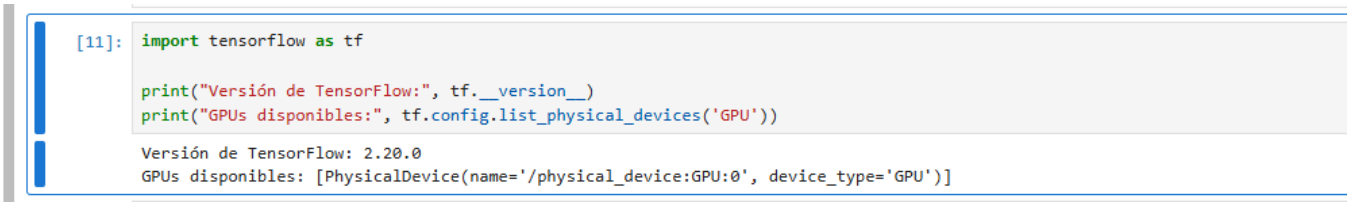


Ilustración 51 TARJETA GRAFICA, Scream-shot imagen obtenida localmente

2.5 Instalación de Jupyter Lab, Entorno virtual y validación de Tarjeta grafica

Una vez se tiene WSL instalado se procede a la instalación de Jupyter-Lab usando el comando “Pip Install Jupyter-notebook”, luego de instalado se creó un entorno virtual “python3 -m venv brook”. Posterior se manda a llamar el programa usando el comando “Jupyter-notebook”, se ejecutará el programa el cual llevará a un enlace de Local host que permitirá ejecutar Jupyter desde el navegador Web.



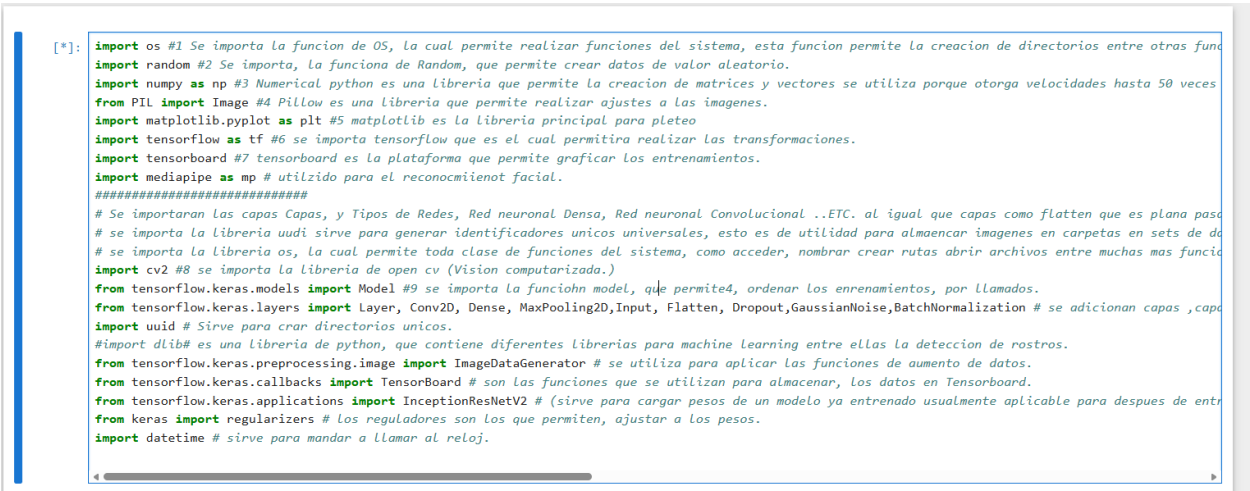
```
[11]: import tensorflow as tf

      print("Versión de TensorFlow:", tf.__version__)
      print("GPUs disponibles:", tf.config.list_physical_devices('GPU'))

      Versión de TensorFlow: 2.20.0
      GPUs disponibles: [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

Ilustración 52 Jupyter-Notebook comprobación de tarjeta grafica

En la Ilustración 51 como se puede observar se manda a imprimir en Jupyter las GPU disponibles y se imprime que ha encontrado la GPU: 0 que significa que nuestro entorno de programación ha logrado encontrar satisfactoriamente la GPU.

Una vez visto que Jupyter funciona bien tanto desde la versión para Windows (Anaconda Jupyter-Lab) y desde WSL, se puede proceder a programar.

Librerías



```
[*]: import os #1 Se importa la funcion de OS, la cual permite realizar funciones del sistema, esta funcion permite la creacion de directorios entre otras func
     import random #2 Se importa, la funciona de Random, que permite crear datos de valor aleatorio.
     import numpy as np #3 Numerical python es una libreria que permite la creacion de matrices y vectores se utiliza porque otorga velocidades hasta 50 veces
     from PIL import Image #4 Pillow es una libreria que permite realizar ajustes a las imagenes.
     import matplotlib.pyplot as plt #5 matplotlib es la libreria principal para pleteo
     import tensorflow as tf #6 se importa tensorflow que es el cual permitira realizar las transformaciones.
     import tensorboard #7 tensorboard es la plataforma que permite graficar los entrenamientos.
     import mediapipe as mp # utilzido para el reconocmiienot facial.
     ##############################
     # Se importaran las capas Capas, y Tipos de Redes, Red neuronal Densa, Red neuronal Convolucional ..ETC. al igual que capas como flatten que es plana pasa
     # se importa la libreria uudi sirve para generar identificadores unicos universales, esto es de utilidad para almaencar imagenes en carpetas en sets de da
     # se importa la libreria os, la cual permite toda clase de funciones del sistema, como acceder, nombrar crear rutas abrir archivos entre muchas mas funcio
     import cv2 #8 se importa la libreria de open cv (Vision computarizada.)
     from tensorflow.keras.models import Model #9 se importa la funciohn model, que permite4, ordenar los enrenamientos, por llamados.
     from tensorflow.keras.layers import Layer, Conv2D, Dense, MaxPooling2D,Input, Flatten, Dropout,GaussianNoise,BatchNormalization # se adicionan capas ,capa
     import uuid # Sirve para crar directorios unicos.
     #import dlib# es una libreria de python, que contiene diferentes librerias para machine learning entre ellas la deteccion de rostros.
     from tensorflow.keras.preprocessing.image import ImageDataGenerator # se utiliza para aplicar las funciones de aumento de datos.
     from tensorflow.keras.callbacks import TensorBoard # son las funciones que se utilizan para almacenar, los datos en Tensorboard.
     from tensorflow.keras.applications import InceptionResNetV2 # (sirve para cargar pesos de un modelo ya entrenado usualmente aplicable para despues de entr
     from keras import regularizers # los reguladores son los que permiten, ajustar a los pesos.
     import datetime # sirve para mandar a llamar al reloj.
```

Ilustración 53 Librerias Imagen extraida de captura de pantalla de codigo en Jupyter-Notebook

Estas son las librerías usadas para entrenar el modelo de Machine Learning que se empleó en este proyecto.

2.7 Dataset

Para entrenar la red Siamés (Rostros-Placas) de este proyecto se necesita el ADN de toda inteligencia artificial la cual son los Dataset, o set de datos. Para este modelo se necesitaron:

- Datos Positivos
- Datos Negativos
- Datos Ancla

Pero antes de definir tanto los positivos como los negativos y anclas, es necesario definir las “Clases”. Las clases para un set de datos son la cantidad de datos diferentes que se espera llevar a la red neuronal para que esta tome las decisiones.

2.8 Clases Para Rostros

Para los rostros se cuenta con 2 Rostros de estudiantes (Br Brookling Moncada y Br Larry Ruiz) y se tomó 5 Rostros adicionales los cuales fueron sustraídos de internet, se tomaron figuras públicas para el set de datos, las clases se compondrían de la siguiente forma:

- Keanu Reeves (Figura Publica Actor).
- LARRY (Estudiante Uní)
- Barack Obama. (Figura Publica tomada de set de datos para Rostros de Internet).
- BROOKLING (Estudiante Uní).
- Robert Downey Jr. (Actor Figura Publica).
- Tom Hiddleston (Actor Figura Publica).

Se tomaron solamente 6 Clases por eficiencia de entrenamiento, el mismo está basado en el proyecto de Google Facenet, que para su entrenamiento se estima fueron usados de 100 hasta 200 Millones de Rostros (Sandberg, 2018).

2.9 Captura de Datos

Para los Rostros de Br. Brookling y Br. Larry fue necesario el uso de la cámara. Se tomaron fotos enfocadas al rostro, posterior se utilizó código para la detección del rostro se aumentó el Bounding Box de la detección del rostro y se guardaron los rostros en una carpeta.

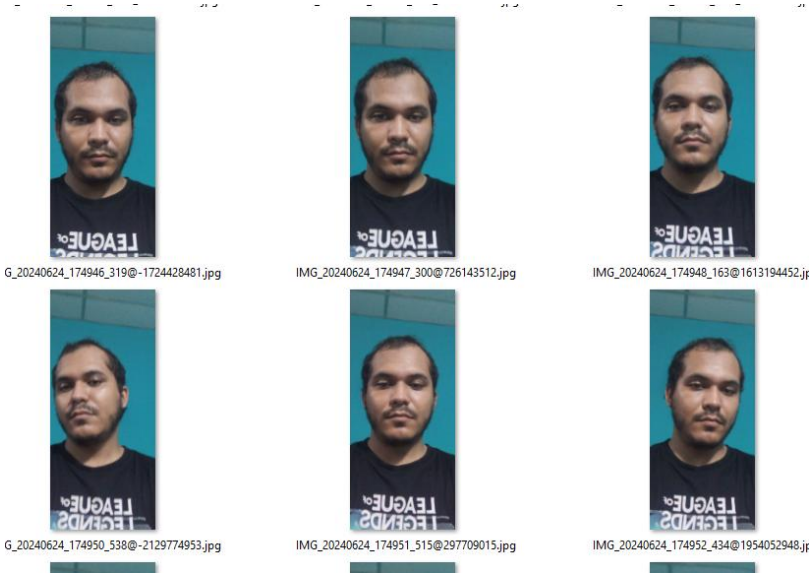


Ilustración 54 Captura de Rostros sin Recortar Brookling

Ilustración 55 Captura de Rostros sin Recortar Larry

2.10 Set de Datos Negativos

Una vez definidas las clases, se requiere aun apoyo de clases negativas, por lo cual, las clases negativas fueron tomadas del Sitio Web “LFW” que contiene más de 1000 Rostros de personas famosas y este viene ya con un tamaño reducido.

Ilustración 56 LFW recuperado de: https://www.kaggle.com/datasets/ashfaqsyed/labelled-faces-in-the-wild

Una vez descargado el set de Datos se descomprime y se tiene lo siguiente:

Ilustración 57 Set de Datos negativos

2.11 Descarga de Clases de Famosos.

Para las clases de “Famosos” se implementó el siguiente código, el cual permite descargar imágenes de internet usando Python, estas imágenes se guardan en un directorio, y posterior se utiliza el recortador de rostros, se utilizó la librería “bing_image_downloader”.

```
from bing_image_downloader import downloader

famosos = [
    "Emma Watson",
    "Tom Hiddleston",
    "Chris Hemsworth",
    "Natalie Portman",
    "Barack Obama",
    "Emma Stone",
    "Rihanna",
    "Robert Downey Jr",
    "Cristiano Ronaldo",
    "Elon Musk",
    "Scarlett Johansson",
    "Lionel Messi",
    "Taylor Swift",
#"Keanu Reeves"
]

# Carpeta raíz donde se guardarán las imágenes
output_dir ='C:/Users/Brookling/Documents/NUEVAS CLASES/clases2'

# Número de imágenes por persona
imagenes_por_persona = 100

# Descargar imágenes por cada famoso
for nombre in famosos:
    print(f"Descargando imágenes de: {nombre}")
    downloader.download(
        nombre,
        limit=imagenes_por_persona,
        output_dir=output_dir,
        adult_filter_off=True,
        force_replace=False,
        timeout=60
    )

print(" Descarga completada.")
```

Ilustración 58 Código para descargar imágenes

2.12 Recortador de Rostros

Para el recortador de rostros se utilizó la librería de “MediaPipe” la cual permite reconocer rostros y mandar a imprimir un rectángulo o cuadrado según se configure en el rostro detectado, para ello se pasó cada rostro de un directorio y se creó un directorio nuevo para los nuevos rostros.

```python
import cv2
import mediapipe as mp
import os
# Rutas
carpeta_entrada ='C:/Users/Brookling/Documents/NUEVAS CLASES/clases2/Natalie Portman'
carpeta_salida = 'C:/Users/Brookling/Documents/NUEVAS CLASES/Natalie Portman'
# Crear carpeta de salida si no existe
os.makedirs(carpeta_salida, exist_ok=True)
# Inicializar MediaPipe
mp_face_detection = mp.solutions.face_detection
face_detection = mp_face_detection.FaceDetection(min_detection_confidence=0.2)
# Margen extra para ampliar el bounding box
margen = 0.4
# Procesar cada imagen en la carpeta de entrada
for nombre_archivo in os.listdir(carpeta_entrada):
    ruta_imagen = os.path.join(carpeta_entrada, nombre_archivo)

    # Leer imagen
    imagen = cv2.imread(ruta_imagen)
    if imagen is None:
        print(f"No se pudo leer {nombre_archivo}")
        continue
    # Convertir a RGB para MediaPipe
    rgb = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)
    resultados = face_detection.process(rgb)
    if resultados.detections:
        for i, detection in enumerate(resultados.detections):
            bbox = detection.location_data.relative_bounding_box
            ih, iw, _ = imagen.shape
            x = int(bbox.xmin * iw)
            y = int(bbox.ymin * ih)
            w = int(bbox.width * iw)
            h = int(bbox.height * ih)

            # Ampliar bounding box con margen
            x1 = int(max(x - w * margen, 0))
            y1 = int(max(y - h * margen, 0))
            x2 = int(min(x + w * (1 + margen), iw))
            y2 = int(min(y + h * (1 + margen), ih))
            # Recortar imagen
            recorte = imagen[y1:y2, x1:x2]
            # Guardar con nombre único
            nombre_salida = f"{os.path.splitext(nombre_archivo)[0]}_rostro_{i}.jpg"
            ruta_salida = os.path.join(carpeta_salida, nombre_salida)
            cv2.imwrite(ruta_salida, recorte)
            print(f"Guardado: {ruta_salida}")
    else:
        print(f"No se detectaron rostros en {nombre_archivo}")
```

Ilustración 59 Código Recortador de Rostros

Una vez realizado el paso de cada imagen por el recortador de rostros las imágenes de rostros se verán así:

Ilustración 60 Imagen recortada

La imagen recortada ya solamente es el rostro y se añade un poco de espacio arriba más un poco de espacio abajo, por si algún motivo no se tomara en cuenta datos como: Barba, Cabello, Frente. De ser así, posiblemente el modelo no tendría suficiente información para poder aprender los rostros, esto se observó en este proyecto. Al recortar demasiado los rostros al punto de ojos a boca, el modelo le cuesta mucho más el aprendizaje, por lo cual, se recorta la imagen del rostro completo (No se adiciona el cuerpo ni tampoco información más que solo el rostro).

2.13 Clases para placas

Para las clases de placas se tomaron fotografías de Vehículos (Autos y Motos), se definió para el entrenamiento de Placas y este proyecto la cantidad de 5 clases al igual que los rostros. Para las placas se tomaron fotografías a las cuales posterior se aplicó un filtro que permite recortar solamente la placa.

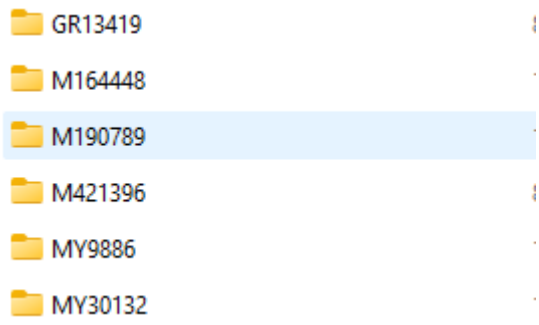


Ilustración 61 Clases para Placas

Una vez se tienen las imágenes capturas (Se podrán ver a detalle en anexos), se inicia el proceso de Recorte de Placas.

2.13.1 Recorte de Placas

Para el recorte de placas, se usaron librerías como “Easy OCR” o “Tesseract OCR” pero nunca se alcanzó un nivel de precisión lo suficientemente estable para la detección de placas, por ende, se usó un modelo preentrenado en Tensorfllow. El modelo fue extraído de Git-Hub (KURATA, 2025), una vez ya implementado el modelo se procede a recortar las placas con el detector de rectángulos.

```
modelpath = 'C:/Users/Brookling/placa detection/detect.tflite'
lblpath   = 'C:/Users/Brookling/placa detection/labelmap.pbtxt'
input_folder  = 'C:/Users/Brookling/Documents/documentos/PLACAS APRO
output_folder = 'C:/Users/Brookling/Documents/NUEVAS CLASES PLACAS/M
os.makedirs(output_folder, exist_ok=True)
min_conf = 0.5
# ---------------- CARGAR MODELO ----------------
interpreter = tf.lite.Interpreter(modelpath)
interpreter.allocate_tensors()
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
height = input_details[0]['shape'][1]
width  = input_details[0]['shape'][2]
float_input = (input_details[0]['dtype'] == np.float32)
input_mean = 127.5
input_std  = 127.5
# ---------------- CARGAR LABELS ----------------
with open(lblpath, 'r') as f:
    labels = [line.strip() for line in f.readlines()]
# ---------------- PROCESAR IMÁGENES ----------------
for img_name in os.listdir(input_folder):
    if not img_name.lower().endswith(('.jpg', '.png', '.jpeg')):
        continue
    img_path = os.path.join(input_folder, img_name)
    frame = cv2.imread(img_path)
    if frame is None:
        print(f"No se pudo leer {img_name}")
        continue
```

```
imH, imW, _ = frame.shape
image_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
image_resized = cv2.resize(image_rgb, (width, height))
input_data = np.expand_dims(image_resized, axis=0)
if float_input:
    input_data = (np.float32(input_data) - input_mean) / input_std
interpreter.set_tensor(input_details[0]['index'], input_data)
interpreter.invoke()
boxes   = interpreter.get_tensor(output_details[1]['index'])[0]
classes = interpreter.get_tensor(output_details[3]['index'])[0]
scores  = interpreter.get_tensor(output_details[0]['index'])[0]
recorte_id = 0
for i in range(len(scores)):
    if scores[i] > min_conf and scores[i] <= 1.0:

        ymin = int(max(0, boxes[i][0] * imH))
        xmin = int(max(0, boxes[i][1] * imW))
        ymax = int(min(imH, boxes[i][2] * imH))
        xmax = int(min(imW, boxes[i][3] * imW))
        recorte = frame[ymin:ymax, xmin:xmax]
        if recorte.size == 0:
            continue
        output_name = f"{os.path.splitext(img_name)[0]}_{recorte_id}.jpg"
        output_path = os.path.join(output_folder, output_name)
        cv2.imwrite(output_path, recorte)
        recorte_id += 1
        print(f"[OK] Guardado: {output_name}")

print("Procesamiento terminado")
```

Ilustración 62 Recortador de Placas

2.14 Filtrado de Imágenes

Una vez se tienen tanto los rostros y placas recortadas, se debe de filtrar las imágenes de manera **manual** y se deben de tomar solo las imágenes que tengan buena calidad dado que una imagen muy borrosa (de baja calidad) solamente entorpecería el proceso de entrenamiento,

2.14.1 Filtrado de Rostros

Para el proyecto se realizó el filtrado por clases hasta obtener la cantidad de 150 Imágenes por clase. Dado al uso del modelo de Red Siamés una de sus ventajas es que no se requiere de tantas imágenes para aprender, al igual que cabe destacar que se está usando el ancla el cual es la misma data que las imágenes positivas, al igual que dentro del proyecto en el entrenamiento se usa una estrategia de programación llamada “Aumento de datos” que permite de manera aleatoria adicionar características (Filtros) a cada imagen, lo que aumenta significativamente el data set.

2.14.2 Filtrado de Placas

Para el filtrado de las placas se filtraron las imágenes hasta obtener 150 Imágenes por clase.

Ilustración 63 proceso de Filtración

Como se puede observar en la ilustración, la importancia del filtrado es evitar que vaya información incorrecta al entrenamiento.

2.15 Entrenamiento

Una vez se tiene el Set de datos listos y ya se cuenta con las librerías se procede a cargar la información en Jupyter, primero se define en que carpeta se encuentra el directorio donde están las clases.

```
8]: directorio='C:/Users/Brookling/Documents/documentos/Rostros Clases DEF' # se declara el directorio en donde estaran las imagenes en este caso las clases.
print(directorio)# se manda a imprimir directorio.

C:/Users/Brookling/Documents/documentos/Rostros Clases DEF
```

Ilustración 64 Clases (Recuperado localmente)

Posterior que ya se ha definido la ruta en donde se encuentran las nuevas clases, se manda a imprimir los directorios de dichas clases:

```
Rostros Clases DEF

[6]: clases= os.listdir(directorio) # se utiliza las funciones del sistema os. para poder ver los directorios.
print(clases)# se manda a imprimir las clases.

['Keanu Reeves', 'LARRY', 'Barack Obama', 'BROOKLING', 'Robert Downey Jr', 'Tom Hiddleston']

[7]: clases_0=clases[0]
```

Ilustración 65 Directorio de clases

Las imágenes pasan por un proceso de filtrado, se mandan a llamar usando la función de “tf.io.read_file”, se ajustan 3 canales (RGB), posterior se hace un reajuste de tamaño a 105x105 que es el tamaño con el cual funciona la red Siamés, luego se ***normaliza*** (En vez de datos de 1 Byte) se divide entre 255, para que la información sea entre 0 y 1, posterior se retorna la imagen.

```
def load_and_preprocess_image(file_path): #se crea funcion para convertir imagenes de entrada en formato que se requiere.
    image = tf.io.read_file(file_path) # se lee la imagen usando tf. para transformarla en formato tensor.
    image = tf.image.decode_jpeg(image, channels=3) # se codifica a jpeg, con 3 canales (RGB)
    image = tf.image.resize(image, [105, 105])  # Ajusta el tamaño según sea necesario
    image = tf.cast(image, tf.float32) / 255.0  # Normaliza la imagen
    return image# retorna la imagen ya transformada.
```

Ilustración 66 Filtrado de imágenes

Posterior al filtrado de imágenes se procede a crear arreglos vacíos, los cuales almacenaran información de Ancla, Positivo y Negativos. Se utiliza la función de la librería OS para tomar las imágenes de cada carpeta usando un ciclo For, y la manera de asignar los positivos y anclas es tomando la carpeta de Clases, se asigna esta como positiva, posterior se usa un condicional el cual toma datos de la misma clase del positivo, pero tomando una imagen diferente a la ya tomada por el positivo por cada ciclo de **for,** por lo tanto el ancla y el positivo son las mismas, pero cada imagen es diferente dado a que debe existir cierta diferencia. Para el caso de los **Negativos** existen 2 vías, la primera es que usando un condicional IF se toma una imagen de

una clase completamente diferente al positivo y ancla, por lo tanto cualquier imagen que no sea el ancla y el positivo será un negativo, también se puede añadir una carpeta llena de negativos, y de esta se toma un porcentaje de negativos, con el fin de fortalecer el entrenamiento.

```
def clases_imagen(directorio): # Se crea clase para extraer los contenidos de las clases.
    tripl=[] # Se crea un arreglo vacio.
    posi=[]# se crea arreglo vacio.
    nega=[]# se crea un arreglo vacio.
    ancl=[]# se crea un arreglo vacio, se crea, para poder almacenar listas a futuro.
    ancla_imagen=[] # se crea un arreglo vacio.
    clases= os.listdir(directorio)# se extrae la lista de clases.


    for cls in clases:# se crea una funcion ciclica con for, para la extraccion de las imagenes.
        class_dir = os.path.join(directorio, cls)# cls es una funcion classmethod.
        print(class_dir)# se manda a imprimir, las clases, para vizualizarlas.
        print(len(class_dir))
        class_imagen = os.listdir(class_dir)# clases de imagen.
        if len(class_imagen) < 2: # si hay almenos mas de 2 clases se continua
            continue
        #class_imagen = class_imagen[:num_imagen]
        for i in range(len(class_imagen)): # se crea una variable (i) dentro de un rango (tamaño de la clase).
            ancla_imagen=os.path.join(class_dir, class_imagen[i])# Se toman las imagenes de clases, y se convierten en el ancla.
            ################
            misma_clase=[img for img in class_imagen if img !=class_imagen[i] ] # se busca una clase igual triplete positivo.
            ###################################
            positivo_path= os.path.join(class_dir, random.choice(misma_clase)) # se elige y  se une a una clas
            ########################################

            negativo_path= random.choice([c for c in clases if c !=cls])# se busca una imagen diferente en posicion a la posicion del triple
            negativo_dir=os.path.join(directorio, negativo_path)# Se une al arreglo negativo.
            negativo_imagen_path=os.path.join(negativo_dir, random.choice(os.listdir(negativo_dir)))#tomara valores aleatorios para los nega
            ################################
            ancla_imagen=(ancla_imagen) # Se realiza el guardado utilizando la misma variable
            positivo_path=(positivo_path)# se guarda con el nombre de la misma variable
            negativo_imagen_path=(negativo_imagen_path)# se actualiza la variable.
            ######################################
            ancl.append((ancla_imagen))# Se crea un apendize.
            posi.append((positivo_path))#Se crea un apendice
            nega.append((negativo_imagen_path))#Se crea un apendice-
    return ancl, posi, nega # retorna 3 valores, la funcion una ancla, positiva y negativa.
```

Ilustración 67 Creación de Set de Datos Positivo; negativo y Ancla

Una vez ya los datos están separados en Positivo, Negativo y Ancla, se procede con el **Aumento de datos (**como se vio previamente en el Marco Teórico), el aumento de dato es usado para hacer pequeños ajustes a las imágenes, (Desde meter ruido, subir el brillo, girarlas hacer efectos espejo etc.).

```
# Zoom aleatorio centrado
def random_zoom(image, zoom_range=(0.9, 1.0)):
    # Factor aleatorio de zoom
    scale = tf.random.uniform([], zoom_range[0], zoom_range[1])
    # Recorte central (zoom-in)
    image = tf.image.central_crop(image, central_fraction=scale)
    # Redimensionar al tamaño original
    return image
```

Ilustración 68 Random Zoom

```python
def augment_image(image): # se crea la funcion de aumento de datos.
    # Se tienen las mejoras en cada imagen,  con cada caracteristica usando random.
    noise = tf.random.normal(shape=tf.shape(image), mean=0.0, stddev=0.1, dtype=tf.float32) # se inserta ruido gauseano.
    image = image + noise # se suma la imagen de entrada mas el ruido.
    image = tf.clip_by_value(image, 0.0, 1.0)# Se asegura que las imagenes estan normalizadas, se requiere que esten den
    #image = tf.image.random_flip_left_right(image)# Invierte la imagen de izquierda a derecha.
    #image = tf.image.random_flip_up_down(image)# invierte la imagen de manera vertical
    #image=  tf.keras.layers.RandomRotation(factor=0.05)# rota un poco la imagen creando un pequeño angulo de inclinacio
    image = tf.image.random_brightness(image, max_delta=0.2)# selecciona un brillo ramdom.
    image = tf.image.random_contrast(image, lower=0.8, upper=1)# selecciona un contraste randoml.
    image = tf.image.random_saturation(image, lower=0.8, upper=1)# la saturacion lo que logra, es poner la imagen o muy
    image = tf.image.random_hue(image, max_delta=0.2) # crea aleatoriamente un rango de matiz, en las imagenes.
    #image = tf.image.rot90(image, tf.random.uniform(shape=[], minval=0, maxval=4, dtype=tf.int32))# rota las imagenes.
    return image # retorna la imagen despues de aplicar todos estos filtros.
```

Ilustración 69 Aumento de Datos

En la ilustración se puede observar diferentes tipos de aumento de datos, algunos han sido desactivados dado a que no todos convienen tal como la rotación de la imagen sea Vertical, 90°,180° o un giro Random.

```python
ancla_imagen = tf.data.Dataset.from_tensor_slices(ancla) # se utiliza la funcion slices para desplazar las imagenes.
ancla_imagen = ancla_imagen.map(lambda x: load_and_preprocess_image(x))# la imagen se envia an ormalizar.
##################################
positivo_imagen = tf.data.Dataset.from_tensor_slices(positivo)# el data set. se mueve, en positivo.
positivo_imagen = positivo_imagen.map(lambda x: load_and_preprocess_image(x)) # el uso de .map y lamda, aplica una funci
##################################
negativo_imagen = tf.data.Dataset.from_tensor_slices(negativo)
negativo_imagen = negativo_imagen.map(lambda x: load_and_preprocess_image(x))

#################################################################################
positivo_aug = positivo_imagen.map(lambda x: augment_image(x)) # se realiza la normalizacino
negativo_aug = negativo_imagen.map(lambda x: augment_image(x)) # se realiza la noramlizacion.
ancla_aug = ancla_imagen.map(lambda x: augment_image(x)) # se realiza la normalizacino.
#################################################################################
# Crear datasets de positivos
positives = tf.data.Dataset.zip((ancla_aug, positivo_aug, tf.data.Dataset.from_tensor_slices(tf.ones(len(ancla_aug)))))
negatives = tf.data.Dataset.zip((ancla_aug, negativo_aug, tf.data.Dataset.from_tensor_slices(tf.zeros(len(ancla_aug)))))
data = positives.concatenate(negatives) #se concatena en un arreglo.
```

Ilustración 70 Etiquetado de Set de Datos

Se etiquetan los datos, si el positivo y el ancla coinciden entonces es 1 y si el ancla y el negativo no coinciden (Se espera nunca lo hagan) es 0.

Una vez etiquetados los datos, se usa una memoria cache para guardar los datos de manera rápida, y se utiliza una función iteradora, esto para lograr que cada época de entrenamiento tome un set de imágenes diferentes.

```python
# #construir data pipelines.
data = data.cache() #se utiliza cache para guardar los datos de manera mas rapida.
data = data.shuffle(buffer_size= 1024)# Shuffle mezcla aleatoriamente los datos, y el buffer size, mientras mas gran
```

```python
ejemplos=data.as_numpy_iterator()# la funcion interador, permite, iterar los valores en este caso las imagenes.
ejemplo_iterador=ejemplos.next()# el punto next, mueve las imagenes
len(ejemplo_iterador)# se calcula el tamaño usando len, esto se realiza, para determinar si el valor es correcto.
```

Ilustración 71 Iterador y Data cache

Posterior a la iteración se crearon nuevas variables las cuales fueron los datos de entrenamiento y los datos de Test. para los datos de entrenamiento se toma un 70% de las imágenes usadas de entrenamiento, para Test se usa el 30% del restante para posterior hacer pruebas y determinar si el modelo ha sido entrenado correctamente, de ser positivo y funcionar se podrá convertir el modelo y enviar a la Raspberry Pi.

```
# Datos que se usaran para entrenamiento.
train_data= data.take(round(len(data)*.7)) # Se toma el 70 % de los datos, para entrenamiento.
#train_data = train_data.cache()
train_data= train_data.batch(16) #Se crea un tamaño de bache de 16, esto dependera de que tanta persicion se queira tene
#train_data = train_data.prefetch(tf.data.AUTOTUNE)
train_data= train_data.prefetch(8)#
train_data# se envia  a llamar a la funcion.
len(train_data)# se envia a imprimir el tamaño.
```

```
76
```

```
# Datos que se usaran para TEST
test_data= data.skip(round(len(data)*.7))
test_data= test_data.take(round(len(data)*.3))
test_data = test_data.cache()
test_data= test_data.batch(16)
#test_data = test_data.prefetch(tf.data.AUTOTUNE)
test_data= test_data.prefetch(8)
test_data
len(test_data)
```

Ilustración 72 Train Data y Test Data

Una vez creado los datos de **entrenamiento** y los datos de **Test** se proceden a crear la estructura de red convolucional.

```
def make_embedding():
    inp = Input(shape=(105,105,3), name='input_image') # imagen de entrada con (105 x150 pixesles y 3 canales (RGB).
    #Anterior penalizacion : 0.01
    # Bloque 1
    c1 = Conv2D(64, (10,10), activation='relu', kernel_regularizer=regularizers.l2(1e-4))(inp) # Primera convolucion,
    m1 = MaxPooling2D((2,2), padding='same')(c1) # Se usa maxpooling y se mantiene el tamaño.

    # Bloque 2
    c2 = Conv2D(128, (7,7), activation='relu', kernel_regularizer=regularizers.l2(1e-4))(m1) # Segunda convolucion, c
    m2 = MaxPooling2D((2,2), padding='same')(c2)# se mantiene el tamaño.

    # Bloque 3
    c3 = Conv2D(128, (4,4), activation='relu', kernel_regularizer=regularizers.l2(1e-4))(m2)# tercer bloque de convol
    m3 = MaxPooling2D((2,2), padding='same')(c3)

    # Bloque final
    c4 = Conv2D(256, (4,4), activation='relu', kernel_regularizer=regularizers.l2(1e-4))(m3)
    f1 = Flatten()(c4)
    dro = Dropout(0.5)(f1)
    out = Dense(4096, activation='sigmoid')(dro)

    model = Model(inputs=inp, outputs=out, name='embedding_model')
    return model

# Crear y guardar
embedding_model = make_embedding()
embedding_model.summary()
```

Ilustración 73 Convolución estructura

En la **ilustración 73** se tiene la estructura de la red convolucional que se emplea para las redes Siamesas, se tiene la función "**Make_embedding**" la cual si se llegase a ejecutar directamente genera **Embeddings**, posterior se aplicaron criterios para ajustarla a una red siamesa completa.

Se tiene una imagen de entrada la cual debe de coincidir con un tamaño de 105 x 105 y 3 Canales RGB. Posterior ingresa al primer bloque de convolución donde se tienen 64 núcleos de resultado o filtros y un tamaño de filtro de 10x10, se utiliza el Max-Pooling para mantener la tendencia de reducir la imagen a la mitad y posterior se emplea el Padding para mantener el mismo tamaño rellenando de 0 los espacios vacíos, se emplean los **Regularizadores** para evitar que los pesos crezcan demasiado y generen falsos resultados.

El proceso de convolución continua capa por capa hasta que se extraen las características más importantes de la imagen, se emplea el llamado "Dropout" el cual apaga las neuronas aleatoriamente, esto con el objetivo que las neuronas aprendan a generar más interconexiones entre ellas y no dependen de un solo modo de aprendizaje, esto las fortalece. Posterior el resultado final se **aplana** de pasar de ser una matriz de NxN dimensiones pasa a una sola dimensión, dicha dimensión es tomada por una red densa la cual toma los valores y criterios con una función de activación sigmoide de 0 a 1.

```
def make_siamese_model():
    input_image = Input(shape=(105,105,3), name='input_img')
    validation_image = Input(shape=(105,105,3), name='validation_img')

    embedding_model = make_embedding()
    inp_emb = embedding_model(input_image)# ingreso de la imagen.
    val_emb = embedding_model(validation_image)# validacion de imagen.

    distances = DistanceLayer(name='distance')([inp_emb, val_emb])
    output = Dense(1, activation='sigmoid')(distances)

    model = Model(inputs=[input_image, validation_image], outputs=output, name='SiameseNetwork')
    return model

siames_red = make_siamese_model()
siames_red.summary()
```

Ilustración 74 Red Siamés.

Una vez se genera la red convolucional se realiza otra función la cual se conoce como **red siamesa**, que manda a crear una imagen de entrada y una imagen de validación. Dichas imágenes las pasa por la red convolucional, posterior la red mide las diferencias y esto se envía a una red densa con una función de activación.

```
]: # --- Distance Layer ---
   class DistanceLayer(Layer):
       def call(self, inputs):
           input_embedding, validation_embedding = inputs
           return tf.math.abs(input_embedding - validation_embedding)
```

Ilustración 75 Distance Layer función donde se genera la diferencia

Para el entrenamiento se hizo uso del modelo Custom donde se debe de declarar la perdida, al igual que se facilitan los registros para hacer uso e inicio de Tensorboard y la implementación del **Early Stopping** que permite que después de cierto número de entrenamientos, en donde la red no mejore, se detenga esta misma y se quede con los pesos mejor entrenados.

```
print(f'Epoch loss: {loss.numpy()} | Epoch recall: {recall_result} | Epoch precision: {precision_result}')
if (recall_result >=0.98) and (precision_result >=0.98):
    print("Registro para Early Stopping")
    count= count+1
    print(count)
    if(count==5):
        print("Early Stopping ha encontrado el mejor entrenamiento")
        print(epoch)
        break
else:
    count=0
```

Ilustración 76 Early Stopping

Para el **Early Stopping**, al momento de definir épocas, se podría poner un número muy alto de 100 hasta 1000 Épochs (épocas), y el Early Stopping solamente tomara las que necesita ya sean 20, 50, 100…Etc.

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
86/86 ━━━━━━━━━━━━━━━━━━━━ 14s 163ms/step
Epoch loss: 0.0003406389441806823 | Epoch recall: 0.998678982257843 | Epoch precision: 1.0
Registro para Early Stopping
5
Early Stopping ha encontrado el mejor entrenamiento
21
```

Ilustración 77 Ejemplo de Early Stopping.

```
[ ]: EPOCHS=50
     train(train_data,EPOCHS)
```

```
Epoch 1/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 338ms/step
Batch 1:
  y_true shape: (16,)
  y_pred shape: (16, 1)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 223ms/step
Batch 2:
  y_true shape: (16,)
  y_pred shape: (16, 1)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 247ms/step
Batch 3:
  y_true shape: (16,)
  y_pred shape: (16, 1)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 218ms/step
```

Ilustración 78 Entrenamiento de Modelo

2.16 Tensor Board

Tensor Board como se ha mencionado previamente es una herramienta visual de Tensor Flow permite ver el entrenamiento en tiempo real y sobre todo permite ver de manera Grafica ver: Perdida, Recall y Precisión.

2.16.1 Graficas Reconocimiento de Rostros

2.16.1.1Perdida

Para el modelo de Reconocimiento de Rostros se tiene la siguiente curva que explica la perdida en el tiempo mientras más baje la perdida significa que más aprende el modelo.

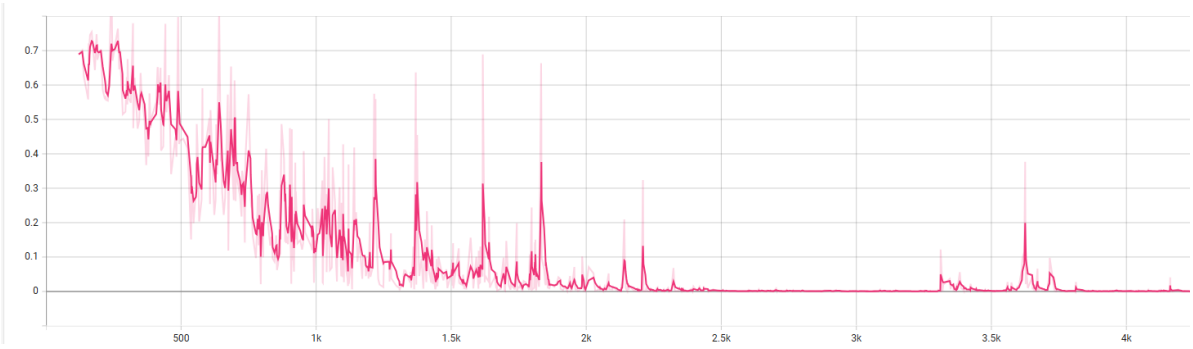


Ilustración 79 Perdida Reconocimiento de facial

El Eje Vertical Representa la **perdida** y el Eje Horizontal representa los Steps o pasos, el número de pasos va a depender de la cantidad de imágenes que tenga que analizar la red neuronal, si fueran 1000 imágenes por un tamaño de bache de 16 cada época necesitaría al menos 62-63 Steps por cada época.

2.16.1.2 Precisión

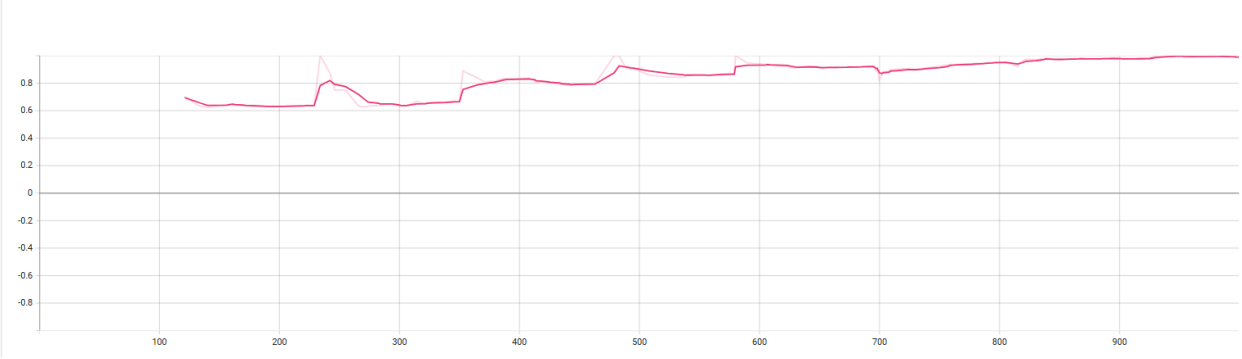


Ilustración 80 Precisión Entrenamiento de Reconocimiento Facial

La precisión nos indica el total de la cantidad de veces que el modelo dice haber detectado una persona; es decir, un resultado positivo; sin embargo, no verifica completamente que todos sean verdaderos. Ejemplo: El modelo dice haber detectado 10 personas, pero en realidad solo 5 son verdaderos, entonces tiene una precisión de 0.5. Útil para evitar falsos positivos. Una manera más fácil de comprender el concepto es mediante la siguiente interrogante: De todas las predicciones que hizo el modelo de IA, ¿cuántas fueron correctas?

2.16.1.3 Recall

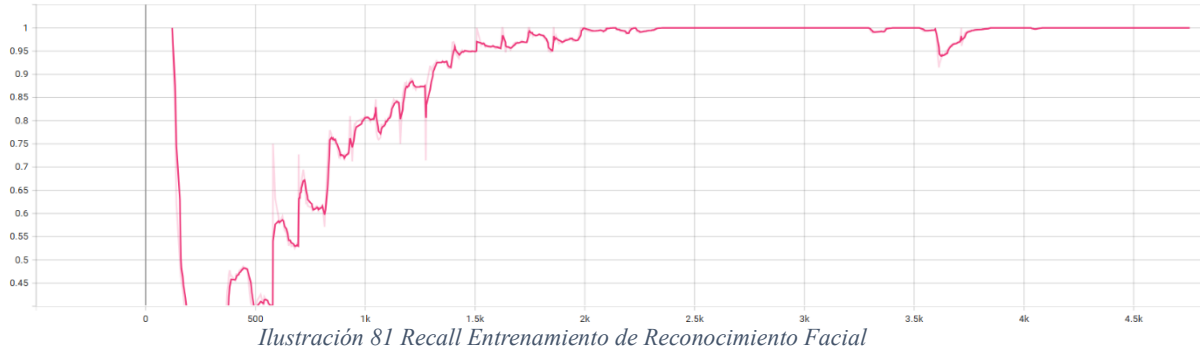


Ilustración 81 Recall Entrenamiento de Reconocimiento Facial

El Recall se parece bastante a la precisión, pero no son lo mismo. El Recall nos muestra de manera exacta el número de veces que verdaderamente el modelo ha acertado en la detección. **Ejemplo:** De 10 personas detectadas solo 8 son verdaderas, entonces hay un recall de 0.8. Útil para evitar falsos negativos. Una manera más fácil de comprender el concepto es mediante la siguiente interrogante: De todos los casos positivos reales, ¿cuántas detectó correctamente el modelo?

2.16.2 Gráficos Reconocimiento de Placas

2.16.2 .1 Perdida

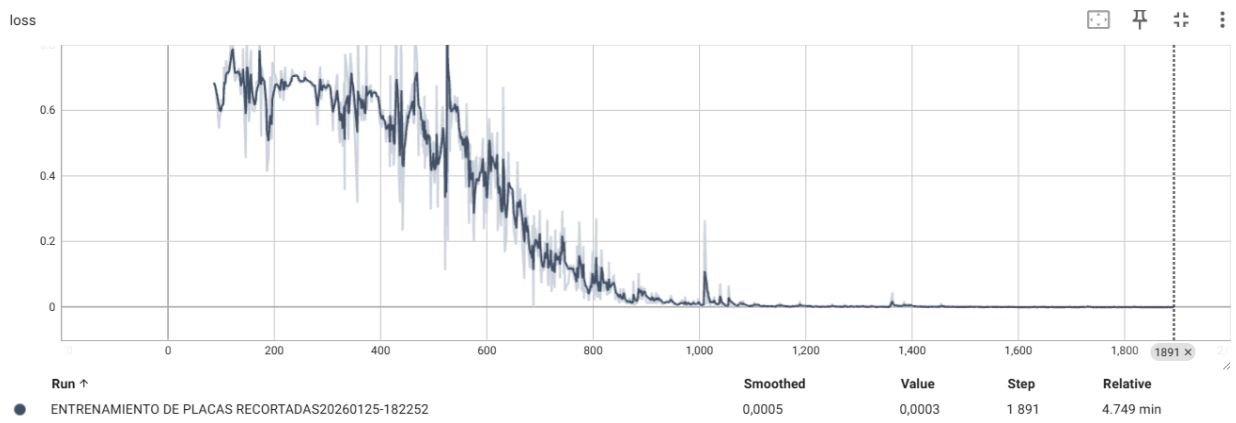


Ilustración 82 Pérdida Plate Recognition

2.16.2.2 Precisión

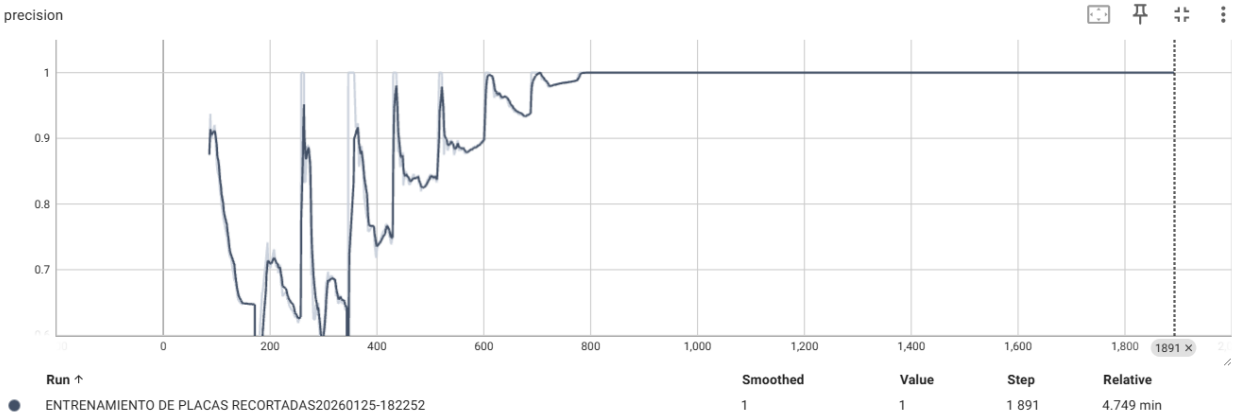


Ilustración 83 Precisión plate recognition

2.1.6.2.3 Recall

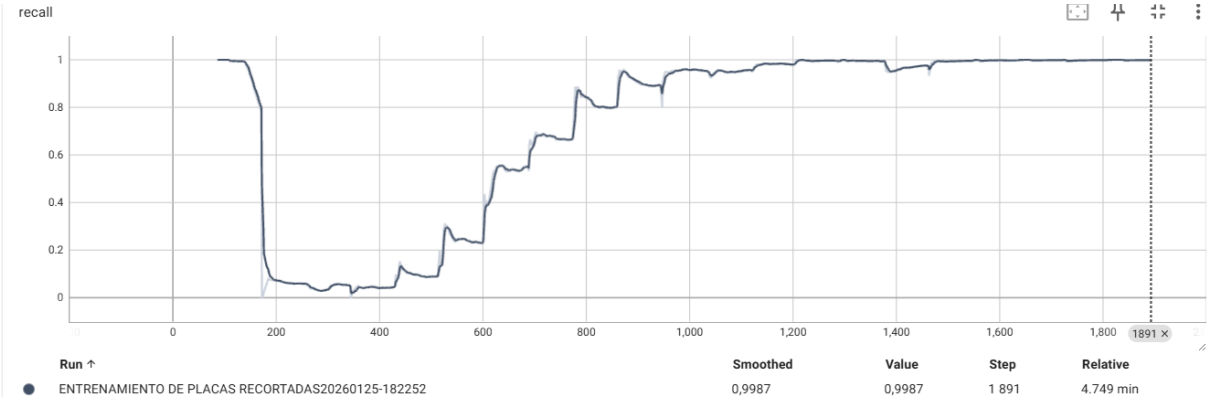


Ilustración 84 Recall plate recognition-

2.17 Test Data

Una vez entrenados los modelos, se utiliza Data test para poder comparar que el modelo funcione correctamente.

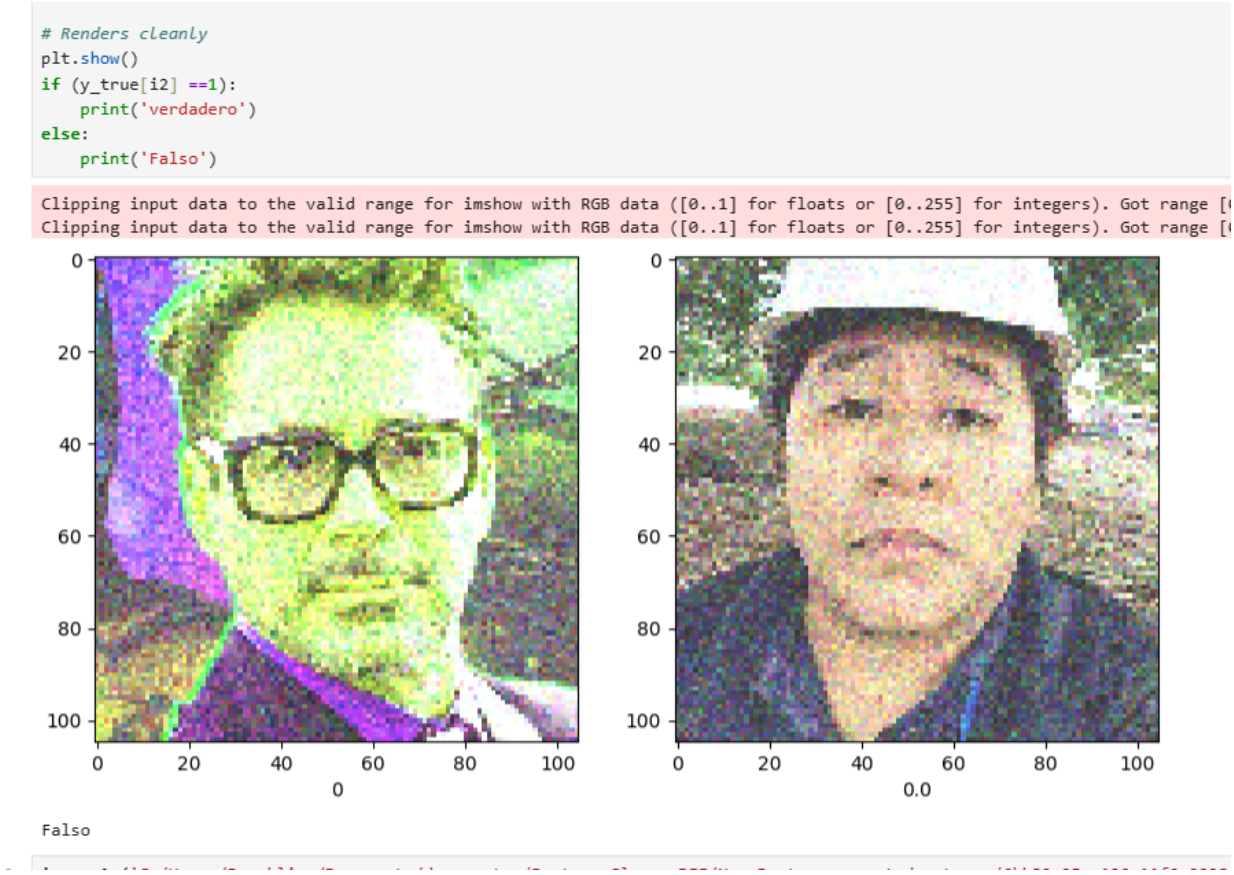


Ilustración 85 Test Data

Como se puede apreciar en la **ilustración 85** se envía a comparar 2 Imágenes de Data test, el modelo logra reconocer apropiadamente que no son iguales, posterior se cargan otras dos imágenes en Data test que deberían ser iguales y se obtiene el siguiente resultado.

Como se puede apreciar en la **ilustración 86** se tienen dos imágenes de la misma persona en 2 momentos diferente de tiempo, con dos diferentes tipos de lentes y cambios físicos en el pelo y barba y pese a ello el modelo ha encontrado las similitudes.

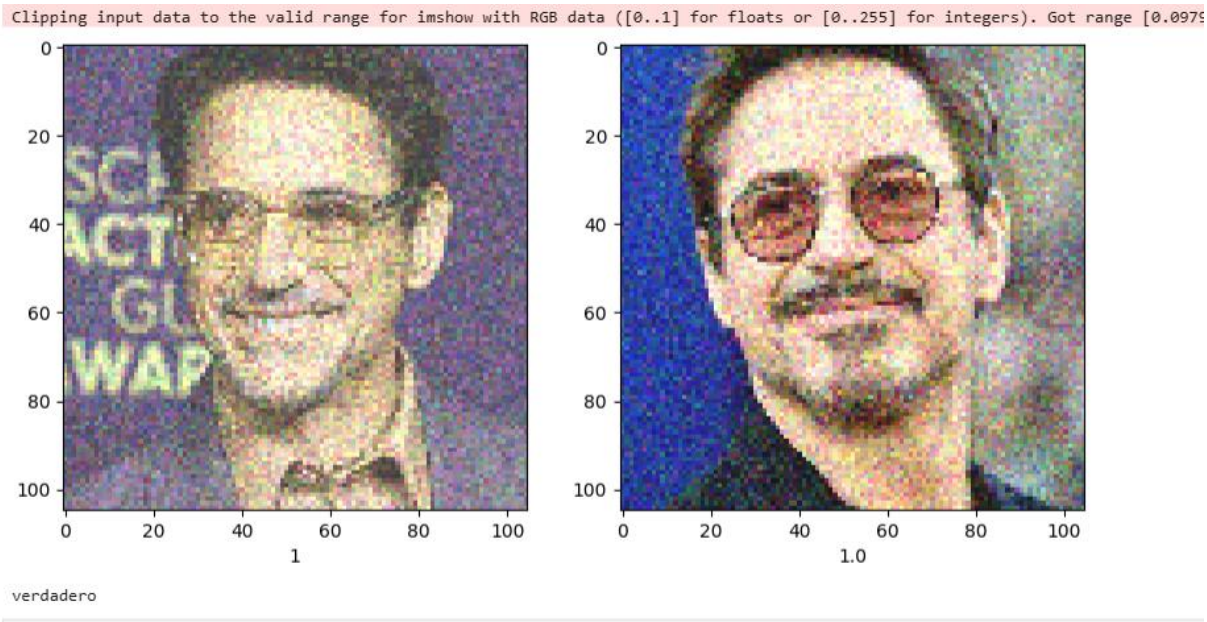


Ilustración 86 Test Data imágenes Verdadero

2.18 Conversión a TF-Lite

Una vez ya entrenado el modelo y que este ha demostrado funcionar el siguiente paso a hacer es realizar la conversión de Tensor Flow a TF-Lite, el cual es el formato que es compatible con Raspberry Pi, para ello se emplean los siguientes códigos:

```
siames_red.export('saved_model/siames_red_nov_Tesis')


converter = tf.lite.TFLiteConverter.from_saved_model('saved_model/siames_red_nov_Tesis')
tflite_model = converter.convert()

# 3. Guardar el modelo TFLite en disco
with open("siames_red_nov_Tesis.tflite", "wb") as f:
    f.write(tflite_model)
```

```
INFO:tensorflow:Assets written to: saved_model/siames_red_nov_Tesis/assets
INFO:tensorflow:Assets written to: saved_model/siames_red_nov_Tesis/assets
Saved artifact at 'saved_model/siames_red_nov_Tesis'. The following endpoints are available:
```

Ilustración 87 Conversión a TF Lite

Una vez usado los siguientes códigos se exporta el formato de **Tensor Flow** a **Tensor Flow-Lite**, y se genera un Archivo. TFLite, el cual se encuentra listo para ser usado en Raspberry PI.

Solamente se debe de buscar el archivo en los directorios donde se ejecuta Jupyter y se podrá encontrar, deberá tener el mismo nombre con el que fue creado.

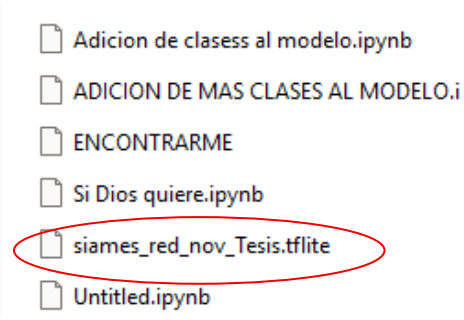


Ilustración 88 Archivo Convertido a TF Lite

Como se aprecia en la figura se encontró el archivo ya convertido en TF-Lite por lo cual ya puede ser empleado el modelo (Preentrenado) en la tarjeta de desarrollo Raspberry Pi.

Capítulo III: Programación en Raspberry Pi

3.1 Entorno-Virtual

Para la programación en **Raspberry Pi 5** el primer paso es tener los modelos Preentrenados en Tf-Lite, posterior, se deberán de declarar las librerías necesarias para lograr la programación, al igual que la programación para el servidor que funciona como BD que es Mongo-DB.

Para ello, lo primero que se debe de realizar en Raspberry Pi, es la apertura de Jupyter-Notebook, para ello, en Debian 12 BookWorm, para evitar daños en el sistema, solicita que el usuario creé entornos virtuales, dichos entornos, deben de ser implementados, si no, desde la consola de líneas no permitirá ejecutar programas como Jupyter.

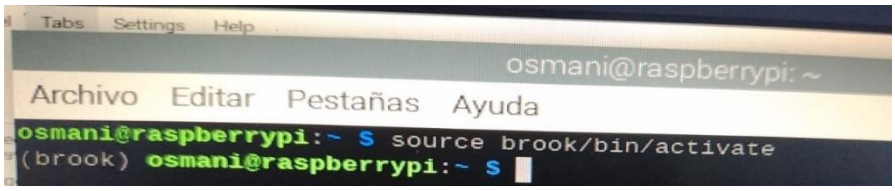


Ilustración 89 Activación de Entorno Virtual

Para crear el entorno virtual se utiliza el siguiente comando "**python3 -m venv brook**". Una vez ejecutado este comando para activar el entorno virtual se ejecuta el siguiente comando "**source brook/bin/actívate**". Con el entorno virtual ya activado se puede proceder a hacer uso de programas de desarrollo al igual que la instalación de programas y dependencias con la herramienta Pip.

3.2 Programación en Jupyter-Notebook Raspberry Pi.

Para la programación en Rasbperry Pi, se debe de tener en claro el concepto de programación no es nada más que la integración del modelo Pre-entrenado de placas, al igual que rostros, ejecutándolo en tiempo real usando una cámara, para posterior enviar los datos al servidor y estos ser registrados en la BD del servidor, es por ello que el primero paso es la adecuación del código.

Para ello, se está utilizando una programación que como primer paso permite cargar el modelo preentrenado y se realiza la prueba comparando 2 imágenes, con rutas ya definidas, posterior se realizó el uso de este código en funciones, en ejecución de tiempo real + la adición del server y BD.

3.2.1 Programación de Reconocimiento Facial En Tiempo Real.

Para el código lo primero que se debe de realizar es cargar las librerías las cuales son las siguientes:

```
import tflite_runtime.interpreter as tflite # se importa TFLIte para pod
import cv2# se importa OPEN CV para poder trabajar con vision computariz
import numpy as np # se importa NUMPY la funcion de Arreglos de numeros
import dlib # se importa Dlib, la cual permite usar funciones de reconoc
from PIL import Image# Se importa  Pillow, la cual nos permite, directam
import os
import time
```

Ilustración 90 Librerías para Face-Recognition

Se carga la librería “**Tflite”**, esta permite la compatibilidad de Tensorflow con la tarjeta de desarrollo Raspberry Pi, “Dlib”, para la detección de rostros, “os” para funciones de sistema, **Pillow (PIL),** en Raspberry “Pillow” librería que permite editar imágenes, dicha librería junto a OpenCv son las piedras angulares en Raspberry Pi para el proyecto.

```
def load_image(path, target_size=(105,105)):
    img= Image.open(path).convert('RGB')
    img= img.resize(target_size)
    img_array = np.array(img, dtype=np.float32)/ 255.0
    img_array = np.expand_dims(img_array, axis=0)
    return img_array
```

Ilustración 91 Función de preparación de imágenes para pasar por red siamés.

Dado a que se trabajó un modelo preentrenado de una red siamesa, se debe de emular la estructura para dicha red, en la cual ingresan 2 imágenes, por lo cual, se debió de

preparar cada imagen en un tamaño 105x105, 3 a como se realizó para el entrenamiento (**Ver Ilustración 92**).

3.2.2 Distancias Euclidianas

Posterior se creó una función la cual usa las distancias Euclidianas (Dichas funciones son de las más usadas para Face-Recognition).

```
[4]: def euclidean_distance(emb1, emb2):
         return np.sqrt(np.sum((emb1-emb2)**2))
```

Ilustración 92 Distancias Euclidianas

Una vez realizado esto se debe cargar el modelo en Tensorflow-Lite, se debe de indicar en que directorio (carpeta) se encuentra el modelo.

```
interpreter= tflite.Interpreter(model_path='/home/osmani/TESIS COMPLETE/ENTREGA FINAL/MODELOS TF LITE/siames_red_nov_Tesis.tflite')
interpreter.allocate_tensors()
```

Ilustración 93 Ruta de Modelo TF Lite

3.2.3 Comparación de Rostros

Una vez cargado el modelo TF-LITE se creó la función para comprar rostros.

```
def comparar_rostros(image):

    input_details=interpreter.get_input_details()
    output_details = interpreter.get_output_details()
    Dir_prueba='/home/osmani/TESIS COMPLETE/ENTREGA FINAL/RESULTADOS POSITIVOS'
    Dir_image=image
    mejor_score = float('inf')
    mejor_imagen = None
    for image in os.listdir(Dir_prueba):
        validacion=os.path.join(Dir_prueba,image)
        img1 =load_image(Dir_image)
        img2 = load_image(validacion)
        # Expandir dimensiones para enviar al modelo
        interpreter.set_tensor(input_details[0]['index'],img1)
        interpreter.invoke()
        embedding1 = interpreter.get_tensor(output_details[0]['index'])

        interpreter.set_tensor(input_details[0]['index'],img2)
        interpreter.invoke()
    embedding2 = interpreter.get_tensor(output_details[0]['index'])
    # 7 Interpretar elZZZZZ resultado (Ejemplo con distancia euclidiana)
    dist= euclidean_distance(embedding1, embedding2)
    similaridad=1-dist
    #print(f"Distancia Euclidiana entre imagenes: {similaridad,image}")
    threshold = 0.8
    if similaridad > threshold:
        print(f"{similaridad,image}:Las imagenes son Similares segun la Red siamesa ")
        data_server(image,similaridad)
    #else:
        #print(f" las imagenes No son Similaares sen gun la Red Siamesa{similaridad,image}")
```

Ilustración 94 Función de comparación de Rostros

Como se puede apreciar en la **ilustración 94** es el código que permite comparar los rostros. El primer paso es obtener las entradas y Salidas del modelo preentrenado, de igual manera se declara la dirección de la carpeta donde se encuentran los rostros a los cuales la imagen en tiempo real se comparará para detectar el parecido.

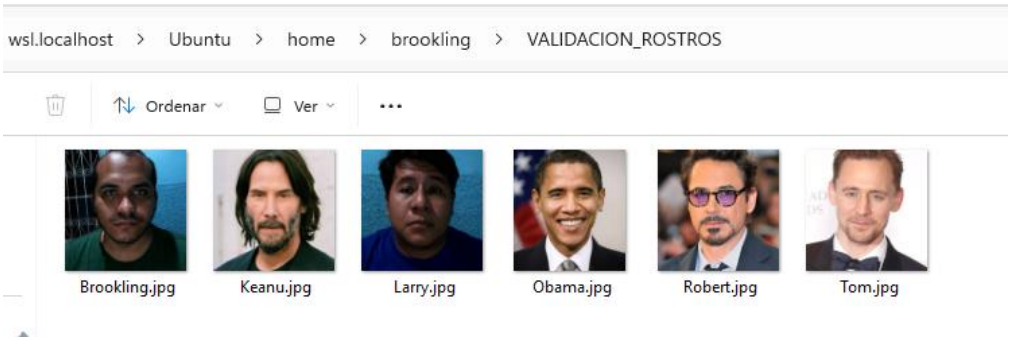


Ilustración 95 Directorio de Imágenes a comparar

Una vez declarada la carpeta en donde se encuentran las clases a comparar se procede a llamar a las distancias euclidianas, las cuales arrojaran un valor de entre 0 a 1 en donde, la configuración, el valor 0 es el más lejano y el valor 1 es el más cercano, para ello se emplea la variable **Threshold** el cual permite filtrar. Por ejemplo, una persona que no está dentro de estas clases se pone frente a la cámara el resultado por imagen sería muy bajo, por lo cual, al no pasar ninguna imagen al valor mínimo de disparo, se enviara a imprimir “Sin Resultado”, mientras si una de las personas de estas clases se pone en frente de la cámara disparará el resultado y se envía a guardar los resultados de la imagen comparada.

3.2.4 Adición de Camara al código

Una vez se tiene la programación cargada para el modelo preentrenado con la función **Comparar Rostros,** se procede a cargar el código, el cual permitirá con la cámara hacer las comparaciones en tiempo real.

Para ello se conecta vía USB al puerto 2.0 a la Raspberry Pi 5. Para la apertura de cámara en la Raspberry Pi es muy importante contar con el siguiente comando: **ls /dev/video*** esto mostrará automáticamente todas las entradas de video, deberá aparecer algo como lo siguiente:

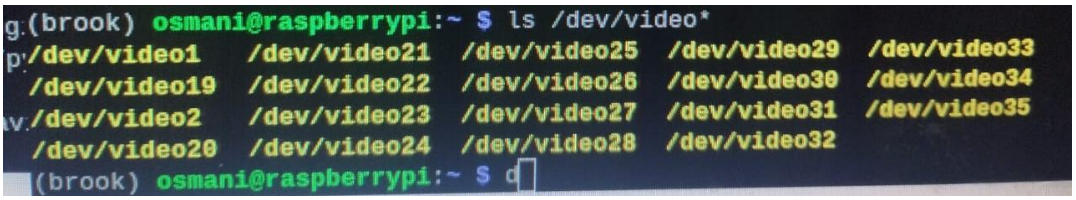


Ilustración 96 Lista de Cámaras

3.2.5 Apertura de Camara

Para la apertura de cámara se emplea un código simple (El cual ya se ha empleado previamente en este trabajo monográfico).

```
cap = cv2.VideoCapture(0)
while True:# se ingresa aun ciclo para aperturar la camara.
    # Leer un frame
    ret, frame = cap.read()
    frame = cv2.rotate(frame, cv2.ROTATE_180)
    cv2.imshow('Camara OpenCV',frame)
            #cv2.destroyWindows('esperando')
            # Presiona 'q' para salir


    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Liberar la cámara y cerrar la ventana
cap.release()
cv2.destroyAllWindows()
```

Ilustración 97 Apertura de Camara

Para la apertura de la cámara es muy importante el cambiar el dato que está en paréntesis en **cv2.VideoCapture(),** esto debido a que en dependencia del número de cámaras se debe de ajustar para llegar a la cámara correctamente.

Nota: Se usa Rotate dado a que la cámara que se está utilizando muestra las imágenes al revés.

Una vez se tiene que la cámara funcionó correctamente se crea un código en donde se llama a la función de **Comparador de rostros**, cabe destacar que se adiciono un condicional el cual si no hay un rostro detectado se envía a imprimir una imagen que se llama “Esperando Rostro”, es una pequeña medida que adiciona automatización y una imagen agradable a la vista. Una vez se posa el rostro, esta imagen desparece.

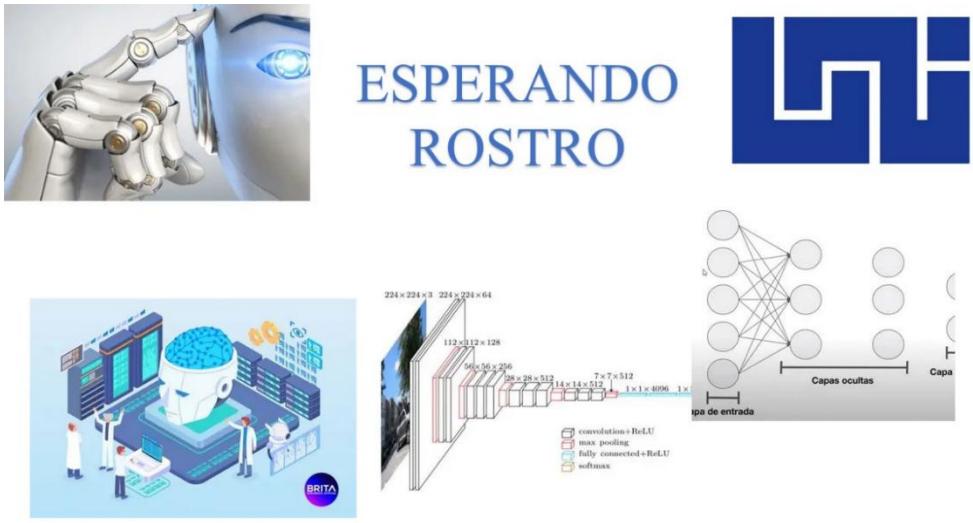


Ilustración 98 Wait_ Image Imagen de espera

La programación en tiempo real es la siguiente, la cual crea un directorio, donde se envían las imágenes que captura la cámara.

```
5]: def real_time_tesis():
        ultimo_tiempo=0
        detector = dlib.get_frontal_face_detector()
        resultado='/home/osmani/TESIS COMPLETE/resultado/prueba.jpg'
        # Iniciar la captura de video (0 para la cámara predeterminada)
        cap = cv2.VideoCapture(0)# captura el video debera, colocarse la caamara que esta co
        wait_image='/home/osmani/Descargas/wait_rostro.jpeg'# Esta imagen servira como una s
        wait_image=cv2.imread(wait_image)# se lee la imagen.
        wait_image= cv2.resize(wait_image, (600, 400))# se hace un resize porque la imagen o
        while True:# se ingresa aun ciclo para aperturar la camara.

            # Leer un frame
            tiempo_actual=time.time()
            ret, frame = cap.read()
            frame = cv2.rotate(frame, cv2.ROTATE_180)
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            # Suavizar la imagen para reducir ruidqo
            faces = detector(gray)
            if len(faces)>0 :
                #cv2.imwrite(resultado, frame)
                cv2.destroyWindow('esperando')
                # Dibujar rectángulos alrededor de cada rostro detectado
                for face in faces:
                    x, y, w, h = (face.left(), face.top(), face.width(), face.height())
                    recorte = cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
                    cv2.imwrite(resultado, recorte)

                    ########################################
                    # Mostrar el frame
                    #comparar_rostros(frame)
                    rostro_recortado = frame[y:y+h, x:x+w]  # Extraer solo el rostro
                    comparar_rostros(resultado)

                    cv2.imshow('Camara OpenCV',frame)
            else:
                cv2.imshow('esperando',wait_image)
            ##############################
```

Ilustración 99 Real Time Tesis (Rostros)

3.2.6 Programación para el Plate-Recognición (Detection)

Para la programación con las Placas, se utilizaron los mismos principios que la programación de rostros, la única variable es la librería para la detección del objetivo, en vez de rostros en este punto son “Rectángulos”, por ende, lo más destacado a resaltar dentro de la programación es el uso del detector de rectángulos que se abordó en el Capítulo 2.

Al integrar el modelo preentrenado de detección de placas más el reconocimiento de Placas se cierra un circuito el que permite llegar desde la etapa de captura de datos hasta la etapa de Clasificación. Haciendo uso de la red Siamesa el código de placas a diferencia del código de rostros usa 2 Modelos preentrenados, el modelo preentrenado hecho en este trabajo monográfico y el modelo preentrenado de github, descargado de repositorios que permite la detección de Rectángulos o Plate Detection.

De igual manera que el código previamente explicado para rostros, se crea una carpeta (directorio) donde se contienen las placas a comparar y se envía a una carpeta el resultado de la cámara al pasar por el sistema.

Capítulo IV: Base de Datos y Servidor

Para la base de datos y servidor se utilizó MongoDB el cual funge como base de datos con un servidor integrado. Para la activación del servidor y alimentación de la BD se usó la Raspberry Pi, para la visualización de datos se usó **Compass** en el computador. El propósito de la BD es recibir los datos en tiempo real de cada comparación (Face-Recognition o Plate-Recognition) que haga el sistema ya sean positivos o negativos, al igual que el propósito del servidor es poder ver dichos datos desde una página web o en esta oportunidad un programa de versión gratuita, dado a esto, se preparó el siguiente código.

4.1 Código en Raspberry PI (MongoDB)

```
#Libreria para BD
from pymongo import MongoClient
from datetime import datetime  # Agregado para usar datetime.now()
cliente = MongoClient("mongodb://localhost:27017/")
# 2. Crear una base de datos nueva (Mongo la crea cuando se use)
nueva_db = cliente["ROSTROS"]
```

Ilustración 100 Dirección IP de MongoDB

Para el cliente en MongoDB se utilizó el Local-host, cabe mencionar que la dirección IP puede ser cambiada al igual que adicionar capas de seguridad, por practicidad se decidió usar el local Host, al igual que no incluir contraseñas.

Se procedió a crear una función de programación, pero antes de ella se definen los criterios o la información que será registrada en la base de datos.

```
#data a server, se extrae la informacion que ira al servidor, por este medio
def data_server(name,score):
    # Elimina la extensión
    nombre_sin_ext = os.path.splitext(name)[0]
    # Divide usando el guión bajo "_"
    partes = nombre_sin_ext.split("_")
    # Asignamos a variables
```

Ilustración 101 Función MongoDB Database-Server

4.2 Criterios para la BD (Face-Recognition).

4.2.1 Criterios para Rostros:

```
documento = {
"Fecha": datetime.now(),
"Nombre": nombre,
"Apellido": apellido,
"ESTATUS":status,
"Carnet": carnet,
"Edad": edad,
"Carrera":carrera,
"% Parecido":score,
"prueba": "ACK"    }
resultado = nueva_coleccion.insert_one(documento)
# 6. Confirmar que se insertó correctamente
print("Documento insertado con _id:", resultado.inserted_id)
```

Ilustración 102 Criterio de BD-Servidor para Rostros

Para los rostros al ser un control de acceso se está tomando que para cada validación positiva se tome fecha y hora, nombre, apellidos, estatus (Estudiante, Docente, Trabajador…etc.), a como este registrado en el sistema de la universidad.

4.2.2 Criterios para Placas:

```
placa=partes[6]
#################################
nueva_coleccion = nueva_db["PLACAS_UNI"]
documento = {
"Fecha": datetime.now(),
"Nombre": nombre,
"Apellido": apellido,
"ESTATUS":status,
"PLACA": carnet,
"Edad": edad,
"Carrera":carrera,
"% Parecido":score,
"Placa":placa,
"prueba": "ACK"    }
resultado = nueva_coleccion.insert_one(documento)
# 6. Confirmar que se insertó correctamente
print("Documento insertado con _id:", resultado.inserted_id)
```

Ilustración 103 Criterios BD-servidor para Placas

Como se puede observar en la **Ilustración 103** los criterios para la placa son iguales que los rostros, solamente se añade el arreglo “Placa”.

Posterior ya definidos los criterios, la programación de comparación de Placas y comparación de Rostros deben alimentar las bases de datos.

4.3 Compass

Compas es la herramienta utilizada para conectar gráficamente (GUI) la BD de Mongo db, permite vía conexión IP, visualizar, editar, adicionar, exportar, al igual que, segmentar la base de dato.

Para ello se debe de instalar el programa en el computador, descargándolo desde la página oficial, una vez descargado se deberá ejecutar.

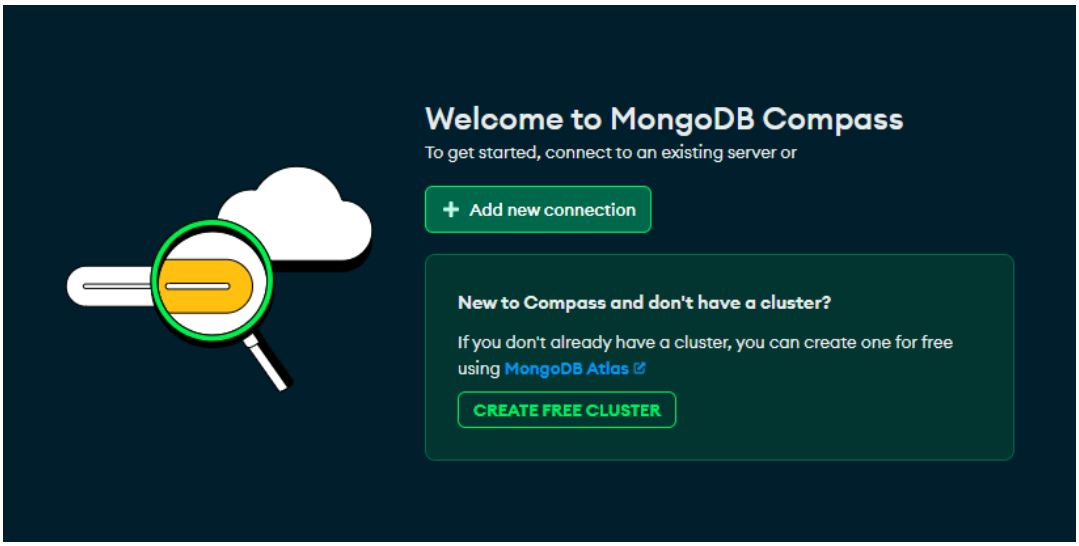


Ilustración 104 Mongodb Compass

Una vez se tiene el programa abierto, se debe de ingresar la IP a la cual se apuntara, como el programa se esta ejecutando en la Raspberry PI, se debe de conocer, la Ip de la raspberry, para ello se usara el siguiente comando “Ifconfig”, el cual muestra toda la información ip, se deberá obtener algo como lo siguiente:

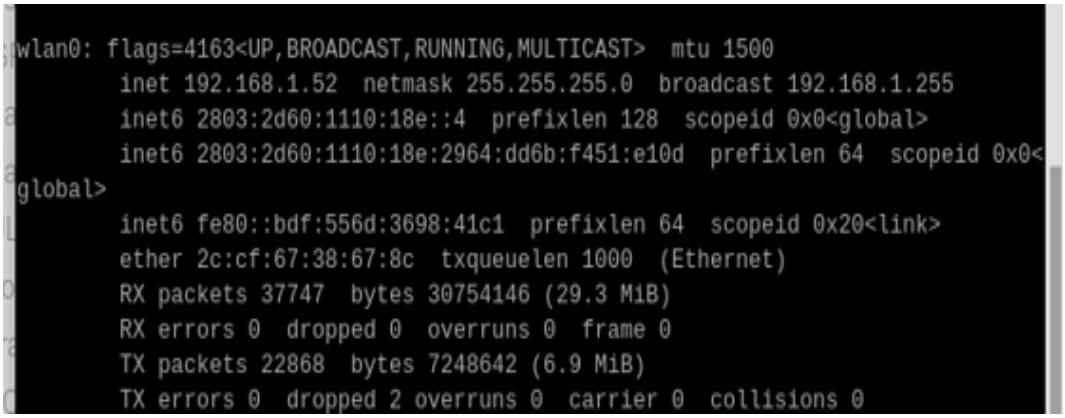


Ilustración 105 Dirección Ip de Raspberry Pi

Cabe destacar que, como la configuración del Raspberry Pi referente a su dirección ip, se encuentra en constante cambio por el protocolo TCP/IP, se debe de ajustar la dirección IP de MongoDB en Compass.

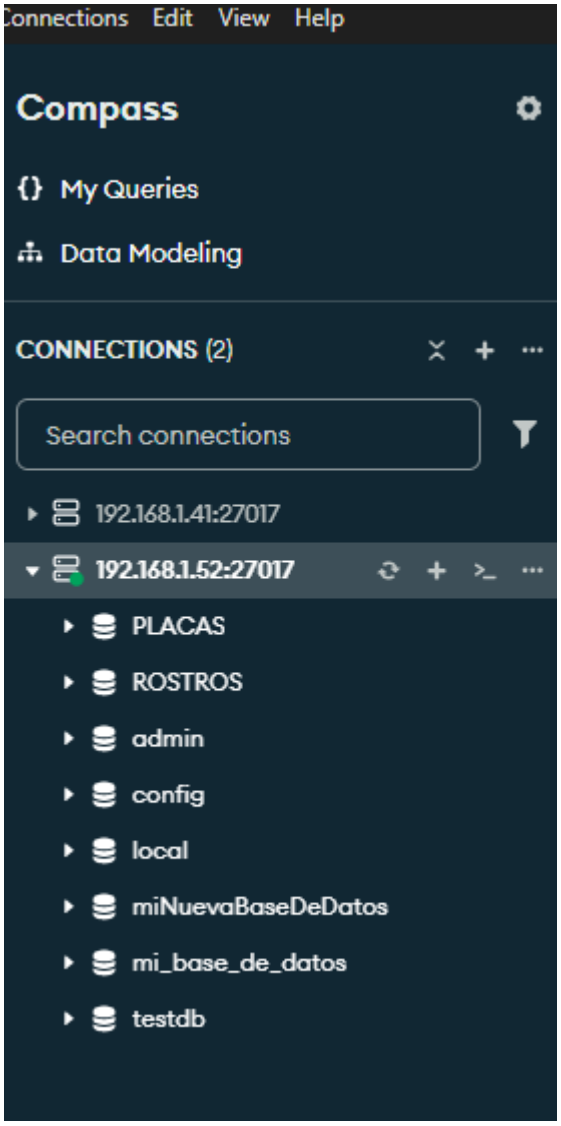


Ilustración 106 COMPASS MONGO DB

Como se puede apreciar en la ilustración se tienen diferentes bases de datos, desde el programa compass se puede acceder a cada una de ellas, obtener información, ver en tiempo real.

Ilustración 107 Compas BD-Server

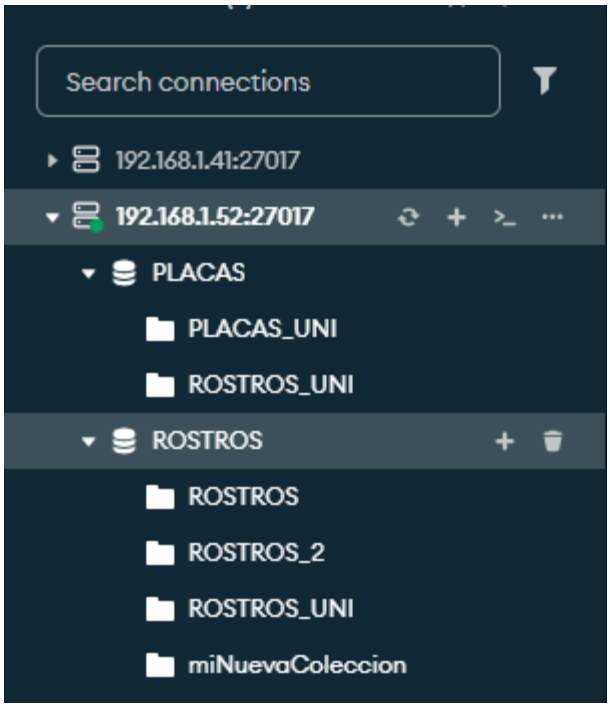


Ilustración 108 MONGO DB CONECTADO

Se logran ver todas las BD creadas (desde pruebas hasta las versiones que ya están definidas como definitivas en programación).

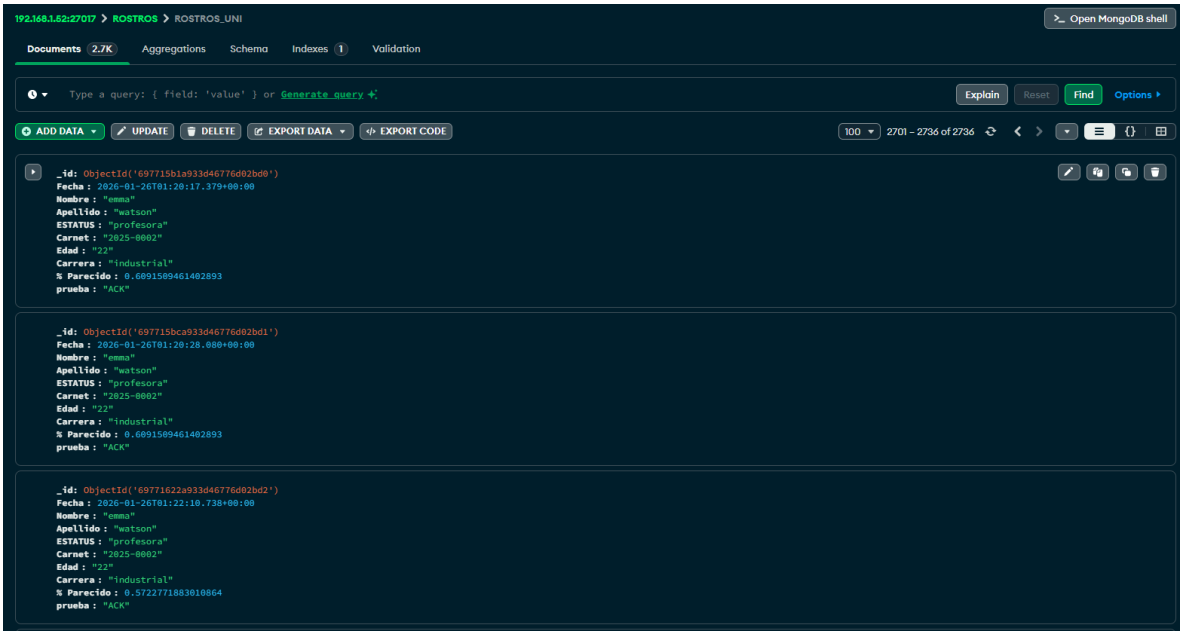


Ilustración 109 BD Rostros MongoDB Compass

Como se puede apreciar en la ilustración desde Compass se puede ver toda la data, se puede filtrar, se puede eliminar, exportar, actualizar, al igual que aperturar una comunicación directa con la Raspberry para crear nuevas bases de datos desde MongoDB-Shell.

Capítulo V Resultados

5.1 Reconocimiento facial

Ya una vez explicado cómo funciona el código, se procederá a mostrar la ejecución de este mismo, para ello, se usaron las clases para rostros en tiempo real, los resultados se pueden ver impresos en las líneas de código al igual que en la base de datos.

5.1.1 Prueba 1.

Persona a probar Br. Larry.

Como se puede apreciar en la siguiente ilustración, en la salida cruda se tiene la comparación en tiempo real de la cámara con todas las imágenes de la ruta (directorio) en donde se encuentran imágenes de las diferentes clases, y como se podrá apreciar en la ilustración, solamente se obtiene un resultado con “Larry”, los valores de las demás imágenes se mantienen en el orden del 0.00 mientras que la imagen Larry

osciló en las pruebas de captura desde 0.67, hasta 0.98, para ello se puede ajustar el Threshold.

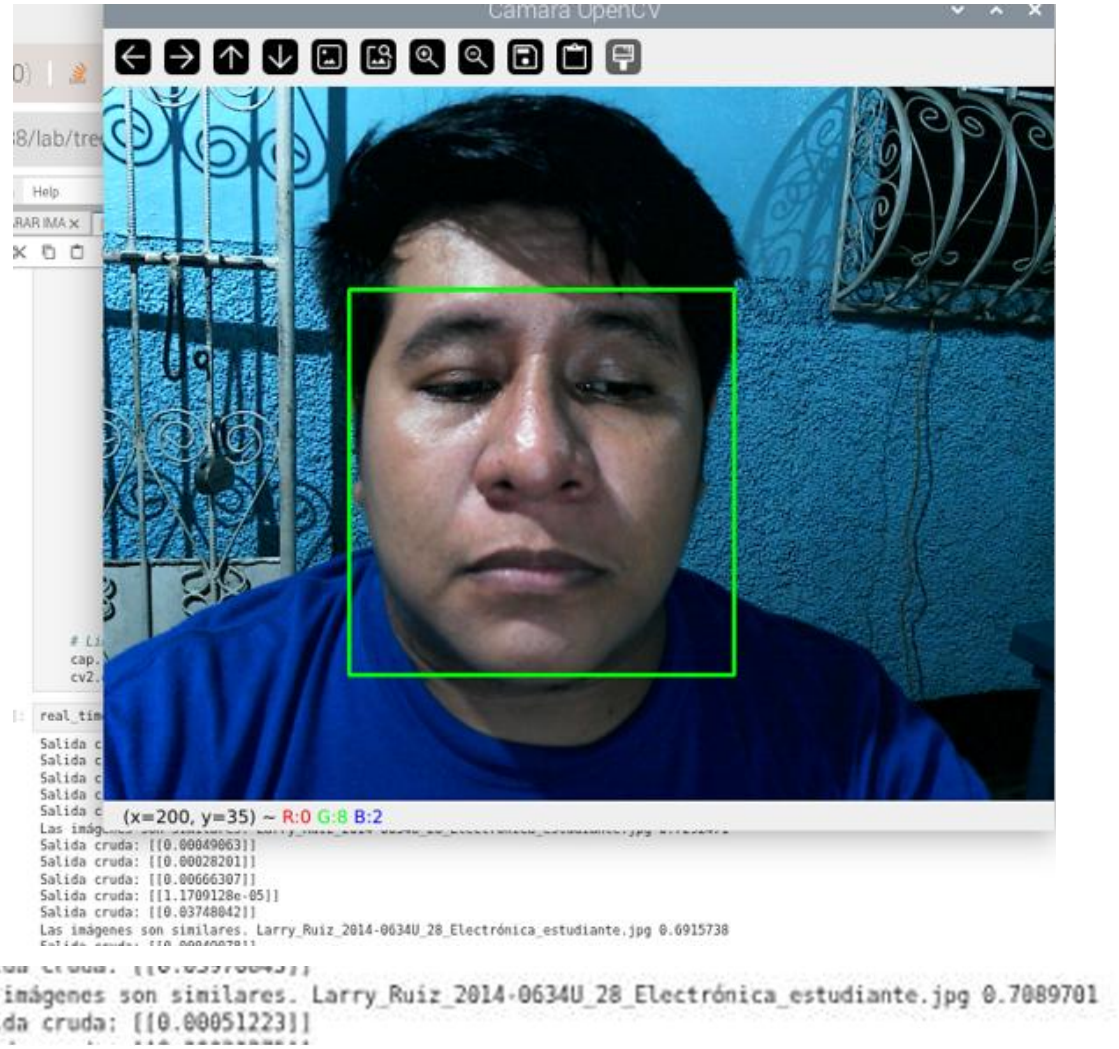


Ilustración 110 Time Real Face Recognition

5.1.2 Prueba 2

Posterior se hizo la prueba con el rostro del Br. Brookling Moncada, y se obtuvieron los siguientes resultados.

Como se puede ver en la ilustración, el resultado que se obtiene es bastante bueno, se tiene un porcentaje de 0.99% de similitud, tomando en cuenta que es una imagen en tiempo real al igual que se está comparando con las imágenes de la carpeta y solo la clase Brookling dispara el Threshold.

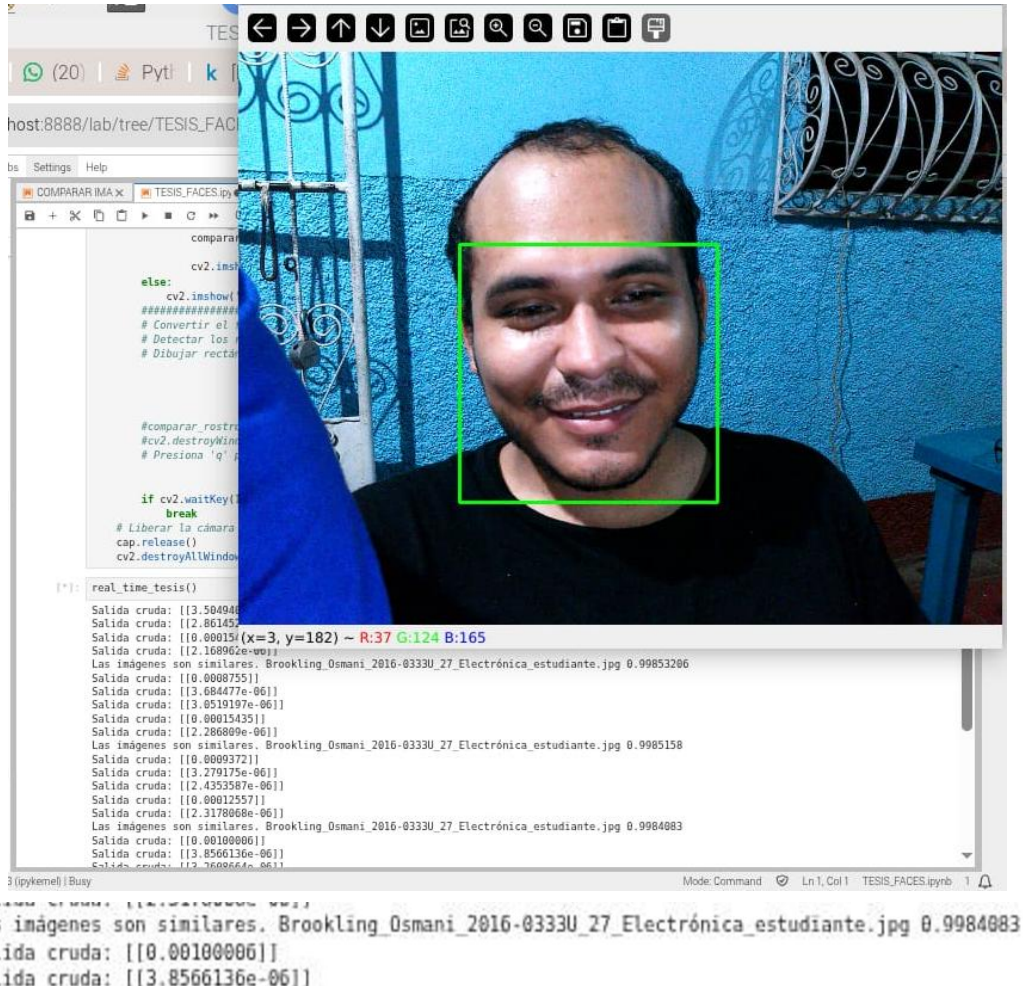
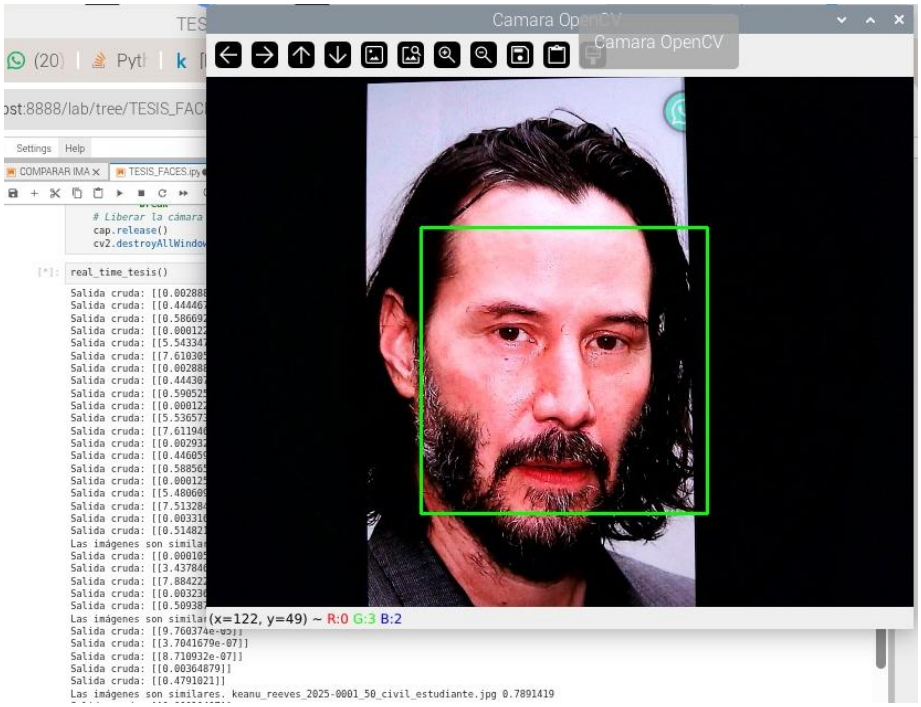


Ilustración 111 Prueba de Reconocimiento facial en tiempo Real

5.1.3 Prueba 3

Posterior se realizó una prueba adicional, la persona a probar como no se tuvo de manera física se buscó una imagen de internet, la clase seleccionada fue la de “Keanu Reeves” el cual es clasificado como estudiante de Ing. Civil.

Como se aprecia en la siguiente **Ilustración 112** se mantiene como resultado adecuado, dado a que la prueba se realizó con una imagen desde el teléfono.



Las imágenes son similares. keanu_reeves_2025-0001_50_civil_estudiante.jpg 0.78577733

Ilustración 112 Tesis Rostros Prueba tiempo Real Keanu Reevs

5.2 Plate-Recognition

5.2.1 Prueba 1

(0.9743654727935791, 'Larry_Ruiz_2014-0634u_28_electrónica_estudiante_MY30132.jpg')
(0.9395849108695984, 'Larry_Ruiz_2014-0634u_28_electrónica_estudiante_MY30132.jpg')
(0.9742885828018188, 'Larry_Ruiz_2014-0634u_28_electrónica_estudiante_MY30132.jpg')
(0.9377584792673588, 'Larry_Ruiz_2014-0634u_28_electrónica_estudiante_MY30132.jpg')

Ilustración 113 Prueba Plate Recognition Placa MY30132

Para las placas se mandó como condicional imprimir solamente la imagen que logre pasar el umbral de Threshold, por lo cual, se tiene que la imagen de prueba supera dicho Umbral, con un valor de parecido con el registro de placa “MY30132” es de 0.98 Aprox, por lo cual se podría decir que es una excelente clasificación.

5.2.2 Prueba 2

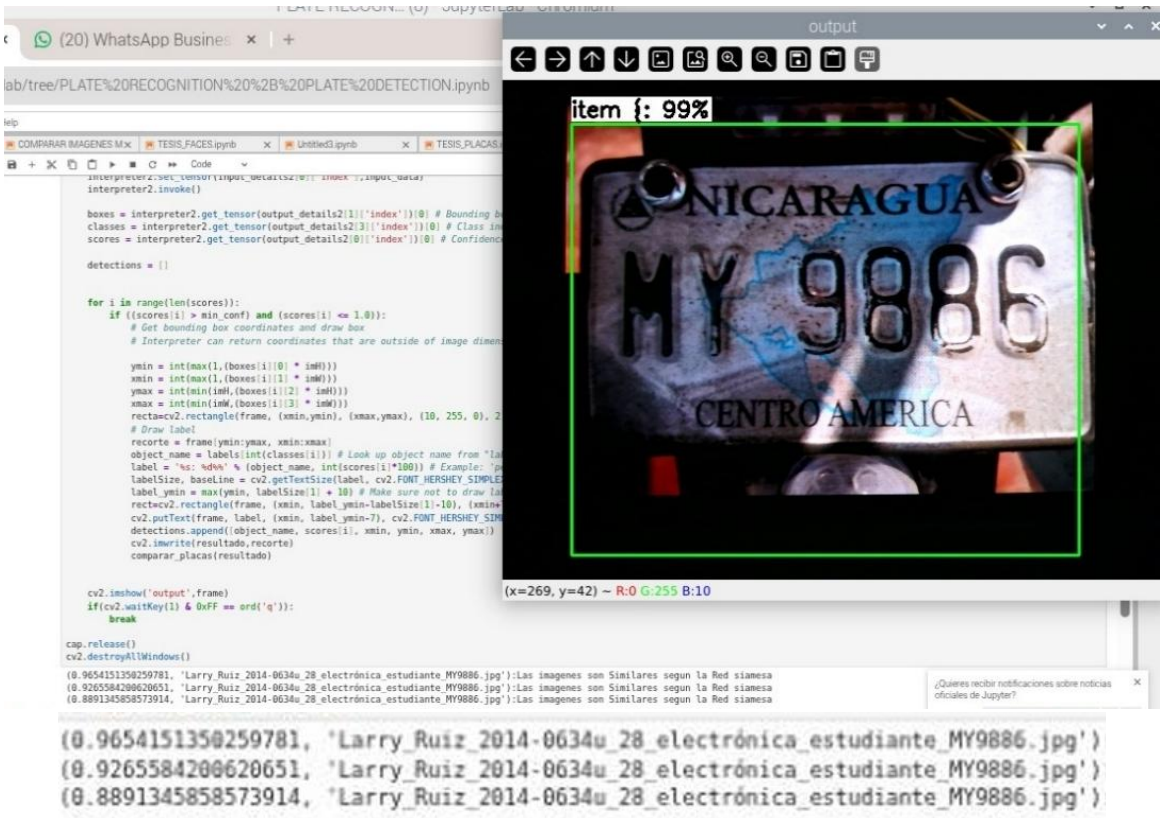
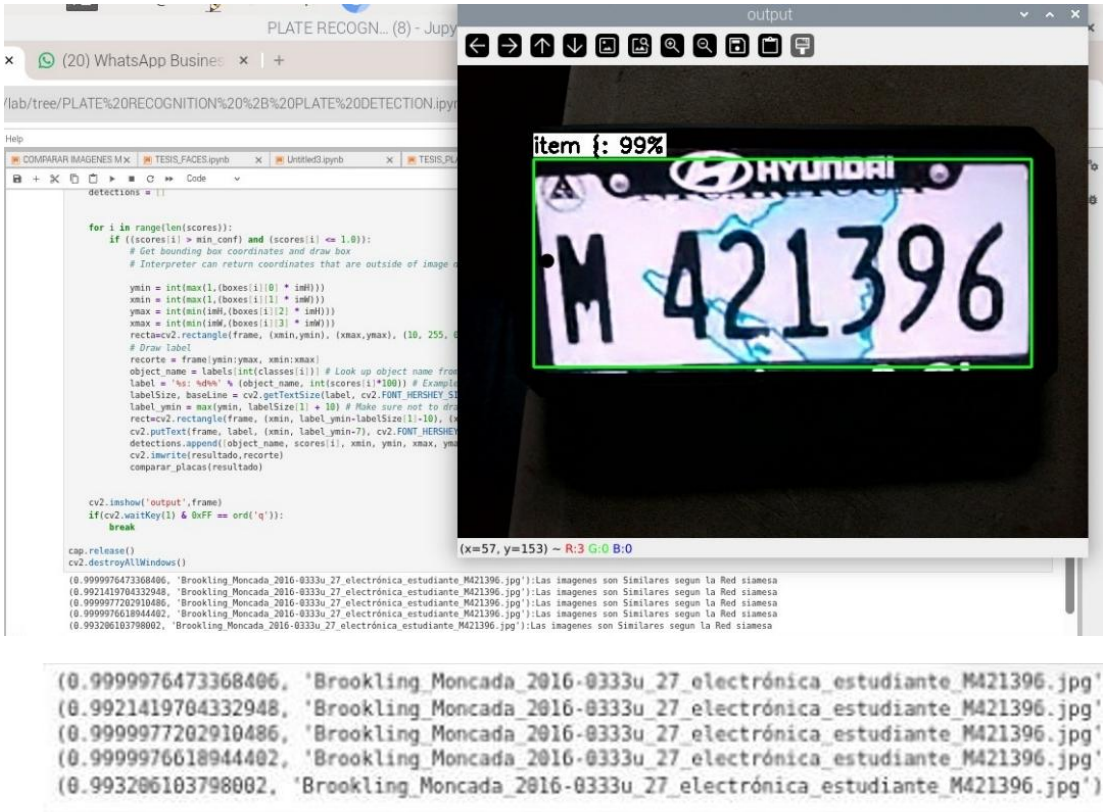


Ilustración 114 Prueba de Placa Numero 2

Como se puede apreciar en la ilustración para esta placa el resultado se mantiene de 88% a 92-96%, lo cual indicia que el modelo está detectando y reconociendo de manera adecuada.

5.2.3 Prueba 3

Para la tercera prueba, se tiene una detección de rectángulos satisfactoriamente, al igual que una detección y reconocimiento satisfactoria de 0.99% de reconocimiento.




```
(0.9999976473368406, 'Brookling_Moncada_2016-0333u_27_electrónica_estudiante_M421396.jpg')
(0.9921419704332948, 'Brookling_Moncada_2016-0333u_27_electrónica_estudiante_M421396.jpg')
(0.9999977202910486, 'Brookling_Moncada_2016-0333u_27_electrónica_estudiante_M421396.jpg')
(0.9999976618944402, 'Brookling_Moncada_2016-0333u_27_electrónica_estudiante_M421396.jpg')
(0.993206103798002, 'Brookling_Moncada_2016-0333u_27_electrónica_estudiante_M421396.jpg')
```

Ilustración 115 Plate Recognition M421396

5.3 Resultados Mongo-DB

Para visualizar la información se ingresó a Compass donde se pueden apreciar los siguientes registros almacenados desde la programación de reconocimiento

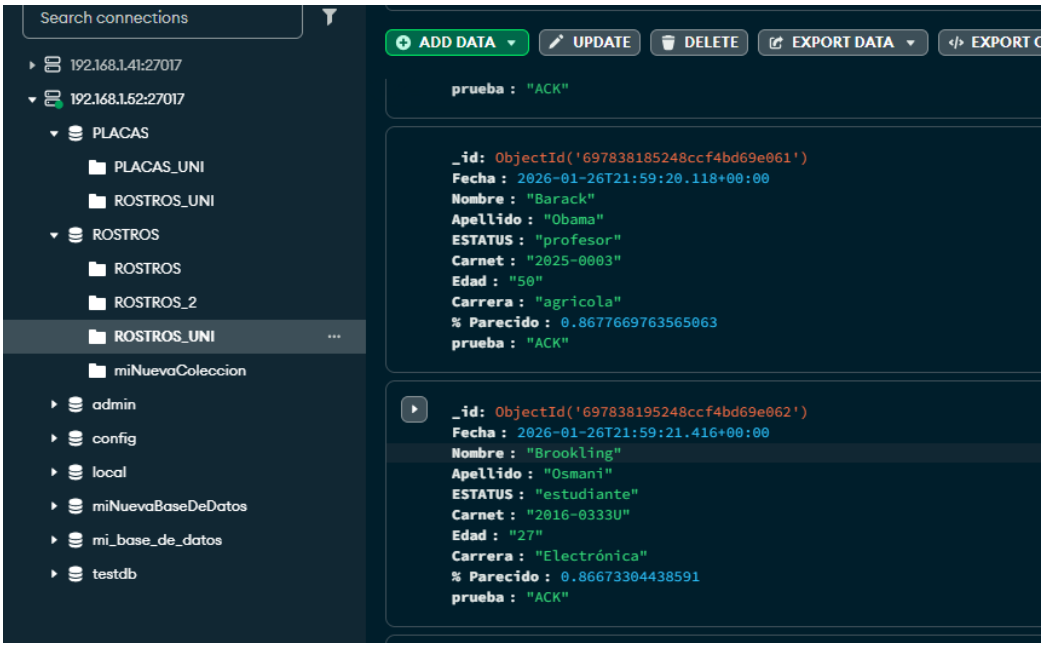


Ilustración 116 Compass Resultados Placas

Conclusiones

Se desarrolló un prototipo que nos permitió adecuadamente lograr detección y reconocimiento tanto de placas como de rostros utilizando la tarjeta de desarrollo Raspberry Pi 5; además, se utilizó una cámara que nos permite trabajar en ambiente semi controlado al proporcionar luz desde esta misma para lograr una mejor captura y reconocer el objeto/persona en cuestión de manera correcta.

Se usó un código de programación tanto para reconocimiento facial y se usó otro código de programación para el reconocimiento de placas mediante un detector de rectángulos. Se realizaron varios ajustes tanto en el entrenamiento, umbrales de detección, bounding box, entre otros, para así lograr resultados satisfactorios que están plasmados en el apartado de resultados.

Se estructuró el servidor de MongoDb de manera que se pudiera adecuar de acuerdo a las necesidades del proyecto en cuestión. Esto se logró mediante los registros enviados que fueron obtenidos durante la detección y con ayuda de Python enviar los datos para que fueran registrados en dicho servidor. Con ayuda de la interfaz gráfica Compass (GUI) logramos observar en tiempo real estos registros y también acceder al historial registrado.

Se configuró la base de datos que registra las entradas de personas y las placas vehiculares asociadas también al personal que ingresa. Se usó la base de datos MongoDb, más puntualmente PyMongo, que nos facilitó trabajar en el mismo lenguaje de programación implementado durante toda esta tesis.

Recomendaciones

1. Para Raspbery Pi se recomienda, el uso de memorias de Alta velocidad (M.2) , tener una conexión eléctrica estable, para evitar la corrupción de datos, comúnmente en memorias micro SD, evitable con SSD M.2, hacer uso de disipadores y abanicos para evitar daños térmicos.
2. Realizar respaldos del sistema completo, por si llegase a haber un error crítico.
3. En la carga del set de datos, se recomienda para los rostros no recortar de la frente a la boca, si no tomar la mayor cantidad de información del rostro, de ser posible desde la barbilla hasta la cabeza, para hacer más robusto al modelo.
4. Para el aumento de datos se recomienda no ser agresivo con los valores, dado a que un sobre ajuste en el set de datos tal como: ruido, brillo, nitidez o giro, puede causar que la imagen se distorsione tanto que el modelo realmente no aprenda y solo genere Overfitting.
5. Dentro de la metodología para lograr determinar si existe un error en la conversión a Tf-Lite, la mejor solución fue dentro del entrenamiento realizar predicciones, anotar el resultado, posterior realizar predicciones en un modelo Tf-Lite aun en el computador y anotar resultados, por último pasar el modelo a la Raspberry y anotar los resultados, en toda la secuencia con las mismas imágenes el resultado debe ser el mismo, con esto se determina que la conversión es exitosa.
6. Se recomienda para mayor escalabilidad exportar la red de extracción de embeddings, en vez de la red siamés, pues esto permite mayor facilitad al guardar los datos vectoriales y compararlos en una base de datos.
7. En caso de ser implementado este proyecto, se recomienda, el uso de código para detección de profundidad para rostros, esto añade una capa de seguridad importante, de este modo solo una persona real podrá ser reconocida.
8. Se recomienda entrenar, ajustar (calibrar) los modelos de Face y Plate recognition con clases pequeñas, una vez la estructura es funcional se puede escalar, hasta el nivel que el hardware lo permita.
9. Hacer uso de Hard Negatives, es una metodología que permite hacer robusta la red ante los falsos positivos.

Bibliografia

AMAZON AWS. (23 de 12 de 2023). *AWS AMAZON*. Obtenido de https://aws.amazon.com/es/what-is/api/

ARIMETRICS. (s.f.). Obtenido de https://www.arimetrics.com/glosario-digital/framework

AWAN, A. A. (MARZO de 2022). *RADAR*. Obtenido de RADAR: https://www.datacamp.com/tutorial/tutorial-for-recurrent-neural-network

AWS. (2023). Obtenido de https://aws.amazon.com/es/what-is/machine-learning/

AWS. (2023). *AWS AMAZON*. Obtenido de https://aws.amazon.com/es/what-is/artificial-intelligence/

AWS. (2023). *AWS. AMAZON*. Obtenido de https://aws.amazon.com/es/compare/the-difference-between-structured-data-and-unstructured-data/

AWS AMAZON. (s.f.). Obtenido de https://aws.amazon.com/es/what-is/database/

AWS AMAZON. (2023). *AWS*. Obtenido de https://aws.amazon.com/es/nosql/

AWS AMAZON. (s.f.). *AWS*. Obtenido de https://aws.amazon.com/es/what-is/sql/

Block, K. (2020). *Blogthinkbig*. Obtenido de https://blogthinkbig.com/los-padres-de-la-inteligencia-artificial-no-son-del-siglo-xxi

Cañadas, R. (14 de 02 de 22). *Abdatum.* Obtenido de https://abdatum.com/tecnologia/dataset

Ciberseguridad. (s.f.). Obtenido de https://ciberseguridad.com/guias/nuevas-tecnologias/machine-learning/perceptron/

Coppola, M. (12 de abril de 2023). *Hubspot*. Obtenido de https://blog.hubspot.es/website/frontend-y-backend

CristinaOrtega. (s.f.). *QuestionPro*. Obtenido de https://www.questionpro.com/blog/es/modelos-de-machine-learning/

Daniele, A. (28 de Diciembre de 2023). *Wired*. Obtenido de https://es.wired.com/articulos/conoce-a-frank-rosenblatt-creador-de-la-primera-inteligencia-artificial

Dev, S. (07 de 03 de 2024). *BUILT IN*. Obtenido de https://builtin.com/articles/overfitting-vs-underfitting

DIGIZONES LAB. (s.f.). Obtenido de https://digizonelabs.com/que-es-un-framework/

Domino. (s.f.). Obtenido de https://domino.ai/data-science-dictionary/anaconda

ebac. (21 de septiembre de 2023). *EBAC.* Obtenido de https://ebac.mx/blog/frameworks

ERICK, R. (26 de marzo de 2023). *Transistores.com*. Obtenido de https://transistores.info/que-es-un-automata-como-funciona-y-para-que-sirve/

Fernández, E. C. (22 de 06 de 2022). *tokioschool*. Obtenido de https://www.tokioschool.com/noticias/backpropagation-redes-neuronales/

García, F. (12 de abril de 2023). *CodigoSQL*. Obtenido de https://codigosql.top/bases-de-datos/dato/

Huet, P. (13 de Abril de 2023). *OpenWebiners*. Obtenido de https://openwebinars.net/blog/que-son-las-redes-neuronales-y-sus-aplicaciones/

IBM. (04 de 02 de 2024). *IBM*. Obtenido de https://www.ibm.com/es-es/topics/api

IBM. (s.f.). *IBM*. Obtenido de https://www.ibm.com/es-es/topics/neural-networks

IBM. (s.f.). *IBM*. Obtenido de https://www.ibm.com/mx-es/topics/artificial-intelligence

ICHI.PRO. (2020-2024). *ICHI.PRO*. Obtenido de https://ichi.pro/es/que-es-un-perceptron-conceptos-basicos-de-las-redes-neuronales-216397265290640

ICHIPRO. (2024). Obtenido de https://ichi.pro/es/perceptron-explicacion-implementacion-y-un-ejemplo-visual-66582574588625

Islas, X. (08 de Febrero de 2021). *Crehana*. Obtenido de https://www.crehana.com/blog/transformacion-digital/que-es-perceptron-algoritmo/

JESUS. (7 de ENERO de 2020). *DATASMARTS*. Obtenido de https://datasmarts.net/es/que-es-un-optimizador-y-para-que-se-usa-en-deep-learning/

KELTA, Z. (SEPTEMBER de 2022). *RADAR* . Obtenido de https://www.datacamp.com/blog/yolo-object-detection-explained

Koidan, K. (21 de 07 de 2023). *learnsql*. Obtenido de https://learnsql.es/blog/que-es-sql/

Maldeadora. (2017). *Platzi*. Obtenido de https://platzi.com/blog/bases-de-datos-que-son-que-tipos-existen/

Maldeadora. (2018). *Platzi.* Obtenido de https://platzi.com/blog/que-es-frontend-y-backend/

Mathworks (Matlab(. (s.f.). *Mathworks*. Obtenido de https://es.mathworks.com/discovery/neural-network.html

NA8. (12 de 12 de 2017). *APRENDE MACHINE LEARNING*. Obtenido de https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/

Na8. (12 de Septiembre de 2018). *AprendeMachineLarning.com*. Obtenido de https://www.aprendemachinelearning.com/breve-historia-de-las-redes-neuronales-artificiales/

Navarro, S. (12 de 04 de 2024). *KeepCoding.* Obtenido de https://keepcoding.io/blog/tensores-todo-lo-que-necesitas-saber/#:~:text=Los%20tensores%20son%20una%20generalizaci%C3%B3n%20de%20

vectores%20y,es%20un%20tensor%20bidimensional%20o%20de%20segundo%20orden.

Navarro, S. (07 de 04 de 2024). *KEEPCODING*. Obtenido de https://keepcoding.io/blog/underfitting-vs-overfitting/

Navarro, S. (17 de 04 de 24). *KEEPCODING.* Obtenido de https://keepcoding.io/blog/que-son-datasets/

Oracle. (s.f.). Obtenido de https://www.oracle.com/mx/database/what-is-database/

Paredes, J. (01 de Junio de 2023). *EBAC*. Obtenido de https://ebac.mx/blog/jupyter-notebook

Pires, L. (06 de 05 de 2024). *ESET*. Obtenido de https://www.eset.com/latam/blog/cultura-y-seguridad-digital/quien-fue-claude-shannon-y-como-revoluciono-la-era-digital/

Redes Neuronales Artificiales . (2023). Obtenido de https://red-neuronal-artificial.gitbook.io/neuronas/contenido/1.-introduccion-a-las-redes-neuronales./1.1-arquitectura-de-una-neurona-artificial/perceptron

Rubio, N. M. (3 de febrero de 2022). *Psicologia y mente*. Obtenido de https://psicologiaymente.com/cultura/test-turing

Team, D. S. (28 de Marzo de 2021). *Data Science*. Obtenido de https://datascience.eu/es/aprendizaje-automatico/funcion-de-activacion-relu/

Tecnologia, I. y. (17 de 02 de 2023). *Unir.net*. Obtenido de https://www.unir.net/ingenieria/revista/backpropagation/

telefonia tech. (s.f.). *aiofthings*. Obtenido de https://aiofthings.telefonicatech.com/recursos/datapedia/datos-semi-estructurados#:~:text=Datos%20semi-estructurados%20no%20tienen%20un%20esquema%20definido.%20No,se%20les%20conoce%20como%20no%20relacionales%20o%20NoSQL.

THE 365 TEAM. (21 de 04 de 2023). *365 DATA SCIENCE*. Obtenido de https://365datascience.com/tutorials/machine-learning-tutorials/overfitting-underfitting/

THE DATA SCHOOL. (s.f.). Obtenido de https://thedataschools.com/que-es/data-set/

The Data School . (2024). *The Data School* . Obtenido de https://thedataschools.com/que-es/data-set/

Tokio. (07 de 08 de 2018). *TokloSchool*. Obtenido de https://www.tokioschool.com/noticias/alan-turing/

Unicoos. (20 de 09 de 2023). Obtenido de https://www.unicoos.com/blog/alan-turing/#:~:text=En%201950%2C%20Turing%20public%C3%B3%20un%20art%C3%ADculo%20seminal%20titulado,plante%C3%B3%20la%20famosa%20pregunta%3A

%20%C2%AB%C2%BFPueden%20pensar%20las%20m%C3%A1quinas%3F%C2%BB.

Velasco, R. (28 de Noviembre de 2021). *SZ SOFT ZONE*. Obtenido de https://www.softzone.es/programas/lenguajes/que-es-sqlite/

writer. (12 de Diciembre de 2022). *Pcweb.info*. Obtenido de https://pcweb.info/datos-semiestructurados-definicion-que-son-tipos-ventajas-y-desventajas/

AWS. (2023). Obtenido de https://aws.amazon.com/es/what-is/machine-learning/

AWS. (2023). *AWS AMAZON*. Obtenido de https://aws.amazon.com/es/what-is/artificial-intelligence/

Bergmann, D. (2025). *IBM*. Obtenido de https://www.ibm.com/es-es/think/topics/zero-shot-learning

Bogado, U. (19 de Septiembre de 2023). *NodoExo*. Obtenido de https://nodoexo.com/redes-neuronales-convolucionales/

Bogado, U. (19 de Septiembre de 2023). *NodoExo*. Obtenido de https://nodoexo.com/redes-neuronales-convolucionales/

IBM. (s.f.). *IBM*. Obtenido de https://www.ibm.com/mx-es/topics/artificial-intelligence

IBM. (s.f.). *IBM*. Obtenido de https://www.ibm.com/es-es/topics/neural-networks

Jawabreh, A. (31 de Marzo de 2023). Obtenido de https://medium.com/the-modern-scientist/multi-task-cascaded-convolutional-neural-network-mtcnn-a31d88f501c8

KEITA, Z. (abril de 2024). *Data Camp*. Obtenido de https://www.datacamp.com/es/tutorial/introduction-to-convolutional-neural-networks-cnns

Kelta, Z. (abril de 2024). *DataCamp*. Obtenido de https://www.datacamp.com/es/tutorial/introduction-to-convolutional-neural-networks-cnns

Luca, D. D. (2025). *Damían De Luca*. Obtenido de https://damiandeluca.com.ar/la-importancia-de-cuda-en-el-desarrollo-de-inteligencia-artificial

MacFarland, A. (9 de Diciembre de 2022). *Unite.ai*. Obtenido de https://www.unite.ai/es/%C2%BFQu%C3%A9-es-el-aumento-de-datos%3F/#:~:text=El%20aumento%20de%20datos%20se%20puede%20definir%20como,modelos%20de%20aprendizaje%20profundo%20para%20generar%20nuevos%20datos.